Data clustering using controlled consensus in complex networks

Thiago H. Cupertino *, Jean Huertas, Liang Zhao

Institute of Mathematical Sciences and Computing, University of São Paulo, Av. Trabalhador São-carlense 400, São Carlos, São Paulo 13560-970, Brazil

ARTICLE INFO

Article history:
Received 3 April 2012
Received in revised form 18 December 2012
Accepted 20 February 2013
Communicated by M. Sato-Ilic
Available online 28 March 2013

Keywords:
Data clustering
Network-based learning
Complex networks
Pinning control
Consensus on networks

ABSTRACT

Recently, many network-based methods have been developed and successfully applied to cluster data. Once the underlying network has been constructed, a clustering method can be applied over its vertices and edges. In this paper, the concept of pinning control in complex networks is applied to cluster data. Firstly, an adaptive method for constructing sparse and connected networks is proposed. Secondly, a dissimilarity measure is computed via a dynamic system in which vertices are expected to reach a consensus state regarding a reference trajectory. The reference is forced into the system by pinning control. A theoretical analysis was carried out to prove the convergence of the dynamic system under certain parameter constraints. The results using real data sets have showed that the proposed method performs well in the presence of clusters with different sizes and shapes comparing to some well-known clustering methods.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

Data clustering consists in dividing an input data set into some groups following a desired characteristic: data items within the same group are more similar than data items belonging to different groups. This is an unsupervised learning task, that is, data items are not labeled and the number of groups is not known a priori. The applications of data clustering includes the characterization of groups of customers based on purchasing patterns, categorization of Web documents, clustering of genes and proteins sharing similar features, image analysis, among others [1].

Most data clustering methods based on static models, such as k-means [2], CLARANS [3], DBSCAN [4], CURE [5] and ROCK [6] are very efficient in some cases, although not suitable for all cases. An example for which such algorithms are not appropriate is clustering data sets that contain groups of different spatial shapes, sizes and densities, as those methods generally assume that the clusters are hyper-spherical and have similar sizes.

On the other hand, network-based data clustering methods are able to detect groups with arbitrary shapes because networks are powerful tools to represent topological relations among objects. These methods usually consist of two stages: construction of a network from the original vector based data set, and partition of the network into sub-networks, each one representing a data cluster. In a network, the concept of cluster is a densely connected group of vertices. The connections among different clusters are sparse. A representative method for data clustering based on networks is called CHAMELEON [7], which uses the concept of k-Nearest Neighbor (k NN) on a sparse network representing the data set. The algorithm searches for the topological structure of the input data, and so it is able to identify data clusters with different shapes. However, the network formation by using k NN presents two main drawbacks: (1) the resulting network is not necessarily connected, and (2) the resulting network may be dense. In these cases, it is difficult to correctly divide the network into meaningful sub-networks.

There also exist many other methods for detecting clusters (or communities) in complex networks. This field is dedicated to the study of large-scale networks with nontrivial connection patterns. A well-known method is based on an iterative removal of edges that present high value of a measure called betweenness [8], resulting in a divisive hierarchical tree of communities. Also, the method of optimal modularity [9] considers the community structure as a statistical arrangement of edges, which can be quantified by using a measure known as modularity [8]. Another method of community detection is based on the concept of collective intelligence [10]. In this method, all vertices are randomly arranged in a circle, so that the angles of each vertex are gradually updated according to the angles of their neighbors. At end of the process, the vertices reach a stationary state in which vertices belonging to the same community lie grouped together. In [11], the authors proposed a method based on particle competition in which particles move through the vertices and compete among themselves to dominate as many vertices as possible. Eventually, each particle will dominate a different community. In [12], the
authors applied the same concept of agents competition to cluster handwritten images of alphanumeric digits. Another method of community detection [13,14] uses a distance measure of complex networks based on a random walk of a Brownian particle in the network. Methods that use the idea of synchronization of coupled oscillators in a network [15], considers that each group of vertices that synchronize in similar times represent a data cluster. When the synchronization is achieved via pinning control [16,17], the communities are synchronized to a common state either in the phase space or in time by controlling some vertices in the network. For a comprehensive and recent review of community detection in graphs refer [18].

The behavior of dynamic systems comprising coupled network of oscillators has been extensively studied [19–22]. Two strongly related concepts have been introduced: consensus and synchronization. Consensus occurs in a coupled network when a common state is reached by all vertices. If each vertex is represented by an oscillator, consensus can be implemented through synchronization. An important strategy to achieve consensus is the “pinning control”, in which control is applied to a small portion of vertices. The synchronized trajectory or the desired homogeneous state can be directly derived from the consensus time concept. Given a pinned vertex, $s(t)$ is the reference trajectory and $g_i$ quantifies the pinning control gain for vertex $i$. Derivation from Eqs. (2) to (3) requires that function $h$ be linear, as it is considered in this work. The reference trajectory is described by an independent oscillator $\tilde{s}(t) = f(s(t))$. Therefore $g_i = 0$ if vertex $i$ is not pinned. The task here is to drive the dynamic complex network to $s(t)$ as $t \to \infty$ by pinning some vertices.

2. Consensus and pinning control

In a network of coupled dynamic systems, the term “consensus” stands for reaching an agreement regarding a certain quantity of interest that depends on the states of all agents [21]. The basic idea of consensus is that each agent updates its own state based on the states of its neighbors. At the end of the process, all agents reach a common value [22].

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

$$x_i(t) = \alpha x_i(t) - c \sum_{j=1}^{n} a_{ij} [x_j(t) - x_i(t)],$$

(1)

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

The linear system represented by Eq. (1) is the distributed consensus algorithm proposed in [27]. This system converges to a common state via local interactions. Assuming that the network is unweighted, the sum of the states of all vertices is invariant and the consensus is reached asymptotically; thus, the collective decision $\alpha$ is equal to the average of the initial states of all vertices, that is, $\alpha = 1/n \sum_{i=1}^{n} x_i(0)$. Consensus algorithms with such a property of invariant sum of the states of the vertices are called average-consensus algorithms.

There have been attempts to control the dynamics of a complex network to an arbitrarily desired state, as an equilibrium point or a periodic orbit of the network [23]. One of them, called pinning control, consists in injecting only a small number of local feedback controllers in the network. Considering that the network $\mathcal{N}$ represents a communication network of $n$ coupled oscillators, the continuous time evolution of the $i$th oscillator can be described by

$$x_i(t) = f(x_i(t)) - c \sum_{j=1}^{n} a_{ij} [x_j(t) - x_i(t)],$$

(2)

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

In the pinning control, only a small fraction of vertices in the network is expected to be controlled. Without loss of generality, let the first $p$ vertices be selected to be pinned. Thus, adding control and using the Laplacian matrix $L$, Eq. (2) can be rewritten as follows:

$$x_i(t) = f(x_i(t)) - c \sum_{j=1}^{n} a_{ij} [x_j(t) - x_i(t)] + u_i(t),$$

(3)

where $u_i(t) = g_i [h(x_i(t)) - s(t)]$ is the local feedback controller, $s(t)$ is the reference trajectory and $g_i$ quantifies the pinning control gain for vertex $i$. Derivation from Eqs. (2) to (3) requires that function $h$ be linear, as it is considered in this work. The reference trajectory is described by an independent oscillator $\tilde{s}(t) = f(s(t))$. Therefore $g_i = 0$ if vertex $i$ is not pinned. The task here is to drive the dynamic complex network to $s(t)$ as $t \to \infty$ by pinning some vertices.

3. The proposed dissimilarity measure based on consensus time

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

From Eq. (3), the pinning control can be reduced to the consensus problem in the presence of pinned vertices with a reference fixed trajectory $s(t) = \bar{x}$, where $\bar{x}$ is the desired stationary state. Such a system, with internal coupling function $h(x) = x$ and linear self-feedback $f(x_i(t)) = \alpha x_i(t)$, $\alpha > 0$, is defined by the following discrete-time equation:

$$x_i(t + 1) = \alpha x_i(t) - c \sum_{j=1}^{n} a_{ij} x_j(t) + c u_i(t),$$

(4)

where $c > 0$ is the step size and $u_i(t) = g_i [\bar{x} - x_i(t)]$. For the controlled vertex, $g_i = 0 > 0$ is the control gain; for all other vertices, $g_i = 0$. By using the discretization procedure from Eqs. (3) to (4), one must assure that the convergence is achieved within a time

---

1 Network $\mathcal{N}$ can be described via its Laplacian matrix $L = [d_i] = D_c - A$, where $D_c = \text{diag}(d_1, \ldots, d_n)$ is the degree matrix of $\mathcal{N}$ and $d_i = \sum_{j=1}^{n} a_{ij}$.
t < ∞. As stated in the previous section, the reference trajectory s (t) is achieved as t → ∞ for continuous systems. However, in this work, we consider the discrete system as in Eq. (4) and set a discretization error, for example e = 10^{-3}, to achieve consensus in a reasonable amount of time. This process, although can be seen too rough, is effective for clustering as it is shown in the results section.

Each vertex in the network is likely to converge to the desired state x at different and finite times by pinning a single vertex (for example, x1(0) = x and g1 = z > 0) and satisfying the parameters constraints which are further addressed in the convergence analysis section. All other vertices must start at the same initial state: x2(0), x3(0), ..., xn(0) = β, β ≠ x.

As an example, consider the toy network consisting of 8 vertices depicted in Fig. 1. The parameter setting is x1 = x = 0, β = 1, α = 0.05, ε = z = 0.2.

In this simple example, it can be seen that each vertex reaches x at different times. Specifically, vertices that lie in the same densely connected group as vertex 1 (1–4) represented by lines green and magenta, converge faster than the other densely connected group (5–8), which is represented by lines blue and red.

Fig. 3 shows the consensus dissimilarities between some pairs of neighbors. It must be noted that the calculations result in asymmetric values. In Section 5.2, this issue will be treated as matrix D which must be symmetrized before being used in the data clustering task.

4. Convergence analysis

In order to achieve the desired results when calculating the consensus time dissimilarity, the proposed system in Eq. (4) must be asymptotically convergent. Some works concerning the use of transversal Lyapunov exponents to prove synchronization [28] or control [29] of coupled chaotic maps can be found in the literature. However, the parameter's conditions to achieve an asymptotically stable behavior have showed to be too strong for discrete-time systems, differing the pin control task [30]. On the other hand, for the linear system proposed in this work we are able to establish some specific conditions on the parameter's values to guarantee the desired asymptotic behaviour.

Theorem. Consider a pinned discrete-time system with n coupled elements (in which only a single element is pinned) whose dynamics are governed by Eq. (4). The origin of the system is asymptotically stable for any initial condition x_i(0) if, and only if, the following constraints hold:

\[ 0 < e < \frac{C}{l_{\text{max}}}, \]

\[ e l_{\text{max}} - C < a < e l_{\text{max}} + C, \]

\[ \frac{a}{e} - l_{\text{min}} - C < z < \frac{a}{e} - l_{\text{min}} + C, \]

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

Proof. The system in Eq. (4) can be rewritten as X(t+1) = AX(t), where X ∈ ℝ^n is a state vector containing all vertices and A ∈ ℝ^{n×n} is a symmetric matrix in the following form:

\[ A = \begin{bmatrix} -e l_{1,1} + z & e l_{1,2} & \cdots & e l_{1,n} \\ e l_{2,1} & -e l_{2,2} & \cdots & e l_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ e l_{n,1} & e l_{n,2} & \cdots & -e l_{n,n} \end{bmatrix}, \]

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

The origin of the discrete system in Eq. (4) is asymptotically stable, i.e., the system converges to zero, if, and only if, A is stable or, in other words, if, and only if, r(A) < 1, where r(A) = λ_{max} is the spectral radius of A [31].

The spectral radius (λ_{max}) of matrix A can be limited if the matrix entries are constrained in some interval [a, b]. Let A be a real and symmetric n × n matrix, n ≥ 2, with entries lying on the interval [−a, a], a > 0. Consider λ_{max} and λ_{min} its largest and smallest eigenvalues, respectively, and s(A) = λ_{max} − λ_{min}, the spread of A.
When \( s(A) \) is symmetric to the origin [32],

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

(9)

Therefore, the spectral radius of \( A \) can be limited to \( r(A) < 1 \) by doing \( \lambda_{\text{max}} < 1 \) and \( \lambda_{\text{min}} > -1 \), i.e., \( s(A) < 2 \). So, to guarantee \( r(A) < 1 \), the condition of Eq. (9) is applied to confine each element of the matrix in Eq. (8). Considering the case in which \( N \) is even (i.e., \( s(A) \leq an\sqrt{2} \)) and using \( C = \sqrt{2}/n \), the following results are found on each matrix entry:

For \( a = e_l_j \):

\[
e_l_j n\sqrt{2} < 2 \\
e < \frac{C}{|l_{\text{jmax}}|},
\]

retrieving Eq. (5).

For \( a = a-e_l_j \):

\[
(a-e_l_j)n\sqrt{2} < 2 \\
a < e_l_{\text{jmax}} + C,
\]

and the negative case:

\[
(a-e_l_j)n\sqrt{2} > -2 \\
a > e_l_{\text{jmin}} - C,
\]

retrieving Eq. (6).

For \( a = a-e(l_i+j+z) \):

\[
(a-e(l_i+j+z))n\sqrt{2} < 2 \\
z > \frac{a}{e} - l_{\text{jmax}} - \frac{C}{e},
\]

and the negative case:

\[
(a-e(l_i+j+z))n\sqrt{2} > -2 \\
z < \frac{a}{e} - l_{\text{jmin}} + \frac{C}{e},
\]

retrieving Eq. (7).

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

**Remark.** The above theorem gives sufficient conditions for the discrete-time system in Eq. (4) to be convergent. However, the implications of different possible values to the parameters when applying the system to compute the consensus time dissimilarities and performing data clustering should be discussed. Parameter \( e \) is responsible for the speed of convergence: the greater its value, the faster the convergence of the system. If the system converges too fast, all dissimilarities between all pairs of vertices may be very similar. In this case, the consensus measure will be inappropriate for clustering, once in the clustering task one expects low dissimilarity between close vertices and high dissimilarity between far vertices. Parameter \( a \) acts in the same way. When applying this method convenient values should be chosen for these parameters accordingly to the intervals of convergence. In all simulations performed in this work, the parameters’ values are chosen to achieve high clustering accuracies, that is, setting small values for \( e \) and \( a \), while at the same time to have a strong pinning control (relative large value for \( z \)).

As an illustrative example, for the network depicted in Fig. 1, where \( l_{\text{jmin}} = 0.28, l_{\text{jmax}} = 0.22 \) and \( l_{\text{jmin}} = 0.05 \), the following limits have been found:

\[
0 < e < 3.54, -0.12 < a < 0.22, \quad \text{for } e = 0.2, \\
-0.85 < z < 0.85, \quad \text{for } a = 0.05.
\]

### 5. Data clustering based on consensus time dissimilarity

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

1. Building of a network from the original data set by using the method described in Section 5.1.
2. Detection of data clusters on the constructed network, as described in Section 5.2.

#### 5.1. Network construction

Basically, a data set can be mapped into a network by using one of the following methods: (i) each vertex, representing a data item, is connected to its \( k \)-nearest neighbors (the \( k \)-most similar data items); (ii) each vertex is connected to all vertices within a predefined distance; (iii) more similar vertices have higher probability to be connected than fewer similar vertices. However the drawback of these methods is that they may construct either disconnected or densely connected networks. As a consequence, they are not suitable for reproducing reliable network clusters that correspond to the expected data clusters. In order to overcome this problem, we propose a method of network construction based on the Single-Linkage (SL) [33] clustering heuristic that is capable of constructing connected and sparse networks which, at the same time, tend to keep the cluster structure of the original data set. The steps of the proposed algorithm are described as follows:

1. Select a data set of size \( n \);
2. Create a disconnected network, that is, without any edges, in which each vertex represents an instance from the data set;
3. Put each vertex \( i \) into a different vertex group \( G_i \) (\( n \) initial groups);
4. Construct an adjacency matrix using a similarity measure, for example the Euclidean distance, to represent distances among all groups;
5. Find the two closest groups\(^2\) and denote them by \( G_1 \) and \( G_2 \);
6. Calculate the average dissimilarity among vertices inside each group \( G_1 \) and \( G_2 \), and denote them by \( d_1 \) and \( d_2 \), respectively;
7. Select the \( k \)-most similar pairs of vertices that connect \( G_1 \) and \( G_2 \), and create an edge between each selected pair if its dissimilarity is smaller than a threshold\(^3\) defined as follows:

\[
d_{\text{thr}} = \gamma \max(d_1, d_2),
\]

where \( \gamma > 0 \). This step joins \( G_1 \) and \( G_2 \) into a larger group;
8. Update the adjacency matrix by calculating the dissimilarities between the group formed in step 7 and all other groups;
9. If the number of groups is larger than one, return to step 5;

As an example of the proposed algorithm for network construction, Fig. 4 shows the results for an artificial data set composed of three clusters of different sizes and densities. In this example, three different values for parameter \( k \) are used: \( k = 3, 5 \) and 20. It can be observed that for all cases it results in a connected network with a fair distribution of edges among

\(^2\) According to the SL heuristic, the dissimilarity between two groups is computed as the dissimilarity between the two closest vertices that connect both groups.

\(^3\) The threshold value is based on the assumption that vertices of the same group generally form a uniform density of dissimilarities. If no pair satisfies the threshold condition, an edge is created between the most similar pair to guarantee a connected network. A connected network is necessary for the convergence of all vertices when the dissimilarity measure is computed by using the consensus time dissimilarity as it is explained in Section 3.
clusters, that is, the connections inside a cluster are dense while the connections inter-clusters are sparse.

**Remark.** As it has been mentioned, the above algorithm for network construction uses a combination of the SL and \(k\) NN algorithms. These specific choices have two main purposes. First, the SL heuristic states that the distance between two groups of vertices is given by the distance between their two closest vertices. By doing so, it is guaranteed that when two groups are being connected the connection starts by the closest vertices, that is, the border vertices of each group. As it is reasonable to consider border vertices as a natural division or frontier among different groups, we suppose that, when joining different groups, instead of connecting far away vertices it is better to connect border vertices. Second, it is necessary to define how many vertices will be connected. As in most common clustering cases one may not have the data distribution for each cluster, or even the real number of clusters, we use a parameter \(k\) to set the number of vertices to be connected. As it can be seen in Fig. 4, \(k\) is responsible for strengthening the intra-cluster connections, while keeping inter-cluster connections sparse. Thus, parameter \(k\) is used for model selection.

5.2. Cluster detection

After creating a network by applying the algorithm of the previous subsection, data clusters can be detected. First, the consensus time dissimilarity (Section 3) is calculated for the given network to construct a dissimilarity matrix \(D\). The basic idea is that, for the same cluster, it takes vertices similar time to reach a common state, resulting in similar values in \(D\). Thus, applying a clustering algorithm in \(D\), a hierarchical structure of clusters is obtained. For this purpose, matrix \(D\) must be transformed into a symmetric dissimilarity matrix as input. A simple approach to do that is to take the average dissimilarity between \(d_{ij}\) and \(d_{ji}\). Therefore the symmetric dissimilarity matrix \(D_s\) is defined as

\[
D_s = \frac{(D + D^T)}{2}.
\]
In a brief, the steps to detect clusters using the proposed dissimilarity measure are described as follows:

1. Calculate the asymmetric dissimilarity matrix D as described in Section 3;
2. Transform D into D∗, according to Eq. (11);
3. Apply a hierarchical clustering method on D∗;
4. Choose a partition from the dendrogram.

6. Simulation results and comparisons

The effectiveness and efficiency of the proposed clustering method have been evaluated by simulating 13 real and diverse data sets selected from the UCI machine learning repository [34] as shown in Table 1. As can be seen in this table, the selected encompassed diversity on data domains as well as considered different number of classes, attributes and set sizes (they vary from 3 to 15, 4 to 91 and 132 to 2310, respectively). These data sets contain multi-cluster data, and many of them present complex data distribution with high mixing among clusters. In the simulations, eventual categorical attributes, such as in Balance data set, were treated as numerical.

Before proceeding to parameter analysis and comparison to other techniques, different techniques for hierarchical clustering were tested (step 3 of Section 5.2). To cluster data on matrix D∗, we tested many hierarchical techniques such as SL, Weighted-Linkage (WL) [35], Average-Linkage (AL) [36] and Complete-Linkage (CL) [37]. The accuracy was measured using the Adjusted Rand Index (ARI) [38,39]. Basically, the ARI is the Rand index [40] corrected for chance. Given two clusterings, say \( C_1 \) and \( C_2 \), the ARI can be noted that there exists a high dependency between the cluster dispersion and the optimal value for parameter \( k \).

The influence of parameter \( k \) for network construction (step 7 in Section 5.1) was analyzed in relation to the given input data set. As has been stated before, parameter \( k \) is used for model selection as one may not known a priori the distribution of the data set he/she is dealing with. To analyze its influence on the proposed method, we simulated all data sets in Table 1 optimizing \( k \) in the set \( \{1, 2, ..., 30\} \) by the grid method. The hierarchical technique used for clustering (step 3 of Section 5) was the AL. In all simulations, \( \alpha = 0, \beta = 1, \gamma = 3, \epsilon = 0.0025, a = 0.02 \) and \( z = 0.2 \).

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

Fig. 5 shows the results normalized in the interval [0,1]. It can be noted that there exists a high dependency between the cluster dispersion and the optimal value for parameter \( k \). Actually, the Pearson’s correlation coefficient between both curves is 0.60. This measure lies in the interval [−1, 1], being −1 a total anticorrelation and 1 a perfect linear dependency. Therefore, the value 0.60 means a high correlation between the \( k \) value and the data set cluster dispersion. In other words, when clusters are separated, the usage of a high value for parameter \( k \) can construct networks with well-defined clusters, that is, a large number of connections inside clusters and sparse connections among different clusters. On the other hand, when clusters are less separated or present a mixture among them, the usage of a small value for parameter \( k \) can alleviate the construction of networks with high mixing among clusters.

### Table 1

Information of real data sets used in simulations.

<table>
<thead>
<tr>
<th>Domain</th>
<th># Instances</th>
<th># Attributes</th>
<th># Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hayes-Hoth</td>
<td>132</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Iris</td>
<td>150</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Teaching</td>
<td>151</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Wine</td>
<td>178</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>Image</td>
<td>210</td>
<td>19</td>
<td>7</td>
</tr>
<tr>
<td>Glass</td>
<td>214</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>E. Coli</td>
<td>336</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Libras</td>
<td>360</td>
<td>91</td>
<td>15</td>
</tr>
<tr>
<td>Balance</td>
<td>625</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Vowel</td>
<td>990</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>Yeast</td>
<td>1484</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Wine Q. (Red)</td>
<td>1599</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>Segment</td>
<td>2310</td>
<td>19</td>
<td>7</td>
</tr>
</tbody>
</table>

### Table 2

Clustering accuracy of the proposed method applying different classical hierarchical clustering techniques. The value in parenthesis shows the optimal value of parameter \( k \) used in network construction. The best results are in boldface. Last line shows the averaged results followed by the standard deviation.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Consensus +WL</th>
<th>Consensus +AL</th>
<th>Consensus +CL</th>
<th>Consensus +SL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hayes-Hoth</td>
<td>0.10 (22)</td>
<td>0.10 (5)</td>
<td>−0.01 (1)</td>
<td>0.02 (11)</td>
</tr>
<tr>
<td>Iris</td>
<td>0.88 (23)</td>
<td>0.94 (1)</td>
<td>0.04 (29)</td>
<td>0.57 (13)</td>
</tr>
<tr>
<td>Teaching</td>
<td>0.04 (2)</td>
<td>0.03 (1)</td>
<td>0.00 (3)</td>
<td>0.00 (1)</td>
</tr>
<tr>
<td>Wine</td>
<td>0.42 (23)</td>
<td>0.43 (1)</td>
<td>0.04 (30)</td>
<td>0.12 (1)</td>
</tr>
<tr>
<td>Image</td>
<td>0.48 (15)</td>
<td>0.51 (3)</td>
<td>0.03 (30)</td>
<td>0.31 (11)</td>
</tr>
<tr>
<td>Glass</td>
<td>0.27 (20)</td>
<td>0.26 (3)</td>
<td>0.04 (30)</td>
<td>0.21 (2)</td>
</tr>
<tr>
<td>E. Coli</td>
<td>0.60 (10)</td>
<td>0.51 (6)</td>
<td>0.02 (4)</td>
<td>0.04 (1)</td>
</tr>
<tr>
<td>Libras</td>
<td>0.39 (11)</td>
<td>0.41 (1)</td>
<td>0.03 (27)</td>
<td>0.06 (2)</td>
</tr>
<tr>
<td>Balance</td>
<td>0.11 (7)</td>
<td>0.18 (4)</td>
<td>0.00 (1)</td>
<td>0.00 (1)</td>
</tr>
<tr>
<td>Vowel</td>
<td>0.22 (16)</td>
<td>0.21 (2)</td>
<td>0.00 (30)</td>
<td>0.01 (16)</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.17 (21)</td>
<td>0.17 (2)</td>
<td>0.00 (23)</td>
<td>0.00 (23)</td>
</tr>
<tr>
<td>Wine Q. Red</td>
<td>0.03 (2)</td>
<td>0.02 (1)</td>
<td>0.00 (1)</td>
<td>0.00 (1)</td>
</tr>
<tr>
<td>Segment</td>
<td>0.58 (27)</td>
<td>0.62 (1)</td>
<td>0.00 (30)</td>
<td>0.01 (30)</td>
</tr>
<tr>
<td>Average</td>
<td>0.33 ± 0.26</td>
<td>0.34 ± 0.26</td>
<td>0.02 ± 0.02</td>
<td>0.09 ± 0.16</td>
</tr>
</tbody>
</table>

6.1. Influence of parameter \( k \)

The influence of parameter \( k \) for network construction (step 7 in Section 5.1) was analyzed in relation to the given input data set. As has been stated before, parameter \( k \) is used for model selection as one may not known a priori the distribution of the data set he/she is dealing with. To analyze its influence on the proposed method, we simulated all data sets in Table 1 optimizing \( k \) in the set \( \{1, 2, ..., 30\} \) by the grid method. The hierarchical technique used for clustering (step 3 of Section 5) was the AL. In all simulations, \( \alpha = 0, \beta = 1, \gamma = 3, \epsilon = 0.0025, a = 0.02 \) and \( z = 0.2 \).

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

Fig. 5 shows the results normalized in the interval [0,1]. It can be noted that there exists a high dependency between the cluster dispersion and the optimal value for parameter \( k \). Actually, the Pearson’s correlation coefficient between both curves is 0.60. This measure lies in the interval [−1, 1], being −1 a total anticorrelation and 1 a perfect linear dependency. Therefore, the value 0.60 means a high correlation between the \( k \) value and the data set cluster dispersion. In other words, when clusters are separated, the usage of a high value for parameter \( k \) can construct networks with well-defined clusters, that is, a large number of connections inside clusters and sparse connections among different clusters. On the other hand, when clusters are less separated or present a mixture among them, the usage of a small value for parameter \( k \) can alleviate the construction of networks with high mixing among clusters.

6.2. Comparisons

Eight well-known clustering techniques were simulated for comparison purposes. Five of them are traditional techniques: AL, WL, CL, SL and \( k \)-means [41,42]; the other three are based on eigenanalysis of graph-Laplacian: unnormalized spectral clustering (Spec1) [43], normalized spectral clustering according to Shi and Malik (Spec2) [44] and normalized spectral clustering according to Ng et al. (Spec3) [45].
In all experiments, the proposed method was simulated with the following parameter setting: $\gamma = 0$, $\beta = 1$, $\epsilon = 0.0025$, $\alpha = 0.02$ and $z = 0.2$. Networks were constructed using the method described in Section 5.1 with parameters $\gamma = 3$ and $k$ varying from 1 to 30. We reported here the results using AL, which resulted the best accuracies. For $k$-means, the results were averaged over 100 simulations and each simulation was optimized by selecting randomly and uniformly 10% of data to the initial seed setting. For the spectral techniques, the network construction algorithm in Section 5.1 was applied.

Table 3 shows the results in terms of ARI with the optimal value of parameter $k$ for model selection. Table 4 shows the rank of the
techniques for each data set. To compute the rank, the accuracy achieved by a technique in a data set is ranked among all other methods for the same set. In the last line is the averaged rank over all data sets. The smaller the average rank, the better were the accuracies achieved by the technique. In the tested data sets, the proposed clustering method achieved the best rank (2.38), followed by Spec2 (2.92) and Spec3 (3.54).

7. Conclusions

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

Acknowledgments

The authors would like to acknowledge the São Paulo State Research Foundation (FAPESP) and the Brazilian National Council for Scientific and Technological Development (CNPq) for the financial support given to this research.

References

Thiago H. Cupertino received the B.S. degree on Computer Engineering in 2009, from University of São Paulo, Brazil. In 2008, he was an interchange student at the Polytechnic University of Valencia, Spain, granted with a Santander scholarship. In 2009, he was an interchange student at the Seville School of Engineering, Spain, granted with a European Erasmus Eubranex scholarship. He is currently a Ph.D. candidate at the Institute of Computer Sciences and Mathematics, University of São Paulo, Brazil. His current research interests include data clustering and classification, network learning and nonlinear dynamic systems.

Jean Huertas received the B.S. degree on Systems Engineering in 2007, from University of San Agustin, Peru, and the M.Sc. degree in Computer Science from the University of São Paulo at the Institute of Computer Sciences and Mathematics, Brazil, in 2011. He is currently working as a software engineer at zAgile, Inc. His research interests include artificial neural networks, complex networks, bioinformatics and pattern recognition.

Liang Zhao received the B.S. degree in 1988, from Wuhan University, China, and both the M.Sc. and the Ph.D. degrees from Aeronautic Institute of Technology, Brazil, in 1996 and 1998, respectively, all in computer science. From 1988 to 1993, he worked as a software engineer. He joined the University of São Paulo, Brazil, in 2000, where he is currently an associate professor in the Department of Computer Science. From 2003 to 2004, he was a visiting researcher in the Department of Mathematics, Arizona State University, USA. His research interests include artificial neural networks, nonlinear dynamic systems, complex networks, bioinformatics and pattern recognition. He has received a Brazilian Research Productivity Award and he is an Associate Editor of IEEE Transactions on Neural Networks.