Neural Networks 31 (2012) 53–72

Contents lists available at SciVerse ScienceDirect

Neural Networks

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

An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification

François Fouss *, Kevin Francoisse, Luh Yen, Alain Pirotte, Marco Saerens

Université catholique de Louvain, ICTEAM & LSM, Louvain-la-Neuve and Mons, Belgium

ARTICLE INFO

Article history:
Received 6 December 2010
Received in revised form 24 August 2011
Accepted 2 March 2012

Keywords:
Kernels on graphs
Graph mining
Collaborative recommendation
Semisupervised classification

ABSTRACT

This paper presents a survey as well as an empirical comparison and evaluation of seven kernels on graphs and two related similarity matrices, that we globally refer to as "kernels on graphs" for simplicity. They are the exponential diffusion kernel, the Laplacian exponential diffusion kernel, the von Neumann diffusion kernel, the regularized Laplacian kernel, the commute-time (or resistance-distance) kernel, the random-walk-with-restart similarity matrix, and finally, a kernel first introduced in this paper (the regularized commute-time kernel) and two kernels defined in some of our previous work and further investigated in this paper (the Markov diffusion kernel and the relative-entropy diffusion matrix). The kernel-on-graphs approach is simple and intuitive. It is illustrated by applying the nine kernels to a collaborative-recommendation task, viewed as a link prediction problem, and to a semisupervised classification task, both on several databases. The methods compute proximity measures between nodes that help study the structure of the graph. Our comparisons suggest that the regularized commute-time and the Markov diffusion kernels perform best on the investigated tasks, closely followed by the regularized Laplacian kernel.

1. Introduction

A function $k(\cdot,\cdot)$: $\Omega \times \Omega \rightarrow \mathbb{R}$, for some *input set* or *space* Ω and real numbers $\mathbb{R}$, can be considered a similarity measure (i.e., enjoying natural properties of similarity like the triangular property Chebotarev & Shamis, 1998a; Theodoridis & Koutroumbas, 2008) if, given two objects x and y in Ω, the real number $k(x, y)$ characterizes in an intuitively adequate manner the similarities and differences (i.e., distance, proximity) of x and y. When the objects are expressed in a vector (or inner-product) space, one such simple and classical similarity measure is the inner product of $\mathbf{x}$ and $\mathbf{y}$: $k(x, y) = k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^{\mathrm{T}}\mathbf{y}$.

Positive definite functions $k(\cdot,\cdot)$ – or kernel functions (Scholkopf & Smola, 2002; Shawe-Taylor & Cristianini, 2004) – enjoy two important properties: (i) computing $k(x, y)$ is equivalent to computing the inner product of some transformation of objects x and y into another, high-dimensional, space (often called the "feature space", although this terminology is a bit confusing) where the data is more likely to be well-separated, while not requiring the explicit computation of the mapping from input space to feature space (only the pairwise similarities $k(x, y)$ are required), and (ii) kernels allow to compute similarities between structured objects that cannot be naturally represented by a simple set of numbers. The latter property is illustrated in this paper, with the main objective of computing similarities between objects represented as nodes of a graph.

Thus, a useful kernel (Scholkopf & Smola, 2002; Shawe-Taylor & Cristianini, 2004) is expected to capture an appropriate measure of similarity for a particular task (like classification or collaborative recommendation, as investigated in this paper) and to be computed efficiently directly from the original objects using a kernel function.

For each kernel function $k(\cdot,\cdot)$ and set of objects $x_1, x_2, \ldots, x_n \in \Omega$, the $n \times n$ symmetric matrix $\mathbf{K}$ ($[\mathbf{K}]_{ij} = k_{ij} = k(x_i, x_j)$) computing the kernel function for $n \times n$ pairs of objects is called a *kernel matrix*. It satisfies the following properties, that are all alternative definitions for symmetric positive semidefinitiveness (see, e.g., Meyer, 2000): (a) $\mathbf{z}^{\mathrm{T}}\mathbf{K}\mathbf{z} \geqslant 0$ for all $\mathbf{z} \in \mathbb{R}^n$; (b) all the eigenvalues of $\mathbf{K}$ are non-negative; (c) $\mathbf{K}$ is a Gram, or inner product, matrix, and (d) $\mathbf{K}$ can be viewed as a diagonal matrix $\mathbf{\Lambda} \geq 0$ in another coordinate system ($\mathbf{K} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}$ where $\mathbf{U}$ is an orthonormal matrix).

Various types of kernels are described in Shawe-Taylor and Cristianini (2004), namely polynomial kernels, Gaussian kernels, anova kernels, kernels on graphs, kernels on sets, kernels on real numbers, string kernels, and randomized kernels. This paper focuses on *kernels on graphs*: it gives an overview of the field, and it analyzes the results of applying various kernels to a collaborative-recommendation task, viewed as a link prediction

* Corresponding author. Tel.: +32 65323216; fax: +32 65323305.
E-mail addresses: francois.fouss@uclouvain.be (F. Fouss), kevin.francoisse@uclouvain.be (K. Francoisse), luh.yen@uclouvain.be (L. Yen), alain.pirotte@uclouvain.be (A. Pirotte), marco.saerens@uclouvain.be (M. Saerens).

0893-6080/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2012.03.001

problem, and to a semisupervised classification task. Kernels on graphs define similarity measures between graph nodes—the objects of interest. They should not be confused with so-called "graph kernels" that compute similarities between graphs (Gartner, 2008; Vishwanathan, Schraudolph, Kondor, & Borgwardt, 2010). As stated in Vishwanathan et al. (2010), comparing nodes in a graph involves constructing a kernel between nodes, while comparing graphs involves constructing a kernel between graphs. In both cases, the research goal consists in defining kernels that capture the semantics inherent in the graph structure and that are reasonably efficient to evaluate.

Mining structured data, like graphs and relational databases, has been the focus of a growing interest (see for instance Džeroski, 2003 and the other papers in that special issue on multi-relational data mining, or Wilson, Hancock, & Luo, 2005). Kernels on graphs allow to address certain issues about graph structure. Indeed, a number of interesting measures, like a distance measure between graph nodes, are easily derived from a kernel. Distances between pairs of nodes allow, for example, to determine the items that are most similar to a given item, which facilitates labeling and clustering items. Intuitively-appealing measures of similarity between nodes are relatively easy to define. Still, their usefulness is clearly related to their ability to perform adequately in practice. This is precisely the objective of this paper: explore and evaluate nine such measures on real-world tasks of collaborative recommendation and classification.

The similarity measures studied in this paper integrate all paths (direct and indirect) between graph nodes. They have the nice property of increasing when the number of paths connecting two nodes increases and when the "length" of paths decreases. In short, the more short paths connect two nodes, the more similar those nodes are. On the contrary, the usual "shortest path" (also called "geodesic" or "Dijkstra" distance) between nodes of a graph does not necessarily decrease when connections between nodes are added and thus it does not capture the fact that strongly connected nodes are at a smaller distance than weakly connected ones. The similarity measures of this paper are usually easy to compute and have an intuitive interpretation, although they do not scale well for large graphs. Moreover, it is well-known that the distance measure naturally derived from a kernel matrix is Euclidean (see Section 2.2.4), that is, the nodes of the graph can be mapped to a Euclidean space preserving distances between nodes. This property can be used to, e.g., visualize the graph in a lower-dimensional space via a principal-component analysis (Saerens, Fouss, Yen, & Dupont, 2004; Scholkopf, Smola, & Muller, 1998; Shawe-Taylor & Cristianini, 2004) or a multidimensional scaling (Borg & Groenen, 1997; Cox & Cox, 2001; Mardia, Kent, & Bibby, 1979) to define a subspace projection that enjoys some optimality property. More generally, kernel variants of multivariate statistical techniques can be used to explore the structure of the graph (Shawe-Taylor & Cristianini, 2004).

The contributions of this paper are threefold. (1) We introduce a new kernel on graphs called the regularized commute-time kernel and we further investigate two graph similarity matrices: the Markov diffusion kernel and the relative-entropy diffusion matrix (Fouss, Yen, Pirotte, & Saerens, 2006). (2) We provide a survey as well as (3) a comprehensive experimental comparison of those methods and of a number of kernels that were proposed in the literature, namely the exponential diffusion kernel (Kondor & Lafferty, 2002), the Laplacian exponential diffusion kernel (Kondor & Lafferty, 2002; Smola & Kondor, 2003), the von Neumann diffusion kernel (Shawe-Taylor & Cristianini, 2004), the regularized Laplacian kernel (Ito, Shimbo, Kudo, & Matsumoto, 2005; Smola & Kondor, 2003), the commute-time (or resistance-distance) kernel (Fouss, Pirotte, Renders, & Saerens, 2007; Saerens et al., 2004), and the random-walk-with-restart similarity matrix (Pan, Yang, Faloutsos, & Duygulu, 2006; Tong, Faloutsos, & Pan, 2006, 2008). A discussion of the rationale behind these kernels is also provided. The experiments indicate that three methods obtain good and stable performance, and that they outperform the other ones. Still, our objective here is not to develop state-of-the-art collaborative-recommendation or semisupervised classification systems; rather, this paper aims at (i) providing a survey of the various graph similarity measures proposed in the literature, (ii) illustrating their usefulness for nontrivial database mining tasks, and (iii) comparing them.

We primarily investigate the extent to which the methods studied capture similarities between nodes. Thus the kernel-based scoring rules for both experimental tasks (recommendation and semisupervised classification) were directly based on simple averages or sums of these computed similarities. Of course, we expect that a finer tuning of similarities will improve the global results. The methods experimented here are, in fact, not kernel-based but rather similarity-based—that is, the methods do not require the similarity matrix to be a kernel matrix. Most of the similarity matrices tested are indeed positive semidefinite and therefore they can be regarded as kernel matrices, but that property is not exploited by the methods. For other state-of-the-art collaborative recommendation techniques, see for instance (Cheung, Tsui, & Liu, 2004; Delannay & Verleysen, 2008; Hofmann, 2004; Koren, 2008; Koren, Bell, & Volinsky, 2009; Paterek, 2007; Salakhutdinov, Mnih, & Hinton, 2007) and the references therein. The four last models were specifically developed in the context of the Netflix Prize. Concerning state-of-the-art semisupervised classification, the current trend is to design the kernel for the task at hand (see for instance Johnson & Zhang, 2008; Kunegis & Lommatzsch, 2009; Lu, Jain, & Dhillon, 2009; Shaw & Jebara, 2009); for a comprehensive survey of semisupervised classification, including graph-based classification, see Abney (2008); Chapelle, Scholkopf, and Zien (2006); Zhu (2008); and Zhu and Goldberg (2009).

This paper is an extended and improved follow-up of an earlier short conference paper (Fouss et al., 2006) presenting preliminary results. The main extensions in this paper concern (a) a more comprehensive literature review (Section 3), (b) the investigation of two additional algorithms (the random-walk-with-restart similarity matrix in Section 4.8 and the regularized commute-time kernel in Section 4.9) and of their link with the other kernels, (c) the introduction of the cosine method and the repositioning of the Basic method, previously called MaxF (Section 5.4), (d) a nested cross-validation (instead of a simple cross-validation) for assessing the performance of the algorithms (Section 5), (e) a thorough redesign of all experiments, and (f) the application of the algorithms to semisupervised classification (Section 6).

Besides this introduction, the paper is structured as follows. Section 2 reviews some relevant basic concepts from the theory of multidimensional scaling – borrowed from the support-vector-machine community (at least for finite-dimension spaces) – defining distance measures and optimal subspace projections from kernels on graphs (Borg & Groenen, 1997; Cox & Cox, 2001; Mardia et al., 1979). Section 3 reviews related work on similarity measures and kernels on graphs. Section 4 briefly presents the nine kernels investigated. Sections 5 and 6 introduce the experiments that illustrate the kernel approach: the comparison between the nine kernels on two collaborative-recommendation tasks (Section 5) and on seven semisupervised classification tasks (Section 6). Section 7 provides a general discussion of the experimental results. Section 8 concludes the paper.

2. Theory review

This section briefly introduces the mathematical notations and the basic theory behind kernel matrices on graphs. In a very general setting, once some similarity measure between the nodes of a graph is shown to be a kernel, then a number of interesting properties automatically follow almost for free. More precisely, from a kernel matrix, it is possible to define a data matrix where each row represents a node of the graph (a node vector) as an r-dimensional vector in a Euclidean space (the feature space). These node vectors are such that their inner products are the elements of the kernel matrix. Through those node vectors, the graph can be viewed as a cloud of points in this r-dimensional space and standard multivariate statistical analysis technique become applicable.

2.1. Adjacency and Laplacian matrix of a graph

Consider a weighted, undirected, graph G with symmetric weights $w_{ij} > 0$ between pairs of nodes i and j linked by an edge. The elements a_{ij} of the adjacency matrix $\mathbf{A}$ of the graph are defined as usual as $a_{ij} = w_{ij}$ if node i is linked to node j and $a_{ij} = 0$ otherwise. The Laplacian matrix is $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D} = \mathbf{Diag}(a_{i\bullet})$ is the (generalized) degree matrix, with diagonal entries $d_{ii} = [\mathbf{D}]_{ii} = a_{i\bullet} = \sum_{j=1}^{n} a_{ij}$, if the graph has n nodes in total. The volume of the graph is defined as $V_G = \mathrm{vol}(G) = \sum_{i=1}^{n} d_{ii} = \sum_{i,j=1}^{n} a_{ij}$. The weight w_{ij} of the edge connecting node i and node j can be set so that the more important the relation or affinity is between node i and node j, then the larger the value of w_{ij} is. As for notations, the name of column vectors is written in bold lowercase while that of matrices is in bold uppercase.

If the graph is connected, that is, if every node is reachable from every node, then $\mathbf{L}$ has rank $n - 1$ (Chung, 1997). If $\mathbf{e}$ is a column vector all of whose elements are "1" (i.e., $\mathbf{e} = [1, 1, \ldots, 1]^{\mathrm{T}}$, where T denotes the matrix transpose) and $\mathbf{0}$ is a column vector all of whose elements are "0", $\mathbf{Le} = \mathbf{0}$ and $\mathbf{e}^{\mathrm{T}}\mathbf{L} = \mathbf{0}^{\mathrm{T}}$ hold: $\mathbf{L}$ is doubly centered. The null space of $\mathbf{L}$ is thus the one-dimensional space spanned by $\mathbf{e}$. Moreover, $\mathbf{L}$ is symmetric and positive semidefinite (see for instance Chung, 1997). Notice that, if the graph is not connected, it can be decomposed into closed subsets of nodes and the analysis can be applied independently on those subsets. Efficient algorithms for finding connected components are readily available (Jungnickel, 2008; Sedgewick, 1990).

2.2. Kernels on graphs

A real symmetric matrix that is not positive semidefinite can be changed to a positive semidefinite one by, e.g., adding the identity matrix $\mathbf{I}$ multiplied by a suitably positive constant to the original matrix (see, for example, Roth, Laub, Buhmann, & Müller, 2003; Rousseuw & Molenberghs, 1993). Indeed, if $\mathbf{M}$ is a symmetric matrix, the matrix $(\mathbf{M} + \alpha\mathbf{I})$, with $\alpha \geqslant |\lambda_{\min}|$ (where $\lambda_{\min}$ is the lowest eigenvalue of $\mathbf{M}$) is positive semidefinite. This holds because $\mathbf{M}$ and $(\mathbf{M} + \alpha\mathbf{I})$ have the same eigenvectors $\mathbf{u}_i$ associated with eigenvalues λ_i for $\mathbf{M}$ and $(\lambda_i + \alpha)$ for $(\mathbf{M} + \alpha\mathbf{I})$. Thus, by choosing $\alpha \geqslant |\lambda_{\min}|$, all the eigenvalues are non-negative and $(\mathbf{M} + \alpha\mathbf{I})$ is positive semidefinite.

Nine similarity matrices, defined from the adjacency matrix $\mathbf{A}$, are described in Section 4, and tested on real-world data in Sections 5 and 6. The main concepts basically come from the fields of multidimensional scaling (Borg & Groenen, 1997; Cox & Cox, 2001; Mardia et al., 1979) and kernel methods (Gartner, 2008; Scholkopf & Smola, 2002; Shawe-Taylor & Cristianini, 2004).

2.2.1. The data matrix associated to a kernel matrix

Let us first recall a fundamental spectral-decomposition theorem from linear algebra, underlying kernel-based work (see, for example, Mardia et al., 1979; Noble & Daniels, 1988):

Any $n \times n$ real positive semidefinite matrix $\mathbf{K}$ of rank r can be expressed as $\mathbf{K} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}} = \mathbf{U}\mathbf{\Lambda}^{1/2}(\mathbf{U}\mathbf{\Lambda}^{1/2})^{\mathrm{T}} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$, where $\mathbf{\Lambda}$ is the diagonal matrix containing the non-negative eigenvalues of $\mathbf{K}$, $\mathbf{U}$ is the $n \times r$ orthonormal matrix whose columns are the normalized eigenvectors $\mathbf{u}_i$ of $\mathbf{K}$, and $\mathbf{X} = \mathbf{U}\mathbf{\Lambda}^{1/2}$ is a $n \times r$ matrix of rank r. Thus, the r columns $\mathbf{c}_i = \sqrt{\lambda_i}\,\mathbf{u}_i$ of $\mathbf{X}$ correspond to the orthonormal eigenvectors $\mathbf{u}_i$ of $\mathbf{K}$ multiplied by the square root of their corresponding eigenvalue.

Therefore, if $\mathbf{x}_i = \mathbf{row}_i(\mathbf{X})^{\mathrm{T}}$ is the column vector corresponding to column i of $\mathbf{X}^{\mathrm{T}}$ (transposed row i of $\mathbf{X}$), then the entries of $\mathbf{K}$ are inner products $[\mathbf{K}]_{ij} = k_{ij} = \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$, in accordance with the fact that $\mathbf{K}$ is a kernel. The nodes of the graph are thus represented as vectors $\mathbf{x}_i$ in a r-dimensional Euclidean space. The $\mathbf{x}_i$'s are called *node vectors*, while matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^{\mathrm{T}}$ containing the transposed node vectors as rows is the *data matrix* associated to the kernel matrix on a graph. This embedding is defined up to an isometry, but the coordinates of the node vectors obtained through spectral decomposition have the nice property of being expressed in the principal-component space (uncentered if the kernel is not centered).

The Euclidean space in which the node vectors are defined is the so-called feature space. Many techniques have been used to visualize the nodes in a low-dimensional space (Lee & Verleysen, 2007; Shawe-Taylor & Cristianini, 2004) but this question is not investigated in this paper.

2.2.2. Centering the data matrix

Usually, there is a preference for working with centered vectors, so that the center of gravity of the node vectors $\mathbf{x}_i$ is $\mathbf{0}$. For instance, most multivariate statistical techniques assume that the data has been centered on the center of gravity of the data cloud (Jolliffe, 2002; Mardia et al., 1979). This amounts to subtracting its mean to each feature of the data matrix, therefore centering it. A data matrix is centered if $\mathbf{X}^{\mathrm{T}}\mathbf{e} = \mathbf{0}$, that is, the sum of the elements of each column of $\mathbf{X}$ is 0. It can easily be shown that a symmetric kernel matrix $\mathbf{K}$ is centered (that is, it corresponds to inner products of centered node vectors) if and only if $\mathbf{Ke} = \mathbf{0}$.

A centered kernel matrix $\overline{\mathbf{K}}$ is obtained by centering the node vectors, that is, by applying the symmetric centering matrix $\mathbf{H} = \mathbf{I} - n^{-1}\mathbf{e}\mathbf{e}^{\mathrm{T}}$ (Mardia et al., 1979) to the data matrix $\mathbf{X}$. Indeed, premultiplying a data matrix by $\mathbf{H}$ re-expresses each element as a deviation from its column mean, i.e., $\overline{\mathbf{X}} = \mathbf{HX}$ has its (i, j)th element $x_{ij} - \bar{x}_j$, where $\bar{x}_j$ is the mean of the jth column of $\mathbf{X}$. Thus, the *centered kernel matrix* $\overline{\mathbf{K}}$ is defined as $\overline{\mathbf{K}} = (\mathbf{HX})(\mathbf{HX})^{\mathrm{T}} = \mathbf{HXX}^{\mathrm{T}}\mathbf{H} = \mathbf{HKH}$.

2.2.3. The cosine, or normalized, kernel matrix

Inner products are not always an appropriate similarity measure. In some fields, such as information retrieval (Baeza-Yates & Ribeiro-Neto, 1999; Manning, Raghavan, & Schutze, 2008), the cosine similarity measure is preferred:

$$\cos_{ij} = \frac{\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j}{\|\mathbf{x}_i\|\,\|\mathbf{x}_j\|} = \frac{k_{ij}}{\sqrt{k_{ii}k_{jj}}}. \tag{1}$$

It corresponds to a kernel matrix since its elements are the inner products of the normalized node vectors $\mathbf{x}_i/\|\mathbf{x}_i\|$ (Shawe-Taylor & Cristianini, 2004). In matrix form, $\mathbf{Cos} = \mathbf{Diag}(\mathbf{K})^{-1/2}\mathbf{K}\,\mathbf{Diag}(\mathbf{K})^{-1/2}$, where $\mathbf{Diag}(\mathbf{K})$ is a diagonal matrix containing the diagonal of $\mathbf{K}$.

2.2.4. Natural distance measure associated to a kernel matrix

A *distance measure* between pairs of nodes in the feature space (therefore also corresponding to a Euclidean distance between the

node vectors of the data matrix) can be derived from the kernel matrix:

$$\begin{aligned}\delta_{ij}^2 &= \|\mathbf{x}_i - \mathbf{x}_j\|^2 = (\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}}(\mathbf{x}_i - \mathbf{x}_j)\\ &= \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_i + \mathbf{x}_j^{\mathrm{T}}\mathbf{x}_j - 2\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j = k_{ii} + k_{jj} - 2k_{ij}\\ &= (\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}}\mathbf{K}(\mathbf{e}_i - \mathbf{e}_j) \qquad (2)\end{aligned}$$

where $\mathbf{e}_i$ is a column vector with "0" entries, except in position i whose entry is "1" (i.e., $\mathbf{e}_i = [\underset{1}{0}, \ldots, \underset{i-1}{0}, \underset{i}{1}, \underset{i+1}{0}, \ldots, \underset{n}{0}]^{\mathrm{T}}$).

Since δ_{ij} corresponds to a Euclidean distance in $\mathbb{R}^r$, it satisfies all the basic properties of a distance (positiveness, symmetry, and triangular inequality). Distances between pairs of elements allow, for instance, to use a clustering algorithm to group nodes that are most similar (Shawe-Taylor & Cristianini, 2004; Yen, Fouss, Decaestecker, Francq, & Saerens, 2007, 2008; Yen et al., 2005). In matrix form, $\mathbf{\Delta}^{(2)} = \mathbf{diag}(\mathbf{K})\mathbf{e}^{\mathrm{T}} + \mathbf{e}(\mathbf{diag}(\mathbf{K}))^{\mathrm{T}} - 2\mathbf{K}$, where $\mathbf{diag}(\mathbf{K})$ is a column vector containing the diagonal elements of $\mathbf{K}$, and $\left[\mathbf{\Delta}^{(2)}\right]_{ij} = \delta_{ij}^2$ contains the squared distances (Borg & Groenen, 1997; Cox & Cox, 2001; Mardia et al., 1979).

With the cosine similarity measure, the distance reduces to $\delta_{ij} = \mathrm{sqrt}(2(1 - \cos_{ij}))$. The other way around, given a square Euclidean distance matrix $\mathbf{\Delta}^{(2)}$, the natural centered kernel matrix associated to $\mathbf{\Delta}^{(2)}$ is $\mathbf{K} = -\frac{1}{2}\mathbf{H}\mathbf{\Delta}^{(2)}\mathbf{H}$ (Borg & Groenen, 1997; Cox & Cox, 2001; Mardia et al., 1979).

Once a data matrix $\mathbf{X}$ has been derived from $\mathbf{K}$, standard multivariate statistical analysis methods can be used to investigate the structure of the data cloud. Kernel-based algorithms, such as kernel principal-component analysis, are applied directly to the kernel matrix without computing the data matrix in the feature space (Scholkopf et al., 1998; Shawe-Taylor & Cristianini, 2004).

3. Related work

This section provides an overview of the various similarity measures between nodes of a graph that were proposed in the literature, to the best of our knowledge, and that are investigated in this paper. Similarity between nodes has also been called relatedness (White & Smyth, 2003). The most well-known quantities measuring relatedness are co-citation (Small, 1973) and bibliographic coupling (Kessler, 1963). These are the two baseline measures that are adopted in our collaborative-recommendation experiments.

More sophisticated measures were also suggested. For instance, a similarity measure between nodes of a graph integrating indirect paths, based on the matrix-forest theorem, was proposed in Chebotarev and Shamis (1997, 1998b). While some nice properties were proven about this similarity measure, no experiment investigating its effectiveness was reported. It is in fact the same measure as the regularized Laplacian kernel described in Section 4.4.

A modified regularized Laplacian kernel was proposed in Ito et al. (2005) and Shimbo, Ito, Mochihashi, and Matsumoto (2009) as an extension of the regularized Laplacian kernel, by introducing a new parameter controlling importance and relatedness. It was shown in Ito et al. (2005), Shimbo and Ito (2006) and Shimbo et al. (2009) that the regularized Laplacian kernel overcomes some limitations of the von Neumann kernel (Kandola, Cristianini, & Shawe-Taylor, 2002), when ranking linked documents. This modified regularized Laplacian kernel is also closely related to a graph-regularization framework introduced by Zhou, Bousquet, Lal, Weston, and Scholkopf (2003) (see also Belkin, Matveeva, & Niyogi, 2004; Wang, Wang, Zhang, Shen, & Quan, 2009; Yajima & Kuo, 2006; Zhou & Scholkopf, 2004), and extended to directed graphs in Zhou, Huang, and Scholkopf (2005). The resulting kernel on graphs is a normalized version of the "regularized Laplacian kernel" involving the extension of the Laplacian matrix for directed graphs (Chung, 2005). In the same spirit, Chen, Yang, and Tang (2007) studied graph embedding and semisupervised classification by using this directed version of the regularized Laplacian kernel.

The exponential and the von Neumann diffusion kernels, based on the adjacency matrix, were introduced in Kandola et al. (2002) and Shawe-Taylor and Cristianini (2004). They are computed through a power series of the adjacency matrix of the graph. They are therefore closely related to graph regularization models. Indeed, a general method for constructing natural families of kernels over discrete structures, based on the matrix-exponentiation idea, was introduced in Kondor and Lafferty (2002) and Smola and Kondor (2003) (see also Lafferty & Lebanon, 2005. The focus of that work is on generating kernels on graphs, with the proposal of a special class of exponential kernels called diffusion kernels (see Sections 4.1 and 4.2). That work can be considered as the first attempt to explicitly define kernels on graphs.

The "commute-time" kernel was introduced in Fouss et al. (2007) and Saerens et al. (2004); it was inspired by the work of Chandra, Raghavan, Ruzzo, Smolensky, and Tiwari (1989) and Klein and Randic (1993). It takes its name from the average commute time, that is, 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 returning to the starting node. The commute-time kernel is defined as the inner product in a Euclidean space where the nodes are exactly separated by the commute-time distance (proportional to the resistance-distance, see Chandra et al., 1989; Klein & Randic, 1993). This kernel performs well for collaborative recommendation, as documented in Fouss et al. (2007). The same commute-time embedding was defined in Ham, Lee, Mika, and Scholkopf (2004), Qiu and Hancock (2005, 2007) and Yen et al. (2005) as well as (Brand, 2005), preserving the commute-time distance. It was applied to image segmentation and multi-body motion tracking (Qiu & Hancock, 2005, 2007), to dimensionality reduction of manifolds (Ham et al., 2004), to clustering (Yen et al., 2005), and to collaborative filtering (Brand, 2005; Fouss et al., 2007), with interesting results. The commute-time kernel is also closely related to the "Fiedler vector" (Fiedler, 1975b; Mohar, 1992), widely used for graph partitioning (Chan, Ciarlet, & Szeto, 1997; Pothen, Simon, & Liou, 1990) and clustering (Donetti & Munoz, 2004), as detailed in Fouss et al. (2007), but also to discrete Green functions (Ding, Jin, Li, & Simon, 2007). Finally, a family of dissimilarity measures reducing to the shortest-path distance at one end and to the commute-time (or resistance) distance at the other end was proposed in Yen, Mantrach, Shimbo, and Saerens (2008) and subsequently in Chebotarev (2011) while a correlation or co-betweenness measure between nodes was developed in Kolaczyk (2009) and Mantrach et al. (2010). The commute-time kernel is described in Section 4.5.

An intuitively appealing distance measure between nodes of a graph was proposed in Latapy and Pons (2005), Nadler, Lafon, Coifman, and Kevrekidis (2005, 2006) and Pons and Latapy (2006) based on a continuous-time diffusion process, called the "diffusion distance". We defined a discrete-time counterpart of their distance and used it to define the "Markov diffusion kernel" (see Fouss et al., 2006 and Section 4.6 of this paper). An application of the diffusion distance to dimensionality reduction and graph visualization was described in Lafon and Lee (2006). The natural embedding induced by the diffusion distance was called "diffusion map" by Nadler et al. (2005, 2006). It is related to correspondence analysis (Yen, Saerens, & Fouss, 2011).

Other papers attempted to adapt the well-known PageRank procedure (Brin & Page, 1998; Page, Brin, Motwani, & Winograd, 1998) to define meaningful similarities between nodes. A random-walk with a restart procedure was proposed in Pan et al. (2006) and Tong et al. (2008). A random-walk process starting from a node of

interest, controlled by a precomputed correlation matrix between nodes, was described in Pucci, Gori and Maggini (2006). A random-walk-with-restart kernel is presented in Section 4.8.

A kernel approach for recommendation tasks based on one-class support-vector machines with kernels generated from a Laplacian matrix was proposed in Yajima and Kuo (2006). Their preliminary experiments on the MovieLens dataset produced interesting results with the commute-time kernel, the regularized commute-time kernel, and the regularized Laplacian kernel. For a special choice of the parameters of the model, their technique reduces to computing a simple alignment of the kernel with a binary vector encoding the selected items of the active user. This scoring technique is in fact equivalent to a regularization technique used in semisupervised classification of graph nodes (Belkin et al., 2004; Johnson & Zhang, 2007, 2008; Wang et al., 2009; Zhou et al., 2003, 2005; Zhu, Ghahramani, & Lafferty, 2003) and it is the method of choice in our semisupervised classification experiments (see Section 6.2).

Several attempts to define similarity measures for directed graphs were proposed as well. For instance (Chung, 2005) proposed to start from the transition-probability matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$ where $\mathbf{D}$ is, as before, a diagonal matrix containing the outdegrees of graph nodes. Then, they compute the left eigenvector $\boldsymbol{\pi}$ of $\mathbf{P}$, $\boldsymbol{\pi}^{\mathrm{T}}\mathbf{P} = \lambda\boldsymbol{\pi}^{\mathrm{T}}$, which corresponds to the stationary distribution of the Markov chain. If the diagonal matrix containing the elements of $\boldsymbol{\pi}$ on its diagonal is denoted by $\boldsymbol{\Pi} = \mathbf{Diag}(\boldsymbol{\pi})$, the Laplacian matrix of the directed graph is then defined by $\mathbf{L} = \boldsymbol{\Pi} - (\boldsymbol{\Pi}\mathbf{P} + \mathbf{P}^{\mathrm{T}}\boldsymbol{\Pi})/2$. It has a number of interesting properties generalizing those of the usual Laplacian matrix for undirected graphs. In particular, it is positive semidefinite and therefore it defines a kernel matrix. Still another extension of the Laplacian matrix to directed graphs has recently been proposed by Boley, Ranjan, and Zhang (2011). It corresponds to $\mathbf{L} = \mathbf{D} - \mathbf{A}$. The authors show how the average commute times for a strongly connected directed graph are related to this asymmetric Laplacian matrix as a direct extension of similar well-known formulas for undirected graphs. Finally, yet another attempt to define similarities between nodes on a directed graph based on the matrix-forest theorem was described in Agaev and Chebotarev (2000, 2001).

Finally, in Chung's extension to directed graphs Chung (2005), the normalized Laplacian matrix can be defined as $\tilde{\mathbf{L}} = \boldsymbol{\Pi}^{-\frac{1}{2}}\mathbf{L}\boldsymbol{\Pi}^{-\frac{1}{2}}$. The regularized normalized Laplacian kernel matrix $(\mathbf{I} - \alpha\tilde{\mathbf{L}})^{-1}$ also appears in the context of semisupervised classification of labeled nodes of a graph (Zhou et al., 2005), while the regularized Laplacian kernel matrix is used for directed graph embedding in Chen et al. (2007). Based on the same idea, (Zhao & Tang, 2007) proposed a directed contextual distance and defined a directed graph from which the Laplacian matrix is computed. It was then used for ranking and clustering images. For undirected graphs, it can be shown that these kernel matrices reduce to variants of the regularized Laplacian kernel. Since all the graphs investigated in our experiments are undirected, we do not provide comparisons with those kernels.

4. Kernels on graphs

This section describes the nine kernels on graphs, already mentioned in Section 3. Several derived measures (a distance measure, the cosine normalized kernels, the PCA-based reduced-rank kernels) are also computed from these kernels and compared on the collaborative-recommendation task in Section 5.

4.1. The exponential diffusion kernel

The so-called *exponential diffusion kernel* $\mathbf{K}_{\mathrm{ED}}$, introduced in Kondor and Lafferty (2002), is defined as

$$\mathbf{K}_{\mathrm{ED}} = \sum_{k=0}^{\infty} \frac{\alpha^k \mathbf{A}^k}{k!} = \exp(\alpha\mathbf{A}) \tag{3}$$

where $\mathbf{A}$ is the adjacency matrix of the graph and exp is the matrix exponential. Element $a_{ij}^k = [\mathbf{A}^k]_{ij}$ of matrix $\mathbf{A}^k$ ($\mathbf{A}$ to the power k) is the number of paths between nodes i and j with exactly k transitions or steps. Thus, the kernel integrates a contribution from all paths between node i and node j, discounting paths according to their number of steps. It favors shorter paths between two nodes by giving them a heavier weight. The discounting factor is $\alpha^k/k!$. Other choices for that factor lead to other kernels, like the von Neumann diffusion kernel, described in Section 4.3.

The $\mathbf{K}_{\mathrm{ED}}$ matrix is clearly positive semidefinite since the exponential of $\mathbf{A}$ amounts to replacing $\boldsymbol{\Lambda}$ (a diagonal matrix containing the eigenvalues of $\mathbf{A}$) by $\exp(\boldsymbol{\Lambda})$ in the spectral decomposition of $\mathbf{A}$ (the matrix exponential of a diagonal matrix is a diagonal matrix whose (i, i) element is the exponential of the corresponding element in the original matrix, therefore always positive).

4.2. The Laplacian exponential diffusion kernel

A meaningful alternative to $\mathbf{K}_{\mathrm{ED}}$ relies on a diffusion model (Smola & Kondor, 2003) that substitutes the adjacency matrix with minus the Laplacian matrix in Eq. (3). It is closely related to continuous-time Markov chains (Kelly, 1979; Norris, 1997; Parzen, 1962). As an intuitive insight, suppose that a quantity $x_i(t)$ is defined on each node i of the graph at time t and that it diffuses to neighboring nodes with a symmetric diffusion rate a_{ij}. Thus, during a small time interval δt, an amount $a_{ij}x_i\delta t$ is transferred from node i to node j, proportional to δt, a_{ij}, and $x_i(t)$. The balance equation (input and output during δt) is given by

$$x_i(t + \delta t) = x_i(t) + \sum_{j=1}^{n} a_{ji}x_j\delta t - \sum_{j=1}^{n} a_{ij}x_i\delta t. \tag{4}$$

Since $\mathbf{A}$ is symmetric, this leads to, for $\delta t \to 0$,

$$\begin{aligned}\frac{dx_i(t)}{dt} &= \sum_{j=1}^{n} a_{ji}x_j - a_{i\bullet}x_i = \sum_{j=1}^{n}(a_{ji}x_j - a_{i\bullet}\delta_{ij}x_j)\\ &= -\sum_{j=1}^{n}(a_{i\bullet}\delta_{ij} - a_{ij})x_j\end{aligned} \tag{5}$$

where δ_{ij} is the Kronecker delta. The elements $l_{ij} = (a_{i\bullet}\delta_{ij} - a_{ij})$ are the entries of the Laplacian matrix. In matrix form,

$$\frac{d\mathbf{x}(t)}{dt} = -\mathbf{L}\,\mathbf{x}(t). \tag{6}$$

This system of differential equations leads to the following solution

$$\mathbf{x}(t) = \exp(-\mathbf{L}t)\mathbf{x}_0 \tag{7}$$

where $\mathbf{x}_0$ is the initial vector $\mathbf{x}$ at time $t = 0$ and exp is the matrix exponential. This leads to the *Laplacian exponential diffusion kernel* $\mathbf{K}_{\mathrm{LED}}$, introduced in Kondor and Lafferty (2002) and Smola and Kondor (2003), and defined as

$$\mathbf{K}_{\mathrm{LED}} = \exp(-\alpha\mathbf{L}) \tag{8}$$

which is similar to Eq. (3), with minus the Laplacian matrix replacing the adjacency matrix. If we substitute $\mathbf{x}_0 = \mathbf{e}_j$ in Eq. (7), then the equation shows that column j of $\mathbf{K}_{\mathrm{LED}}$ corresponds to quantity $\mathbf{x}$ at time $t = \alpha$ when the initial vector is $\mathbf{e}_j$, that is, a unit quantity starting diffusing from node j at $t = 0$.

4.3. The von Neumann diffusion kernel

The *von Neumann diffusion kernel* $\mathbf{K}_{\mathrm{VND}}$ (Kandola et al., 2002; Shawe-Taylor & Cristianini, 2004) differs from the exponential

diffusion kernel by its exponential discounting rate α^k, instead of $\alpha^k/k!$:

$$\mathbf{K}_{\mathrm{VND}} = \sum_{k=0}^{\infty} \alpha^k \mathbf{A}^k = (\mathbf{I} - \alpha\mathbf{A})^{-1} \tag{9}$$

$\mathbf{K}_{\mathrm{VND}}$ is well defined if $0 < \alpha < \rho(\mathbf{A})^{-1}$ where $\rho(\mathbf{A})$ is the spectral radius of the symmetric matrix $\mathbf{A}$ (that is, its largest eigenvalue in absolute value). In this case, $\mathbf{K}_{\mathrm{VND}}$ is positive definite. It was shown in Ito et al. (2005) that this kernel applied to document ranking defines a link-analysis measure intermediate between co-citation/bibliographic coupling (Kessler, 1963; Small, 1973) and HITS (Kleinberg, 1999). Indeed, von Neumann kernels subsume Kleinberg's HITS importance at the limit of their parameter range. Because they reduce to co-citation relatedness at the other end of the parameter, von Neumann kernels provide a spectrum of link analysis measures between the two established measures of importance and relatedness (Shimbo et al., 2009).

4.4. The regularized Laplacian kernel

The *regularized Laplacian kernel* (Chebotarev & Shamis, 1997, 1998b; Ito et al., 2005; Smola & Kondor, 2003) is also called the normalized random walk with restart kernel in Mantrach et al. (2011) because of its links with the random walk with restart model (Pan et al., 2006; Tong et al., 2006, 2008) – see Section 4.8. It differs from the von Neumann diffusion kernel only by substituting the negated Laplacian matrix $-\mathbf{L}$ for the adjacency matrix $\mathbf{A}$:

$$\mathbf{K}_{\mathrm{L}} = \sum_{k=0}^{\infty} \alpha^k (-\mathbf{L})^k = (\mathbf{I} + \alpha\mathbf{L})^{-1} \tag{10}$$

where $0 < \alpha < \rho(\mathbf{L})^{-1}$ and $\rho(\mathbf{L})$ is the spectral radius of $\mathbf{L}$. $\mathbf{K}_{\mathrm{L}}$ is clearly positive definite. The regularized Laplacian kernel has been shown to provide better performance than the von Neumann kernel in a bibliographic citation task (Ito et al., 2005).

This similarity measure has an interesting interpretation in terms of the matrix-forest theorem (Chebotarev & Shamis, 1997, 1998b). Let $\mathcal{F}^i$ be the set of all spanning forests rooted at node i of graph G and $\mathcal{F}^{ij}$ be the set of those spanning rooted forests for which nodes i and j belong to the same tree rooted at i. A spanning rooted forest is an acyclic subgraph of G that has the same nodes as G and one marked node (a root) in each component. It is shown in Chebotarev and Shamis (1997, 1998b) that the matrix $(\mathbf{I}+\mathbf{L})^{-1}$ exists and that $[(\mathbf{I}+\mathbf{L})^{-1}]_{ij} = \epsilon(\mathcal{F}^{ij})/\epsilon(\mathcal{F}^i)$ where $\epsilon(\mathcal{F}^{ij})$ and $\epsilon(\mathcal{F}^i)$ are the total weights of forests that belong to $\mathcal{F}^{ij}$ and $\mathcal{F}^i$ respectively. The elements of this matrix are therefore called "relative forest accessibilities" between nodes. This interpretation can be generalized to the matrix $(\mathbf{I} + \alpha\mathbf{L})^{-1}$ where parameter α weights the number of edges belonging to the forests and limits the size of forests in terms of number of edges (see Chebotarev & Shamis, 1997; Chebotarev & Shamis, 1998b for more detail). This matrix can be shown to be a similarity measure.

This kernel has also been used for semisupervised classification in the following regularization framework (Belkin et al., 2004; Sindhwani, Belkin, & Niyogi, 2006; Wang et al., 2009; Zhou et al., 2003, 2005; Zhu et al., 2003). Let a class membership y_i^c be associated to each node i with value equal to 1 when node i belongs to class c and 0 otherwise. Label consistency is assumed, namely that neighbor nodes are likely to have the same class label (Kolaczyk, 2009; Zhou et al., 2003). This assumption is also usually made in spatial statistics (Cressie, 1993; Haining, 2003) and often called the "cluster assumption", "guilt by association", or "structural autocorrelation".

A reasonable smoothed, predicted, class membership value, $\widehat{y}_i^c$, can be obtained by minimizing

$$\underbrace{\sum_{i=1}^{n} (\widehat{y}_i^c - y_i^c)^2}_{\text{quality of fit}} + \alpha \underbrace{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}(\widehat{y}_i^c - \widehat{y}_j^c)^2}_{\text{smoothness}}. \tag{11}$$

Other supervised classification methods, such as k-nearest neighbors, also rely on such a label consistency assumption. This loss function consists of two parts: a penalty accounting for the deviation of the predicted value from the real value (quality of fit term) and a smoothness term imposing that nearby nodes have similar predicted values (such a smoothing term was already suggested in Hall, 1970). With $\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}(\widehat{y}_i^c - \widehat{y}_j^c)^2 = 2\widehat{\mathbf{y}}^{\mathrm{T}}\mathbf{L}\widehat{\mathbf{y}}$ (Hall, 1970), minimizing this loss function in terms of the predicted values yields

$$\widehat{\mathbf{y}}^c = (\mathbf{I} + \alpha\mathbf{L})^{-1}\mathbf{y}^c \tag{12}$$

which aims to simply compute the sum of similarities with every node of the graph belonging to class c—an alignment with the binary vector encoding class labels[1]. This simple sum-of-similarities procedure is used in Section 6.2 for the semisupervised classification (SC) experiments. For other choices of the quadratic penalty and smoothing terms, minimization of the loss function results in the product of a graph similarity matrix, similar to the kernels on graphs discussed in this paper, with a membership vector $\mathbf{y}^c$ (Zhou et al., 2003). Interestingly, the same technique also appears in collaborative recommendation with one-class support vector machine, for a special choice of the parameters (Yajima & Kuo, 2006).

Finally, a modified regularized Laplacian kernel was proposed in Ito et al. (2005) that introduces a parameter controlling importance and relatedness. A modified Laplacian matrix is defined as $\mathbf{L}_\gamma = \gamma\mathbf{D} - \mathbf{A}$ with $0 < \gamma \leq 1$. The *modified regularized Laplacian kernel* $\mathbf{K}_{\mathrm{RL}}$ (Ito et al., 2005) is thus computed as

$$\mathbf{K}_{\mathrm{RL}} = \sum_{k=0}^{\infty} \alpha^k (-\mathbf{L}_\gamma)^k = (\mathbf{I} + \alpha\mathbf{L}_\gamma)^{-1} \tag{13}$$

It was shown that, for $\alpha > 0$ and $0 < \gamma \leq 1$, if the series (13) converges, $\mathbf{K}_{\mathrm{RL}}$ is positive semidefinite and that it yields a measure intermediate between relatedness and importance (Ito et al., 2005). Since $\mathbf{K}_{\mathrm{RL}}$ is more general than $\mathbf{K}_{\mathrm{L}}$, it will be used in our experiments. Indeed, $\mathbf{K}_{\mathrm{RL}} = \mathbf{K}_{\mathrm{L}}$ when $\gamma = 1$.

4.5. The commute-time kernel

Consider a discrete Markov chain defining a random walk on graph G and let $s(t)$ be a random variable representing the state of the Markov chain at time step t. Thus, if the process is in state $i \in \{1, \ldots, n\}$ at time t, then $s(t) = i$. Let the probability of being in state i at time t be denoted as $x_i(t) = \mathrm{P}(s(t) = i)$ and define $\mathbf{P}$ as the transition-probability matrix with entries $p_{ij} = [\mathbf{P}]_{ij} = \mathrm{P}(s(t+1) = j | s(t) = i)$. Consider a Markov model whose transition probabilities are $p_{ij} = a_{ij}/a_{i\bullet}$ with $a_{i\bullet} = \sum_{j=1}^{n} a_{ij}$. In other words, to any state or node $s(t) = i$, a probability of jumping to an adjacent node $s(t+1) = j$ is associated that is proportional to the weight $a_{ij} \geqslant 0$ of the edge connecting i and

[1] Strictly speaking, in the context of semisupervised classification, this least square should be defined on the training set only (labeled samples). However, initially, the framework of Zhou et al. (2003) considers that, when facing unlabeled samples, the function should be close to 0 as well. In other words, unlabeled samples are considered as not belonging to the class of interest. See, e.g., Bengio, Delalleau, and Le Roux (2006) for the model taking unlabeled samples into account.

j, and then normalized (this corresponds to a standard random-walk model on a graph). Transition probabilities depend only on the current state and not on past ones (first-order Markov chain). Since the graph is undirected and connected, the Markov chain is reversible and irreducible (every state can be reached from every state). In matrix form, the evolution of the Markov chain is characterized by $\mathbf{x}(t+1) = \mathbf{P}^{\mathrm{T}}\mathbf{x}(t)$, which provides the state probability distribution $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^{\mathrm{T}}$ at time t given the initial probability density $\mathbf{x}(0) = \mathbf{x}_0$ at $t = 0$. After t steps, the probability distribution is given by $\mathbf{x}(t) = (\mathbf{P}^{\mathrm{T}})^t\mathbf{x}_0$.

The commute-time kernel (Fouss et al., 2007; Saerens et al., 2004) takes its name from the average commute time $n(i, j)$, which is the average number of steps that a random walker, starting in node $i \neq j$, takes before entering node j for the first time and then going back to i. The average commute time (Fouss et al., 2007; Klein & Randic, 1993; Saerens et al., 2004) can be computed as

$$n(i, j) = V_G\,(\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}}\mathbf{L}^{+}(\mathbf{e}_i - \mathbf{e}_j) \tag{14}$$

where every node i of the graph is represented by a basis vector $\mathbf{e}_i$ in the Euclidean space $\mathbb{R}^n$ and V_G is the volume of the graph. $\mathbf{L}^{+}$ is the Moore–Penrose pseudoinverse of the Laplacian matrix of the graph and it is positive semidefinite. Thus, Eq. (14) is a Mahalanobis distance between the nodes of the graph. It is referred to as the "commute-time distance" or the "resistance distance" because of a close analogy with the effective resistance in electrical networks (Chandra et al., 1989; Fouss et al., 2007; Klein & Randic, 1993). However, it has been shown that this resistance distance is in fact meaningless when dealing with large graphs (von Luxburg, Radl, & Hein, 2010) so that it should be used with caution.

It can be shown that the elements of $\mathbf{L}^{+}$ are inner products of the node vectors in the Euclidean space where these node vectors are exactly separated by commute-time distances (Brand, 2005; Saerens et al., 2004). In other words, the elements of $\mathbf{L}^{+}$ can be viewed as similarity measures between nodes. Hence the *commute-time kernel* $\mathbf{K}_{\mathrm{CT}}$ (Fouss et al., 2007; Saerens et al., 2004) is defined as

$$\mathbf{K}_{\mathrm{CT}} = \mathbf{L}^{+} \tag{15}$$

with no parameter tuning necessary. The dominant eigenvector of $\mathbf{K}_{\mathrm{CT}}$ is in fact the Fiedler vector (Fouss et al., 2007) known to convey some important information concerning the structure of the graph (Chung, 1997; Fiedler, 1975a, 1975b, 1989). A slight modification of this kernel, the sigmoid commute-time kernel, worked remarkably well in a node-clustering task (community detection, Yen et al., 2007; Yen, Fouss et al., 2008).

Because of some issues with the `pinv` function in Matlab when dealing with large sparse matrices, the following formula computing the pseudoinverse of the Laplacian matrix (see, e.g., Rao & Mitra, 1971, Chapter 10), was used in all our experiments:

$$\mathbf{L}^{+} = \left(\mathbf{L} - \mathbf{e}\mathbf{e}^{\mathrm{T}}/n\right)^{-1} + \mathbf{e}\mathbf{e}^{\mathrm{T}}/n. \tag{16}$$

This formula exploits the fact that $\mathbf{L}$ is centered and thus $\mathbf{e}$, a column vector full of 1's, spans its null space.

There is of course a close relationship between the commute-time kernel and the regularized Laplacian kernel. Indeed, if $\mathbf{L}$ has eigenvalues λ_i (in decreasing order), $(\mathbf{L} + \alpha^{-1}\mathbf{I})$ has corresponding eigenvalues $(\lambda_i + \alpha^{-1})$ and both matrices share the same eigenvectors $\mathbf{u}_i$. Remember that $\mathbf{L}$ and its Moore–Penrose pseudoinverse $\mathbf{L}^{+}$ have the same set of eigenvectors and inverse eigenvalues (zero eigenvalues are equal to zero in both matrices). The Laplacian matrix of a connected graph has rank $n - 1$ and therefore it has exactly one eigenvalue equal to 0 (Chung, 1997). Thus a spectral decomposition of the corresponding kernel yields $\mathbf{L}^{+} = \sum_{i=1}^{n-1}\lambda_i^{-1}\mathbf{u}_i\mathbf{u}_i^{\mathrm{T}}$ (by definition of the Moore–Penrose pseudoinverse) and $(\mathbf{I} + \alpha\mathbf{L})^{-1} = \alpha^{-1}\sum_{i=1}^{n}(\lambda_i + \alpha^{-1})^{-1}\mathbf{u}_i\mathbf{u}_i^{\mathrm{T}}$. Since the normalized eigenvector of $\mathbf{L}$ corresponding to eigenvalue $\lambda_n = 0$ is $\mathbf{u}_n = \mathbf{e}/\sqrt{n}$, the nth term for the commute-time kernel is simply 0, while it is $\mathbf{e}\mathbf{e}^{\mathrm{T}}/n$ for the regularized Laplacian kernel. Therefore, this last term simply adds a constant value $(1/n)$ to all the elements of the matrix and, if α is large, $\lambda_i + \alpha^{-1}$ will be close to λ_i, so that the two kernels essentially differ by a multiplication factor and a constant value added to all the elements of the matrix.

4.6. The Markov diffusion kernel

This kernel introduced in Fouss et al. (2006) is based on a diffusion distance defined in Latapy and Pons (2005), Nadler et al. (2005, 2006) and Pons and Latapy (2006) between nodes of a graph in a diffusion model. We adapted their definition of diffusion distance to periodic Markov chains (as nonperiodic Markov chains are unrealistic for collaborative recommendation) to define a valid kernel on a graph (Fouss et al., 2006; Yen et al., 2011). It was shown in Nadler et al. (2005, 2006) that the low-dimension representation of the data by the first few eigenvectors of the corresponding transition-probability matrix is optimal under a given least-square error criterion involving the diffusion distance. The intuition behind the diffusion distance is the following: two nodes are considered similar when they diffuse in a similar way through the graph, and therefore when they "influence" the other nodes in a similar manner. An application of the diffusion distance to dimensionality reduction and graph visualization was described in Lafon and Lee (2006).

More precisely, the average visiting rate $\bar{x}_{ik}(t)$ in state k after t steps (the probability of finding the random walker in state k during the first t steps) for a process that started in state i at time $t = 0$ is computed as follows:

$$\bar{x}_{ik}(t) = \frac{1}{t}\sum_{\tau=1}^{t}\mathrm{P}(s(\tau) = k|s(0) = i). \tag{17}$$

From this quantity, the diffusion distance at time t between node i and node j is defined as follows

$$d_{ij}(t) = \sum_{k=1}^{n}\left(\bar{x}_{ik}(t) - \bar{x}_{jk}(t)\right)^2 \tag{18}$$

which corresponds to the sum of the squared differences between the average visiting rates after t transitions when starting from node i and node j at time $t = 0$. This is a natural definition which quantifies the dissimilarity between two nodes based on the evolution of the probability distribution. Thus, if two nodes influence (diffuse through) the graph in exactly the same way, the distance between them is zero. Of course, when $i = j$, $d_{ij}(t) = 0$.

This diffusion distance was proposed by Nadler et al. (2005) and by Latapy and Pons (2005) in the context of diffusion processes. Actually, their original definition, namely $d_{ij}(t) = \sum_{k=1}^{n}\left(x_{ik}(t) - x_{jk}(t)\right)^2$, involves $x_{ik}(t)$, the probability of finding the random walker in state k at time t, when starting from node i at time 0 (i.e., $\mathbf{x}(0) = \mathbf{e}_i$) instead of $\bar{x}_{ik}(t)$ as in Eq. (18). Their definition does not work with periodic Markov chains that are relevant to collaborative recommendation (with a bipartite graph). The definition in Latapy and Pons (2005), Nadler et al. (2005, 2006) and Pons and Latapy (2006) also adds a weighting factor proportional to the inverse of the stationary distribution of the Markov chain, therefore putting more weight on low-density nodes. We do not include that factor here because experiments showed that it does not improve performance for the applications studied in Sections 5 and 6. Still another variant of this diffusion kernel was recently introduced in Yen et al. (2011).

We first compute $\bar{x}_{ik}(t)$ from Eq. (17):

$$\bar{x}_{ik}(t) = \frac{1}{t}\sum_{\tau=1}^{t} \mathrm{P}(s(\tau)=k|s(0)=i) \tag{19}$$

$$= \frac{1}{t}\sum_{\tau=1}^{t} \mathbf{e}_k^{\mathrm{T}}(\mathbf{P}^{\mathrm{T}})^{\tau}\mathbf{e}_i = \mathbf{e}_k^{\mathrm{T}}\left[\frac{1}{t}\sum_{\tau=1}^{t}\mathbf{P}^{\tau}\right]^{\mathrm{T}}\mathbf{e}_i \tag{20}$$

since $\mathrm{P}(s(t)=k|s(0)=i) = \mathbf{e}_k^{\mathrm{T}}\mathbf{x}(t) = \mathbf{e}_k^{\mathrm{T}}(\mathbf{P}^{\mathrm{T}})^t\mathbf{e}_i$ when $\mathbf{x}_0 = \mathbf{e}_i$. By defining $\mathbf{Z}(t) = \frac{1}{t}\sum_{\tau=1}^{t}\mathbf{P}^{\tau}$, we obtain

$$\bar{x}_{ik}(t) = \mathbf{e}_k^{\mathrm{T}}\mathbf{Z}^{\mathrm{T}}(t)\mathbf{e}_i. \tag{21}$$

We now turn to the evaluation of the Markov diffusion distance (Eq. (18)):

$$d_{ij}(t) = \sum_{k=1}^{n}\left(\bar{x}_{ik}(t)-\bar{x}_{jk}(t)\right)^2 = \sum_{k=1}^{n}\left(\mathbf{e}_k^{\mathrm{T}}\mathbf{Z}^{\mathrm{T}}(t)\mathbf{e}_i - \mathbf{e}_k^{\mathrm{T}}\mathbf{Z}^{\mathrm{T}}(t)\mathbf{e}_j\right)^2 \tag{22}$$

$$= \|\mathbf{Z}^{\mathrm{T}}(t)(\mathbf{e}_i-\mathbf{e}_j)\|^2 = (\mathbf{e}_i-\mathbf{e}_j)^{\mathrm{T}}\mathbf{Z}(t)\mathbf{Z}^{\mathrm{T}}(t)(\mathbf{e}_i-\mathbf{e}_j) \tag{23}$$

which has exactly the same form as Eq. (2). We immediately deduce the form of the *Markov diffusion* ($\mathbf{K}_{\mathrm{MD}}$) *kernel*

$$\mathbf{K}_{\mathrm{MD}}(t) = \mathbf{Z}(t)\mathbf{Z}^{\mathrm{T}}(t) \quad \text{with } \mathbf{Z}(t) = \frac{1}{t}\sum_{\tau=1}^{t}\mathbf{P}^{\tau} \tag{24}$$

Notice that in order to evaluate $\mathbf{Z}(t)$, we use a trick similar to the one used in PageRank (Langville & Meyer, 2005): a dummy absorbing state linked to all the states of the Markov chain is created with a very small probability of jumping to this state. This amounts to subtracting some small quantity from every element of $\mathbf{P}$, with the result that matrix $\mathbf{P}$ is now substochastic (if this was not the case, $(\mathbf{I}-\mathbf{P})$ would not be invertible) and that $\mathbf{Z}(t)$ admits the following analytical form:

$$\mathbf{Z}(t) = \frac{1}{t}(\mathbf{I}-\mathbf{P})^{-1}(\mathbf{I}-\mathbf{P}^t)\mathbf{P}. \tag{25}$$

4.7. The relative-entropy diffusion matrix

Instead comparing two probability distributions with a squared distance as in Eq. (18), the symmetric relative entropy is more appropriate (see, e.g., Cover & Thomas, 2006; Kapur & Kesavan, 1992). As it is well-known, the symmetric relative entropy (also called the Kullback–Leibler divergence) between two probability distributions $\mathbf{p}$ and $\mathbf{q}$ is defined as (Cover & Thomas, 2006; Kapur & Kesavan, 1992):

$$D(\mathbf{p}:\mathbf{q}) = \sum_{k=1}^{n}\left[p_k\log\frac{p_k}{q_k} + q_k\log\frac{q_k}{p_k}\right]. \tag{26}$$

Letting $p_k = \bar{x}_{ik}(t)$ and $q_k = \bar{x}_{jk}(t)$ yields

$$d_{ij}(t) = \sum_{k=1}^{n}\left[\bar{x}_{ik}(t)\log\frac{\bar{x}_{ik}(t)}{\bar{x}_{jk}(t)} + \bar{x}_{jk}(t)\log\frac{\bar{x}_{jk}(t)}{\bar{x}_{ik}(t)}\right]. \tag{27}$$

From Eq. (21), $\bar{x}_{ik}(t) = \mathbf{e}_k^{\mathrm{T}}\mathbf{Z}^{\mathrm{T}}(t)\mathbf{e}_i = \mathbf{e}_i^{\mathrm{T}}\mathbf{Z}(t)\mathbf{e}_k$, so that Eq. (27) becomes

$$\begin{aligned} d_{ij}(t) &= \sum_{k=1}^{n}[\mathbf{e}_i^{\mathrm{T}}\mathbf{Z}(t)\mathbf{e}_k\log(\mathbf{e}_i^{\mathrm{T}}\mathbf{Z}(t)\mathbf{e}_k) - \mathbf{e}_i^{\mathrm{T}}\mathbf{Z}(t)\mathbf{e}_k\log(\mathbf{e}_j^{\mathrm{T}}\mathbf{Z}(t)\mathbf{e}_k) \\ &\quad + \mathbf{e}_j^{\mathrm{T}}\mathbf{Z}(t)\mathbf{e}_k\log(\mathbf{e}_j^{\mathrm{T}}\mathbf{Z}(t)\mathbf{e}_k) - \mathbf{e}_j^{\mathrm{T}}\mathbf{Z}(t)\mathbf{e}_k\log(\mathbf{e}_i^{\mathrm{T}}\mathbf{Z}(t)\mathbf{e}_k)] \\ &= \sum_{k=1}^{n}[\mathbf{e}_i^{\mathrm{T}}\mathbf{Z}(t)\mathbf{e}_k\mathbf{e}_k^{\mathrm{T}}\log(\mathbf{Z}^{\mathrm{T}}(t))\mathbf{e}_i - \mathbf{e}_i^{\mathrm{T}}\mathbf{Z}(t)\mathbf{e}_k\mathbf{e}_k^{\mathrm{T}}\log(\mathbf{Z}^{\mathrm{T}}(t))\mathbf{e}_j \\ &\quad + \mathbf{e}_j^{\mathrm{T}}\mathbf{Z}(t)\mathbf{e}_k\mathbf{e}_k^{\mathrm{T}}\log(\mathbf{Z}^{\mathrm{T}}(t))\mathbf{e}_j - \mathbf{e}_j^{\mathrm{T}}\mathbf{Z}(t)\mathbf{e}_k\mathbf{e}_k^{\mathrm{T}}\log(\mathbf{Z}^{\mathrm{T}}(t))\mathbf{e}_i] \\ &= \mathbf{e}_i^{\mathrm{T}}\mathbf{Z}(t)\log(\mathbf{Z}^{\mathrm{T}}(t))\mathbf{e}_i - \mathbf{e}_i^{\mathrm{T}}\mathbf{Z}(t)\log(\mathbf{Z}^{\mathrm{T}}(t))\mathbf{e}_j \\ &\quad + \mathbf{e}_j^{\mathrm{T}}\mathbf{Z}(t)\log(\mathbf{Z}^{\mathrm{T}}(t))\mathbf{e}_j - \mathbf{e}_j^{\mathrm{T}}\mathbf{Z}(t)\log(\mathbf{Z}^{\mathrm{T}}(t))\mathbf{e}_i \\ &= (\mathbf{e}_i-\mathbf{e}_j)^{\mathrm{T}}\left[\mathbf{Z}(t)\log(\mathbf{Z}^{\mathrm{T}}(t))\right](\mathbf{e}_i-\mathbf{e}_j) \\ &= (\mathbf{e}_i-\mathbf{e}_j)^{\mathrm{T}}\frac{\left[\mathbf{Z}(t)\log(\mathbf{Z}^{\mathrm{T}}(t)) + \log(\mathbf{Z}(t))\mathbf{Z}^{\mathrm{T}}(t)\right]}{2}(\mathbf{e}_i-\mathbf{e}_j) \end{aligned} \tag{28}$$

where we used $\sum_{k=1}^{n}\mathbf{e}_k\mathbf{e}_k^{\mathrm{T}} = \mathbf{I}$ (the identity matrix) and the element-wise logarithm is taken on each element of the matrix. The last line follows from the fact that any matrix $\mathbf{M}$ can be expressed as the sum of its symmetric part $(\mathbf{M}+\mathbf{M}^{\mathrm{T}})/2$ and its skew-symmetric part $(\mathbf{M}-\mathbf{M}^{\mathrm{T}})/2$ and that the distance cancels out the skew-symmetric part: $(\mathbf{e}_i-\mathbf{e}_j)^{\mathrm{T}}(\frac{\mathbf{M}-\mathbf{M}^{\mathrm{T}}}{2})(\mathbf{e}_i-\mathbf{e}_j) = 0$. Eq. (28) takes the form of a Mahalanobis distance, with the important difference that $[\mathbf{Z}(t)\log(\mathbf{Z}^{\mathrm{T}}(t)) + \log(\mathbf{Z}(t))\mathbf{Z}^{\mathrm{T}}(t)]$ is not positive semidefinite in general.

This leads to the *relative-entropy diffusion* ($\mathbf{K}_{\mathrm{RED}}$) *matrix*:

$$\mathbf{K}_{\mathrm{RED}}(t) = \mathbf{Z}(t)\log(\mathbf{Z}^{\mathrm{T}}(t)) + \log(\mathbf{Z}(t))\mathbf{Z}^{\mathrm{T}}(t)$$

$$\text{with } \mathbf{Z}(t) = \frac{1}{t}\sum_{\tau=1}^{t}\mathbf{P}^{\tau} \tag{29}$$

This similarity matrix is not positive semidefinite and therefore is not a kernel.

4.8. The random-walk-with-restart similarity matrix

Pan et al. (2006) (see also Tong et al., 2006; Tong et al., 2008) introduced a similarity matrix inspired by the well-known PageRank algorithm (Brin & Page, 1998; Page et al., 1998). This model has been applied to various interesting applications, including center-piece subgraph discovery and content-based image retrieval (Pan et al., 2006; Tong et al., 2006, 2008). The same idea was also introduced by Pucci et al. (2006) and Gori and Pucci (2006, 2007) in the context of collaborative recommendation.

Like the diffusion kernel, the model considers a random walker jumping from some node i to some neighbor node j with a probability proportional to the edge weight $p_{ij} = \mathrm{P}(s(t+1)=j|s(t)=i) = a_{ij}/a_{i\bullet}$. In addition, at each step of the random walk, the random walker has some probability $(1-\alpha)$ of returning to node i instead of continuing to neighbor nodes. In other words, the probability distribution of finding the random walker on each node at time t is provided by

$$\begin{cases}\mathbf{x}(0) = \mathbf{e}_i \\ \mathbf{x}(t+1) = \alpha\mathbf{P}^{\mathrm{T}}\mathbf{x}(t) + (1-\alpha)\mathbf{e}_i\end{cases} \tag{30}$$

where $\alpha \in]0,1[$. Thus, random walkers have the opportunity to restart with probability $(1-\alpha)$ from node i at each time step, instead of continuing their random walk. Considering the steady-state solution $\mathbf{x}(t+1) = \mathbf{x}(t) = \mathbf{x}_\infty$ and extracting the probability distribution $\mathbf{x}$ of finding the random walker on each node when starting from node i yields

$$\mathbf{x}_\infty = (1-\alpha)\,(\mathbf{I}-\alpha\mathbf{P}^{\mathrm{T}})^{-1}\mathbf{e}_i \tag{31}$$

which corresponds, up to a scaling factor, to column i of the matrix $(\mathbf{I}-\alpha\mathbf{P}^{\mathrm{T}})^{-1}$. Notice that since the matrix $\alpha\mathbf{P}$ is substochastic, the inverse of $(\mathbf{I}-\alpha\mathbf{P})$ exists if the Markov chain is regular. Vector $\mathbf{x}_\infty$ can be viewed as containing a similarity between node i and the other nodes of the graph.

Since $\mathbf{x}_\infty$ is column i of matrix $(\mathbf{I}-\alpha\mathbf{P}^{\mathrm{T}})^{-1}$, the *random-walk-with-restart* ($\mathbf{K}_{\mathrm{RWR}}$) *matrix* (Gori & Pucci, 2006; Pan et al., 2006; Pucci et al., 2006; Tong et al., 2006, 2008) can be defined as the

Table 1
The nine kernels with the value range tested for their parameters.

Name	Abbr.	Equation	Par.	Tested values
Exponential diffusion kernel	K_{ED}	(3)	α	$10^{-6}, 10^{-5}, \ldots, 10^{2}$
Laplacian exponential diffusion kernel	K_{LED}	(8)	α	$10^{-6}, 10^{-5}, \ldots, 10^{2}$
Von Neumann diffusion kernel	K_{VND}	(9)	α	$(10^{-6}, 10^{-5}, \ldots, 10^{-1}, 0.99) \times \|\mathbf{A}\|_2^{-1}$
Regularized Laplacian kernel	K_{RL}	(13)	γ	$10^{-6}, 10^{-5}, \ldots, 1$ with $\alpha = 1$
			α	$10^{-6}, 10^{-5}, \ldots, 10^{5}$
Commute-time kernel	K_{CT}	(16)	/	/
Markov diffusion kernel	K_{MD}	(24) and (25)	t	1, 2, . . . , 10, 50, 100
Relative-entropy diffusion matrix	K_{RED}	(29)	t	1, 2, . . . , 10, 50, 100
Random-walk-with-restart similarity matrix	K_{RWR}	(32)	α	$10^{-6}, 10^{-5}, \ldots, 0.99$
Regularized commute-time kernel	K_{RCT}	(34)	α	$10^{-6}, 10^{-5}, \ldots, 0.99$

matrix whose i'th row contains the similarities to node i (the matrix is transposed):

$$\mathbf{K}_{RWR} = (\mathbf{I} - \alpha\mathbf{P})^{-1} \tag{32}$$

with the same form as the fundamental matrix $(\mathbf{I} - \mathbf{P})^{-1}$ of the Markov chain. $\mathbf{K}_{RWR}$ can be rewritten as

$$\mathbf{K}_{RWR} = (\mathbf{D}^{-1}(\mathbf{D} - \alpha\mathbf{A}))^{-1} = (\mathbf{D} - \alpha\mathbf{A})^{-1}\mathbf{D} \tag{33}$$

since $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$. Notice that $\mathbf{K}_{RWR}$ is not a kernel since it is not symmetric.

4.9. The regularized commute-time kernel

Remember that the Laplacian matrix, whose pseudoinverse is called the commute-time kernel in Section 4.5, is not invertible. Instead of taking the pseudoinverse of the matrix, a simple regularization framework can be applied as in Section 4.4. One such regularization leads to what we call the *regularized commute-time* ($\mathbf{K}_{RCT}$) *kernel*:

$$\mathbf{K}_{RCT} = (\mathbf{D} - \alpha\mathbf{A})^{-1} \tag{34}$$

with $\alpha \in]0, 1[$. Since $\mathbf{K}_{RCT}$ is the matrix inverse of the sum $((1 - \alpha)\mathbf{D} + \alpha\mathbf{L})$ of a positive definite matrix and a positive semidefinite matrix, it is positive definite and it is a valid kernel. This kernel is a slight extension of the modified Laplacian matrix $(\gamma\mathbf{D} - \mathbf{A})$, $\gamma \in]0, 1[$ (Ito et al., 2005), equivalent to changing the range of γ to $\gamma > 1$.

Another justification of the interest of this kernel is as follows (Mantrach et al., 2011). Consider the following random-walk model starting at node i

$$\begin{cases} \mathbf{x}(0) = \mathbf{e}_i \\ \mathbf{x}(t+1) = \mathbf{P}^{\mathrm{T}}\mathbf{x}(t). \end{cases} \tag{35}$$

The column vector $\mathbf{x}(t)$ contains the probability distribution of finding the random walker in each state of the finite Markov chain. Thus, the random walker starts at node i and diffuses through the network. Let us define the similarity vector between node i and other nodes of the network by:

$$\mathbf{sim}_i = \sum_{\tau=0}^{\infty} \alpha^{\tau}\mathbf{D}^{-1}\mathbf{x}(\tau) \tag{36}$$

with $\alpha \in]0, 1[$. The weighting factor $\mathbf{D}^{-1}$ compensates for the fact that $\mathbf{x}(t)$ converges to the stationary distribution, which is proportional to the diagonal elements of $\mathbf{D}$ with undirected graphs (Ross, 1996). A similarity measure based on $\mathbf{x}(t)$ only (without weighting factor) would therefore favor the nodes with a high outdegree $[\mathbf{D}]_{ii}$. Eq. (36) thus accumulates, with a damping factor α (later visits are less important than earlier ones), the probability of visiting each node when starting from node i. From another point of view, it also corresponds to an "evaporating" random walk where the random walker has a $(1-\alpha)$ probability of disappearing at each step. Eq. (36) cumulates the presence rate in each node during this evaporating random walk, when starting from node i.

We easily find

$$\mathbf{sim}_i = \sum_{\tau=0}^{\infty} \alpha^{\tau}\mathbf{D}^{-1}\mathbf{x}(\tau) = \mathbf{D}^{-1}\sum_{\tau=0}^{\infty} \alpha^{\tau}\left(\mathbf{P}^{\mathrm{T}}\right)^{\tau}\mathbf{e}_i \tag{37}$$

$$= \mathbf{D}^{-1}(\mathbf{I} - \alpha\mathbf{P}^{\mathrm{T}})^{-1}\mathbf{e}_i = \mathbf{D}^{-1}\left((\mathbf{D} - \alpha\mathbf{A})\mathbf{D}^{-1}\right)^{-1}\mathbf{e}_i \tag{38}$$

$$= (\mathbf{D} - \alpha\mathbf{A})^{-1}\mathbf{e}_i = \mathbf{col}_i(\mathbf{K}_{RCT}) \tag{39}$$

where we used $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$ and the fact that $\mathbf{D}$ is diagonal and $\mathbf{A}$ is symmetric (undirected graph).

Thus, the (i, j) element of $\mathbf{K}_{RCT}$ can be interpreted as the discounted cumulated probability of visiting node j when starting from node i. The regularized commute-time kernel (Eq. (34)) differs from the random-walk-with-restart similarity matrix (Eq. (33)) only in the fact that the latter post-multiplies the former by $\mathbf{D}$. The regularized commute-time kernel matrix was already presented in Zhou et al. (2003) in the context of semisupervised classification, through a regularization framework along the same lines as in Eq. (12).

4.10. Parameter tuning and efficiency

Table 1 lists the kernel-based algorithms. The "Equation" column refers to the number in this paper of the equation used in the experiments to compute each kernel (those equations are bowed in the text). Most algorithms need a value for some parameters (given in the "Par." column). Those values (see the "Tested values" column) were tuned in the experiments following a nested cross-validation procedure detailed in Section 5.3 for recommendation and in Section 6.2 for classification.

Efficiently inverting matrices can have a significant influence on performance for large graphs. Also, inverse matrices are often dense and may not fit in main memory. Cholesky factorizations address those issues (see, e.g., Fouss et al., 2007). Incomplete (lower-rank) Cholesky matrix factorizations (see, e.g., Fine & Scheinberg, 2001) can be efficiently computed and remain sparse if the original matrix is sparse. From the factorization, every column of the matrix inverse can be obtained by simple backsubstitution. Therefore, a recommendation to a specific user or the classification of an unlabeled node of interest by kernel alignment can be obtained quite efficiently from the factorization. Notice also that columns of the kernel matrix can be obtained by solving a linear system of equations and that this does not require the computation of the inverse. However, kernel-based methods are not suited for large-scale graphs (millions of nodes) since they require a matrix inverse or factorization.

5. Testing the kernels on collaborative recommendation

This section reports on experimental work aiming at (i) comparing the accuracy of the various kernels and (ii) comparing their performance with that of the latent-class model (Cheung et al.,

2004; Hofmann, 2004), a well-established and effective recommendation method. Recommender systems provide their users with ranked lists of items that they will appreciate, based on past preferences, history of choices, and demographic information. For example, imagine a simple movie database with three sets of elements *people*, *movie*, and *movie_category*, and two relationships *has_watched*, between *people* and *movie*, and *belongs_to*, between *movie* and *movie_category*. Computing similarities between people and movies allows to recommend lists of movies for people to watch. The idea originates (see the survey in Adomavicius & Tuzhilin, 2005) mainly from research on machine learning, information retrieval (Salton, 1989), cognitive science (Rich, 1979), forecasting theories (Armstrong, 2001), marketing (Lilien, Smith, & Moorthy, 1992), and management (Murthi & Sarkar, 2003). Recommender systems emerged as a research area in the mid-1990s, with the first papers on collaborative filtering (Hill, Stead, Rosenstein, & Furnas, 1995; Resnick, Neophytos, Mitesh, Bergstrom, & Riedl, 1994; Shardanand & Maes, 1995).

The nine algorithms presented in Section 4 were tested on a collaborative-recommendation task with two concrete datasets. A weighted undirected graph is associated with a database in the following way: database elements correspond to nodes of the graph and database links correspond to edges. In the movie database, *people* and *movies* correspond to nodes of the (bipartite) graph, and *has_watched* links appear as edges connecting the corresponding nodes. In the corresponding user-item matrix, the weight of an edge is set to 1 if there is a link between the corresponding items and to 0 otherwise. Recommending items to a user amounts to predicting missing user-item links in the graph. Matrices for typical recommendation applications are sparse. Therefore the assumption that the corresponding graph is connected may not apply.

5.1. Datasets

With the *MovieLens (ML)* web-based recommender system[2], visitors of the web site are invited to rate movies that they watched and to ask for recommendations about movies that they should watch, given those that they and other users have already watched. We selected a sample of the ML database as suggested in Sarwar, Karypis, Konstan, and Riedl (2002). Enough users (i.e., 943 of them) were randomly selected to obtain 100,000 ratings, retaining only users who had rated at least 20 movies on a total of 1682 movies. The resulting user-movie matrix contains about 6.3% of "1" values and 93.7% of "0" values.

The *Book-Crossing (BC)* used in this paper (collected in 2004[3] from the Book-Crossing community[4]) contains 278,858 users providing 1149,780 ratings about 271,379 books. We selected from it a sample of 2222 books, 1028 users, and 109,374 ratings, retaining only users who had rated at least 40 books and books that had been rated by at least 20 users. The user-book matrix contains about 4.8% of "1" values and 95.2% of "0" values, which makes it sparser than the matrix created from the MovieLens database.

Our experiments discarded the numerical ratings provided in the datasets (that is, we retained only the fact that a person did or did not rate an item). Indeed, few numerical ratings are provided for the BC dataset while most ratings in the ML dataset are $4^{'}$s or $5^{'}$s on a 1-to-5 preference scale, which reduces their significance for discrimination. The graphs computed from the preprocessed datasets are connected. Isolated users or items appearing during the cross-validation splits (see Section 5.3) were automatically removed in order to work with connected graphs.

[2] http://www.movielens.umn.edu.

[3] http://www.informatik.uni-freiburg.de/~cziegler/BX/.

[4] http://www.bookcrossing.com.

5.2. Direct and indirect recommendation

We used three methods to test the kernels. We call them, respectively, the "direct method", the "user-based indirect method", and the "item-based indirect method". Both indirect methods are based on a nearest-neighbor technique that requires a measure of closeness or similarity. Important considerations involved in choosing a similarity measure (Johnson & Wichern, 2002) include the nature of variables (discrete, continuous, or binary), the scale of measurements (nominal, ordinal, interval, or ratio), and specific knowledge about the subject matter. The nearest-neighbor scoring algorithm is one of the simplest and oldest methods for general classification tasks (Duda, Hart, & Stork, 2001). It classifies an unknown pattern as belonging to the class of the nearest example in the dataset as measured by a similarity metric. K-nearest-neighbor techniques generalize the idea to taking into account k nearest examples to decide on class membership.

The *direct method* computes similarities $\mathrm{sim}(p_0, i)$ with each scoring algorithm between a given user p_0 requesting recommendations and items i. As an answer, a ranked list of items is suggested to user p_0 from among those that p_0 did not rate.

The *user-based indirect method* suggests items by applying a two-step process:

(1) identify users with rating patterns most similar to those of user p_0 requesting recommendations;
(2) use the ratings from those users to compute recommendations.

More precisely, the user-based indirect method uses the scoring algorithms like the direct method to compute similarities $\mathrm{sim}(p_0, p)$ between user p_0 and users p. Then, it computes, from the k nearest-neighbor users of p_0, the preference or predicted value for each item. The preference for item i_0 and user p_0 is the sum, weighted by $\mathrm{sim}(p_0, p)$, of the values of the link weight (0 or 1) of item i_0 for the k users p closest to p_0 (k nearest neighbors):

$$\mathrm{pred}(p_0, i_0) = \frac{\sum_{p=1}^{k} \mathrm{sim}(p_0, p) w_{pi_0}}{\sum_{p=1}^{k} \mathrm{sim}(p_0, p)} \tag{40}$$

where w_{pi_0} is "1" if user p rated item i_0 and "0" otherwise. The higher the preference $\mathrm{pred}(p_0, i_0)$, the stronger the recommendation that p_0 will like i_0.

The *item-based indirect method* works as follows:

(1) first, for each item, determine a set of its nearest neighbor items;
(2) use these neighborhoods and the ratings of the requesting user p_0 to compute recommendations.

This method corresponds to the methodology proposed by the SUGGEST approach (Karypis, 2001). It first applies scoring algorithms to compute similarities $\mathrm{sim}(i_0, i)$ between a given item i_0 and other items i. Then, for user p_0 and item i_0, it computes, from the k nearest items of i_0 (k nearest neighbors), the preference or predicted value of i_0 for p_0 as a sum, weighted by $\mathrm{sim}(i_0, i)$, of the values of the link weight (0 or 1) of p_0 and items i:

$$\mathrm{pred}(p_0, i_0) = \frac{\sum_{i=1}^{k} \mathrm{sim}(i_0, i) w_{p_0 i}}{\sum_{i=1}^{k} \mathrm{sim}(i_0, i)} \tag{41}$$

where $w_{p_0 i}$ is defined as above. Items are suggested to user p_0 in decreasing order of predicted values, among those not rated by p_0.

Choosing a value for k. When applying an indirect method to compute a ranked list of items to be suggested to a user, a value

for the number k of neighbors must be chosen. For each scoring algorithm and both indirect methods, we systematically varied k in the range (10, 20, . . . , 100). We decided to limit the maximum number of neighbors to 100, a fairly large number. Since the neighbors are sorted by decreasing similarity, as explained earlier in this section, the weight given to the ith neighbor (a user or an item, depending on the method) corresponds to a similarity value (as computed by the scoring algorithm) that is higher than the weight given to the $(i + 1)$th neighbor. In other words, the farther the neighbor, the smaller its contribution to the final predicted value (as computed by Eqs. (40) and (41)). Parameter k is tuned by using an internal cross-validation (see Section 5.3).

5.3. Performance evaluation

The performance of the various scoring algorithms is evaluated by applying a standard nested (or double) cross-validation based on links. This involves tuning parameters, namely, the parameter of the kernels (see Section 4), the number of neighbors (see Section 5.2), and the number of latent classes (see Section 5.4). Evaluation consists in determining the measure providing the best results (see Section 2.2) among (1) the original kernel, (2) the centered kernel matrix, (3) the cosine kernel matrix, (4) the centered cosine kernel matrix, and (5) the distance measure. This evaluation is performed with an internal nine-fold cross-validation and performance is averaged on an external ten-fold cross-validation.

More precisely, each dataset (containing links between users and items) was divided into ten subsets of links. Each subset (i.e., ten percent of the links) was in turn used for performance estimation in the external cross-validation (it is referred to as the "external test set"). The nine remaining subsets, merged to form the "external training set", were also used for the internal cross-validation to tune the parameters. On these nine subsets, an internal cross-validation was performed mirroring the external cross-validation, i.e., one of these nine subsets was used in turn as an "internal test set" while the remaining data subsets were merged, forming an "internal training set". We chose as values for the parameters those providing the best average performance on the internal nine-fold cross-validation and we used in turn those values for performance evaluation on the external ten-fold cross-validation.

For indirect methods, we used two ways to work with the original kernel matrix: (1) apply the transformation to the whole kernel matrix (for example, centering), extract the user–user matrix or the item–item matrix, and proceed from there, and (2) extract the user–user matrix or the item–item matrix from the original kernel matrix, apply the transformation (for example, centering) on this submatrix, and proceed from there. Both ways were tested in the nested cross-validation procedure.

Thus, each scoring algorithm provides, for each person, a set of proximity measures indicating preferences about items. In particular, for kernels, the kernel matrix is computed from the adjacency matrix of the bipartite graph containing person and item nodes, and containing a (i, j) link if user i rated item j. From that information and for each person, we extracted a ranked list of all the items that the person has not yet rated, according to the training set. In that list, the items closest to the person, in terms of the proximity measure, are considered the most relevant.

Accuracy. To evaluate the accuracy of the scoring algorithms, we compared their performance by applying a recall measure (Herlocker, Konstan, Terveen, & Riedl, 2004). For each user, the test set (external or internal) contains a set of items that the user actually rated and that were removed from the training set. Those items are part of the ranked list supplied by each scoring algorithm which contains all the items that the user did not rate, according to the training set. Ideally, these items should be ranked first since they were rated by the user. The recall accuracy measure (called "recall score" in the sequel) was used to compare the ranking provided by each algorithm (where all the items are ranked) and the "ideal ranking" (where each item from the test set is ranked before an item not belonging to the test set).

More precisely, the recall score is the average (on all users) of the percentage of items from the test set that appear among the top k of the ranked list, for some given k. A recall score of 100% indicates that the scoring algorithm always positions the items in the test set among the top k of the ranked list (assuming that the number of items in the test set is smaller than k). This measure should be as high as possible for good performance. We report the recall for the top 10 (Recall 10) and the top 20 (Recall 20) items (for a total of 1682 items in the ML dataset and 2222 books in the BC dataset).

We did not consider the case where new elements (i.e., new users or new items) are added into the dataset: only the case where "new" links (forming the test set) are added is considered. Iterative procedures could be developed to this aim for the various scoring algorithms.

5.4. Other (non kernel-based) scoring algorithms

The kernel-based algorithms were compared to four standard techniques: the baseline (Basic) algorithm, the standard binary algorithm (Bin), the cosine algorithm (Cos), and the latent-class algorithm (LCl). The latter three algorithms were chosen for comparison since they provide the best results among a number of standard methods (Fouss et al., 2007): Katz' method, shortest-path algorithm, maximum flow algorithm, and correspondence analysis. More detail about these methods as well as their results obtained on the MovieLens dataset are presented in Fouss et al. (2007). The last three algorithms depend on a parameter that is optimized in the inner loop of the nested cross-validation

The *baseline algorithm (Basic)* serves as a reference for comparison.

(1) In its direct form (see Section 5.2), Basic simply recommends items according to the number of times that they have been rated. Thus the ranking is the same for all users. This is similar to basing decisions only on a priori probabilities in supervised classification.
(2) In its user-based indirect form, the user–user similarity matrix to recommend items is simply the usual "bibliographic-coupling matrix" $\mathbf{W}\mathbf{W}^{\mathrm{T}}$ of bibliometry (Ito et al., 2005; Kessler, 1963) where $\mathbf{W}$ is the user-item adjacency matrix. Two users are similar if the corresponding element of $\mathbf{W}\mathbf{W}^{\mathrm{T}}$ is high, i.e., if there are many items that have been rated by both users. From these similarity measures ($\mathrm{sim}(p_i, p_j)$ = element i, j of $\mathbf{W}\mathbf{W}^{\mathrm{T}}$), a ranked list of items is then computed with Eq. (40).
(3) Symmetrically, in its item-based indirect form, the item–item similarity matrix to recommend items is the "co-citation matrix" $\mathbf{W}^{\mathrm{T}}\mathbf{W}$. Two items are similar if the corresponding element of $\mathbf{W}^{\mathrm{T}}\mathbf{W}$ is high, i.e., if there are many users who have rated both items. From these similarity measures ($\mathrm{sim}(i_k, i_l)$ = element k, l of $\mathbf{W}^{\mathrm{T}}\mathbf{W}$), a ranked list of items is then computed with Eq. (41).

Binary algorithm (Bin). Pairs of items are compared on the basis of the presence or absence of certain features (e.g., rating a particular movie or a particular book). The presence or absence of a feature can be described with a binary variable, with value 1 if the feature is present (i.e., person i rated item j) and value 0 if the feature is absent (i.e., person i did not rate item j). We now detail the procedure followed by applying Bin in the user-based indirect

method (see Section 5.2); applying Bin in the item-based indirect method is similar and does not require further explanations.

Each user i is characterized by a binary vector $\mathbf{v}_i$ (whose dimension is the total number of items) encoding the items rated by that user. The k nearest neighbors of person i are extracted by taking the k nearest $\mathbf{v}_j$ according to some binary similarity measure $\text{sim}(i, j) = s(\mathbf{v}_i, \mathbf{v}_j)$. In Fouss et al. (2007), we performed systematic comparisons between eight such measures s (listed in Johnson & Wichern, 2002, p. 674) for different values of k (= 10, 20,…,100). The best recall scores were obtained with the measure "ratio of 1–1 matches to mismatches with 0–0 matches excluded". Notice that Bin can only be used as an indirect method with bipartite graphs. The number of nearest neighbors is optimized in the inner loop of the nested cross-validation.

Cosine algorithm (Cos). The cosine coefficient between variables i and j measures the strength and the direction of a linear relationship between two variables. It is defined as $\text{sim}(i, j) = \cos(i, j) = (\mathbf{v}_i^T \mathbf{v}_j)/(\|\mathbf{v}_i\| \, \|\mathbf{v}_j\|)$ (Dunham, 2003; Sarwar, Karypis, Konstan, & Riedl, 2001). If variables are items (i.e., considering the item-based indirect method based on finding neighbors to items) represented as vectors in the user space (item i is characterized by vector $\mathbf{v}_i$), two items are considered more similar if the angle between their corresponding vectors in the user space is smaller. Like Bin, Cos is an indirect method that can only be used with bipartite graphs. The number of nearest neighbors is optimized in the inner loop of the nested cross-validation.

Latent-class algorithm (LCl). Latent-variable models provide an important tool for the analysis of multivariate data. Latent variables appear in many fields to which statistical methods are applicable, especially in social sciences (Bartholomew, 1999). Many patterns (represented by latent variables) that are of interest to social scientists cannot be observed directly. Examples are preferences, attitudes, behavioral intentions, and personality traits. Such parameters can only be estimated indirectly by means of observable indicators, such as questionnaires designed to extract information on, for example, a specific attitude or preferences of a person.

A latent-variable model, called "latent-class model", was developed in the context of collaborative filtering. It can be considered a sound and state-of-the-art recommendation technique (Cheung et al., 2004; Hofmann, 2004). Basically, it is a clustering model assuming that the preferences of a user are established through a latent variable. In this model, a latent-class variable $z \in \mathbb{Z} = \{z_1, \ldots, z_q\}$ (q being the number of latent classes or patterns) is associated with each observation (x, y) where x represents a user and y an item. The key assumption of the latent-class model is that x and y are independent, conditional on z. The standard procedure for maximum likelihood estimation in latent-class models is the Expectation–Maximization (EM) algorithm (introduced in Dempster, Laird, & Rubin, 1977). The number of latent classes is optimized in the inner loop of the nested cross-validation.

5.5. Results and discussion

Thus, for each person and each cross-validation split, we first selected the items that were not rated (according to the training set). Then, we ranked them with each scoring algorithm. Finally, we compared those rankings with the test set by using the recall score described in Section 5.3.

Results are summarized in Table 2 (for the MovieLens dataset) and Table 3 (for the Book-Crossing dataset), which show the recall, considering either the top 10 of the ranked list (Recall 10) or the top 20 of the ranked list (Recall 20) for the best configuration (in terms of the parameter of the scoring algorithm, the number of neighbors, and the applied derived measure) of each of the thirteen scoring algorithms (nine kernels on a graph, and Bin, Cos, Basic and LCl). The standard deviation of the results (STD) across the ten external cross-validation sets is also reported. The smaller the STD is, the more stable is the performance of the algorithm across different data samples. We used a paired t-test to determine whether there is a significant difference (with a p-value smaller than 0.05) between the results of the algorithms. The best results overall, for each measure of performance, are displayed in bold, based on the t-test.

5.5.1. MovieLens dataset

Table 2 shows that, with the direct method to rank movies for each user, the best Recall 10 is obtained by K_{MD} (20.73) while the best Recall 20 is obtained by LCl (31.55). For the user-based indirect method, the best results are obtained by K_{CT} and K_{MD} for both recalls (21.41, 32.09 and 21.40, 31.98, respectively—differences are not significant according to the paired t-tests). For the item-based indirect method, the best results are obtained by K_{RED} and K_{MD} for Recall 10 (19.85 and 19.72 respectively) and by Bin for Recall 20 (31.16), followed by K_{RED} (30.63), and K_{MD} (30.58).

Overall, for all three methods (see results in bold), the best scores are obtained by K_{CT} (21.41 for Recall 10 and 32.09 for Recall 20) and K_{MD} (21.40 for Recall 10 and 31.98 for Recall 20). The best recall scores for a standard scoring algorithm are obtained by Cos in the user-based indirect method (20.70 for Recall 10 and 31.18 for Recall 20), closely followed by Bin.

5.5.2. Book-Crossing dataset

Table 3 shows that, with the direct method, the best results are obtained by K_{RCT}, K_{RL}, and K_{CT} for both recalls (7.35, 7.36, and 7.26, respectively, for Recall 10, and 11.74, 11.58, and 11.73, respectively, for Recall 20). For the user-based indirect method, the best results are obtained by four kernels (i.e., K_{RL}, K_{RCT}, K_{CT}, and K_{MD}) for both recalls (7.09, 7.12, 7.07, and 7.09, respectively, for Recall 10; 11.23, 11.39, 11.39, and 11.31, respectively, for Recall 20). For the item-based indirect method, the best recall scores are obtained by K_{CT} (8.84 for Recall 10 and 13.58 for Recall 20) together with a standard scoring algorithm, Bin (8.79 for Recall 10 and 13.50 for Recall 20).

Overall (see results in bold), the best scores are obtained by the K_{CT} kernel and the Bin standard scoring algorithm, with the item-based indirect method.

5.5.3. General discussion

Table 4 shows, for the four settings (i.e., Recall 10 and Recall 20 for both datasets), the best score for each algorithm and the method applied to obtain it. Remember that Bin and Cos can only be used in indirect methods, and LCl only directly.

For the ML dataset, with the exception of K_{RED} (for both recalls) and Bin (for Recall 20), the best results (in 21 of 24 cases—ignoring LCl that can be used only by the direct method) are obtained with the user-based indirect method. For the BC dataset, with the exception of K_{ED}, K_{LED}, and K_{VND} (all three for both recalls), the best results (in 18 of 24 cases—ignoring again LCl) are obtained with the item-based indirect method.

This interesting observation suggests a possible link between the sparsity of the dataset and, more generally, the structure of its graph, and the decision about which indirect method is better suited (remember that 6.3% of the potential links are effective in the ML dataset for only 4.8% in the BC dataset). Still, admittedly, the evidence is weak.

We observe that K_{CT} produces the best scores in all four settings, virtually comparable with K_{MD} on both settings with the ML dataset and Bin on both settings with the BC dataset. The results also indicate that the best kernels on graphs are competitive with the standard algorithms. For example, K_{MD}, K_{RCT} obtain slightly

Table 2
Average performance, on the MovieLens dataset, for the thirteen scoring algorithms and the three methods (direct, user-based indirect, and item-based indirect) for using them.

	K_{ED}	K_{LED}	K_{VND}	K_{RL}	K_{RCT}	K_{CT}	K_{MD}	K_{RED}	K_{RWR}	Bin	Cos	Basic	LCI
Direct method													
Recall 10	18.52	18.06	15.05	18.94	19.75	17.24	20.73	20.39	16.57	/	/	10.98	20.29
STD	0.29	0.37	0.45	0.32	0.44	0.44	0.32	0.28	0.45	/	/	0.27	0.35
Recall 20	27.84	26.95	23.05	28.38	30.29	26.08	30.96	30.70	25.99	/	/	17.41	31.55
STD	0.49	0.54	0.53	0.37	0.57	0.51	0.31	0.41	0.57	/	/	0.46	0.49
User-based indirect method													
Recall 10	19.89	20.55	19.87	21.17	20.97	**21.41**	**21.40**	19.96	20.49	20.69	**20.70**	17.54	/
STD	0.41	0.40	0.40	0.43	0.37	0.37	0.37	0.37	0.31	0.41	0.33	0.33	/
Recall 20	29.93	31.09	29.83	31.70	31.28	**32.09**	**31.98**	30.07	30.68	31.07	**31.18**	26.34	/
STD	0.63	0.52	0.57	0.59	0.55	0.58	0.53	0.45	0.42	0.54	0.48	0.46	/
Item-based indirect method													
Recall 10	12.45	13.49	12.57	16.82	18.08	19.08	19.72	19.85	18.98	19.50	17.79	6.26	/
STD	0.24	0.52	0.31	0.40	0.39	0.26	0.32	0.54	0.46	0.47	0.49	0.27	/
Recall 20	22.14	22.51	22.16	27.15	28.66	29.58	30.58	30.63	29.20	31.16	28.71	11.90	/
STD	0.44	0.72	0.38	0.43	0.45	0.33	0.39	0.51	0.60	0.57	0.47	0.26	/

Table 3
Average performance, on the Book-Crossing dataset, for the thirteen scoring algorithms and the three methods (direct, user-based indirect, and item-based indirect) for using them.

	K_{ED}	K_{LED}	K_{VND}	K_{RL}	K_{RCT}	K_{CT}	K_{MD}	K_{RED}	K_{RWR}	Bin	Cos	Basic	LCI
Direct method													
Recall 10	4.52	5.98	4.51	7.36	7.35	7.26	7.12	6.22	4.74	/	/	3.70	5.83
STD	0.15	0.27	0.15	0.15	0.15	0.18	0.29	0.31	0.16	/	/	0.13	0.18
Recall 20	7.37	9.87	7.36	11.58	11.74	11.73	11.46	9.81	7.79	/	/	6.28	9.78
STD	0.15	0.26	0.16	0.20	0.26	0.33	0.23	0.35	0.99	/	/	0.95	0.17
User-based indirect method													
Recall 10	6.45	6.33	6.42	7.09	7.12	7.07	7.09	6.79	6.16	6.66	6.35	4.84	/
STD	0.13	0.22	0.13	0.21	0.11	0.17	0.18	0.20	0.15	0.12	0.16	0.12	/
Recall 20	10.33	10.33	10.28	11.23	11.39	11.39	11.31	10.77	9.95	10.64	10.28	8.05	/
STD	0.11	0.29	0.13	0.26	0.24	0.25	0.19	0.15	0.24	0.19	0.20	0.17	/
Item-based indirect method													
Recall 10	5.92	5.80	5.94	8.40	8.72	**8.84**	8.70	8.62	8.58	**8.79**	8.03	5.32	/
STD	0.24	0.26	0.21	0.45	0.27	0.22	0.30	0.35	0.22	0.26	0.23	0.26	/
Recall 20	9.61	8.97	9.67	12.77	13.47	**13.58**	13.35	13.17	13.25	**13.50**	12.36	8.58	/
STD	0.39	0.30	0.38	0.50	0.42	0.44	0.39	0.47	0.21	0.34	0.35	0.22	/

Table 4
Summary of the best accuracy results, for each scoring algorithm and both recall scores, with the method providing the best result.

	K_{ED}	K_{LED}	K_{VND}	K_{RL}	K_{RCT}	K_{CT}	K_{MD}	K_{RED}	K_{RWR}	Bin	Cos	Basic	LCI
MovieLens dataset													
Recall 10	19.89	20.55	19.87	21.17	20.97	**21.41**	**21.40**	20.39	20.49	20.69	20.70	17.54	20.29
method	*user*	*user*	*user*	*user*	*user*	*user*	*user*	*dir*	*user*	*user*	*user*	*user*	*dir*
Recall 20	29.93	31.09	29.83	31.70	31.28	**32.09**	**31.98**	30.70	30.68	31.16	31.18	26.34	31.55
method	*user*	*user*	*user*	*user*	*user*	*user*	*user*	*dir*	*user*	*item*	*user*	*user*	*dir*
Book-Crossing dataset													
Recall 10	6.45	6.33	6.42	8.40	8.72	**8.84**	8.70	8.62	8.58	**8.79**	8.03	5.32	5.83
method	*user*	*user*	*user*	*item*	*item*	*item*	*item*	*item*	*item*	*item*	*item*	*item*	*dir*
Recall 20	10.33	10.33	10.28	12.77	13.47	**13.58**	13.35	13.17	13.25	**13.50**	12.36	8.58	9.78
method	*user*	*user*	*user*	*item*	*item*	*item*	*item*	*item*	*item*	*item*	*item*	*item*	*dir*

better results than the Bin method that performs best among the standard algorithms. However they do not outperform the standard methods, which is a bit disappointing. While the results show that the kernels methods accurately capture some structural information, they do not provide a significant added value over standard methods.

Table 5 attempts a fair comparison between the nine kernel-based algorithms by normalizing the numbers (the "initial scores") in Table 4. The numbers in each row i of the twelve first rows of Table 5 are computed as follows (the "baseline score" is the score obtained by the Basic algorithm):

(1) new score = (initial score − baseline)/baseline.
(2) final score = (new score−average(i))/std(i), where average(i) is the mean score on row i and std(i) is the standard deviation on row i.

The "Average CR score" row is the average of the first twelve results. The "CR ranking" row ranks the algorithms according to these average scores.

Table 5 suggests that the best kernel-based algorithm (in terms of average CR score) is K_{MD} (0.97), followed by K_{RCT} (0.77), and by K_{CT} (0.70). A paired t-test shows that there is no significant difference (with a p-value smaller than 0.05) between K_{MD} and

Table 5
Overall scores for ranking the algorithms on the collaborative-recommendation task (the higher the value, the better the algorithm).

	K_{ED}	K_{LED}	K_{VND}	K_{RL}	K_{RCT}	K_{CT}	K_{MD}	K_{RED}	K_{RWR}
ML-*dir*10	0.09	−0.16	−1.78	0.31	0.75	−0.60	1.28	1.09	−0.97
ML-*dir*20	0.01	−0.33	−1.82	0.22	0.95	−0.66	1.21	1.11	−0.70
ML-*user*10	−1.18	−0.13	−1.21	0.85	0.53	1.22	1.21	−1.06	−0.23
ML-*user*20	−1.17	0.15	−1.29	0.84	0.36	1.28	1.16	−1.01	−0.32
ML-*item*10	−1.39	−1.06	−1.36	0.01	0.42	0.74	0.95	0.99	0.71
ML-*item*20	−1.31	−1.21	−1.31	0.05	0.46	0.72	0.99	1.00	0.61
BC-*dir*10	−1.28	−0.11	−1.29	1.00	0.99	0.92	0.80	0.08	−1.11
BC-*dir*20	−1.30	0.01	−1.30	0.90	0.98	0.98	0.84	−0.02	−1.08
BC-*user*10	−0.71	−1.02	−0.79	0.95	1.03	0.90	0.95	0.17	−1.46
BC-*user*20	−0.79	−0.79	−0.87	0.80	1.08	1.08	0.94	−0.01	−1.46
BC-*item*10	−1.30	−1.39	−1.29	0.49	0.72	0.81	0.70	0.65	0.62
BC-*item*20	−1.22	−1.55	−1.19	0.40	0.76	0.82	0.70	0.61	0.65
Average CR score	−0.96	−0.62	−1.30	0.54	0.77	0.70	0.97	0.28	−0.38
CR ranking	8	7	9	4	2	3	1	5	6

K_{CT} ranges of scores. Two other algorithms exhibit above-average (i.e., positive) CR scores: K_{RL} (0.54) and K_{RED} (0.28), while the last four algorithms are below average: K_{ED} (−0.96), K_{LED} (−0.62), K_{VND} (−1.30), and K_{RWR} (−0.38). Notice that K_{MD}, K_{RCT}, and K_{RL} are the only kernels with all their scores positive, providing argument for the stability of these kernels. K_{MD} and K_{RCT} can therefore be considered as the best kernels on these collaborative recommendation tasks.

6. Testing the kernels on semisupervised classification

This section reports on another experimental comparison of the accuracy of the various kernels on a semi-supervised classification task. Their performance was also compared to that of a state-of-the-art semisupervised random-walk-based classification method, namely the $\mathcal{D}$-walks model (Callut, Francoisse, Saerens, & Dupont, 2008). In a practical classification setting, labeled data is typically hard to obtain, since it often depends on expensive human labor. Unlabeled data is more easily available. For example, large databases of text documents are readily available but explicitly classified text databases are much scarcer. Semisupervised classification algorithms take advantage of whatever labeled data is available to improve the classification accuracy. For a comprehensive survey of semisupervised classification, see Abney (2008), Chapelle et al. (2006) and Zhu (2008); Zhu and Goldberg (2009).

The main objective is to infer labels for the unlabeled nodes from the labeled ones (Macskassy & Provost, 2007), that is, to classify unlabeled nodes. In addition to available node labels, the graph topology provides useful information. Several values of the labeling rate (i.e., the proportion of nodes with known labels) were considered (ranging from 10% to 90%). In order to assess the ability of the algorithms to determine correct labels, some labels were removed from the original dataset and the removed data was used as a test set (the remaining data constituting the so-called training set). For each labeling rate considered, 50 random deletions of labels were performed thus creating 50 test sets of unlabeled nodes on which performance was averaged.

6.1. Datasets

Datasets, seven in total, comprise partially graphs (NewsGroup, Cora, WebKB, IMDb) described below. They are real-world representations of networked data. They come from several domains that have been the subject of prior study in machine learning (Macskassy & Provost, 2007). For each dataset, local consistency of node labeling is assumed, i.e., it is assumed that nodes within the same neighborhood tend to share the same label. This could be assessed by using autocorrelation indexes, such as Moran's I or Geary's C, as proposed in the spatial statistics literature (Cressie, 1993; Haining, 2003).

Table 6
The IMDb dataset.

Class	Size
High-revenue	572
Low-revenue	597
Total	1169
Majority class proportion	51.1%
Number of edges	40,564

Table 7
The CORA dataset.

Class	Size
Case-based	402
Genetic algorithms	551
Neural networks	1064
Probabilistic methods	529
Reinforcement learning	335
Rule learning	230
Theory	472
Total	3583
Majority class proportion	29.7%
Number of edges	22,516

The collaborative Internet Movie Database (*IMDb*) has had several applications, like making recommendations for movies to watch or classifying movies into categories. Here, the classification problem deals with predicting movie popularity (whether a movie is a likely blockbuster, award winner, b-movie, or stinker). It exploits a graph of movie nodes linked when they share a production company. Edge weights denote the number of production companies that two movies have in common. The dataset is summarized in Table 6.

The *Cora* dataset is a graph of 3583 nodes collected from research papers on machine learning labeled with one of seven different topics. Two papers are linked if they share an author or if one paper cites the other one. The composition of the dataset is summarized in Table 7.

The *WebKB* data consists of four datasets of web pages gathered from four university departments, with each page manually labeled as belonging to one of six authorship categories: course, department, faculty, project, staff, and student. Pages are linked by co-citation (x and y are linked if x references z and y references z). The composition of the dataset is summarized in Table 8.

The *Newsgroup* dataset[5] comprises about 20,000 unstructured documents, taken from 20 Usenet newsgroups. News articles are labeled as belonging to one of five categories out the 20 original ones. The graph of documents was built by sampling

[5] Available from http://people.csail.mit.edu/jrennie/20Newsgroups/.

Table 8
Composition of the WebKB dataset.

Class	Size			
	Cornell	Texas	Washington	Wisconsin
Course	54	51	170	83
Department	25	36	20	37
Faculty	62	50	44	37
Project	54	28	39	25
Staff	6	6	10	11
Student	145	163	151	155
Total	346	334	434	348
Majority class proportion	41.9%	48.8%	39.2%	44.5%
Number of edges	26,832	32,988	30,462	33,250

Table 9
Composition of the Newsgroup dataset.

Class	Size
Computer/graphics	200
Computer/pchardware	200
Motor/auto	200
Religion/atheism	200
Politics/mideast	200
Total	1000
Majority class proportion	20.0%
Number of edges	328,904

about 200 documents at random from five different topics (see Table 9). The links between documents were extracted from the five samples by eliminating stopwords, by stemming using a TF–IDF (term frequency–inverse document frequency) technique, by computing mutual information, and finally by computing the document–document matrix, as described in Yen et al. (2007). The composition of the dataset is summarized in Table 9.

6.2. Classification rule and performance evaluation

The kernels studied in this paper are compared on the seven datasets just introduced with the following methodology. The classification of unlabeled nodes is performed with a simple kernel-alignment strategy based on a sum of similarities. More precisely, each class c and its nodes are described by an n-dimensional indicator vector $\mathbf{y}^c$ containing a "1" entry when the corresponding node belongs to class c and a "0" entry otherwise, that is, when the node is unlabeled or belongs to another class. The similarity of node i with the set of nodes belonging to class c is the value of the i-entry of the column vector $\mathbf{K}\mathbf{y}^c$ where $\mathbf{K}$ is the kernel matrix. This similarity is computed for each class c in turn. Then, each node is assigned to the class with largest similarity. The column vector containing the estimated class index of all nodes is thus $\widehat{\mathbf{y}}^* = \arg\max_c(\mathbf{K}\mathbf{y}^c)$. For the kernels that are inverses of matrices, if only one node is to be labeled, a number of systems of linear equations can be solved instead of inverting the matrix (Mantrach et al., 2011). As discussed in Section 4.4, this simple kernel alignment rule is based, for some particular kernels, on the optimization of a loss function taking available class labels and smoothness into account (Belkin et al., 2004; Belkin, Niyogi, & Sindhwani, 2006; Wang et al., 2009; Yajima & Kuo, 2006; Zhou et al., 2003, 2005; Zhu et al., 2003). For a theoretical analysis of this regularization framework, see Johnson and Zhang (2007, 2008).

This classification methodology is arguably somewhat primitive, but the simple decision rule permits to check which kernels accurately capture similarity and class membership. The task thus consists in classifying unlabeled nodes—classifiers assign the most appropriate class to each unlabeled node. The classification accuracy was computed for several values of the labeling rate (10%, 20%, …, 90%), i.e., the proportion of nodes with known label. The labels of remaining nodes were removed and the corresponding sets of nodes were used as test sets. For each labeling rate, 50 random deletions of node labels (50 runs) were performed and classification rate was averaged on those 50 runs.

For each run, a ten-fold internal cross-validation was performed in order to tune the parameters of each classifier (typically, the parameters of the kernels, as detailed in Table 1). The performance on each run was assessed on the remaining unlabeled nodes with the tuned parameter.

6.3. Other (non kernel-based) scoring algorithms

The kernel-based alignment procedure was compared to two algorithms, NetKit (Macskassy & Provost, 2007), as a baseline, and the $\mathcal{D}$-walks approach (Callut et al., 2008).

NetKit is based on a relational learning approach (Macskassy & Provost, 2007). This general framework builds a model based on three components: a local classifier to generate class priors, a relational classifier, which relies on the relations in the network to guess the class membership, and a so-called collective inferencing component which 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. For our experiments, testing all the module configurations would have been prohibitive. For the baseline, we chose the parameters that generally provide the best results and are quite robust according to (Macskassy & Provost, 2007). More precisely, the local classifier inducer uses the class prior, the relational classifier inducer uses the "weighted vote relational neighbor classifier", and a relaxation labeling is used for the collective inferencing. The first two instantiations do not take any parameter as they imply only class priors and direct nearest neighbors (no descriptor on the nodes). As for the relaxation labeling (Chakrabarti, Dom, & Indyk, 1998), parameters were set to default values (described in Section 4.2.2 of (Macskassy & Provost, 2007)), that were shown to be quite robust (Macskassy & Provost, 2007).

$\mathcal{D}$-walks rely on bounded random walks performed on the input graph seen as a Markov chain (Callut et al., 2008). A betweenness measure, based on passage times during random walks of bounded length, is derived for each class (label category). Unlabeled nodes are assigned to the category for which the betweenness is highest. This approach 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 while remaining competitive with other techniques. It has a unique hyper-parameter, the walk length, that can be tuned during cross-validation.

6.4. Results and discussion

Classification results for each kernel, and for NetKit and $\mathcal{D}$-walks on all datasets are shown in Figs. 1–7. Some methods

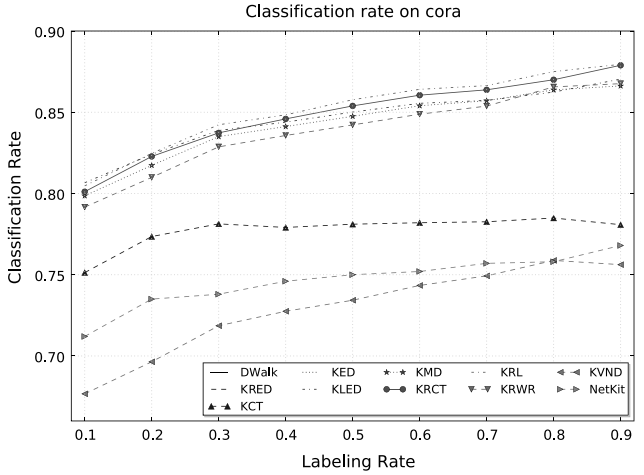

Fig. 1. Average classification rates for the CORA dataset as a function of labeling rate.

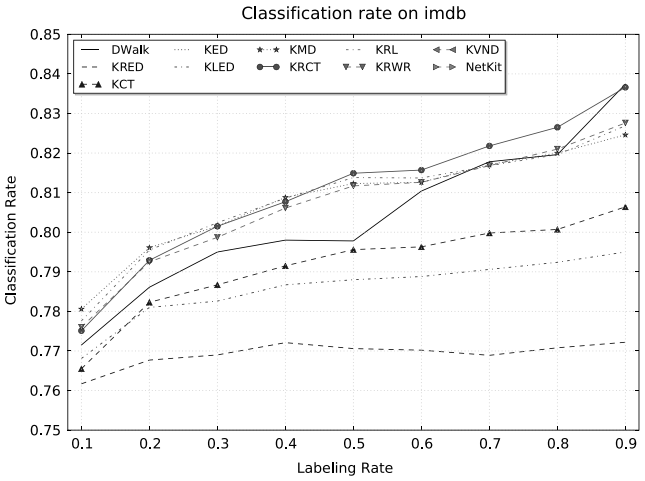

Fig. 2. Average classification rates for the IMDb dataset as a function of labeling rate.

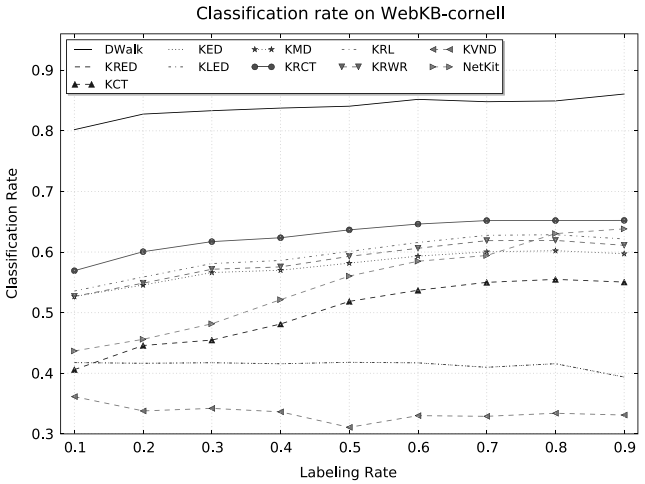

Fig. 3. Average classification rates for the WebKB Cornell dataset.

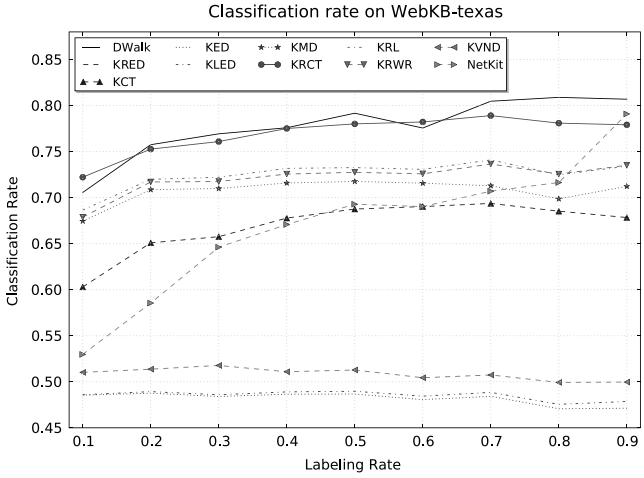

Fig. 4. Average classification rates for the WebKB Texas dataset.

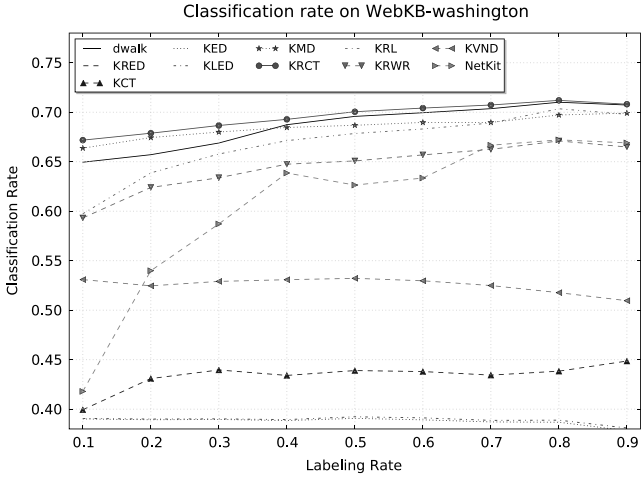

Fig. 5. Average classification rates for the WebKB Washington dataset.

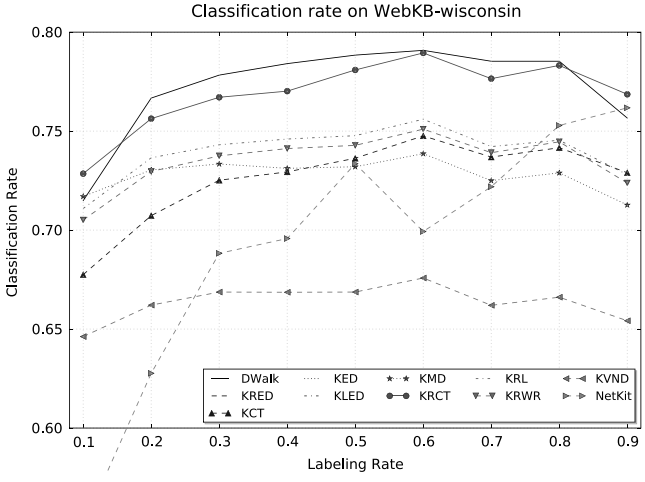

Fig. 6. Average classification rates for the WebKB Wisconsin datasets.

that perform poorly do not appear in the charts. Thus results for methods K_{ED} and K_{RED} in Fig. 1 and those for methods K_{ED}, K_{VND}, and NetKit in Fig. 2 are not shown. The same holds for method K_{RED} in both Figs. 3 and 4, for method K_{RED} in Fig. 5, for methods K_{RED}, K_{ED}, and K_{LED} in Fig. 6, and for method K_{ED} in Fig. 7. Note that the curves for K_{RED} and K_{ED}, as well as for K_{RCT} and $\mathcal{D}$-walks, overlay in Fig. 1, like those of K_{LED} and K_{ED} in Fig. 3.

Table 10
Area under curve score for each method and each dataset.

	K_{ED}	K_{LED}	K_{VND}	K_{RL}	K_{RCT}	K_{CT}	K_{MD}	K_{RED}	K_{RWR}	$\mathcal{D}$-walks	NetKit
Cora	0.100	0.677	0.585	**0.682**	0.679	0.623	0.674	0.209	0.671	0.680	0.597
IMDb	0.401	0.629	0.482	0.647	**0.648**	0.633	0.647	0.615	0.646	0.643	0.559
WebKB-Cornell	0.331	0.331	0.266	0.477	**0.503**	0.402	0.462	0.010	0.470	**0.672**	0.436
WebKB-Texas	0.385	0.388	0.407	0.581	**0.617**	0.538	0.567	0.030	0.578	**0.624**	0.536
WebKB-Washington	0.310	0.311	0.420	0.537	**0.557**	0.347	0.548	0.010	0.517	0.550	0.490
WebKB-Wisconsin	0.190	0.351	0.532	0.593	**0.617**	0.582	0.583	0.030	0.590	**0.621**	0.557
NewsGroup	0.160	0.616	0.623	0.691	**0.694**	0.688	0.684	0.623	0.686	0.693	0.661
Average AUC	0.268	0.471	0.473	0.601	**0.616**	0.544	0.595	0.216	0.594	**0.640**	0.548
Ranking	8	7	6	2	1	5	3	9	4	/	/

Table 11
Overall scores for ranking the algorithms on the semisupervised-classification (SC) task (the higher the value, the better the algorithm).

	K_{ED}	K_{LED}	K_{VND}	K_{RL}	K_{RCT}	K_{CT}	K_{MD}	K_{RED}	K_{RWR}
IMDb	−2.17	0.39	−1.24	0.58	0.58	0.45	0.58	0.26	0.58
Cora	−1.97	0.61	0.15	0.61	0.58	0.39	0.58	−1.49	0.55
WebKB-Cornell	−0.13	−0.13	−0.55	0.75	0.94	0.05	0.67	−2.30	0.70
WebKB-Texas	−0.36	−0.36	−0.22	0.69	0.87	0.41	0.64	−2.33	0.67
WebKB-Washington	−0.49	−0.49	0.17	0.77	0.91	−0.24	0.88	−2.19	0.66
WebKB-Wisconsin	−1.23	−0.45	0.38	0.67	0.75	0.59	0.64	−1.99	0.64
NewsGroup	−2.60	−0.02	0.14	0.50	0.54	0.50	0.50	−0.02	0.46
Average SC score	−1.28	−0.06	−0.17	0.65	0.74	0.31	0.64	−1.44	0.61
SC ranking	8	6	7	2	1	5	3	9	4

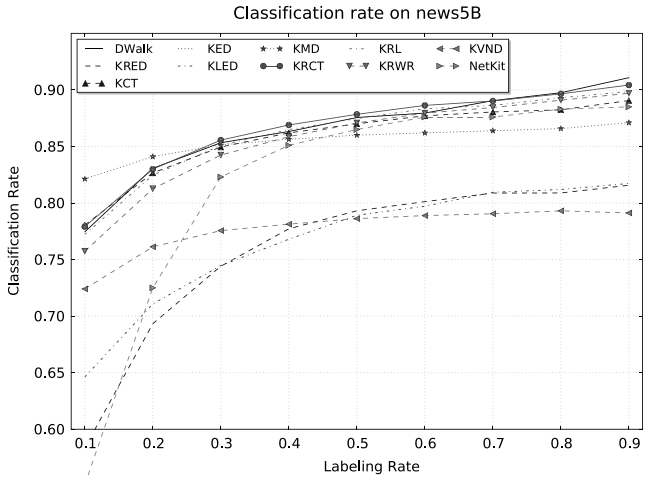

Fig. 7. Average classification rates for the `Newsgroup` dataset.

To facilitate comparisons, a summary table containing the relative "area under curves" (AUC score)[6] of Figs. 1–7 is shown as Table 10. For each dataset (i.e., for each row in the table), the best result among all kernels appears in bold. In addition, if the best performing method is not a kernel, the best result among all methods also appears in bold. The last two rows show, for each method, the AUC score averaged on all datasets as well as the ranking of kernel methods.

K_{RCT} and K_{RL} clearly outperform the other kernels on every dataset, with a small advantage for K_{RCT}. More precisely, K_{RCT} produces the best results on almost every dataset except for `CORA`, while K_{RL} obtains the best results on `CORA` and comes second (after K_{RCT}) on the other datasets. They appear to be very stable across datasets (as also pointed out in collaborative recommendation). Both are competitive with the $\mathcal{D}$-walks approach, except for the `WebKB-Cornell` dataset which looks somewhat atypical. Both K_{RWR} and K_{MD} are competitive but never get better results than the two leading kernels.

Some other kernels perform less well. K_{CT}, which provides rather good results on collaborative recommendation, lies behind on almost every dataset, except for the `Newsgroup` one. K_{LED} gets decent results on the `CORA`, `IMDb`, and `Newsgroup` datasets, but performs badly on the four `WebKB` datasets, making it unstable to use in other situations. This observation is in accordance with the results provided by K_{LED} on collaborative recommendation. Other kernels, such as K_{ED}, K_{VND}, and K_{RED} provide poor and unstable results on almost every dataset, confirming their poor results on collaborative recommendation.

We observe that the best kernel-based methods, while competitive in most cases, do not perform better than the semisupervised $\mathcal{D}$-walks approach. This indicates that kernel methods appear to be useful general-purpose tools that deserve investigation, but also that they do not outperform domain-specific techniques, at least on the explored datasets.

Similar to Table 5 for collaborative recommendation, Table 11 shows a summary of the scores obtained on the semisupervised classification experiments by the various kernel-based algorithms (see Section 5.5 for details about computing scores). Each row corresponds to one dataset, for a fixed labeling rate of 30%. The "baseline score" is the score obtained by NetKit. The "Average SC score " row computes, for each scoring algorithm, the average of its 7 results, with the same weight for all the criteria (i.e., the rows). The "SC ranking" row ranks the algorithms according to their average score. Table 11 shows that the best kernel-based algorithm (in terms of its average SC score) is K_{RCT} (0.74), followed by K_{RL} (0.65), and by K_{MD} (0.64). A paired t-test shows that there is a significant difference (with a p-value smaller than 0.05) between the results provided by the best algorithm (i.e., K_{RCT}) and the results provided by the other ones. Two more algorithms provide scores that are above average (i.e., with a positive average SC score): K_{RWR} (0.61) and K_{CT} (0.31), while the last four algorithms provide scores which are below average: K_{ED} (−1.28), K_{LED} (−0.06), K_{VND} (−0.17), and K_{RED} (−1.44).

Note that the rankings based on AUC scores, shown in Table 10, are clearly confirmed by those based on overall scores in Table 11 (except for K_{LED} and K_{VND} whose rankings – i.e., 6 and 7 – are reversed).

[6] This AUC is calculated by computing the definite integral using the trapezoidal approximation technique (Dahlquist & Bjorck, 1974; Stoer & Bulirsch, 2002).

Table 12
Overall scores for ranking the algorithms on collaborative recommendation (CR) and semisupervised classification (SC) (the higher the value, the better the algorithm).

	K_{ED}	K_{LED}	K_{VND}	K_{RL}	K_{RCT}	K_{CT}	K_{MD}	K_{RED}	K_{RWR}
Average CR score	−0.96	−0.62	−1.30	0.54	0.77	0.70	0.97	0.28	−0.38
CR ranking	8	7	9	4	2	3	1	5	6
Average SC score	−1.28	−0.06	−0.17	0.65	0.74	0.31	0.64	−1.44	0.61
SC ranking	8	6	7	2	1	5	3	9	4
Average score	−1.12	−0.34	−0.74	0.60	**0.76**	0.51	**0.81**	−0.58	0.12
Final ranking	9	6	8	3	2	4	1	7	5

7. General discussion

Table 12 shows scores obtained by combining Tables 5 and 11. More precisely, the "Average score" is computed by averaging "Average CR" and "Average SC" scores (thus giving the same weight to both tasks). A "Final ranking" based on the "Average score" is assigned to each kernel. These results suggest that K_{MD} is the best method, with an average score of 0.81, closely followed by K_{RCT} (with a score of 0.76), K_{RL} (with a score of 0.60), and K_{CT} (with a score of 0.51). Still, (1) we cannot conclude that K_{MD} provides significantly better results than K_{RCT} (using a paired t-test to determine if there is a significant difference between the algorithms, $p < 0.05$) and (2) the results of Table 12 confirm that the choice of the best kernel is application-dependent (K_{MD} for collaborative recommendation and K_{RCT} for semisupervised classification); the most application-independent method (i.e., with the closest CR and SC scores) is K_{RCT} (0.77 and 0.74).

Three kernels provide quite stable and good overall performance on all datasets, independently of the application, namely K_{MD}, K_{RCT}, and K_{RL} (see Table 1 for definitions). All the other kernels perform significantly worse on at least one task in our experiments. However, these best kernel-based methods, while competitive, do not always perform better than more domain-specific techniques, at least for the investigated datasets. This indicates that both the choice of kernels as well as their performance in comparison to more domain-specific techniques are to a certain extend domain-, or even data-, dependent.

8. Conclusion and further work

This paper investigates several kernel-based procedures to compute similarities between elements of a database or, more generally, nodes of an undirected weighted graph. These similarity measures can be used to compare items belonging to database tables that are not necessarily directly connected. They depend on the strength of the connectivity between items. Our experiments show that some kernels perform well in comparison with standard algorithms, namely the Markov diffusion kernel, the regularized Laplacian kernel, and the regularized commute-time kernel (see Table 1 for definitions). These three kernels perform consistently well. They have a nice and intuitive interpretation in terms of random walk (the Markov diffusion kernel and the regularized commute-time kernel) or in terms of the matrix-forest theorem (the regularized Laplacian kernel).

Another interesting property of the kernel-based methods is that they induce a data matrix on which standard multivariate statistical analysis methods can be applied for mining graphs or databases and that this data matrix need not be computed (the kernel matrix is sufficient). This allows to generalize useful techniques, like clustering, principal-component analysis, discriminant analysis, and canonical correlation analysis to the study of graphs or databases.

Further work will investigate the interest of adding more features of users and items, and ratings on the edges of the adjacency matrix. Other kernel-based multivariate statistical techniques, like discriminant analysis or canonical correlation analysis, will also be studied in the context of graph mining. Finally, various approximation schemes allowing to perform semisupervised classification on large graphs will be studied.

References

Abney, S. (2008). *Semisupervised learning for computational linguistics*. Chapman and Hall, CRC.

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 734–749.

Agaev, R., & Chebotarev, P. (2000). The matrix of maximum out forests of a digraph and its applications. *Automation and Remote Control*, *61*(9), 1424–1450.

Agaev, R., & Chebotarev, P. (2001). Spanning forests of a digraph and their applications. *Automation and Remote Control*, *62*(3), 443–466.

Armstrong, J. (2001). *Principles of forecasting, a handbook for researchers and practitioners*. Kluwer Academic.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval*. Addison-Wesley.

Bartholomew, D. J. (1999). *Latent variable models and factor analysis* (second ed.) Arnold.

Belkin, M., Matveeva, I., & Niyogi, P. (2004). Tikhonov regularization and semi-supervised learning on large graphs. In *Proceedings of the IEEE international conference on acoustics, speech, and signal processing. ICASSP2004* (pp. 1000–1003).

Belkin, M., Niyogi, P., & Sindhwani, V. (2006). Manifold regularization: a geometric framework for learning from examples. *Journal of Machine Learning Research*, *7*, 2399–2434.

Bengio, Y., Delalleau, O., & Le Roux, N. (2006). Label propagation and quadratic criterion. In O. Chapelle, B. Scholkopf, & A. Zien (Eds.), *Semi-supervised learning* (pp. 193–216). MIT Press.

Boley, D., Ranjan, G., & Zhang, Z.-L. (2011). Commute times for a directed graph using an asymmetric Laplacian. *Linear Algebra and its Applications*, *435*(2), 224–242.

Borg, I., & Groenen, P. (1997). *Modern multidimensional scaling: theory and applications*. Springer.

Brand, M. (2005). A random walks perspective on maximizing satisfaction and profit. In *Proceedings of the 2005 SIAM international conference on data mining*.

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, *30*(1–7), 107–117.

Callut, J., Francoisse, K., Saerens, M., & Dupont, P. (2008). Semi-supervised classification from discriminative random walks. In *Lecture notes in artificial intelligence*: *Vol. 5211. Proceedings of the 19th European conference on machine learning, ECML 2008* (pp. 162–177). Berlin: Springer-Verlag.

Chakrabarti, S., Dom, B., & Indyk, P. (1998). Enhanced hypertext categorization using hyperlinks. In *Proceedings of the 1998 ACM SIGMOD international conference on management of data* (pp. 307–318).

Chan, T., Ciarlet, P., & Szeto, W. (1997). On the optimality of the median cut spectral bisection graph partitioning method. *SIAM Journal on Scientific Computing*, *18*(3), 943–948.

Chandra, A. K., Raghavan, P., Ruzzo, W. L., Smolensky, R., & Tiwari, P. (1989). The electrical resistance of a graph captures its commute and cover times. In *Annual ACM symposium on theory of computing* (pp. 574–586).

Chapelle, O., Scholkopf, B., & Zien, A. (2006). *Semi-supervised learning*. MIT Press.

Chebotarev, P. (2011). A class of graph-geodetic distances generalizing the shortest-path and the resistance distances. *Discrete Applied Mathematics*, *159*(5), 295–302.

Chebotarev, P., & Shamis, E. (1997). The matrix-forest theorem and measuring relations in small social groups. *Automation and Remote Control*, *58*(9), 1505–1514.

Chebotarev, P., & Shamis, E. (1998a). On a duality between metrics and s-proximities. *Automation and Remote Control*, *59*(4), 608–612.

Chebotarev, P., & Shamis, E. (1998b). On proximity measures for graph vertices. *Automation and Remote Control*, *59*(10), 1443–1459.

Chen, T., Yang, Q., & Tang, X. (2007). Directed graph embedding. In *Proceedings of the international joint conference on artificial intelligence. IJCAI* (pp. 2707–2712).

Cheung, K., Tsui, K., & Liu, J. (2004). Extended latent class models for collaborative recommendation. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, *34*, 143–148.

Chung, F. R. (1997). *Spectral graph theory*. American Mathematical Society.

Chung, F. R. (2005). Laplacians and the Cheeger inequality for directed graphs. *Annals of Combinatorics*, *9*, 1–19.

Cover, T. M., & Thomas, J. A. (2006). *Elements of information theory* (second ed.) John Wiley & Sons.

Cox, T., & Cox, M. (2001). *Multidimensional scaling* (second ed.) Chapman and Hall.

Cressie, N. (1993). *Statistics for spatial data, revised edition*. Wiley.

Dahlquist, G., & Bjorck, A. (1974). *Numerical methods*. Prentice-Hall.

Delannay, N., & Verleysen, M. (2008). Collaborative filtering with interlaced generalized linear models. *Neurocomputing*, *71*(7–9), 1300–1310.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society. Series B*, *39*, 1–38.

Ding, C., Jin, R., Li, T., & Simon, H. (2007). A learning framework using Green's function and kernel regularization with application to recommender system. In *Proceedings of the international conference on knowledge discovery in databases. KDD2007* (pp. 260–269).

Donetti, L., & Munoz, M. (2004). Detecting network communities: a new systematic and efficient algorithm. *Journal of Statistical Mechanics: Theory and Experiment*, P10012.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification* (second ed.) John Wiley & Sons.

Dunham, M. (2003). *Data mining: introductory and advanced topics*. Prentice Hall.

Džeroski, S. (2003). Multi-relational data mining: an introduction. *ACM SIGKDD Explorations Newsletter*, *5*(1), 1–16.

Fiedler, M. (1975a). Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, *23*(2), 298–305.

Fiedler, M. (1975b). A property of eigenvectors of nonnegative symmetric matrices and its applications to graph theory. *Czechoslovak Mathematical Journal*, *25*(100), 619–633.

Fiedler, M. (1989). Laplacian of graphs and algebraic connectivity. *Combinatorics and Graph Theory*, *25*, 57–70.

Fine, S., & Scheinberg, K. (2001). Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, *2*, 243–264.

Fouss, F., Pirotte, A., Renders, J.-M., & Saerens, M. (2007). Random-walk computation of similarities between nodes of a graph, with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, *19*(3), 355–369.

Fouss, F., Yen, L., Pirotte, A., & Saerens, M. (2006). An experimental investigation of graph kernels on a collaborative recommendation task. In *Proceedings of the 6th international conference on data mining. ICDM 2006* (pp. 863–868).

Gartner, T. (2008). *Kernels for structured data*. World Scientific Publishing.

Gori, M., & Pucci, A. (2006). Research paper recommender systems: a random-walk based approach. In *Proceedings of the 2006 IEEE/WIC/ACM international conference on web intelligence* (pp. 778–781).

Gori, M., & Pucci, A. (2007). Itemrank: a random-walk based scoring algorithm for recommender engines. In *Proceedings of the international joint conference on artificial intelligence. IJCAI 2007* (pp. 2766–2771).

Haining, R. (2003). *Spatial data analysis*. Cambridge University Press.

Hall, K. M. (1970). An r-dimensional quadratic placement algorithm. *Management Science*, *17*(8), 219–229.

Ham, J., Lee, D., Mika, S., & Scholkopf, B. (2004). A kernel view of the dimensionality reduction of manifolds. In *Proceedings of the 21st international conference on machine learning. ICML2004*.

Herlocker, J., Konstan, J., Terveen, L., & Riedl, J. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, *22*(1), 5–53.

Hill, W., Stead, L., Rosenstein, M., & Furnas, G. (1995). Recommending and evaluating choices in a virtual community of use. In *Proceedings of ACM CHI'95 conference on human factors in computing systems* (pp. 194–201).

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, *22*(1), 89–115.

Ito, T., Shimbo, M., Kudo, T., & Matsumoto, Y. (2005). Application of kernels to link analysis. In *Proceedings of the eleventh ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 586–592).

Johnson, R., & Wichern, D. (2002). *Applied multivariate statistical analysis* (fifth ed.) Prentice Hall.

Johnson, R., & Zhang, T. (2007). On the effectiveness of Laplacian normalization for graph semi-supervised learning. *Journal of Machine Learning Research*, *8*, 1489–1517. July.

Johnson, R., & Zhang, T. (2008). Graph-based semi-supervised learning and spectral kernel design. *IEEE Transactions on Information Theory*, *54*(1), 275–288.

Jolliffe, I. (2002). *Principal components analysis* (second ed.) Springer-Verlag.

Jungnickel, D. (2008). *Graphs, networks, and algorithms* (third ed.) Algorithms and computation in mathematics.

Kandola, J., Cristianini, N., & Shawe-Taylor, J. (2002). Learning semantic similarity. *Advances in Neural Information Processing Systems*, 657–664.

Kapur, J. N., & Kesavan, H. K. (1992). *Entropy optimization principles with applications*. Academic Press.

Karypis, G. (2001). Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the tenth international conference on information and knowledge management* (pp. 247–254).

Kelly, F. P. (1979). *Reversibility and stochastic networks*. John Wiley.

Kessler, M. M. (1963). Bibliographic coupling between scientific papers. *American Documentation*, *14*(1), 10–25.

Klein, D. J., & Randic, M. (1993). Resistance distance. *Journal of Mathematical Chemistry*, *12*, 81–95.

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM*, *46*(5), 604–632.

Kolaczyk, E. (2009). *Statistical analysis of network data: methods and models*. Springer.

Kondor, R. I., & Lafferty, J. (2002). Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th international conference on machine learning. ICML 2002* (pp. 315–322).

Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceeding of the 14th ACM SIGKDD international conference on knowledge discovery and data mining. KDD 2008* (pp. 426–434).

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *IEEE Computer*, *42*(8), 30–37.

Kunegis, J., & Lommatzsch, A. (2009). Learning spectral graph transformations for link prediction. In *Proceedings of the 26th international conference on machine learning. ICML2009* (pp. 561–568).

Lafferty, J., & Lebanon, G. (2005). Diffusion kernels on statistical manifolds. *Journal of Machine Learning Research*, *6*, 129–163.

Lafon, S., & Lee, A. B. (2006). Diffusion maps and coarse-graining: a unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(9), 1393–1403.

Langville, A., & Meyer, C. D. (2005). A survey of eigenvector methods for web information retrieval. *SIAM Review*, *47*, 135–161.

Latapy, M., & Pons, P. (2005). Computing communities in large networks using random walks. In *Proceedings of the 20th international symposium on computer and information sciences* (pp. 284–293).

Lee, J., & Verleysen, M. (2007). *Nonlinear dimensionality reduction*. Springer.

Lilien, G., Smith, B., & Moorthy, K. (1992). *Marketing models*. Prentice Hall.

Lu, Z., Jain, P., & Dhillon, I. (2009). Geometry-aware metric learning. In *Proceedings of the 26th international conference on machine learning. ICML 2009* (pp. 673–680).

Macskassy, S. A., & Provost, F. (2007). Classification in networked data: a toolkit and a univariate case study. *Journal of Machine Learning Research*, *8*, 935–983.

Manning, C., Raghavan, P., & Schutze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.

Mantrach, A., van Zeebroeck, N., Francq, P., Shimbo, M., Bersini, H., & Saerens, M. (2011). Semi-supervised classification and betweenness computation on large, sparse, directed graphs. *Pattern Recognition*, *44*(6), 1212–1224.

Mantrach, A., Yen, L., Callut, J., Francoisse, K., Shimbo, M., & Saerens, M. (2010). The sum-over-paths covariance kernel: a novel covariance between nodes of a directed graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(6), 1112–1126.

Mardia, K. V., Kent, J. T., & Bibby, J. M. (1979). *Multivariate analysis*. Academic Press.

Meyer, C. D. (2000). *Matrix analysis and applied linear algebra*. SIAM.

Mohar, B. (1992). Laplace eigenvalues of graphs—a survey. *Discrete Mathematics*, *109*, 171–183.

Murthi, B., & Sarkar, S. (2003). The role of the management sciences in research on personalization. *Management Science*, *49*(10), 1344–1362.

Nadler, B., Lafon, S., Coifman, R., & Kevrekidis, I. (2005). Diffusion maps, spectral clustering and eigenfunctions of Fokker–Planck operators. *Advances in Neural Information Processing Systems*, *18*, 955–962.

Nadler, B., Lafon, S., Coifman, R., & Kevrekidis, I. (2006). Diffusion maps, spectral clustering and reaction coordinate of dynamical systems. *Applied and Computational Harmonic Analysis*, *21*, 113–127.

Noble, B., & Daniels, J. (1988). *Applied linear algebra* (third ed.) Prentice-Hall.

Norris, J. R. (1997). *Markov chains*. Cambridge University Press.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1998). The Pagerank citation ranking: bringing order to the web. *Technical report*. Computer System Laboratory. Stanford University.

Pan, J. -Y., Yang, H. -J., Faloutsos, C., & Duygulu, P. (2006). Automatic multimedia cross-modal correlation discovery. In Proceedings of the 10th ACM SIGKDD international conference on knowledge discovery and data mining. Vol. 82. No. 4 (pp. 331–338).

Parzen, E. (1962). *Stochastic processes*. Holden-Day.

Paterek, A. (2007). Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of the KDD cup workshop*.

Pons, P., & Latapy, M. (2006). Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications*, *10*(2), 191–218.

Pothen, A., Simon, H. D., & Liou, K.-P. (1990). Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, *11*(3), 430–452.

Pucci, A., Gori, M., & Maggini, M. (2006). A random-walk based scoring algorithm applied to recommender engines. In *Proceedings of the International Workshop on Knowledge Discovery on the Web, WebKDD 2006* (pp. 127–146).

Qiu, H., & Hancock, E. R. (2005). Image segmentation using commute times. In *Proceedings of the 16th British machine vision conference. BMVC 2005* (pp. 929–938).

Qiu, H., & Hancock, E. R. (2007). Clustering and embedding using commute times. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(11), 1873–1890.

Rao, C., & Mitra, S. (1971). *Generalized inverse of matrices and its applications*. John Wiley & Sons.

Resnick, P., Neophytos, I., Mitesh, S., Bergstrom, P., & Riedl, J. (1994). GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the conference on computer supported cooperative work* (pp. 175–186).

Rich, E. (1979). User modeling via stereotypes. *Cognitive Science*, *3*(4), 329–354.

Ross, S. (1996). *Stochastic processes* (second ed.) Wiley.

Roth, V., Laub, J., Buhmann, J., & Müller, K.-R. (2003). Going metric: denoising pairwise data. In *Advances in neural information processing systems. NIPS 2002. Vol. 15* (pp. 817–824).
Rousseuw, P., & Molenberghs, G. (1993). Transformation of non positive semidefinite correlation matrices. *Communications in Statistics, Theory and Methods*, *22*(4), 965–984.
Saerens, M., Fouss, F., Yen, L., & Dupont, P. (2004). The principal components analysis of a graph, and its relationships to spectral clustering. In *Lecture notes in artificial intelligence*: *Vol. 3201*. *Proceedings of the 15th European conference on machine learning*, ECML 2004, (pp. 371–383). Berlin: Springer-Verlag.
Salakhutdinov, R., Mnih, A., & Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on machine learning. ICML09* (pp. 791–798).
Salton, G. (1989). *Automatic text processing*. Addison-Wesley.
Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the international world wide web conference* (pp. 285–295).
Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2002). Recommender systems for large-scale *e*-commerce: scalable neighborhood formation using clustering. In *Proceedings of the fifth international conference on computer and information technology*.
Scholkopf, B., & Smola, A. (2002). *Learning with kernels*. The MIT Press.
Scholkopf, B., Smola, A., & Muller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *5*(10), 1299–1319.
Sedgewick, R. (1990). *Algorithms in c*. Addison-Wesley.
Shardanand, U., & Maes, P. (1995). Social information filtering: algorithms for automating 'word of mouth'. In *Proceedings of the conference on human factors in computing systems* (pp. 210–217).
Shaw, B., & Jebara, T. (2009). Structure preserving embedding. In *Proceedings of the 26th international conference on machine learning. ICML 2009* (pp. 937–944).
Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge University Press.
Shimbo, M., & Ito, T. (2006). *Kernels as link analysis measures*. John Wiley & Sons, pp. 283–310.
Shimbo, M., Ito, T., Mochihashi, D., & Matsumoto, Y. (2009). On the properties of von Neumann kernels for link analysis. *Machine Learning*, *75*(1), 37–67.
Sindhwani, V., Belkin, M., & Niyogi, P. (2006). The geometric basis of semi-supervised learning. In O. Chapelle, B. Scholkopf, & A. Zien (Eds.), *Semi-supervised learning* (pp. 217–235). MIT Press.
Small, H. (1973). Co-citation in the scientific literature: a new measure of the relationship between two documents. *Journal of the American Society for Information Science*, *24*(4), 265–269.
Smola, A. J., & Kondor, R. (2003). Kernels and regularization on graphs. In *Proceedings of the conference on learning theory. COLT*.
Stoer, J., & Bulirsch, R. (2002). *Introduction to numerical analysis* (third ed.) Springer.
Theodoridis, S., & Koutroumbas, K. (2008). *Pattern recognition* (fourth ed.) Academic Press.
Tong, H., Faloutsos, C., & Pan, J. -Y. (2006). Fast random walk with restart and its applications. In *Proceedings of sixth IEEE international conference on data mining. ICDM 2006* (pp. 613–622).
Tong, H., Faloutsos, C., & Pan, J.-Y. (2008). Random walk with restart: fast solutions and applications. *Knowledge and Information Systems*, *14*(3), 327–346.
Vishwanathan, S., Schraudolph, N., Kondor, R., & Borgwardt, K. (2010). Graph kernels. *Journal of Machine Learning Research*, *11*, 1201–1242.
von Luxburg, U., Radl, A., & Hein, M. (2010). Getting lost in space: large sample analysis of the commute distance. In *Proceedings of the 23th neural information processing systems conference. NIPS 2010* (pp. 2622–2630).
Wang, J., Wang, F., Zhang, C., Shen, H., & Quan, L. (2009). Linear neighborhood propagation and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(9), 1600–1615.
White, S., & Smyth, P. (2003). Algorithms for estimating relative importance in networks. In *Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining. KDD 2003* (pp. 266–275).
Wilson, R. C., Hancock, E. R., & Luo, B. (2005). Pattern vectors from algebraic graph theory. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*, 1112–1124.
Yajima, Y., & Kuo, T.-F. (2006). Efficient formulations for 1-SVM and their application to recommendation tasks. *Journal of Computers*, *1*(3), 27–34.
Yen, L., Fouss, F., Decaestecker, C., Francq, P., & Saerens, M. (2007). Graph nodes clustering based on the commute-time kernel. In *Lecture notes in computer science*: *Vol. 4426*. *Proceedings of the 11th Pacific-Asia conference on knowledge discovery and data mining*, PAKDD 2007, (pp. 1037–1045). LNAI.
Yen, L., Fouss, F., Decaestecker, C., Francq, P., & Saerens, M. (2008). Graph nodes clustering with the sigmoid commute-time kernel: a comprehensive study. *Data & Knowledge Engineering*, *68*(3), 338–361.
Yen, L., Mantrach, A., Shimbo, M., & Saerens, M. (2008). A family of dissimilarity measures between nodes generalizing both the shortest-path and the commute-time distances. In *Proceedings of the 14th SIGKDD international conference on knowledge discovery and data mining. KDD 2008* (pp. 785–793).
Yen, L., Saerens, M., & Fouss, F. (2011). A link analysis extension of correspondence analysis for mining relational databases. *IEEE Transactions on Knowledge and Data Engineering*, *23*(4), 481–495.
Yen, L., Vanvyve, D., Wouters, F., Fouss, F., Verleysen, M., & Saerens, M. (2005). Clustering using a random walk-based distance measure. In *Proceedings of the 13th European symposium on artificial neural networks. ESANN2005* (pp. 317–324).
Zhao, D., & Tang, Z. L. X. (2007). Contextual distance for data perception. In *Proceedings of the eleventh IEEE international conference on computer vision. ICCV. Vol. 57* (pp. 1–8).
Zhou, D., Bousquet, O., Lal, T., Weston, J., & Scholkopf, B. (2003). Learning with local and global consistency. In *Proceedings of the neural information processing systems conference. NIPS 2003* (pp. 237–244).
Zhou, D., Huang, J., & Scholkopf, B. (2005). Learning from labeled and unlabeled data on a directed graph. In *Proceedings of the 22nd international conference on machine learning* (pp. 1041–1048).
Zhou, D., & Scholkopf, B. (2004). Learning from labeled and unlabeled data using random walks. In Rasmussen (Ed.). *Proceedings of the 26th DAGM symposium* (pp. 237–244).
Zhu, X. (2008). Semi-supervised learning literature survey. http://pages.cs.wisc.edu/~jerryzhu/research/ssl/semireview.html.
Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the twentieth international conference on machine learning. ICML 2003* (pp. 912–919).
Zhu, X., & Goldberg, A. (2009). *Introduction to semi-supervised learning*. Morgan & Claypool Publishers.