Classification in Graphs using Discriminative Random Walks

Semi-supervised learning, large graphs, betweenness measure, passage times

Jérôme Callut JEROME.CALLUT@UCLOUVAIN.BE
Kevin Françoisse KEVIN.FRANCOISSE@UCLOUVAIN.BE
Marco Saerens MARCO.SAERENS@UCLOUVAIN.BE
UCL Machine Learning Group (MLG)
Louvain School of Management, IAG,
Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium

Pierre Dupont PIERRE.DUPONT@UCLOUVAIN.BE
UCL Machine Learning Group (MLG)
Department of Computing Science and Engineering, INGI,
Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium

Abstract

This paper describes a novel technique, called $\mathcal{D}$-walks, to tackle semi-supervised classification problems in large graphs. We introduce here a betweenness measure based on passage times during random walks of bounded lengths in the input graph. The class of unlabeled nodes is predicted by maximizing the betweenness with labeled nodes. This approach can deal with directed or undirected graphs with a linear time complexity with respect to the number of edges, the maximum walk length considered and the number of classes. Preliminary experiments on the CORA database show that $\mathcal{D}$-walks outperforms NetKit (Macskassy & Provost, 2007) as well as Zhou et al's algorithm (Zhou et al., 2005), both in classification rate and computing time.

1. Introduction

Mining and learning problems involving structured data such as graphs, trees or sequences have received much attention recently. This paper is concerned with semi-supervised classification of nodes in a graph. Given an input graph with some nodes being labeled, the problem is to predict the missing node labels. This problem has numerous applications such as classification of individuals in social networks, linked documents (*e.g.* patents or scientific papers) categorization or protein function prediction, to name a few. Even when the data are not naturally structured as a graph, it can be convenient to build a neighborhood graph of similar examples from an affinity matrix.

Several approaches have been proposed to tackle semi-supervised classification problems in graphs. Kernel methods (Zhou et al., 2005; Tsuda & Noble, 2004) embed the nodes of the input graph into an Euclidean feature space where decision boundaries can be estimated. In general, these techniques cannot easily scale up to large problems due to their high time complexity. Nevertheless, they obtain good results on problems of moderate size. NetKit (Macskassy & Provost, 2007) is an alternative approach which has been recently proposed. This general network learning framework builds a model based on three components: a local classifier to generate class-priors, a relational classifier which takes advantage of the relations in the network to guess the class membership and a collective inferencing component further refines the class predictions. The main advantage of this framework is that each of the three components can be instantiated with various existing methods making it easily adaptable to many situations. This flexibility comes however with a time-consuming tuning process to optimize performance. Compared to the above mentioned kernel methods, it provides good performance while having a lower time complexity.

The approach proposed in this paper, called $\mathcal{D}$-walks, relies on random walks performed on the input graph seen as a Markov chain. More precisely, a betweenness

6th International Workshop on Mining and Learning with Graphs (MLG), Helsinki, Finland, July 4-5, 2008.

measure, based on passage times during random walks of bounded length, is derived for each class (or label category). Unlabeled nodes are assigned to the category for which the betweenness is the highest. The $\mathcal{D}$-walks approach has the following properties: (i) it has a linear time complexity with respect to the number of edges, the maximum walk length considered and the number of classes; such a low complexity allows to deal with very large graphs, (ii) it can handle directed or undirected graphs, (iii) it can deal with multi-class problems and (iv) it has a unique hyper-parameter that can be tuned efficiently.

This paper is organized as follows: section 2 introduces the $\mathcal{D}$-walks approach for semi-supervised classification in graphs and section 3 shows some experimental results obtained with the proposed technique applied to real-life data.

2. Discriminative random walks

We are given an input graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{E}$ is the set of edges. In the sequel, $n = |\mathcal{N}|$ denotes the number of nodes and $m = |\mathcal{E}|$ the number of edges in the graph. Let also A denote the $n \times n$ adjacency matrix of $\mathcal{G}$. Since the graph G may be directed and weighted, the matrix A is not necessarily symmetric nor binary. The graph $\mathcal{G}$ is assumed partially labeled. The nodes in the *labeled set* $\mathcal{L} \subset \mathcal{N}$ are assigned to a category from a discrete set $\mathcal{Y}$. The *unlabeled set* is defined as $\mathcal{U} = \mathcal{N} \setminus \mathcal{L}$. The label of a node $q \in \mathcal{L}$ is written y_q and $\mathcal{L}_y$ denotes the set of nodes in class y, with $n_y = |\mathcal{L}_y|$.

Random walks in a graph can be modeled by a discrete-time Markov chain $\{X_t \in \mathcal{N}\}_{t \in \mathbb{N}}$ (MC) describing the sequence of nodes visited during the walk. The random variable X_t represents the state of the MC reached at the discrete time index t. Since each state of the Markov chain corresponds to a distinct node of the graph, the state set of the MC is simply the set of nodes $\mathcal{N}$. The terms *nodes* and *states* are thus used interchangeably in this paper. The transition probability to state q' at time $t+1$, given that the current state is q at time t, is defined as:

$$P[X_{t+1} = q' \mid X_t = q] = p_{qq'} \triangleq \frac{a_{qq'}}{\sum_{q' \in \mathcal{N}} a_{qq'}}. \quad (1)$$

Thus, from any state q, the (homogeneous) probability to jump to state q' is proportional to the weight $a_{qq'}$ of the edge from q to q'. These transition probabilities are conveniently stored in a $n \times n$ row-stochastic transition matrix $P = \{p_{qq'}\}_{q,q' \in \mathcal{N}}$.

We consider *discriminative random walks* ($\mathcal{D}$-walks, for short) in order to define a betweenness measure used for classifying unlabeled nodes. A $\mathcal{D}$-walk is a random walk starting in a labeled node and ending when any node having the same label (possibly the starting node itself) is reached for the first time.

Definition 1 ($\mathcal{D}$-walk) *Given a MC defined on the state set $\mathcal{N}$, a class $y \in \mathcal{Y}$ and a discrete length $l > 1$, a $\mathcal{D}$-walk is a sequence of state $q_0, \ldots, q_l$ such that $y_{q_0} = y_{q_l} = y$ and $y_{q_t} \neq y$ for all $0 < t < l$.*

The notation $\mathcal{D}_l^y$ refers to the set of all $\mathcal{D}$-walks of length l, starting and ending in a node of class y. We also consider $\mathcal{D}_{\leq L}^y$ referring to all limited $\mathcal{D}$-walks *up to* a given length L. The betweenness function $B_L(q, y)$ measures how much a node $q \in \mathcal{U}$ is located "between" nodes of class $y \in \mathcal{Y}$. The betweenness $B_L(q, y)$ is formally defined as the expected number of times node q is reached during $\mathcal{D}_{\leq L}^y$-walks.

Definition 2 ($\mathcal{D}$-walk betweenness) *Given an unlabeled node $q \in \mathcal{U}$ and a class $y \in \mathcal{Y}$, the $\mathcal{D}$-walk betweenness function $\mathcal{U} \times \mathcal{Y} \to \mathbb{R}^+$ is defined as follows: $B_L(q, y) \triangleq \mathbb{E}\left[\mathrm{pt}(q) \mid \mathcal{D}_{\leq L}^y\right]$, where $\mathrm{pt}(q)$ is the passage times function $\mathcal{N} \to \mathbb{R}^+$ counting the number of times a node q has been visited.*

This betweenness measure is related to the one proposed by Newman in (Newman, 2005). Our measure is however relative to a specific class y rather than to the whole graph. It also considers random walks up to a given length instead of unbounded walks. Bounding the walk length has two major benefits: (i) better results are generally obtained with respect to unbounded walks[1], as enforcing a certain degree of locality improves discrimination between classes (ii) the betweenness measure can be computed very efficiently.

The betweenness computation relies on forward and backward variables, similar to those used in the Baum-Welch algorithm for HMM parameter estimation (Rabiner & Juang, 1993). Given a state $q \in \mathcal{N}$ and a time $t \in \mathbb{N}^0$, the forward variable $\alpha^y(q, t)$ computes the probability to reach state q after t steps without passing through[2] nodes in class y, while starting initially from any state in class y. The forward variables α's are computed using the recurrence (2).

[1] The optimal walk length estimated with cross-validation (see section 3) is generally falling between 6 and 30, depending on the considered graph.

[2] In contrast with *leaving from* a node q, *passing through* q means to jump from some node q' to q and then to leave from q.

$$\left|\begin{array}{lll} (t=1) \quad \alpha^y(q,1) & = & \sum_{q' \in \mathcal{L}_y} \frac{1}{n_y} p_{q'q} \\ (t \geq 2) \quad \alpha^y(q,t) & = & \sum_{q' \in \mathcal{N} \setminus \mathcal{L}_y} \alpha^y(q',t-1)\, p_{q'q} \end{array}\right. \tag{2}$$

Given a state $q \in \mathcal{N}$ and a time $t \in \mathbb{N}^0$, the backward variable $\beta^y(q,t)$ computes the probability that state q is attained by the process t steps before reaching any node labeled y for the first time. The backward variables β's are computed using the recurrence (3).

$$\left|\begin{array}{lll} (t=1) \quad \beta^y(q,1) & = & \sum_{q' \in \mathcal{L}_y} p_{qq'} \\ (t \geq 2) \quad \beta^y(q,t) & = & \sum_{q' \in \mathcal{N} \setminus \mathcal{L}_y} \beta^y(q',t-1)\, p_{qq'} \end{array}\right. \tag{3}$$

The time complexity of the forward and backward recurrences is $\Theta(mL)$, where m is the number of edges and L is the maximal walk length considered.

The betweenness measure follows from the computation of the forward and backward variables:

$$B_L(q,y) = \frac{\sum_{l=1}^{L} \sum_{t=1}^{l-1} \alpha^y(q,t)\beta^y(q,l-t)}{\sum_{l=1}^{L} \sum_{q' \in \mathcal{L}_y} \alpha^y(q',l)} \tag{4}$$

By rearranging adequately the sums in the numerator of equation (4), the time complexity of the betweenness computation for all unlabeled nodes is $\mathcal{O}(nL)$. The time complexity for computing the betweenness with respect to all classes is thus $\Theta(|\mathcal{Y}|mL)$ as, for each class, the computation time is dominated by the cost of the recurrence computations.

Finally, an unlabeled node $q \in \mathcal{U}$ is assigned to the class with the highest betweenness:

$$\hat{y}_q = \arg\max_{y \in \mathcal{Y}} B_L(q,y) \tag{5}$$

3. Experiments

We report here preliminary experiments performed on the `Cora` dataset (Macskassy & Provost, 2007) containing 3582 nodes classified under 7 categories. As this graph is fully labeled, node labels were randomly removed and used as test set. More precisely, we have considered 9 labeling rates[3]: $\{0.1, 0.2, \ldots, 0.9\}$ and for each rate, 10 independent runs were performed. Comparative performances obtained with NetKit (Macskassy & Provost, 2007) and with the approach of Zhou et al. (Zhou et al., 2005) are also provided. The hyper-parameters of each approach have been tuned using ten-fold cross-validation. For the $\mathcal{D}$-walks approach, the unique hyper-parameter to tune is L (optimal L values typically fall between 6 and 30 on this dataset, showing the interest of bounding the walk length) whereas Zhou et al. approach also needs one smoothing parameter to be tuned. The three components of NetKit have been instantiated with the default choice present in the framework: a weighted-vote for the relational classifier and a relaxation labeling for the collective inference component. Figure 1 shows the correct classification rate on test data obtained by each approach for increasing labeling rates. The $\mathcal{D}$-walk approach clearly outperforms its competitors on these data. The $\mathcal{D}$-walks approach is also the fastest method. It requires typically 1.5 seconds of CPU[4] for every graph classification including the auto-tuning of its hyper-parameter L. NetKit takes about 4.5 seconds per graph classification and our implementation of Zhou et al. approach typically takes several minutes. A larger graph of 7,590 nodes and 18,696,340 edges has been successfully classified in about 11 minutes with the $\mathcal{D}$-walks approach whereas neither NetKit nor Zhou et al. methods can be applied to such a large graph due to memory and time constraints on a standard PC[4].

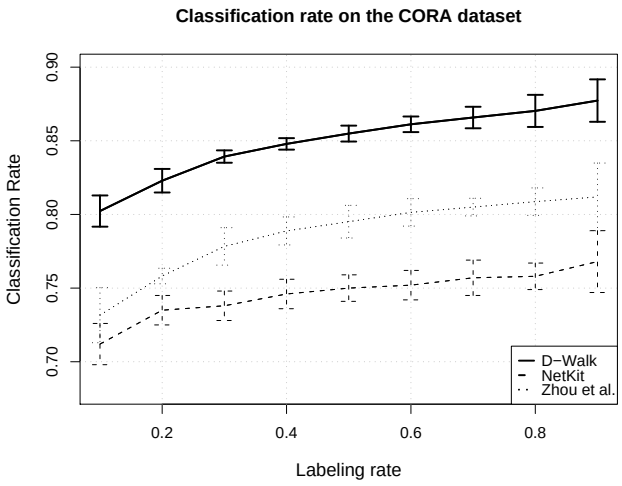

Figure 1. Classification rate of D-walk and two competing methods on the `Cora` dataset. Error bars report standard deviations over 10 independent runs.

References

Macskassy, S. A., & Provost, F. (2007). Classification in networked data: A toolkit and a univariate case

[3]The labeling rate is the proportion of labeled nodes in the graph.

[4]Intel Core 2 Duo 2.2Ghz with 2Gb of virtual memory.

study. *J. Mach. Learn. Res.*, *8*, 935–983.

Newman, M. (2005). A measure of betweenness centrality based on random walks. *Social networks*, *27*, 39–54.

Rabiner, L., & Juang, B.-H. (1993). *Fundamentals of speech recognition*. Prentice-Hall.

Tsuda, K., & Noble, W. S. (2004). Learning kernels from biological networks by maximizing entropy. *Bioinformatics*, *20*, 326–333.

Zhou, D., Huang, J., & Schölkopf, B. (2005). Learning from labeled and unlabeled data on a directed graph. *ICML '05: Proceedings of the 22nd international conference on Machine learning* (pp. 1036–1043). New York, NY, USA: ACM.