Pattern Recognition 44 (2011) 1212–1224

Contents lists available at ScienceDirect

Pattern Recognition

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

Semi-supervised classification and betweenness computation on large, sparse, directed graphs

Amin Mantrach [a,*], Nicolas van Zeebroeck [a], Pascal Francq [b], Masashi Shimbo [c], Hugues Bersini [a], Marco Saerens [b]

[a] IRIDIA Laboratory, Université Libre de Bruxelles, 50 Av. Fr. Roosevelt, B-1050 Brussels, Belgium
[b] ISYS/LSM & Machine Learning Group, Université de Louvain, Place des Doyens 1, B-1348 Louvain-la-Neuve, Belgium
[c] Graduate School of Information Science, Nara Institute of Science and Technology, 8916-5 Takayama, Ikoma, Nara, Japan

ARTICLE INFO

Article history:
Received 7 December 2009
Received in revised form
20 October 2010
Accepted 25 November 2010
Available online 30 November 2010

Keywords:
Graph mining
Semi-supervised classification
Within-network classification
Betweenness centrality
Graph-based classification
Kernel methods
Kernel on a graph
Large-scale graphs

ABSTRACT

This work addresses graph-based semi-supervised classification and betweenness computation in large, sparse, networks (several millions of nodes). The objective of semi-supervised classification is to assign a label to unlabeled nodes using the whole topology of the graph and the labeling at our disposal. Two approaches are developed to avoid explicit computation of pairwise proximity between the nodes of the graph, which would be impractical for graphs containing millions of nodes. The first approach directly computes, for each class, the sum of the similarities between the nodes to classify and the labeled nodes of the class, as suggested initially in [1,2]. Along this approach, two algorithms exploiting different state-of-the-art kernels on a graph are developed. The same strategy can also be used in order to compute a betweenness measure. The second approach works on a trellis structure built from biased random walks on the graph, extending an idea introduced in [3]. These random walks allow to define a biased bounded betweenness for the nodes of interest, defined separately for each class. All the proposed algorithms have a *linear computing time* in the number of edges while providing good results, and hence are applicable to large sparse networks. They are empirically validated on medium-size standard data sets and are shown to be competitive with state-of-the-art techniques. Finally, we processed a novel data set, which is made available for benchmarking, for multi-class classification in a large network: the *U.S. patents citation network* containing 3M nodes (of six different classes) and 38M edges. The three proposed algorithms achieve competitive results (around 85% classification rate) on this large network–they classify the unlabeled nodes within a few minutes on a standard workstation.

1. Introduction

Within-network semi-supervised classification has received a growing focus in recent years (see [4,5] for a comprehensive survey of the topic). In such a setting, one tries to assign a label to the unlabeled nodes of a graph. Since the topology of the entire graph is used (including the unlabeled nodes), the problem is semi-supervised. Despite the growing need for dealing with huge real-world networks, few of the existing methods scale up to large graphs[1] so that semi-supervised classification on large graphs has become one of the current central issues; see the survey [4, Section 6.3]. Indeed, the techniques that scale well [6] are not always competitive when compared to state-of-the-art graph-based metrics [7] such as the regularized Laplacian kernel [8], the sum-over-paths (SoP) covariance [9], the random walk with restart similarity and its normalized version [10,11,7], or the Markov diffusion kernel [12]. A naive application of these graph kernel-based approaches does not scale well since it relies on the computation of a dense similarity matrix, which usually requires a matrix inversion. Techniques approximating the inverse of the matrix usually require some strong properties on the matrix, like the positive semi-definiteness [13], and are only conceivable for medium-size graphs (up to 50,000 nodes)—for larger graphs, a dense similarity matrix cannot be computed and stored into memory.

This paper tackles this problem with two different approaches. The first approach is based on existing, competitive, kernels on a graph, but it explicitly avoids the computation of the pairwise similarities between the nodes (following an idea suggested by Zhou et al. [1,2]). Indeed, as opposed to [11,14,9], Zhou et al. suggest

* Corresponding author. Tel.: +32 650 27 79; fax: +32 650 27 15.
E-mail addresses: amantrac@ulb.ac.be (A. Mantrach), Nicolas.van.Zeebroeck@ulb.ac.be (N. van Zeebroeck), pascal.francq@uclouvain.be (P. Francq), shimbo@is.naist.jp (M. Shimbo), bersini@ulb.ac.be (H. Bersini), marco.saerens@uclouvain.be (M. Saerens).

[1] In this work, we consider that a graph is large scale when the number of nodes exceeds 10^5—one graph investigated in this work has more than 3×10^6 nodes.

0031-3203/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2010.11.019

to avoid computing each pairwise measure and solving a system of linear equations instead. We design two iterative algorithms along this approach, each based on a different state-of-the-art similarity metric: the SoP covariance kernel [9] and the normalized random walk with restart. This kernel on a graph was called regularized commute-time kernel in [7] and is closely related to the modified Laplacian matrix [15], as well as the random walk with restart similarity [10,11]. As suggested initially in [1,2], the algorithms directly approximate the sum of similarities to labeled nodes. The second approach takes its inspiration from the randomized shortest path framework of [16,9] and the $\mathcal{D}-$walk algorithm based on bounded random walks [3]. In this case, a random walk betweenness [17], measuring how well a node is "in-between" each class, is derived from a trellis structure constructed from a biased random walk on the graph.

1.1. Contribution and organization of the paper

This work makes three main contributions:

- It provides three algorithms to address within-network semi-supervised classification tasks on large, sparse, directed graphs. All these algorithms have a *computing time linear in the number of edges* of the graph. Moreover, an algorithm allowing to compute the SoP betweenness centrality [9] is also proposed.
- It validates the three proposed algorithms on eight medium-size standard data sets and compares them to state-of-the-art techniques. Their performances are shown to be competitive in comparison with the other techniques. Results are also computed on a standard large-scale data set [18].
- It introduces a *novel benchmark data set*: The U.S. patents citation network, on which our three algorithms obtain competitive results.

The subsequent part, Section 2, introduces the necessary background and notations. Then, in Section 3, two iterative algorithms are derived from the assumptions of local and global consistency. Further, Section 4 defines a biased bounded betweenness and proposes a forward/backward algorithm to compute it. Section 5 applies our three algorithms to semi-supervised classification tasks and compares the results to various state-of-the-art techniques. A novel benchmark data set is also introduced: the U.S. patents citation network on which our three algorithms are assessed. Section 6 discusses the related work. Finally, the last part of the article includes conclusions and remarks as well as further extensions.

2. Background and notations

Consider a weighted directed graph or network, G, not necessarily strongly connected, with a set of n nodes V (or vertices) and a set of arcs E (or edges). Also consider a set of classes, $\mathcal{L}$. It is assumed that each node belongs to exactly one class—but the class label can be unknown. Moreover, let us define an n-dimensional indicator vector, $\mathbf{y}^c$, containing as entries 1 when the corresponding node belongs to class c and 0 otherwise (in which case the node is unlabeled or belongs to another class). To each arc linking node k and k' a positive number $c_{kk'}>0$, representing the *immediate cost* of following this arc, is associated. The *cost matrix* $\mathbf{C}$ is the matrix containing the immediate costs $c_{kk'}$ as elements.

A random walk on this graph is defined in the standard way. In node k, the random walker chooses the next arc to follow according to transition probabilities representing the probability of jumping from node k to node $k'\in S(k)$, the set of successor nodes (successors S). These transition probabilities will be denoted as $p_{kk'}$ with $k'\in S(k)$. If there is no arc between k and k', we simply consider that $c_{kk'}$ takes an arbitrary large value, denoted by ∞; in this case, the corresponding transition probability will be set to zero, $p_{kk'}=0$. The *natural random walk* on this graph is defined in the following way: it corresponds to a random walk with transition probabilities

$$p_{kk'}=\frac{1/c_{kk'}}{\sum_{k'\in S(k)}(1/c_{kk'})} \tag{1}$$

The corresponding transition matrix will be denoted as $\mathbf{P}$. In other words, the random walker chooses to follow an arc with a probability proportional to the inverse of the immediate cost (apart from the sum-to-one normalization), therefore locally favoring arcs having a low cost. Instead of $\mathbf{C}$, we might be given an adjacency matrix $\mathbf{A}$ with elements $a_{kk'}\geq 0$ indicating the affinity between node k and node k'. In this case, the corresponding costs could be computed from $c_{kk'}=1/a_{kk'}$ and the transition probabilities associated to each node are simply proportional to the affinities (and normalized).

3. Kernel-based semi-supervised classification on large sparse graphs

Three approaches for semi-supervised classification on large sparse graphs are investigated in this paper. The first two approaches (detailed in Sections 3.2 and 3.3) are based on approximating, or bounding, standard kernel-based techniques, and are developed in this section. They will therefore be referred to as *approximate approaches*. The third approach, discussed in detail in Section 4, is a generalization of the discriminative random walks classifier ($\mathcal{D}-$walks, [3]).

3.1. Kernel-based classification

The approximate approaches are kernel-based and adopt the simple following classification procedure (the consistency method), initially proposed by Zhou et al. in [1,2] (see also [19–22]). Based on a regularization framework for the optimization a loss function, this classification procedure takes both available class labels and smoothness into account. The resulting decision procedure is based on a simple sum of similarities (each similarity being provided by an element of the graph kernel matrix) with the labeled nodes (as described in [1,2] for instance). This technique has been used with other kernels than those initially proposed by Zhou et al. with competitive results [9,7]. It corresponds to a simple alignment between the kernel matrix and the class membership vector.

More precisely, suppose that we are given a meaningful proximity matrix $\mathbf{K}$ (usually a graph kernel matrix; see e.g. [23]) providing similarities k_{ij} between each pair of nodes of the graph G. For each node, its total similarity with nodes belonging to class c is contained in the column vector $\mathbf{s}^c=\mathbf{K}\mathbf{y}^c$. Then, each node is assigned to the class showing the largest similarity; the predicted class index is thus provided by $\mathrm{argmax}_c(\mathbf{s}^c)$ for all nodes. In this section, we propose to directly estimate this sum of similarities $\mathbf{s}^c$ for two different metrics, i.e. the *sum-over-paths* (SoP) covariance [9] and the so-called *regularized commute-time kernel* [7], called in this paper the *normalized random walk with restart*.

The SoP covariance kernel is related to other, already available, kernels on a graph [23,12,7], such as, e.g., the commute-time kernel [24,25]. The commute-time kernel is the natural kernel associated to the commute-time distance (also called resistance distance [26]), the average number of steps that a random walker, starting from a given node, takes for entering another node for the first time and afterward going back to the starting node. As explained in [7,24], most of these kernels on a graph define a similarity measure