ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0. Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

Network community detection with edge classifiers trained on LFR graphs

Twan van Laarhoven and Elena Marchiori [*]

Department of Computer Science, Radboud University
Nijmegen, The Netherlands

Abstract. Graphs generated using the Lancichinetti-Fortunato-Radicchi (LFR) model are widely used for assessing the performance of network community detection algorithms. This paper investigates an laternative use of LFR graphs: as training data for learning classifiers that discriminate between edges that are 'within' a community and 'between' network communities. The LFR generator has a parameter that controls the extent to which communities are mixed, and hence harder to detect. We show experimentally that a linear edge-wise weighted support vector machine classifier trained on a graph with more mixed communities also works well when tested on easier graph instances, while it achieves mixed performance on real-life networks, with a tendency towards finding many communities.

1 Introduction

Network community detection is the task of identifying communities in a graph. Informally, a community is a set of nodes, such that there are many edges inside the community and relatively few edges linking it to the rest of the graph. Here we consider the traditional view of community structure of a graph as a partition of its nodes into groups such that each group is a community.

A simple way to find communities in a graph is to identify and remove edges that connect nodes belonging to different communities. The connected components of the resulting graph are the communities (see for instance [1, 2]). The question of community detection then becomes a question of how to pick the set of edges to remove. In this paper we investigate the effectiveness of a direct approach to this task, based on supervised learning. Using supervised learning requires training data, that is, graphs with a known community structure. We consider training data generated using the LFR model [3]. This model accounts for the fact that complex real-life networks are characterized by heterogeneous distributions of degree and community sizes, and that the degrees of nodes, as well as the sizes of communities follow a power law distribution. Constructing LFR graphs is efficient with linear complexity in the number of edges of the graph.

We consider a simple learning setting: linear classifiers acting on local features of an edge. Such features can be efficiently computed since they are based only on the degree of the edge's nodes and the number of triangles containing that edge. They are used in heuristic methods for community detection, e.g. [2].

[*]This work has been partially funded by the Netherlands Organization for Scientific Research (NWO) within the NWO project 612.066.927.

We investigate empirically two main questions: (a) Is there one classifier trained on a specific type of LFR graphs that also generalizes well to graphs generated using other LFR parameter settings? (b) If such a classifier exists, what is its performance on real world graphs? Extensive experiments indicate that one can build such a classifier, e.g. using the linear edge-wise weighted support vector machine. Its generalization performance on the considered real world graphs is shown to be mixed, with a tendency to detect a high number of clusters. In general the results indicate that it is possible to learn a simple supervised model based on few local topological features for identifying community structure in artificial and real world networks. Therefore community detection, that is graph clustering, can be addressed using a supervised setting.

2 Community detection by classifying edges

The problem of detecting communities in a graph $G = (V, E)$ can be formulated as a binary classification task, where the goal is to decide whether each edge $ab \in E$ is 'within' a community or 'between communities'. A classifier is a function h that assigns a score to each edge, where edges with a positive scores are considered to be 'within a community'. Given these scores, we can construct the *reduced subgraph*, containing only edges $ab \in E$ for which $h(\mathbf{x}_{ab}) > 0$. In other words, the reduced graph contains only edges classified as being 'within a community'. The connected components of this reduced graph are considered the graph's communities.

In order to use a classifier on edges, we need to associate a feature vector $\mathbf{x}_{ab}$ to each edge $ab \in E$ connecting two nodes $a, b \in V$.

We use simple features employed in heuristic methods for network analysis: the degrees d_a, d_b of a, b, the number of triangles containing ab, and the mean values of these features. This gives a total of 8 features, all of which can be calculated efficiently. Because the features are local, this calculation can in principle also be done in parallel for different parts of the graph.

We consider undirected graphs, where $ab \in E$ if and only if $ba \in E$. Therefore features should be symmetric, that is, $\mathbf{x}_{ab} = \mathbf{x}_{ba}$. To this end we replace the degrees with minimum and maximum degree values, $\min(d_a, d_b)$ and $\max(d_a, d_b)$.

Several heuristic scoring functions have been used in the literature to rank edges. For instance, in [2] one of the score functions used to rank edges for performing community detection is the fraction of possible triangles that contain an edge ab,

$$s_{\text{Radicchi}}(ab) = \frac{t_{ab} + 1}{\min(d_a - 1, d_b b - 1)},$$

where t_{ab} is the number of triangles containing the edge ab, which is equivalent to the number of paths of length 2 between a and b.

The algorithm then iteratively removes the edge with the lowest score from the graph, until some stopping criterion is reached. At that point, the graph will contain only edges with scores greater than some threshold τ. That is, edges ab such that $s_{\text{Radicchi}}(ab) > \tau$. For the above score function we can rewrite this

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0. Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

condition in the form of a linear classifier,

$$h(\mathbf{x}_{ab}) = \langle (\tau + 1, -\tau, 0, 1), \mathbf{x}_{ab} \rangle > 0,$$

where $\mathbf{x}_{ab} = (1, \min(d_a, d_b), \max(d_a, d_b), t_{ab})$. This expression is in the standard form of a linear classifier, $\langle \mathbf{w}, \mathbf{x} \rangle > 0$.

The Jaccard similarity between the adjacency lists of a and b has also been used as a score function [4]. If $\mathrm{Adj}(a)$ denotes the set of nodes adjacent to a, this Jaccard similarity can be written as

$$s_{\mathrm{Jaccard}}(ab) = \frac{|\mathrm{Adj}(a) \cap \mathrm{Adj}(b)|}{|\mathrm{Adj}(a) \cup \mathrm{Adj}(b)|}.$$

The condition $s_{\mathrm{Jaccard}}(ab) > \tau$ can also be rewritten as a linear classifier, $\langle (0, -\tau, -\tau, 1 - \tau), \mathbf{x}_{ab} \rangle > 0$.

Instead of using such a fixed score function, we can try to learn one based on a training graph. The learning problem looks like a standard linear classification problem. For each edge $ab \in E$ we could use the features $\mathbf{x}_{ab}$ defined above, and labels such that y_{ab} is 1 if a and b are in the same community, and is -1 otherwise.

The problem of finding the weight vector $\mathbf{w}$ is then a linear classification problem that can be solved by, for instance, a weighted Support Vector Machine. We call this the *edge-wise* classification problem. Note that usually incorrectly keeping a between community edge results in a much larger error in the community structure than incorrectly removing a within community edge. Therefore the weight of negative training instances should be higher.

In extreme cases, all edges incident to a node could be classified as between community edges. In that case, the connected component containing that node would be a singleton. However, a single node alone is not a community. We found that we can significantly improve the quality of the clustering if we avoid such singletons. To enforce that all communities consist of at least two nodes, for each node a, we keep the edge with the highest classifier score $h(\mathbf{x}_{ab})$, regardless of whether this score is positive.

3 LFR benchmark as training data

To learn a classifier, we need training data. I.e. one or more graphs with a known community structures. For real world applications, such training data can be hard to come by.

Therefore, we instead use benchmarks specifically constructed to closely resemble real world graphs. A popular example is the LFR benchmark by Lancichinetti, Fortunato, and Radicchi [3], which generates artificial graphs that closely resemble real world graphs. In this benchmark, the size of each community is drawn from a power-law distribution; as is the degree of each node. It has previously been observed that real world graphs also have such a power-law degree distribution Clauset et al. [5]. Therefore, we hope that by using LFR graphs for training data, we can train classifiers that also work well on real-world testing graphs.

The LFR model has several parameters. The most important one is the mixing parameter μ, that controls the fraction of edges that are between communities. Essentially this can be viewed as the amount of noise in the graph. If $\mu = 0$ all edges are within community edges, if $\mu = 1$ all edges are between nodes in different communities.

Other parameters control the number of nodes, the distributions of community sizes, the distribution of degrees, etc.

We used the LRF implementation from `http://sites.google.com/site/santofortunato`. We consider the four benchmarks used in [6], two with 'small communities' of between 10 and 50 nodes, and two with 'large communities' of between 20 and 100 nodes. Each graph has either 1000 or 5000 nodes in total. Furthermore the GN benchmark (see [1]) is considered. This is an instance of LFR with community sizes equal to 32 nodes, and the degree of each node equal to 16. We refer to the resulting 5 classes of graphs as 'Small1000', 'Small5000', 'Big1000', 'Big5000' and 'GN'.

4 Experiments

Since the true community structure is known for the benchmark graphs, we can assess the quality of the resulting clustering by means of the Normalized Mutual Information (NMI) metric, computed between a given output clustering and the true one [7].

As learning method we consider a linear edge-wise weighted SVM, using the LIBLINEAR implementation (available at `www.csie.ntu.edu.tw/~cjlin/liblinear/`). To select the hyper parameters of the learning method, the regularization parameter and the relative weight of within community edges, we use a grid search procedure.

First, for each testing graph we used another graph generated with the same settings as training data. That means that the distribution of training and test graphs is the same. The left plot of figure 1 shows the NMI as a function of the mixing parameter.

The right plot in figure 1 shows the NMI for a classifier that was trained on a big community graph with 1000 nodes, and $\mu = 0.5$. The same classifier was used for all tests. The two plots in figure 1 show that the performance is similar to that obtained with a classifier trained for the specific network generation parameter settings. This shows that the classifier generalizes well to other parameter settings of the LFR benchmark.

Further experiments indicate that, in general, classifiers trained on small communities benchmark graphs also work well for graphs with large communities, and vice-versa. Additionally, classifiers trained with a particular mixing parameter μ will still perform well on graphs using *less* mixing. The converse is not true, however, since graphs with low μ are essentially 'too easy'. They do not have any examples of between community edges with larger number of

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0. Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

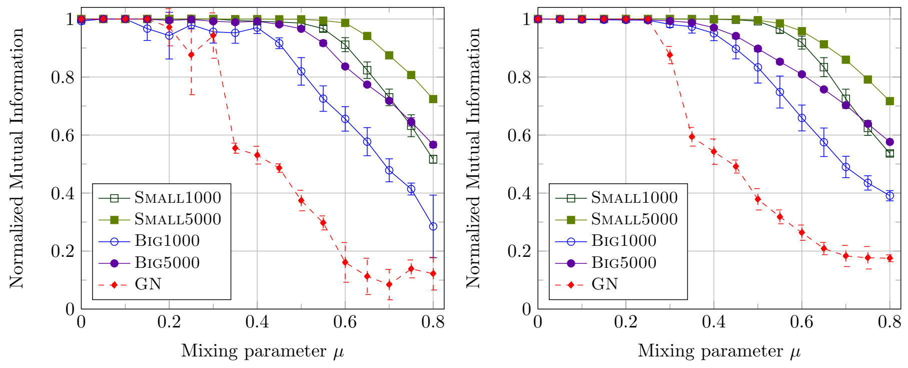

Figure 1: The performance of classifier based community detection on graphs generated with the LFR benchmark, as measured with Normalized Mutual Information. The error bars show the standard deviation across different training and testing datasets. In the left plot a new classifier is trained for each graph. In the right plot the same classifier is used in all cases, trained on a big community graph (BIG1000) with $\mu = 0.5$.

	Normalized Mutual Information				Number of communities	
Dataset	Classifier	R. Weak	R. Strong	Infomap	Actual	Classifier
Zachary	0.649	0	0	0.568	2	4
Football	0.923	0.908	0.201	0.924	12	15
PolBooks	0.522	0	0	0.537	3	9
PolBlogs	0.134	0.014	0.014	0.340	2	322

Table 1: Results on real world datasets.

triangles, which do appear with higher μ.

The reason for this generalization results is that the ‘good’ linear classifiers that are found by the training algorithm are very similar. This suggests that for a particular set of features there might be a large class of graphs for which the same scoring or classification function is optimal.

If the LFR benchmark is representative of real world data, then the classifiers we learned for artificial graphs should also generalize to real world graphs. Since the classifier trained on the BIG1000 dataset with $\mu = 0.5$ generalizes well to other LFR benchmark graphs, it is interesting to examine its performance on real world datasets.

We consider real-life networks employed in previous studies. The Zachary’s karate club; Football: A network of American football games; Political books: A network of books about US politics; Political blogs: Hyperlinks between weblogs on US politics.

We compare the results against those of the infomap method [8], a representative of the state of the art in network community detection [6], and the two methods proposed in [2], because they are similar to the classifier based method.

Indeed they also use the number of triangles and node degrees as features, and make local decisions for each edge. The two methods differ in their stopping criterion, one is based on 'weak communities', the other on 'strong communities', we refer to them as R. Weak and R. Strong, respectively.

Table 1 shows the results. For the Zachary, Football and Political books networks, the results of the classifier are close to or better than those of infomap, and better than those of R. Weak and R. Strong. This does not hold for the Political Blogs network: the actual network is considered to have just 2 communities but the classifier finds a much larger number. Radicchi's method has problems with these graphs, often returning a single cluster that contains all nodes.

5 Conclusion

We investigated how community detection can be approached as a classification problem using as training data LFR graphs. An inherent limitation of the proposed method is the fact that it removes edges in one go, while existing algorithms for community detection based on edge removal incorporate an adaptive mechanism where the score of the edges is updated each time a (set of) edges is removed. How to incorporate such a mechanism in a supervised setting without making the training process too complex is an open issue.

An extended version of this paper and source code are available from `http://cs.ru.nl/~T.vanLaarhoven/learning-communities-2013/`.

References

[1] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12): 7821–7826, 2002.

[2] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2658–2663, 2004.

[3] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, 78(4), 2008.

[4] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466:761–764, 2010.

[5] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Rev.*, 51:661–703, November 2009. ISSN 0036-1445.

[6] A. Lancichinetti and S. Fortunato. Community detection algorithms: A comparative analysis. *Phys. Rev. E*, 80:056117, Nov 2009.

[7] L. Danon, J. Duch, A. Arenas, and A. Díaz-Guilera. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 9008:09008, 2005.

[8] M. Rosvall and C.T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences of the United States of America*, 105(4):1118–1123, January 2008.