Pattern Recognition 41 (2008) 3592 – 3599

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Fast online graph clustering via Erdős–Rényi mixture

Hugo Zanghi[a,*], Christophe Ambroise[b], Vincent Miele[b]

[a]Exalead, 10 place de la Madeleine, 75008 Paris, France
[b]Statistique et Génome (UMR CNRS 8071, INRA 1152), La genopole Tour Evry 2, 523 place des Terrasses, 91000 Evry, France

ARTICLE INFO

Article history:
Received 2 May 2007
Received in revised form 13 June 2008
Accepted 17 June 2008

Keywords:
EM algorithm
Graph clustering
Online

ABSTRACT

In the context of graph clustering, we consider the problem of simultaneously estimating both the partition of the graph nodes and the parameters of an underlying mixture of affiliation networks. In numerous applications the rapid increase of data size over time makes classical clustering algorithms too slow because of the high computational cost. In such situations online clustering algorithms are an efficient alternative to classical batch algorithms. We present an original online algorithm for graph clustering based on a Erdős–Rényi graph mixture. The relevance of the algorithm is illustrated, using both simulated and real data sets. The real data set is a network extracted from the French political blogosphere and presents an interesting community organization.

1. Introduction

In many scientific fields, systems can be modeled using networks to represent data relationships. World-wide-web, gene interactions, social networks, authors citations, are examples of fields where graph representation helps interpreting relationships between the nodes. As for the web, nodes represent web sites or web pages, and each edge represents a hyperlink relating two nodes.

These so-called real networks share some properties [1] such as small-world phenomenon in the sense that most nodes are close to each other (i.e. six degrees of separation in social networks), scale-free distribution of the degrees (distribution of degrees does not depend on the graph size), degree distribution which obeys a power law (presence of hubs), giant component (connected subgraph that contains a majority of the entire graph's nodes), high clustering coefficient (important aggregative trend of a graph) and preferential attachment (new nodes connect with higher probability to nodes that already have a large number of edges). Real networks have also been shown to be structured into communities or groups. Finding these communities of nodes which have a higher within-group density of edges than between-groups can have a significant importance for interpreting the network. For instance, groups within the world-wide-web might correspond to sets of web pages on related topics [2]; groups within social networks might correspond to social units or communities [3]. The mere fact of finding a network that contains tightly knit groups (the so-called community structure) can convey useful information: if for instance a metabolic network were to be divided into such groups, it could provide evidence for a modular view of the network's dynamics, with different groups of nodes performing different functions with some degree of independence [4].

This detection of community structure is a well-studied problem. Proposed approaches for communities detection take their inspiration from classical clustering algorithms, proposing criterion or transformation of the data adapted to this particular problem. Many different techniques have been developed, based on spectral graph theory, random graph model, optimization of centrality measure or other approaches [5].

Spectral clustering usually involves the use of classical clustering algorithms, such as the kmeans, in a space spanned by the top eigenvectors of a matrix describing the graph. First studied in the context of graph theory [6], spectral methods have become quite popular in the machine learning community, where they are used on Gram (inner-product kernel) matrix [7,8].

Random graphs are a possible model for networks where vertices are given and edges are considered as random variables. The simplest and most studied random graph model is the Erdős–Rényi graph, where each pair of nodes is connected with the same probability p. But Erdős–Rényi graphs do not exhibit most of the properties of real networks, particularly the degree distribution and the high clustering coefficient. Thus alternative models have been developed. For example, mechanistic models [9] have been proposed to model network growth. They are mainly based on informative summary statistics, such as the degree distribution, but do not allow for any statistical inference.

In this paper we focus on the Erdős–Rényi mixture model for graph (ERMG or MixNet), which has been proposed by [10] with an

* Corresponding author. Tel.: +33 1 55 35 3011; fax: +33 1 55 35 2627.
E-mail addresses: hugo.zanghi@exalead.com (H. Zanghi), cambroise@genopole.cnrs.fr (C. Ambroise), vincent.miele@genopole.cnrs.fr (V. Miele).

0031-3203/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2008.06.019

associated EM [11] estimation algorithm, and is not to be confused with exponential random graph models for network data (ERGM) which consider distributions ensuing from the exponential family to model the edge distribution [12]. The MixNet model allows to capture the structure of a network and in particular to detect communities.

There exists a strong connection between MixNet, pairwise clustering and block clustering [13,14]. Block clustering searches for homogeneous blocks in a data matrix by simultaneous clustering of rows and columns. When partitions of rows and columns are assumed to be identical, Bernoulli block mixture model and its associated estimation algorithm [15] is equivalent to MixNet. In this context of block clustering for classical binary data, an almost equivalent MixNet algorithm without any underlying statistical model can be dated back to the early work of Govaert [16,17].

There is also a long tradition of developing statistical graph models for social networks [18,19], but unfortunately these models often raise computational issues when dealing with large networks. To find communities in large networks several algorithms have recently been proposed. One of the most effective is a hierarchical agglomeration algorithm proposed by Newman [20], based on the greedy optimization of the quantity known as *modularity*. It runs in time $O((m+n)n)$, or $O(n^2)$ on a sparse graph, where m and n are, respectively, the number of edges and nodes. In Ref. [21], the use of more sophisticated data structures and some shortcuts in the optimization problem enables to solve the communities detection problem in time $O(md\log(n))$ where d is the depth of the "dendrogram" describing the network's community structure.

In this paper, we propose an estimation procedure for the MixNet model, which is able to deal with large networks. It is based on a recursive estimation of the parameters. The kind of recursive algorithm originates in situations where the data are processed object by object without having the entire data set. In such situations online clustering algorithms are an efficient alternative to classical batch algorithms. Online parameter estimation using mixture models has already been studied by many authors [22,23]. More recently Liu et al. [24] have considered, for the modeling of Internet traffic, a recursive EM algorithm for the estimation of Poisson mixture models. Typical clustering algorithms include the online kmeans algorithm [25] or kernel based method such as SAKM [26].

The rest of this paper presents an original online algorithm for graph clustering based on a Erdős–Rényi graph mixture. The first section introduces the model, its online estimation, practical strategies for the initialization and the choice of the number of groups. In the second section extensive simulation illustrates the efficiency of this algorithm and a real data set dealing with the French political blogosphere is studied.

2. Online MixNet algorithm

2.1. Mixture model for random graph

Following Frank and Harary [27], the MixNet model [10] assumes that given a set of n nodes partitioned into Q classes, edges are Bernoulli random variables conditionally independent, given the class of the nodes

$$X_{ij}|\{i\in q, j\in l\}\sim\mathscr{B}(\pi_{ql}).$$

The class vector Z_i is a random vector following a multinomial distribution:

$$Z_i\sim\mathscr{M}(1,\alpha_1,\ldots,\alpha_q).$$

Thus the model parameters are the proportion of the classes and the probability of connectivity between classes.

Daudin et al. [10] tackle the problem of MixNet parameter estimation using an expectation-maximization algorithm [11], with an approximate E-step. The problem induced by the dependency between the nodes of the graph, makes it difficult to compute the probability of the missing data conditionally to the available knowledge of the network structure, $P\{\mathbf{Z}|\mathbf{X}\}$. Let us also note that the likelihood of the model is impossible to compute since it implies a sum over all possible partitions of the graph's nodes. In this paper, we propose a fast alternative algorithm, well adapted for clustering, based on stochastic approximation and the maximization of the complete data log-likelihood.

2.2. Classification log-likelihood

The Classification EM (CEM) algorithm is an iterative clustering algorithm which simultaneously yields the parameters and the classification. In this paper the considered classification log-likelihood is defined by

$$\begin{aligned}L_C(\mathbf{X},\mathbf{Z},\Phi)&=\log P_\Phi(\mathbf{Z})+\log P_\Phi(\mathbf{X}|\mathbf{Z})\\&=\sum_{i,q}z_{iq}\log\alpha_q+\sum_{q,l,i,j}z_{iq}z_{jl}\log(\pi_{ql}^{x_{ij}}(1-\pi_{ql})^{1-x_{ij}}),\end{aligned}$$

where $\Phi=\{\alpha_1,\ldots,\alpha_{Q-1},(\pi_{ql})\}$ represents the parameter vector and (z_{iq}) are indicator variables such that $z_{iq}=1$ if node i belongs to class q and 0 otherwise. Maximizing $L_C(\mathbf{X},\mathbf{Z},\Phi)$ according to $(\mathbf{Z},\Phi)$ is equivalent to maximizing the criterion

$$L_C(\mathbf{X},\mathbf{\Phi})=\max_{\mathbf{Z}}[L_C(\mathbf{X},\mathbf{Z};\mathbf{\Phi})].$$

If the considered graph is directed, undirected, has or has not self-loops, the complete log-likelihood can be expressed in a relative simple form in function of

$$\begin{cases}n_{ql}=\sum_{i,j}x_{ij}z_{iq}z_{jl}, \text{the number of edges having nodes in class } q \text{ and } l,\\ n_q=\sum_i z_{iq}, \text{ the number of nodes of class } q.\end{cases}$$

Considering for the sake of illustration, an undirected graph without self-loops, the classification log-likelihood is given by

$$\begin{aligned}L_C(\mathbf{X},\mathbf{Z})=&\sum_q n_q\log\alpha_q+\sum_{q>l}n_{ql}\log\frac{\pi_{ql}}{1-\pi_{ql}}+\sum_{q>l}n_qn_l\log(1-\pi_{ql})\\&+\sum_q n_{qq}\log\frac{\pi_{qq}}{1-\pi_{qq}}+\frac{1}{2}\left(\sum_q(n_q^2-n_q)\log(1-\pi_{qq})\right).\end{aligned}$$

Further details concerning the derivation of the expression of the classification log-likelihood are given in Appendix A.

2.3. Online estimation

This section describes an original incremental Classification version of the EM algorithm. Incremental algorithms recursively update parameters, using current parameters and new observations.

Let us note $\mathbf{X}^{m-1}$ the adjacency matrix of a graph with $m-1$ nodes and $\mathbf{Z}^{m-1}(\Phi)$, the classification matrix verifying:

$$\mathbf{Z}^{m-1}(\Phi)=\arg\max_{\mathbf{Z}}L_C(\mathbf{X}^{m-1},\mathbf{Z}^{m-1},\Phi).$$

Let Φ^{m-1} be the parameter vector maximizing $L_C^{m-1}(\mathbf{X}^{m-1},\mathbf{Z}^{m-1}(\Phi^{m-2}),\Phi)$ the complete log-likelihood expressed in function of $m-1$

nodes. When a new node and all attached edges $\mathbf{X}_m = \{x_{mj} | j < m\}$ become available, the new complete log-likelihood of Φ is expressed as the sum of the previous complete log-likelihood and a new term function of the edges between the new node and the existing network:

$$L_C^m(\mathbf{X}^m, \mathbf{Z}^m(\Phi^{m-1}), \Phi) = L_C^{m-1}(\mathbf{X}^{m-1}, \mathbf{Z}^{m-1}, \Phi) + \underbrace{\sum_q z_{mq} \left(\log \alpha_q + \sum_l \sum_{j \neq m} z_{jl} \log(\pi_{ql}^{x_{mj}} (1 - \pi_{ql})^{1 - x_{mj}}) \right)}_{L_C(\mathbf{X}_m, \mathbf{Z}^m, \Phi)}.$$

The principle of the recursive algorithm consists in computing the parameter $\Phi^{(m)}$ maximizing $L_C^m(\mathbf{X}^m, \mathbf{Z}^m(\Phi^{m-1}), \Phi)$ and exploiting the fact that the new estimates are function of the old ones. But prior to computing the new estimate of the parameter vector, it is necessary to find the partition $\mathbf{Z}^m(\Phi^{m-1})$ which maximizes $L_C^m(\mathbf{X}^m, \mathbf{Z}^m(\Phi^{m-1}), \Phi)$. A suboptimal solution, which increases the complete log-likelihood, consists of assigning to the new node the most probable class (maximum a posteriori strategy) without changing the class of all the other nodes. Once the new classification matrix is estimated, the maximization of the complete log-likelihood is straightforward and leads to new estimates of the parameter vector.

The recursive algorithm is then described by the two following steps each time a new (m)th node (and corresponding vertices) is considered:

- *Classification step*: Assign each new node $\mathbf{X}_m$ to the class q^* which maximizes

$$L_C(\mathbf{X}_m, q; \mathbf{\Phi}) = \log \alpha_q + \sum_l \sum_{j \neq m} z_{jl} \log(\pi_{ql}^{x_{mj}} (1 - \pi_{ql})^{1 - x_{mj}}).$$

  Thus set z_{mq} equal to 1 if $q = q^*$, 0 otherwise.
- *Estimation step*: Update the parameters for all classes:

$$n_q^{(m)} = n_q^{(m-1)} + z_{mq}, \tag{1}$$
$$n_{ql}^{(m)} = n_{ql}^{(m-1)} + \sum_{j \neq m} z_{mq} z_{jl} x_{mj}, \tag{2}$$
$$\alpha_q^{(m)} = \frac{n_q^{(m)}}{m}, \tag{3}$$
$$\pi_{ql}^{(m)} = \frac{n_{ql}^{(m)}}{n_q^{(m)} n_l^{(m)}}, \tag{4}$$
$$\pi_{qq}^{(m)} = \frac{2 n_{qq}^{(m)}}{n_q^{(m)} (n_q^{(m)} - 1)}. \tag{5}$$

This algorithm increases the complete-log likelihood at each step, and requires at most as many iterations as the number of nodes. It can be considered as a special case of online reinforcement learning [28] where all clusters are strongly dependent.

Still the parameters can be further improved and the complete log-likelihood further increased by revisiting each node a few times. When m the number of iterations is greater than n the size of the network, it is possible to apply the above described recursive principle, and the algorithm can be continued as follows:

- *Classification step*: Find a node $\mathbf{x}_i$, whose class change improves the classification log-likelihood.

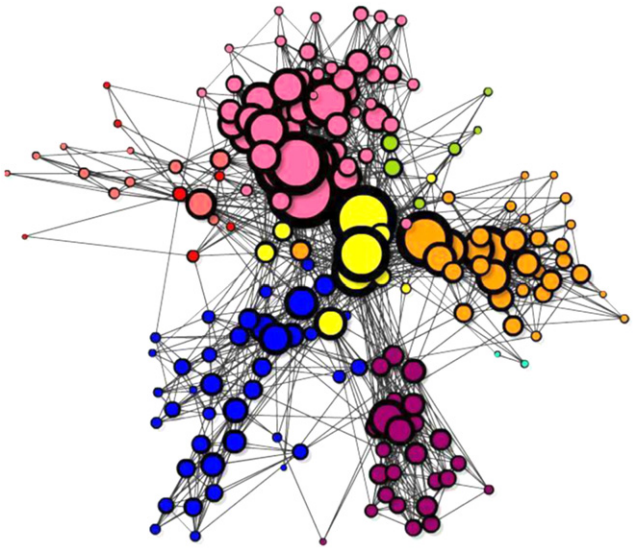

Fig. 1. Network of the blogopole www.blogopole.fr.

- *Estimation step*: Update the parameters for all classes:

$$n_q^{(m)} = n_q^{(m-1)} - z_{iq}^{m-1} + z_{iq}^m, \tag{6}$$
$$n_{ql}^{(m)} = n_{ql}^{(m-1)} + (z_{iq}^m - z_{iq}^{m-1}) \sum_{j \neq i} z_{jl} x_{ij}, \tag{7}$$
$$\alpha_q^{(m)} = \frac{n_q^{(m)}}{n}, \tag{8}$$
$$\pi_{ql}^{(m)} = \frac{n_{ql}^{(m)}}{n_q^{(m)} n_l^{(m)}}, \tag{9}$$
$$\pi_{qq}^{(m)} = \frac{2 n_{qq}^{(m)}}{n_q^{(m)} (n_q^{(m)} - 1)}. \tag{10}$$

When the classification step does not improve the classification log-likelihood, the algorithm can be stopped. Notice that this second phase of the algorithm can be related to relaxation labeling [29,30]. In this context, the label of node i is the class vector. It is changed to the class q maximizing the posterior probability

$$p(Z_{iq} = 1 | \mathbf{X}, \mathbf{Z}_{\setminus i}) = \frac{p(Z_{iq} = 1, \mathbf{Z}_{\setminus i} \mathbf{X})}{p(\mathbf{Z}_{\setminus i}, \mathbf{X})} = \frac{p(Z_{iq} = 1, \mathbf{Z}_{\setminus i}, \mathbf{X})}{\sum_{q=1}^{Q} p(Z_{iq} = 1, \mathbf{Z}_{\setminus i}, \mathbf{X})} = \frac{e^{L_C(\mathbf{X}_i, q; \mathbf{\Phi})}}{\sum_{\ell=1}^{Q} e^{L_C(\mathbf{X}_i, \ell; \mathbf{\Phi})}},$$

and then fixed, before relaxing another node label.

2.4. Affiliation model

An affiliation model is a simple model where two types of edges exist: edges between nodes of the same class and edges between nodes of different classes. Each type of edge has a given probability: $\pi_{qq} = \lambda$ and $\pi_{ql} = \varepsilon$ $(q \neq l)$. Self-loops are not taken into account (see for example Fig. 2). This model allows to explicitly compute the distribution of the degree K_i of node i, which is approximately a mixture of Poisson distribution $\mathcal{P}((n-1)[(1-\alpha_q)\varepsilon + \alpha_q \lambda])$.

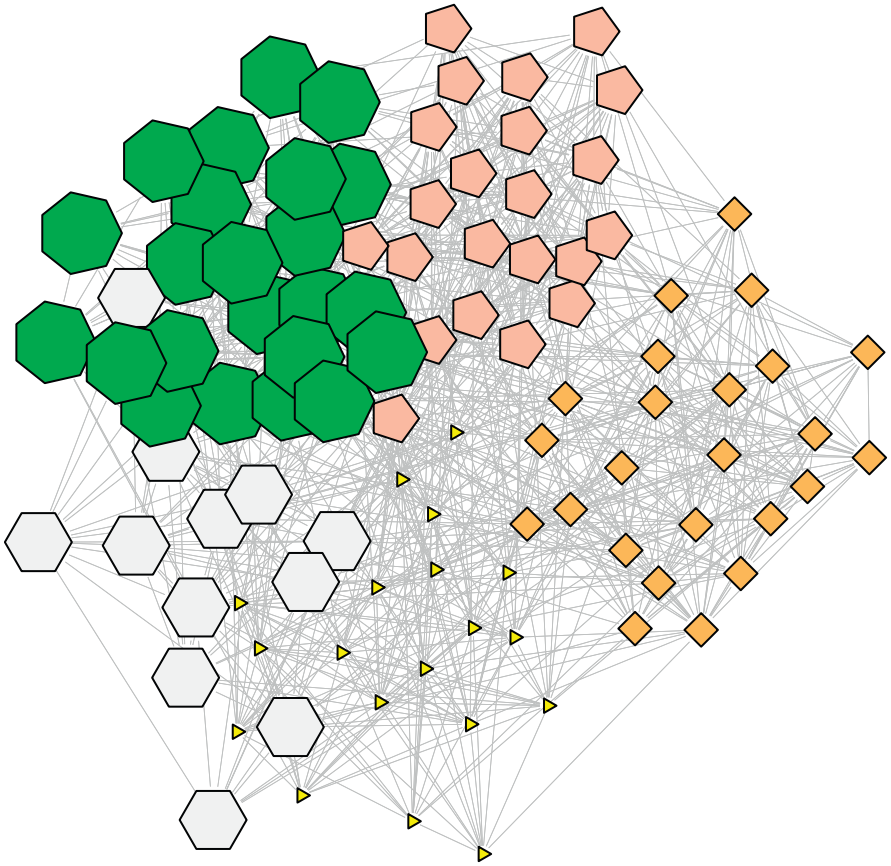

Fig. 2. Simulation of a 100 nodes graph with five classes according to an affiliation model.

Table 1
Statistics of the affiliation model computed from $n_{ql} = \sum_{i>j} x_{ij} z_{iq} z_{jl}$, the number of edges having nodes in class q and l, and $n_q = \sum_i z_{iq}$, the number of nodes of class q

	Neighbours	Not neighbours
Same class	$n_\lambda = \sum_q n_{qq}$	$n_{1-\lambda} = \sum_q \frac{n_q(n_q-1)}{2} - n_\lambda$
Different class	$n_\varepsilon = \sum_{q\neq l} n_{ql}$	$n_{1-\varepsilon} = \frac{n(n-1)}{2} - n_\lambda - n_{1-\lambda} - n_\varepsilon$

When considering this model (in an undirected graph without self-loops), the parameters are $\Phi = \{\alpha_1, \ldots, \alpha_{Q-1}, \lambda, \varepsilon\}$. Observing that $z_{iq} z_{jl} \log(\pi_{ql}^{x_{ij}} (1-\pi_{ql})^{1-x_{ij}})$ takes four different values according to the class of i and j, and their neighbourhood relationship, the classification likelihood can be expressed as

$$L_C(\mathbf{X}, \Phi) = \sum_q n_q \log \alpha_q + n_\lambda \log \lambda + n_{1-\lambda} \log(1-\lambda) + n_\varepsilon \log \varepsilon + n_{1-\varepsilon} \log(1-\varepsilon),$$

where n_λ, $n_{1-\lambda}$, n_ε and $n_{1-\varepsilon}$ are the number of node pairs defined by class and neighbourhood relationship (see Table 1).

The parameters $\hat{\Phi}$ maximizing this criterion for a given partition are as follows:

$$\begin{cases} \hat{\alpha}_q = \dfrac{n_q}{n}, \\ \hat{\lambda} = \dfrac{n_\lambda}{n_\lambda + n_{1-\lambda}}, \\ \hat{\varepsilon} = \dfrac{n_\varepsilon}{n_\varepsilon + n_{1-\varepsilon}}. \end{cases}$$

The adaptation of the proposed algorithm to the affiliation model is straightforward and requires only the computation of the four statistics of the affiliation model (see Table 1) from the $n_{ql}^{(m)}$.

2.5. Initialization and online supervised classification

If one is mainly interested in finding clusters of nodes which have strong interconnection and weak between-connection, the above described algorithm can be simplified and accelerated by working with fixed parameter values. A possible interesting choice for clustering consists in choosing a high value for the probability λ of within-connection and a small one for the probability ε of between-group connection (i.e. $\lambda = 0.8$ and $\varepsilon = 0.05$). This choice implicitly assumes the existence of clique-like clusters. Concerning the proportions, without any additional knowledge, it seems reasonable to consider clusters of the same size $\alpha_1 = \cdots = \alpha_Q = 1/Q$.

Repeating step 2, with a given value of the parameter vector, produces a partition in a finite number of iterations. Starting from a random partition $\mathbf{z}^0$, each iteration considers a randomly chosen node and assigns this node to the cluster which results in the greatest increase of the classification log-likelihood. It is obvious that this relaxation procedure increases the classification log-likelihood at each iteration and converges toward a local maximum since the criterion is upper-bounded.

This minimal clustering algorithm can be used as an effective initialization strategy for the previously described online MixNet algorithm. Notice that as all local optimization algorithms, the proposed online MixNet estimation strategy depends strongly on the initialization. A common way to circumvent this problem consists in

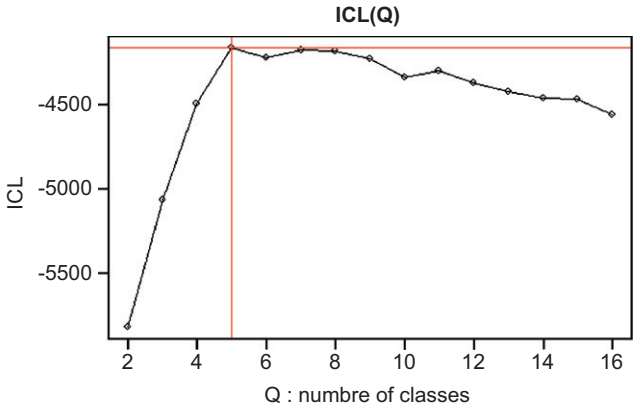

Fig. 3. Integrated Classification Likelihood Criterion in function of the number of clusters computed for the simulated graph of Fig. 2.

testing multiple initialization points and selecting the best partition and parameters in terms of likelihood.

2.6. Choosing the number of clusters

As the algorithm relies on a statistical model, it is possible to use the integrated classification likelihood (ICL) to choose the optimal number of classes [31]. This choice is done by running our online algorithm concurrently for models from 2 to Q classes and selecting the solution which maximizes the ICL criterion.

In our situation, following Daudin [10], the ICL criterion can be written as

$$ICL(Q) = \underbrace{-2L_C(\mathbf{x}, \Phi)}_{\text{A}} + \underbrace{(Q-1)\log(n) + 2\log\left(\frac{n(n-1)}{2}\right)}_{\text{B}},$$

where A is related to the classification log-likelihood, B to the free number of parameters and n is the number of nodes treated. The ICL criterion is essentially the ordinary BIC considering the complete log-likelihood instead of the log-likelihood. Consider, for example, Fig. 3 which exhibits this strategy for synthetic data of Fig. 2. It displays the ICL criterion computed for models with 2–16 clusters and shows a clear maximum for five clusters, which is the real number used for data simulation.

3. Applications

We carried out experiments to assess how well the proposed online clustering algorithm discovers node clusters. We consider simulation experiments using synthetic data generated according to the assumed random graph model, as well as real data coming from the web. The real data consists of over 1100 political blogs collected in order to analyze the French presidential campaign on the blogosphere.

An ANSI `C++` implementation of the CEM algorithm is available in the `MixNet` package. Compilation and installation are compliant with the GNU standard procedure.

The package is free and available at http://stat.genopole.cnrs.fr/software/mixnet/. On-line documentation is also available. `MixNet` is licensed under the GNU General Public License (http://www.gnu.org/licences.html).

3.1. Synthetic data

Four affiliation models have been considered (see Table 2). The difference among the four models is related to their modular structure, which varies from no structure (almost the Erdős–Rényi model) to strong modular structure (low inter-module connectivity and strong intra-module connectivity).

Table 2
Parameters of the four affiliation models of the experiment

Model	λ	ε	Q
1	0.8	0.02	3
2	0.5	0.05	8
3	0.6	0.25	5
4	0.55	0.35	5

The Q modules are mixed in the same proportion. Each model consider $n = 1000$ nodes.

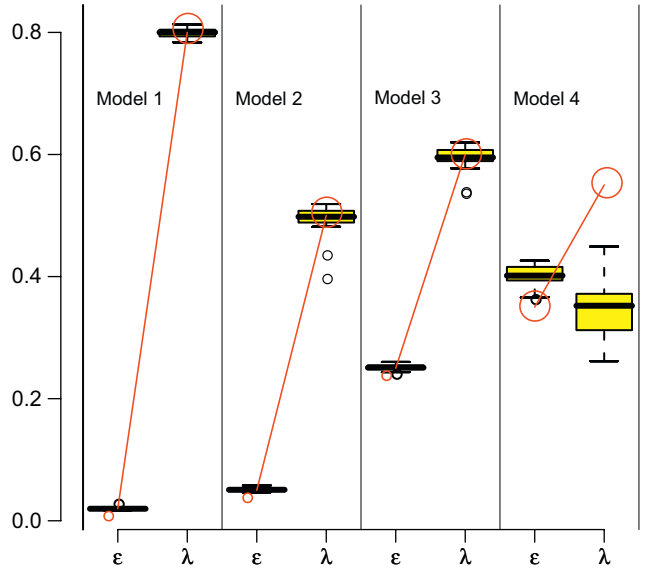

Fig. 4. Boxplot of the parameter estimates for 30 estimations of the 4 models. Each model is described by two boxplots, one for the estimations of ε and the other for the estimations of λ. The circles show the true value of the parameters.

Given the number of nodes n and the class proportions (α_q), the colour of each node is simulated via a multinomial distribution $\mathcal{M}(1, \alpha_1, \ldots, \alpha_Q)$. Conditionally to the node colours, edges between two nodes of the same class are drawn according to a probability λ and edges between nodes of different colours are drawn according to a probability ε.

Comparing the estimated partition with the true partition is not as straightforward as comparing the parameter estimates. In order to evaluate the agreement between these two partitions, we use the adjusted Rand index [32] which lies between 0 and 1. The computation of this index is based on a ratio between the number of node pairs belonging to the same and to different classes when considering the true partition and the estimated partition. Two identical partitions have an adjusted Rand index equal to 1.

We have simulated 30 networks for each model and run the online MixNet algorithm to estimate the model parameters. Fig. 4 shows two boxplots for each experiment: one boxplot for ε and one for λ. Notice that for the first model, the highly structured one, the estimation is very close to the true parameters and exhibits no variance. The estimation of the second and third models shows a small downward bias and a small variance. But the fourth model is more difficult to deal with: the algorithm underestimates λ the probability of within-cluster connection and overestimates ε the probability of between-cluster connection. In summary, the less obvious the structure of the network is, the highest bias we observe in the resulting estimation. Let us also note that in well-structured models (except

Table 3
Means and standard deviation of the parameter estimates of the four models computed over 30 different runs

Model	MixNet				Online MixNet			
	$\hat{\lambda}$	σ_λ	$\hat{\varepsilon}$	σ_ε	$\hat{\lambda}$	σ_λ	$\hat{\varepsilon}$	σ_ε
1	0.800	0.007	0.020	0.002	0.800	0.007	0.020	0.002
2	0.494	0.024	0.051	0.003	0.493	0.025	0.051	0.003
3	0.593	0.018	0.251	0.005	0.594	0.018	0.251	0.005
4	0.385	0.034	0.390	0.011	0.350	0.046	0.402	0.017

Table 4
Means and standard deviation of the Rand index of the four models computed over 30 different runs for graph clustering competitors and MixNet algorithms

Model	PAM		SC		MixNet		Online MixNet	
	$\overline{ri}_{PAM}$	$\sigma_{ri_{PAM}}$	$\overline{ri}_{SC}$	$\sigma_{ri_{SC}}$	$\overline{ri}_{MixNet}$	$\sigma_{ri_{MixNet}}$	$\overline{ri}_{onlineMixNet}$	$\sigma_{ri_{onlineMixNet}}$
1	1.000	0.000	0.998	0.008	1.000	0.000	1.000	0.000
2	0.745	0.067	0.988	0.018	0.990	0.027	0.988	0.030
3	0.523	0.111	0.798	0.144	0.967	0.048	0.963	0.055
4	0.083	0.031	0.056	0.043	0.027	0.055	0.045	0.055

for model 1), the algorithm has a slight tendency to produce biased estimates. This is a phenomenon generally observed with classification versions of the EM algorithm. When using Table 3 to compare the estimates between our online MixNet and MixNet [10], we can observe that both estimations are very close. But the online version runs much faster: Although the computational complexity of both algorithms is $O(n^2)$ (n being the number of nodes of the network), the speed ratio between online MixNet and MixNet is a linear function of the number of nodes which is approximatively equal to $1.5n$.

When considering Table 4, we can observe that the poor estimation of λ, the probability of within-cluster connection also reveals a small Rand index. This means that the poor estimation of λ makes it impossible to retrieve the modular structure of the network.

We also compared the results of the online MixNet algorithm with alternative clustering methods additional to the variational batch version. We consider as competitors, a basic spectral clustering algorithm [8], and a kmeans-like algorithm considering a dissimilarity matrix as input, called PAM (partitioning around medoids).

The spectral clustering algorithm searches for a partition in the space spanned by the eigenvectors of the normalized Laplacian. For the PAM algorithm, we consider a computationally heavy method which builds a dissimilarity matrix based on the shortest paths between all pairs of nodes. Floyd's algorithm is specifically designed to solve this problem. Note that these shortest paths are computed in $O(n^3)$ runtime, thus it does not allow us to use it with huge and dynamic networks.

On the previous generated networks, we have run these algorithms and have computed the Rand index on each of them. When considering Table 4, we can observe that both MixNet algorithms always produce the best nodes classification. It is also noticable that PAM and spectral clustering algorithms seem to be rather efficient on obvious models (model 1 and 2), but deteriorate when the network structure becomes weak. The spectral clustering algorithm remains more accurate than the PAM algorithm.

To conclude, if the results of MixNet and online MixNet are similar, the latter appears to be the more efficient competitor as far as computational cost is concerned.

3.2. French political blogosphere network

We also studied our algorithm on a real data set. The data consist of a single day snapshot of over 1100 political blogs automatically extracted the 14 October 2006 and manually classified by the "Observatoire Présidentielle" project. This project is the result of a collaboration between RTGI SAS and Exalead and aims at analyzing the French presidential campaign on the web.

Table 5
Contingency table comparing true and estimated partitions

True	Estimated					
	DC	A	D	E	G	L
DC	172	32	4	18	3	25
A	2	22	2	5	5	5
D	3	27	165	11	1	26
E	1	2	0	80	1	0
G	5	97	12	66	181	45
L	1	1	3	1	0	82

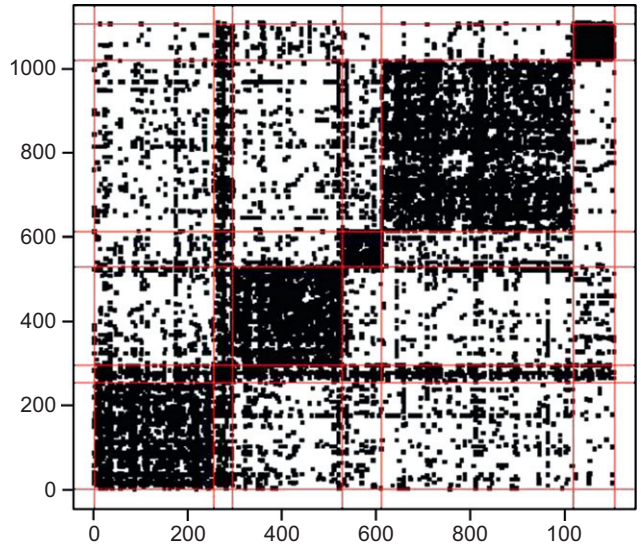

Fig. 5. Adjacency matrix of the blogopole network after reordering according to the estimated partition.

In this data set, nodes represent hostnames (a hostname contains a set of pages) and edges represent hyperlinks between different hostnames. If several links exist between two different hostnames, we collapse them into a single one. Note that intra-domain links can be considered if hostnames are not identical. Finally, in this experimentation we consider that edges are not oriented which is not realistic but which does not affect the interpretation of the groups. This network presents an interesting communities organization due to the existence of several political parties and commentators. We assume that authors of these blogs tend to link, by political affinities, blogs with similar political positions. A sample of 250 blogs of this political blogoshere can be seen in Fig. 1. Six known communities compose this network: **G**auche ("french democrat"), **D**ivers **C**entre (Moderate party), **D**roite (french republican), **E**cologiste (green), **L**iberal (supporters of economic-liberalism) and finally **A**nalysts. Proportions of blogs in these communities are, respectively, 0.36, 0.23, 0.21, 0.08, 0.08 and 0.04.

In this experimentation we are interested in finding six groups of blogs using the online clustering algorithm for a given couple of ε and λ. The number of groups is fixed to six in order to compare the true partition of blogs with the estimated partition running our algorithm. Finding similar partitions validates the assumption that political affinities can be detected using the structure of the political blogosphere. The couple ($\lambda = 0.55, \varepsilon = 0.04$) gives the maximal agreement between the real and estimated partitions with an acceptable Rand index value (0.34).

Table 5 shows a contingency table of the counts of given and estimated blogs classes. Except for the **A** class, we can observe a relative coherence between these two partitions. In fact, the **A** class is a hub class constituted of blogs which links the other classes in order to analyze and comment the French political news. Our algorithm overestimates the number of blogs contained in this class and generates important classification differences. We can also observe that there are classification errors produced by the political proximity of parties. For example, the estimated **E** class has many blogs belonging to the real **G** class. Finally, we illustrate in Fig. 5 the density of links using the adjacency matrix projection of the network after reordering by class estimation. We can observe that there is a higher density of intra-group links than inter-groups which validates our MixNet model. Note that classification errors of the **A** class can be observed creating horizontal and vertical bands by linking blogs of other classes.

4. Conclusion

The proposed online classification EM algorithm classifies the nodes of a network as they are discovered. The algorithm is based on Erdős–Rényi Graph Mixture, which is a well-known model [27], for which we provide a fast estimation procedure. The algorithm runs in $O(n^2)$ and is thus able to handle networks with thousands of nodes. This online estimation strategy allows us to both find communities in a network and a reliable estimation of the model parameters. When the cluster structure is weak the estimates are biased. In the near future, we plan to investigate pure EM online strategies for finding a better estimation for more difficult situations.

Appendix A. Complete log-likelihood of the Erdős–Rényi mixture

The complete log-likelihood of the Bernoulli Erdős–Rényi mixture takes the form

$$L_C(\mathbf{X},\mathbf{Z}) = \underbrace{\sum_i \sum_q z_{iq} \log \alpha_q}_{\log P(\mathbf{Z})} + \underbrace{\sum_{ij}\sum_{ql} z_{iq} z_{jl} x_{ij} \log \frac{\pi_{ql}}{1-\pi_{ql}} + \sum_{ij}\sum_{ql} z_{iq} z_{jl} \log(1-\pi_{ql})}_{\log P(\mathbf{X}|\mathbf{Z})}.$$

If the considered graph is directed, undirected, has or has not self-loops, the complete log-likelihood can be expressed in a relative simple form in function of the $n_q = \sum_i z_{iq}$ and $n_{ql} = \sum_{ij} z_{iq} z_{jl} x_{ij}$.

A.1. Directed with self-loops

If the considered graph has self-loops and is directed, we have

$$L_C(\mathbf{X},\mathbf{Z}) = \sum_q n_q \log \alpha_q + \sum_{ql} n_{ql} \log \frac{\pi_{ql}}{1-\pi_{ql}} + \sum_{ql} n_q n_l \log(1-\pi_{ql}).$$

To simplify the notation let us use $g(\pi_{ql}) = \log \pi_{ql}/(1-\pi_{ql})$ and $h(\pi_{ql}) = \log(1-\pi_{ql})$:

$$L_C(\mathbf{X},\mathbf{Z}) = \sum_q n_q \log \alpha_q + \sum_{ql} n_{ql} g(\pi_{ql}) + \sum_{ql} n_q n_l h(\pi_{ql}).$$

Deriving according to π_{ql} results in the following estimates:

$$\hat{\pi}_{ql} = \frac{n_{ql}}{n_q n_l}.$$

A.2. Directed without self-loops

In order to express the classification log-likelihood in function of the basic statistics, we express the likelihood as a sum of sums over all indexes $ijql$:

$$\begin{aligned} L_C(\mathbf{X},\mathbf{Z}) &= \log P(\mathbf{Z}) + \log P(\mathbf{X}|\mathbf{Z}) \\ &= \sum_q n_q \log \alpha_q + \sum_{i\neq j,ql} z_{iq} z_{jl} (x_{ij} g(\pi_{ql}) + h(\pi_{ql})) \\ &\quad + \sum_{iql} z_{iq} z_{il} (x_{ii} g(\pi_{ql}) + h(\pi_{ql})) - \sum_{iql} z_{iq} z_{il} (x_{ii} g(\pi_{ql}) + h(\pi_{ql})) \\ &= \sum_q n_q \log \alpha_q + \sum_{ij,ql} z_{iq} z_{jl} (x_{ij} g(\pi_{ql}) + h(\pi_{ql})) \\ &\quad - \sum_{iql} z_{iq} z_{il} (x_{ii} g(\pi_{ql}) + h(\pi_{ql})) \\ &= \sum_q n_q \log \alpha_q + \sum_{ql} n_{ql} g(\pi_{ql}) + \sum_{ql} n_q n_l h(\pi_{ql}) - \sum_q n_q h(\pi_{qq}). \end{aligned}$$

Deriving according to π_{ql} results in the following estimates:

$$\hat{\pi}_{ql} = \begin{cases} \dfrac{n_{ql}}{n_q n_l} & \text{if } q \neq l, \\ \dfrac{n_{qq}}{n_q(n_q-1)} & \text{otherwise.} \end{cases}$$

A.3. Undirected without self-loops

We consider the same kind of computation and try to express the likelihood as a sum of sums over all indexes $ijql$. First decompose the sum of two term (one over $q > l$ and one over $q = q$) and then take into account the fact that $\pi_{ql} = \pi_{lq}$. This leads to

$$\begin{aligned} L_C(\mathbf{X},\mathbf{Z}) &= \log P(\mathbf{Z}) + \log P(\mathbf{X}|\mathbf{Z}) \\ &= \sum_q n_q \log \alpha_q + \sum_{i>j,ql} z_{iq} z_{jl} (x_{ij} g(\pi_{ql}) + h(\pi_{ql})) \\ &= \sum_q n_q \log \alpha_q + \sum_{q>l} n_{ql} g(\pi_{ql}) + \sum_{q>l} n_q n_l h(\pi_{ql}) + \sum_q n_{qq} g(\pi_{qq}) \\ &\quad + \frac{1}{2}\left(\sum_q n_q^2 h(\pi_{qq}) - \sum_q n_q h(\pi_{qq}) \right). \end{aligned}$$

Deriving according to π_{ql} results in the following estimates:

$$\hat{\pi}_{ql} = \begin{cases} \dfrac{n_{ql}}{n_q n_l} & \text{if } q \neq l, \\ \dfrac{2n_{qq}}{n_q(n_q-1)} & \text{otherwise.} \end{cases}$$

References

[1] A.-L. Barabasi, Linked: How Everything is Connected to Everything Else and What it Means for Business, Science, and Everyday Life, Plume Books, 2003 URL ⟨http://www.amazon.ca/exec/obidos/redirect?tag = citeulike09-20{&}path = ASIN/0452284392⟩.

[2] G.W. Flake, S. Lawrence, C.L. Giles, F. Coetzee, Self-organization of the web and identification of communities, J. IEEE Comput. 35 (3) (2002) 66–71 URL ⟨citeseer.ist.psu.edu/flake02selforganization.html⟩.

[3] S. Wasserman, K. Faust, D. Iacobucci, Social Network Analysis, Cambridge University Press, Cambridge, UK, 1994 URL ⟨http://www.cambridge.org/catalogue/catalogue.asp?isbn = 9780521387071⟩.

[4] R. Guimera, L.A.N. Amaral, Functional cartography of complex metabolic networks, Nature 433 (2005) 895 URL ⟨http://www.citebase.org/abstract?id = oai:arXiv.org:q-bio/0502035⟩.

[5] M. Newman, Detecting community structure in networks, Eur. Phys. J. B. 38 (2) (2004) 321–330.

[6] M. Fiedler, Algebraic connectivity of graphs, Czech. Math. J. 23 (1973) 298–305.

[7] B. Schölkopf, P. Knirsch, A. Smola, C. Burges, Fast approximation of support vector kernel expansions, and an interpretation of clustering as approximation in feature spaces, in: P. Levi, M. Schanz, R.-J. Ahlers, F. May (Eds.),

Mustererkennung 1998–20. DAGM-Symposium, Informatik aktuell, Springer, Berlin, 1998, pp. 124–132.
[8] A. Ng, M. Jordan, Y. Weiss, On spectral clustering: analysis and an algorithm, in: NIPS 14, 2002.
[9] R. Albert, A.-L. Barabasi, Statistical mechanics of complex networks, Rev. Mod. Phys. 74 (2002) 47 URL ⟨http://www.citebase.org/abstract?id = oai:arXiv.org:cond-mat/0106096⟩.
[10] J.-J. Daudin, F. Picard, S. Robin, A mixture model for random graphs, Statist. Comput. 18 (2) (2008) 151–171.
[11] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum-likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. B 39 (1977) 1–39.
[12] T.A.B. Snijders, P.E. Pattison, G.L. Robins, M.S. Handcock, New specifications for exponential random graph models, Sociol. Methodol. 36 (1) (2006) 99–153.
[13] A. Robles-Kelly, E. Hancock, An EM-Like Algorithm for Motion Segmentation via Eigendecomposition, in: Proceedings of the British Machine Vision Conference, 2001, pp. 123–132.
[14] T.A.B. Snijders, K. Nowicki, Estimation and prediction for stochastic block-structures for graphs with latent block structure, J. Classification 14 (1997) 75–100.
[15] G Govaert, M Nadif, An EM algorithm for the block mixture model, IEEE Trans. Pattern Anal. Mach. Intell. 27 (4) (2005) 643–647.
[16] G. Govaert, Algorithme de classification d'un tableau de contingence, in: First International Symposium on Data Analysis and Informatics, INRIA, Versailles, 1977, pp. 487–500.
[17] G. Govaert, Classification croisée. thèse d'état, université, Ph.D. Thesis, Paris 6, France, 1983.
[18] S. Wasserman, K. Faust, D. Iacobucci, Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences), Cambridge University Press, Cambridge, 1994 URL ⟨http://www.amazon.ca/exec/obidos/redirect?tag = citeulike04-20{&}path = ASIN/0521387078⟩.
[19] M. Handcock, A. Raftery, J. Tantrum, Model based clustering for social networks, J. R. Stat. Soc. A. 170 (2007) 301–354.
[20] M.E.J. Newman, Fast algorithm for detecting community structure in networks, Phys. Rev. E 69 (2004) 066133 URL ⟨http://www.citebase.org/abstract?id = oai:arXiv.org:cond-mat/0309508⟩.
[21] A. Clauset, M.E.J. Newman, C. Moore, Finding community structure in very large networks, August 2004. URL ⟨http://arxiv.org/abs/cond-mat/0408187⟩.
[22] D.M. Titterington, Recursive parameter estimation using incomplete data, J. R. Stat. Soc. B 46 (1984) 257–267.
[23] S. Wang, Y. Zhao, Almost sure convergence of Titterington's recursive estimator for finite mixture models, in: IEEE International Symposium on Information Theory IST.
[24] Z. Liu, J. Almhana, V. Choulakian, R. McGorman, Online EM algorithm for mixture with application to Internet traffic modeling, Comput. Statist. Data Anal. 50 (4) (2006) 1052–1071 available at ⟨http://ideas.repec.org/a/eee/csdana/v50y2006i4p1052-1071.html⟩.
[25] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematics, Statistics and Probability, vol. 1, 1967, pp. 281–296.
[26] H.A. Boubacar, S. Lecoeuche, A new kernel-based algorithm for online clustering, in: ICANN, 2005, pp. 583–588.
[27] O. Frank, F. Harary, Cluster inference by using transitivity indices in empirical graphs, J. Am. Stat. Assoc. 77 (380) (1982) 835–840.
[28] A. Likas, A reinforcement learning approach to on-line clustering, Neural Comput. 11 (8) (1999) 1915–1932.
[29] A. Rosenfeld, R. Hummel, S. Zucker, Scene labeling by relaxation operations, SMC 6 (6) (1976) 420–433.
[30] E. Hancock, J. Kittler, Discrete relaxation, Pattern Recognition 23 (7) (1990) 711–733.
[31] C. Biernacki, G. Celeux, G. Govaert, Assessing a mixture model for clustering with the integrated completed likelihood, IEEE Pattern Anal. Mach. Intell. 22 (7) (2000) 719–725.
[32] L. Hubert, P. Arabie, Comparing partitions, J. Classification 2 (1985) 193–218.

About the Author—HUGO ZANGHI received his Engineer diploma (speciality "Data Mining") from the Université de Technologie de Compiégne (France) in 2006. Since 2006, he is a PhD student at the European search engine Exalead, in Paris. His research interests include web data mining and networks topology.

About the Author—CHRISTOPHE AMBROISE received his PhD in System Control from the Université de Technologie de Compiégne (France) in 1996. From 1998 to 2006, he was assistant professor at the Université de Technologie de Compiégne. He is researcher at the laboratory "Statistic and Genome" of the Genopole in Evry. His current research includes statistical learning applied to microarray analysis and web mining.

About the Author—VINCENT MIELE received his Engineer diploma (speciality "Mathematics and Modelling") of the "Grande Ecole" CUST (France) in 2000; he also had a master in Applied Mathematics in 2000. Since 2001 he is research engineer in the CNRS (National Center for Scientific Research), working at the laboratory "Statistic and Genome" of the Genopole in Evry. His research interests include biological networks topology, in silico nucleosome positioning and software development (efficient algorithms, parallel computing).