An Experimental Investigation of Graph Kernels on Collaborative Recommendation and Semisupervised Classification

François Fouss
Management Department – LSM
Facultés Universitaires Catholiques de Mons (FUCaM), Belgium
francois.fouss@fucam.ac.be

Kevin Françoisse, Luh Yen, Alain Pirotte & Marco Saerens
Information Systems Research Unit (ISYS) – LSM
Université catholique de Louvain (UCL), Belgium
{kevin.fransoisse,luh.yen,alain.pirotte,marco.saerens}@uclouvain.be

October 9, 2008

Abstract

This paper presents a systematic comparison of seven graph kernels and two related graph matrices, namely the exponential diffusion kernel, the Laplacian exponential diffusion kernel, the von Neumann diffusion kernel, the regularized Laplacian kernel, the commute-time kernel, the random-walk-with-restart similarity matrix, and finally, three kernels introduced in this paper: the regularized commute-time kernel, the Markov diffusion kernel, and the cross-entropy diffusion matrix. The power of the kernel-on-a-graph approach is illustrated by comparing the nine kernel-based algorithms on a collaborative-recommendation task and on a semisupervised classification task on several databases. Graph kernels compute proximity measures between nodes that help to study the structure of the graph. These comparisons suggest that the regularized Laplacian, the Markov diffusion, and the regularized commute-time kernels perform best.

1 Introduction

A function $k$ of the form $k: \Omega \times \Omega \rightarrow \mathbb{R}$ that, given two objects represented by $x$ and $y$ in some input space $\Omega$, returns a real number $k(x, y)$ can be considered a similarity measure (i.e., enjoying natural properties of similarity, like the triangular property, among others [14]) if it characterizes in a meaningful manner (i.e., intuitively adequate for applications) the similarities and differences (i.e., distance, proximity) between $x$ and $y$. A simple and classical similarity measure is the inner product of $x$ and $y$, provided that $x$ and $y$ are expressed in a vector (or inner-product) space. Positive definite functions $k$ (or kernel functions) [70, 73] enjoy the property that computing $k(x, y)$ is equivalent to computing the inner product of some transformation of $x$ and $y$ into another space.
In summary, kernel-based algorithms are characterized by at least two important properties: they allow (i) to compute inner products (i.e., similarities) in a high-dimensional space (often called “feature space”) where the data is more likely to be well-separated while not requiring to make explicit the mapping from input space to feature space, and (ii) to compute similarities between structured objects that cannot be naturally represented by a simple set of features. The latter property will be illustrated in this paper, with the general objective of computing similarities between nodes of a graph.

Thus, a useful kernel is expected to capture an appropriate measure of similarity for a particular task (such as a collaborative-recommendation task or a classification task, as investigated in this paper) and to require significantly less computation than would be needed via an explicit evaluation of the corresponding mapping from input space into feature space. In addition, mathematically, a kernel function must satisfy two requirements: (a) it must be symmetric (since the inner product of vectors is symmetric) and (b) it must be positive semidefinite.

An \( n \times n \) symmetric kernel matrix \( K \) (\( K_{ij} = k_{ij} = k(x_i, x_j) \)) computing the kernel function between \( n \times n \) pairs of objects indexed by \( i, j \) satisfies the following properties (that are all alternative definitions for symmetric positive semidefiniteness): (a) \( x^T K x \geq 0 \) for all \( x \in \mathbb{R}^n \); (b) all the eigenvalues of \( K \) are positive; (c) \( K \) is the Gram (inner product) matrix of some collection of linearly independent vectors \( x_1, ..., x_n \in \mathbb{R}^r \) for some \( r \), and (d) \( K \) can be viewed as a diagonal matrix \( \Lambda \) in another coordinate system (\( K = U \Lambda U^T \)).

Various types of kernels were described in [73], such as polynomial kernels, Gaussian kernels, anova kernels, kernels on a graph, kernels on sets, kernels on real numbers, randomized kernels, etc. This paper focuses on graph kernels: it gives an overview of the field, and it reports on applying various graph kernels to a collaborative-recommendation task and to a semisupervised classification task.

Mining structured data, like graphs and relational databases, has been the focus of growing interest (see for instance [1, 80]). Graph kernels 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 but, eventually, their intrinsic usefulness will clearly be related to their ability to perform adequately in practice. This is precisely the main objective of this paper: explore and evaluate nine such measures on real-world tasks of collaborative recommendation and classification.

The kernel-based similarity measures studied in this paper take into account 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 any path 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 distance measure naturally derived from a kernel can easily be shown to be Euclidean (see Section 2.2.4), that is, the nodes of the graph can be embedded in a Euclidean space preserving distances between nodes. This property can be used to visualize the graph in a lower-dimensional space or, more generally, via a principal-component analysis [67, 71, 73] or a discriminant analysis, to define a subspace pro-
jection that enjoys some optimality property.

The contributions of this paper are twofold. First, we introduce three new graph similarity matrices (the first two are also kernels): the Markov diffusion kernel, the regularized commute-time kernel, and the cross-entropy diffusion matrix. Second, we provide a comprehensive experimental comparison of those and a number of kernels that were proposed in the literature, namely the exponential diffusion kernel [44], the Laplacian exponential diffusion kernel [44, 76], the von Neumann diffusion kernel [73], the regularized Laplacian kernel [34, 76], the commute-time kernel [27, 67], and the random-walk-with-restart similarity matrix [58, 77, 78].

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 a graph kernel [7, 20, 51]. Section 3 reviews related work on similarity measures and graph kernels. Section 4 briefly presents the nine kernels to be investigated. Sections 5 introduces the experiments that illustrate the power of the approach: the comparison between the nine kernels on two collaborative-recommendation tasks (Section 6) and on seven semisupervised classification tasks (Section 7). Section 8 provides a global discussion of the experimental results. Section 9 concludes the paper.

A preliminary version of our work on collaborative recommendation appeared in [28]. The main extensions in this paper concern (a) a more comprehensive literature review (Section 3), (b) the definition of two new kernel-based 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 6.4), (d) a nested cross-validation (instead of a simple cross-validation) applied for assessing the performance of the algorithms (Section 6), (e) a thorough redesign of all experiments, and (f) the application of the algorithms to a semisupervised classification task (Section 7).

2 Review of basic relevant theory

This section reviews the basic theory behind graph kernels. In a very general setting, once some proximity measure between the nodes of a graph has been shown to be a kernel, then a number of derived quantities and interesting results 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. 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 \( A \) 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 \( L = D - A \), where \( D = \text{Diag}(a_{ii}) \) is the degree matrix, with diagonal entries \( d_{ii} = [D]_{ii} = a_{ii} = \sum_{j=1}^{n} a_{ij} \), if the graph has
n nodes in total. The volume of the graph is defined as $V_G = vol(G) = \sum_{i=1}^{n} d_i = \sum_{i,j=1}^{n} a_{ij}$.

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.

If the graph is connected, that is, if every node is reachable from every node, then $L$ has rank $n - 1$ [17]. If $e$ is a column vector all of whose elements are “1” (i.e., $e = [1, 1, \ldots, 1]^T$, where $T$ denotes the matrix transpose) and $0$ is a column vector all of whose elements are “0” $Le = 0$ and $e^T L = 0^T$ hold. $L$ is doubly centered. The null space of $L$ is thus the one-dimensional space spanned by $e$. $L$ is symmetric and positive semidefinite (see for instance [17]).

2.2 Kernels on a graph

In this paper, we are interested in intuitively meaningful similarity measures defined on every pair of nodes in a graph. Mathematically, most of these similarity matrices are graph kernels, i.e., positive semidefinite matrices.

A real symmetric matrix that is not positive semidefinite can be changed to a positive semidefinite one by adding the identity matrix $I$ multiplied by a suitably positive value to the original matrix (see, for example, [66]). Indeed, if $M$ is a symmetric matrix, the matrix $(M + \alpha I)$, with $\alpha \geq |\lambda_{\min}|$ (the smallest eigenvalue of $M$) is positive semidefinite. This is because $M$ and $(M + \alpha I)$ have the same eigenvectors $u_i$ associated with eigenvalues $\lambda_i$ for $M$ and $(\lambda_i + \alpha)$ for $(M + \alpha I)$. Thus, by choosing $\alpha \geq |\lambda_{\min}|$, all the eigenvalues are non-negative and $(M + \alpha I)$ is positive semidefinite.

Nine similarity matrices, defined from the adjacency matrix $A$ and whose elements are the values of the kernel function on all pairs of nodes, are defined in Section 4, and tested on real-world data in Sections 6 and 7. We review the main concepts behind this idea in the next sections. These concepts basically come from the fields of multidimensional scaling [7, 20, 51] and kernel methods [70, 73].

2.2.1 The data matrix associated to a kernel matrix

Let us first recall the fundamental spectral-decomposition theorem from standard linear algebra, underlying kernel-based work (see, e.g., [51, 56]):

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

Thus, if $x_i$ is the column vector corresponding to column $i$ of $X^T$, then the entries of $K$ are inner products $[K]_{ij} = k_{ij} = x_i^T x_j$, in accordance with the fact that $K$ is a kernel. The nodes of the graph are thus represented as vectors $x_i$ in a $r$-dimensional Euclidean space and they form a cloud of $n$ points in $\mathbb{R}^r$. The $x_i$’s are called node vectors, while matrix $X = [x_1, x_2, \ldots, x_n]^T$ containing the transposed node vectors as rows is the data matrix associated to the graph kernel.

The Euclidean space in which the node vectors are defined is the so-called feature space. A whole bundle of different algorithms can be used to visualize the nodes in a low-dimensional space [48] 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 centre of gravity of the node vectors \( \mathbf{x}_i \) is 0. For instance, most multivariate statistical techniques assume that the data has been centered on the center of gravity of the data cloud [36, 51]. This amounts to subtracting its mean to each feature of the data matrix, therefore normalizing it. A data matrix is centered if \( \mathbf{X}^T e = 0 \), that is, the sum of the elements of each column of \( \mathbf{X} \) is 0. Now, 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 \( \mathbf{K} e = 0 \).

A centered kernel matrix \( \mathbf{K} \) is therefore defined by centering the node vectors, that is, by applying the symmetric centering matrix \( \mathbf{H} = \mathbf{I} - \frac{1}{n} \mathbf{e} \mathbf{e}^T \) [51] to the data matrix \( \mathbf{X} \). Indeed, premultiplying a data matrix by \( \mathbf{H} \) re-expresses each element of the matrix as a deviation from its column mean, i.e., \( \mathbf{X} = \mathbf{HX} \) has its \( (i, j) \)th element \( x_{ij} - x_j \), where \( x_j \) is the mean of the \( j \)th column of \( \mathbf{X} \). Thus, the centered kernel matrix \( \mathbf{K} \) is defined as \( \mathbf{K} = (\mathbf{H} \mathbf{X})(\mathbf{H} \mathbf{X})^T = \mathbf{HXX}^T \mathbf{H} = \mathbf{HKH} \).

2.2.3 The cosine, or normalized, kernel matrix

Inner product scores are not always an appropriate similarity measure. In some fields, such as information retrieval [6], the cosine similarity measure is preferred:

\[
\cos_{ij} = \frac{\mathbf{x}_i^T \mathbf{x}_j}{||\mathbf{x}_i|| ||\mathbf{x}_j||} = \frac{k_{ij}}{\sqrt{k_{ii}k_{jj}}} \tag{1}
\]

It is a kernel matrix since its elements are the inner products of the normalized node vectors \( \mathbf{x}_i/||\mathbf{x}_i|| \) [73].

In matrix form, \( \mathbf{Cos} = \text{Diag}(\mathbf{K})^{-1/2} \mathbf{K} \text{Diag}(\mathbf{K})^{-1/2} \), where \( \text{Diag}(\mathbf{K}) \) 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 any pair 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:

\[
\delta_{ij}^2 = ||\mathbf{x}_i - \mathbf{x}_j||^2 = (\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j) \\
= \mathbf{x}_i^T \mathbf{x}_i + \mathbf{x}_j^T \mathbf{x}_j - 2\mathbf{x}_i^T \mathbf{x}_j = k_{ii} + k_{jj} - 2k_{ij} \\
= (\mathbf{e}_i - \mathbf{e}_j)^T \mathbf{K} (\mathbf{e}_i - \mathbf{e}_j) \tag{2}
\]

Since \( \delta_{ij} \) corresponds to a Euclidean distance in \( \mathbb{R}^r \), it satisfies all the properties of a distance (positiveness, triangular inequality, etc.). Distances between pairs of elements allow, for instance, to use a clustering algorithm to group the nodes that are most similar [82, 84].

In matrix form, \( \Delta = \text{diag}(\mathbf{K}) e^T + e(\text{diag}(\mathbf{K}))^T - 2\mathbf{K} \), where \( \text{diag}(\mathbf{K}) \) is a column vector containing the diagonal elements of \( \mathbf{K} \), and \( [\Delta]_{ij} = \delta_{ij}^2 \) contains the squared distances [7, 20, 51]. As for notations, the name of column vectors is in bold lowercase while that of matrices is in bold uppercase.

With the cosine similarity measure, the distance reduces to \( \delta_{ij} = \sqrt{2(1 - \cos_{ij})} \). The other way around, given a square Euclidean distance matrix \( \Delta \), the natural centered kernel matrix associated to \( \Delta \) is \( \mathbf{K} = -1/2 \mathbf{H} \Delta \mathbf{H} \) [7, 20, 51].
2.2.5 The principal-component analysis of the kernel matrix

Once a data matrix $X$ has been derived from $K$, standard multivariate statistical-analysis methods can be applied to investigate the structure of the data cloud. For instance, principal-component analysis can provide a low-dimension view of the data, keeping as much variance as possible in terms of the distance induced by the kernel. It is well-known that the principal-component analysis of a centered data matrix $X$ yields, as $k^{th}$ principal axis, the eigenvector $v_k$ of $X^T X$ (which is the variance-covariance matrix since the $x_i$’s are centered). In [27, 67], it is shown that the node vectors $x_i$ are expressed in the principal-component coordinate system. Although this is not investigated in this paper, these node vectors can thus be used as such in order to visualize the graph. This is a consequence of classical multidimensional scaling [7, 20, 51]. 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 [71, 73].

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 is also called relatedness [79] in the literature and the most well-known quantities measuring relatedness are co-citation [75] and bibliographic coupling [41]. These are the two baseline measures that are adopted in our collaborative-recommendation experiments.

More sophisticated measures were suggested as well. For instance, a similarity measure between nodes of a graph integrating indirect paths, based on the matrix-forest theorem, was proposed in [13, 15]. While some nice properties were proven about this similarity measure, no experiment investigating its effectiveness was reported. We therefore investigated this matrix-forest based measure (which actually is the same measure as the regularized Laplacian kernel described in Section 4.4).

A modified regularized Laplacian kernel was proposed in [34] as an extension of the regularized Laplacian kernel, by introducing a new parameter controlling importance and relatedness. Moreover, in [34, 74], it was shown that the regularized Laplacian kernel overcomes some limitations of the von Neumann kernel [37], when ranking linked documents. This modified regularized Laplacian kernel is also closely related to a graph-regularization framework introduced by Zhou & Schölkopf in [87] and extended to directed graphs in [86]. The resulting graph kernel is a normalized version of the “regularized Laplacian kernel” involving the extension of the Laplacian matrix for directed graphs [18]. In the same spirit, Chen et al [16] 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 [37, 73]. They are computed through a power series of the adjacency matrix of the graph; they are therefore closely related to graph regularization models [44]. Indeed, Kondor & Lafferty [44], as well as Smola & Kondor [76], introduced a general method for constructing natural families of kernels over discrete structures, based on the matrix-exponentiation idea. 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).
The “commute-time” kernel was introduced in [27, 67]; it was inspired by the work of [12, 42]. It takes its name from the average commute time, which 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 then going back 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. This kernel performs well in collaborative recommendation, as shown in [27]. At the same time, Qiu & Hancock [61, 62], Ham, Lee, Mika & Schölkopf [31], Yen et al [84] as well as Brand [8] defined the same commute-time embedding, preserving the commute-time distance, and applied it to image segmentation and multi-body motion tracking [61, 62], to dimensionality reduction of manifolds [31], to clustering [84], and to collaborative filtering [8, 27], with interesting results. The commute-time kernel is also closely related to the “Fiedler vector” [25, 52], widely used for graph partitioning [11, 60] and clustering [22], as detailed in [27], but also to discrete Green functions [21]. Finally, a family of dissimilarity measure reducing to the shortest-path distance at one end and to the commute-time (or resistance) distance at the other end was proposed in [83].

Recent papers [47, 54, 55, 59] proposed an intuitively appealing distance measure between nodes of a graph 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” [28]. An application of the diffusion distance to dimensionality reduction and graph visualization was described in [45]. The natural embedding induced by the diffusion distance was called the “diffusion map” by Nadler et al [54, 55].

Other recent papers attempted to adapt the well-known PageRank procedure [9, 57] to define meaningful similarities between nodes. A random-walk with a restart procedure was proposed in [58, 78]. A random-walk process starting from a node of interest, controled by some precomputed correlation matrix between nodes, was described in [29]. A random-walk-with-restart kernel is presented in Section 4.8 and tested with the other kernels in Sections 6 and 7.

Recently, Yajima et al [81] proposed a kernel approach for recommendation tasks based on one-class support-vector machines with graph kernels generated from a Laplacian matrix. Preliminary experiments on the MovieLens dataset produced interesting results with the commute-time kernel, the regularized commute-time kernel, and the regularized Laplacian kernel.

Several attempts to define similarity measures for regular directed graphs were proposed as well. For instance Chung [18] proposed to start from the transition-probability matrix \( P = D^{-1}A \) where \( D \) is, as before, a diagonal matrix containing the outdegrees of the graph nodes. Then, they compute the left eigenvector of \( P \), \( \pi^T P = \lambda \pi \), which corresponds to the stationary distribution of the Markov chain. If the diagonal matrix containing the elements of \( \pi \) is denoted by \( \Pi = \text{Diag}(\pi) \), the Laplacian matrix of the directed graph is then defined by

\[
L = \Pi - \frac{\Pi P + P^T \Pi}{2}
\]  

(3)

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 defines a kernel matrix.

The normalized Laplacian matrix can be defined as \( \tilde{L} = \Pi^{-\frac{1}{2}} L \Pi^{-\frac{1}{2}} \). Zhou et al [86] used the regularized normalized Laplacian kernel matrix \( (I - \alpha \tilde{L})^{-1} \) in the context of semisupervised classification of labeled nodes of a graph while Chen et al
used the regularized Laplacian kernel matrix for directed graph embedding. Based on the same idea, Zhao et al [85] 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.

Yet another attempt to define similarities between nodes on a directed graph based on the matrix-forest theorem was described in [3, 4].

4 Nine kernels on a graph

This section describes in more detail the nine kernel-based algorithms, introduced in Section 3, that are investigated in this paper. Notice that several derived measures (a distance measure, the cosine kernel, the principal components) are also computed from these kernels, and compared on the collaborative-recommendation task in Section 6.

4.1 The exponential diffusion kernel

The so-called exponential diffusion kernel \( K_{ED} \), introduced by Kondor & Lafferty [44], is defined as

\[
K_{ED} = \sum_{k=0}^{\infty} \frac{\alpha^k A^k}{k!} = \exp(\alpha A)
\]

where \( A \) is the adjacency matrix of the graph and \( \exp \) is the matrix exponential. Element \( a_{ij}^k = (A^k)_{ij} \) of matrix \( A^k \) (\( A \) to the power \( k \)) is the number of paths (assuming that direct weights \( a_{ij} \) are interpreted as the number of direct links between the two nodes \( i \) and \( j \)) between node \( i \) and node \( j \) with exactly \( k \) transitions or steps. Thus the kernel integrates a contribution from all paths connecting node \( i \) and node \( j \), discounting paths according to their number of steps. It favors shorter (in terms of number of steps) 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 \( K_{ED} \) matrix is clearly positive semidefinite since the exponential of \( A \) amounts to replacing \( A \) (a diagonal matrix containing the eigenvalues of \( A \)) by \( \exp(A) \) in the spectral decomposition of \( A \) (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 \( K_{ED} \) is a diffusion model [40, 76] that substitutes the adjacency matrix with minus the Laplacian matrix in Equation (4). As an intuitive insight into the model, suppose that a quantity \( x_i \) is defined on each node \( i \) of the graph and that it diffuses to neighboring nodes with a symmetric diffusion rate \( a_{ij} \). Thus, during a small time interval \( \delta t \), an amount \( a_{ij} x_i \delta t \) is transferred from node \( i \) to node \( j \), proportional to both the time interval \( \delta t \), the weight \( a_{ij} \), and to the quantity \( x_i(t) \) present
at node $i$ at time $t$. The balance equation is given by

$$x_i(t + \delta t) = x_i(t) + \sum_{j=1}^{n} a_{ji} x_j \delta t - \sum_{j=1}^{n} a_{ij} x_i \delta t \quad (5)$$

Since $A$ is symmetric, this last equation leads to, for $\delta t \to 0$,

$$\frac{dx_i(t)}{dt} = \sum_{j=1}^{n} (a_{ji} x_j - a_{ij} \delta_{ij} x_j) = - \sum_{j=1}^{n} (a_{ij} \delta_{ij} - a_{ij}) x_j \quad (6)$$

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

$$\frac{dx(t)}{dt} = -L x(t) \quad (7)$$

This system of differential equations leads to the following solution

$$x(t) = \exp(-Lt) x_0 \quad (8)$$

where $x_0$ is the initial vector $x$ at time $t = 0$ and $\exp$ is the matrix exponential. This leads to the Laplacian exponential diffusion kernel, introduced in [44, 76], and defined as

$$K_{LED} = \exp(-\alpha L) \quad (9)$$

which is similar to Equation (4), except that it involves the Laplacian matrix as basis matrix instead of the adjacency matrix. Equation (8) shows that column $i$ of $K_{LED}$ corresponds to the quantity $x$ observed at time $t = \alpha$ when the initial vector is $x_0 = e_i = [0, \ldots, 0, 1, 0, \ldots, 0]^T$.

4.3 The von Neumann diffusion kernel

The von Neumann diffusion kernel ($K_{VND}$) [37, 73] differs from the exponential diffusion kernel by the discounting scheme. The von Neumann diffusion kernel has an exponential discounting rate $\alpha^k$:

$$K_{VND} = \sum_{k=0}^{\infty} \alpha^k A^k = (I - \alpha A)^{-1} \quad (10)$$

$K_{VND}$ is well-defined only if $0 < \alpha < ||A||_2^{-1}$ where $||A||_2$ is the spectral radius of the symmetric matrix $A$. In this case, $K_{VND}$ is positive definite. It was shown in [34] that this kernel applied to document ranking defines a link-analysis measure intermediate between co-citation/bibliographic coupling [41, 75] and HITS [43].
4.4 The regularized Laplacian kernel

The regularized Laplacian kernel [34, 76] differs from the von Neumann diffusion kernel only by substituting the negated Laplacian matrix \(-L\) for the adjacency matrix \(A\):

\[
K_L = \sum_{k=0}^{\infty} \alpha^k (-L)^k = (I + \alpha L)^{-1}
\]

where \(0 < \alpha < ||L||_2^{-1}\) where \(||L||_2\) is the spectral radius of \(L\). \(K_L\) is positive semidefinite. The regularized Laplacian kernel has been shown to provide better performance than the von Neumann kernel in a bibliographic citation task [34].

This similarity measure has an interesting interpretation in terms of the matrix-forest theorem [13, 15]. Let \(F^j\) be the set of all spanning forests rooted at node \(i\) of graph \(G\) and \(F^{ij}\) 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 [13, 15] that the matrix \((I + L)^{-1}\) exists and that \([[(I + L)^{-1}]_{ij}] = \epsilon(F^{ij})/\epsilon(F^i)\) where \(\epsilon(F^{ij})\) and \(\epsilon(F^i)\) are the total weights of forests that belong to \(F^{ij}\) and \(F^i\) respectively. The elements of this matrix are therefore called “relative forest accessibilities” between nodes. This interpretation can be generalized to the matrix \((I + \alpha L)^{-1}\) with a parameter \(\alpha\) weighting the number of edges belonging to the forests as well as limiting the size of forests (in terms of number of edges; see [13, 15] for more detail); it can be shown that this matrix is a similarity measure.

A modified regularized Laplacian kernel was proposed in [34], by introducing a new parameter controlling importance and relatedness. A modified Laplacian matrix is defined as \(L_\gamma = \gamma D - A\) with \(0 < \gamma \leq 1\). The modified regularized Laplacian kernel \(K_{RL}\) is computed as

\[
K_{RL} = \sum_{k=0}^{\infty} \alpha^k (-L_\gamma)^k = (I + \alpha L_\gamma)^{-1}
\]

It was shown that, for \(\alpha > 0\) and \(0 < \gamma \leq 1\), if the series (12) converges, \(K_{RL}\) is positive semidefinite and that it also yields a measure intermediate between relatedness and importance [34].

4.5 The commute-time kernel

Consider a discrete Markov chain 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 us denote the probability of being in state \(i\) at time \(t\) by \(x_i(t) = P(s(t) = i)\) and define \(P\) as the transition-probability matrix with entries \(p_{ij} = [P]_{ij} = P(s(t + 1) = j|s(t) = i)\), where \([P]_{ij}\) is the \(i, j\) element of matrix \(P\). We consider a Markov model for which the transition probabilities are provided by \(p_{ij} = a_{ij}/a_i\) with \(a_i = \sum_{j=1}^{n} a_{ij}\). In other words, to any state or node \(s(t) = i\), we associate a probability of jumping to an adjacent node \(s(t + 1) = j\) that is proportional to the weight \(a_{ij} \geq 0\) of the edge connecting \(i\) and \(j\) (this corresponds to a standard random-walk model on a graph). The transition probabilities depend only on the current state and not on the past ones (first-order Markov chain). Since the graph is undirected and connected, the Markov chain is irreducible, that is, every state can be reached from any other state.
In matrix form, the evolution of the Markov chain is characterized by \( \mathbf{x}(t + 1) = \mathbf{P}^T \mathbf{x}(t) \), which provides the state probability distribution \( \mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^T \) at time \( t \) once the initial probability density \( \mathbf{x}(0) = \mathbf{x}^0 \) at \( t = 0 \) is known. It is clear that after \( t \) steps, the probability distribution will be given by \( \mathbf{x}(t) = (\mathbf{P}^T)^t \mathbf{x}^0 \).

The commute-time kernel \([27, 67]\) takes its name from the \textbf{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 can be computed as \([27, 67]\):

\[
    n(i, j) = V_G (\mathbf{e}_i - \mathbf{e}_j)^T \mathbf{L}^+ (\mathbf{e}_i - \mathbf{e}_j) \tag{13}
\]

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, Equation (13) is a Mahalanobis distance between the nodes of the graph and is referred to as the “commute-time distance” or the “resistance distance” because of a close analogy with the effective resistance in electrical networks \([27]\).

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. In other words, the elements of \( \mathbf{L}^+ \) can be viewed as similarity measures between nodes. Hence the \textbf{commute-time (\( K_{CT} \) kernel)} is defined as

\[
    K_{CT} = \mathbf{L}^+ \tag{14}
\]

with no parameter tuning necessary.

There is of course a close relationship between the commute-time kernel and the regularized Laplacian kernel. Indeed, if \( \mathbf{L} \) has eigenvalues \( \lambda_i \) (in decreasing order), \((\mathbf{L} + \alpha^{-1} \mathbf{I})\) has corresponding eigenvalues \((\lambda_i + \alpha^{-1})\) and both matrices share the same eigenvectors \( \mathbf{u}_i \). Thus, a spectral decomposition of the corresponding kernels yields \( \mathbf{L}^+ = \sum_{i=1}^{n-1} \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^T \) (by definition of the Moore-Penrose pseudoinverse) and \((\mathbf{I} + \alpha \mathbf{L})^{-1} = \alpha^{-1} \sum_{i=1}^{n-1} (\lambda_i + \alpha^{-1})^{-1} \mathbf{u}_i \mathbf{u}_i^T \). Since the normalized eigenvector of \( \mathbf{L} \) corresponding to the last eigenvalue \( \lambda_n = 0 \) is \( \mathbf{u}_n = \mathbf{e}/\sqrt{n} \), the \( n \)th term for the commute-time kernel is simply 0, while it is \( \mathbf{e} \mathbf{e}^T/n \) 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 \( \alpha \) is large, \( \lambda_i + \alpha^{-1} \) will be close to \( \lambda_i \), 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 graph kernel, introduced in \([28]\), is based on a diffusion distance defined in \([47, 54, 55, 59]\) between nodes of a graph in a continuous-time diffusion model. We adapted their definition of diffusion distance to discrete-time processes and to periodic Markov chains (as nonperiodic Markov chains are unrealistic for collaborative recommendation) to define a valid kernel on a graph \([28]\). It was shown in \([54, 55]\) that the low-dimension representation of the data by the first few eigenvectors of the corresponding Markov matrix is optimal under a given mean-square error criterion involving the diffusion distance. An application of the diffusion distance to dimensionality reduction and graph visualization was described in \([45]\).
The average visiting rate \( \pi_{ik}(t) \) in state \( k \) after \( t \) steps for a process that started in state \( i \) at time \( t = 0 \) is computed as follows:

\[
\pi_{ik}(t) = \frac{1}{t} \sum_{\tau=1}^{t} P(s(\tau) = k | s(0) = i)
\]  

(15)

From this quantity, we define the diffusion distance at time \( t \) between node \( i \) and node \( j \) as

\[
d_{ij}(t) = \sum_{k=1}^{n} (\pi_{ik}(t) - \pi_{jk}(t))^2
\]

(16)

which corresponds to the sum of the squared differences between the average visiting rates at node \( k \) 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 the graph in the same way, the distance is zero. Of course, when \( i = j \), \( d_{ij}(t) = 0 \).

This diffusion distance was proposed by Nadler et al [54] and by Latapy and Pons [47] in the context of diffusion processes. Actually, their definition, namely

\[
d_{ij}(t) = \sum_{k=1}^{n} (x_{ik}(t) - x_{jk}(t))^2
\]

involves \( x_{ik}(t) \), the probability of finding the random walker in state \( k \) at time \( t \), when starting in node \( i \) at time \( 0 \) (i.e., \( x(0) = e_i \)) instead of \( \pi_{ik}(t) \) as in Equation (16). Their definition does not work with periodic Markov chains, that are relevant to collaborative recommendation (with a bipartite graph). The definition of [47, 54, 55, 59] 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 because experiments showed that it does not improve the performance for the applications studied in Sections 6, 7.

We first compute \( \pi_{ik}(t) \) from Equation (15):

\[
\pi_{ik}(t) = \frac{1}{t} \sum_{\tau=1}^{t} P(s(\tau) = k | s(0) = i)
\]

(17)

\[
= \frac{1}{t} \sum_{\tau=1}^{t} e_k^T (P^T)^\tau e_i = e_k^T \left[ \frac{1}{t} \sum_{\tau=1}^{t} P^\tau \right]^T e_i
\]

(18)

since \( P(s(\tau) = k | s(0) = i) = e_k^T x(t) = e_k^T (P^T)^\tau e_i \) since \( x(0) = e_i \).

By defining \( Z(t) = \frac{1}{t} \sum_{\tau=1}^{t} P^\tau \), we obtain

\[
\pi_{ik}(t) = e_k^T Z(t) e_i
\]

(19)

We now turn to the evaluation of the Markov diffusion distance (Equation (16)):

\[
d_{ij}(t) = \sum_{k=1}^{n} (\pi_{ik}(t) - \pi_{jk}(t))^2
\]

(20)

\[
= \sum_{k=1}^{n} (e_i^T Z(t) e_i - e_j^T Z(t) e_j)^2
\]

(21)

\[
= ||Z^T(t)(e_i - e_j)||^2
\]

(22)

\[
= (e_i - e_j)^T Z(t) Z^T(t)(e_i - e_j)
\]

(23)
which has exactly the same form as Equation (2). We immediately deduce the form of the Markov diffusion ($K_{MD}$) kernel

$$K_{MD}(t) = Z(t)Z^T(t) \text{ with } Z(t) = \frac{1}{t} \sum_{\tau=1}^{t} P^\tau$$  \hspace{1cm} (24)

Notice that in order to evaluate $Z(t)$, we use a trick similar to the one used in PageRank [46]: 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 aims to subtract some small quantity from every element of $P$, with the result that the matrix $P$ is now substochastic and that $Z(t)$ admits the following analytical form,

$$Z(t) = \frac{1}{t}(I - P)^{-1}(I - P^t)P$$  \hspace{1cm} (25)

### 4.7 The cross-entropy diffusion matrix

Instead of using a least-square distance for comparing two probability distributions as in Equation (16), it is more appropriate to use the symmetric cross-entropy divergence [19, 38], as computed by

$$d_{ij}(t) = \sum_{k=1}^{n} \left[ \tau_{ik}(t) \log \frac{\tau_{ik}(t)}{\tau_{jk}(t)} + \tau_{jk}(t) \log \frac{\tau_{jk}(t)}{\tau_{ik}(t)} \right]$$  \hspace{1cm} (26)

This leads to the cross-entropy diffusion ($K_{CED}$) matrix (since the distance cancels out the asymmetric part of the matrix, the kernel matrix can be symmetrized without change):

$$K_{CED}(t) = Z(t) \log(Z^T(t)) + \log(Z(t))Z^T(t) \text{ with } Z(t) = \frac{1}{t} \sum_{\tau=1}^{t} P^\tau$$  \hspace{1cm} (27)

where the element-wise logarithm is taken on each element of the matrix. We did not prove the positive semidefinitiveness of this similarity matrix; it therefore cannot be considered as a kernel.

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

Pan et al [58] (see also [77, 78]) recently introduced a similarity matrix between nodes inspired by the well-known PageRank algorithm [9, 57]. This model has been applied to various interesting applications, including center-piece subgraph discovery and content-based image retrieval [58, 77, 78]. The same idea was introduced also in [29, 30] in the context of collaborative recommendation.

Like the diffusion kernel, the model considers a random walker jumping from some node $i$ and to some neighbor node $j$ with a probability proportional (apart from normalization) to the edge weight $p_{ij} = P(s(t+1) = j | s(t) = i) = a_{ij}/a_{i\cdot}$. In addition, at each step of the random walk, the random walker has some probability $(1 - \alpha)$ to return 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

\[
\begin{cases}
x(0) = e_i \\
x(t+1) = \alpha P^T x(t) + (1 - \alpha) e_i
\end{cases}
\]  \hspace{1cm} (28)
where $e_i$ is a column vector with “0” entries, except in position $i$ whose entry is “1”.

Considering the steady-state solution $x(t + 1) = x(t) = x$ and extracting the probability distribution $x$ of finding the random walker on each node when starting from node $i$ yields

$$x = (1 - \alpha) (I - \alpha P^T)^{-1} e_i$$

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

Now, since $x$ is column $i$ of matrix $(I - \alpha P^T)^{-1}$, the random-walk-with-restart ($K_{RWR}$) matrix is defined as the matrix whose $i$'th row contains the similarities to node $i$ (the matrix is transposed):

$$K_{RWR} = (I - \alpha P)^{-1}$$

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

$$K_{RWR} = (D^{-1}(D - \alpha A))^{-1} = (D - \alpha A)^{-1} D$$

since $P = D^{-1} A$.

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 could be applied as in Section 4.4. One such regularization leads to what we call the regularized commute-time ($K_{RCT}$) kernel

$$K_{RCT} = (D - \alpha A)^{-1}$$

with $\alpha \in [0, 1]$. Since $K_{RCT}$ is the matrix inverse of the sum $((1 - \alpha)D + \alpha L)$ of a positive definite matrix and a positive semidefinite matrix, it is positive definite and it is a valid kernel.

Another justification of the interest of this graph kernel is as follows. Consider the following random-walk model starting at node $i$

$$\begin{cases} x(0) = e_i \\ x(t + 1) = P^T x(t) \end{cases}$$

The column vector $x(t)$ 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:

$$\text{sim}_{i} = \sum_{\tau=0}^{\infty} \alpha^\tau D^{-1} x(\tau)$$

with $\alpha \in [0, 1]$. The weighting factor $D^{-1}$ compensates for the fact that $x(t)$ converges to the stationary distribution, which is proportional to the diagonal elements of $D$ with undirected graphs [65]. A similarity measure based on $x(t)$ only (without weighting
factor) would therefore favor the nodes with a high outdegree $|D|_{i\cdot}$. Equation (34) thus accumulates, with a damping factor $\alpha$ (late visits are less important than early visits), the probability of visiting each node when starting from node $i$.

We easily find

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

$$= D^{-1} \sum_{\tau=0}^{\infty} \alpha^\tau (P^T)^\tau e_i$$

$$= D^{-1} (I - \alpha P^T)^{-1} e_i$$

$$= D^{-1} ((D - \alpha A) D^{-1})^{-1} e_i$$

$$= (D - \alpha A)^{-1} e_i$$

$$= \text{col}_i (K_{RCT})$$

where we used $P = D^{-1} A$ and the fact that $D$ is diagonal and $A$ is symmetric.

Thus, the $i,j$ element of $K_{RCT}$ can be interpreted as the discounted cumulated probability of visiting node $j$ when starting from node $i$. The regularized commute-time kernel (Equation (32)) differs from the random-walk-with-restart similarity matrix (Equation (31)) only in the fact that the latter post-multiplies the former by $D$: $K_{RCT}$ is not a kernel since it is not symmetric and therefore not positive semidefinite.

5 Testing the kernels

5.1 Introduction of the two case studies

5.1.1 Collaborative recommendation

Recommender systems aim at providing their users with recommendations for items that they will appreciate, based on past preferences, history of purchases, 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 ranked movies for people to watch.

The idea originates (see the state-of-the-art survey of [2] for more detail) mainly from research on machine learning, information retrieval [68], cognitive science [64], forecasting theories [5], marketing [49], and management [53]. Recommender systems emerged as an independent research area in the mid-1990s, with the first papers on collaborative filtering [33, 63, 72].

The nine kernel-based algorithms presented in Section 4 were tested on a collaborative-recommendation task (see Section 6) with two concrete datasets: the MovieLens dataset and the Book-Crossing dataset.

A weighted, undirected, graph $G$ is associated with the database in the following way: database elements correspond to nodes of the graph and database links correspond to edges. In the movie database, this means that each element of the people and movie (or, more generally, item) sets corresponds to a node of the (bipartite) graph, and each has watched link is expressed as an edge connecting the corresponding nodes. For both collaborative-recommendation datasets, the weight of an edge between two nodes
is set to 1 if there is a link between the corresponding elements in the database and to 0 otherwise.

5.1.2 Semisupervised classification

In a practical classification setting, labeled data is typically hard to obtain, since it often requires expensive human labor. On the other hand, unlabeled data is often 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 unlabeled data, in addition to labeled, to increase classification accuracy. The dataset is a graph where some nodes are labeled and some are not.

In the present case, a graph with labels associated to some nodes is available while the other nodes are unlabeled. The graph topology, along with the available node labels, provides useful information for classifying the unlabeled nodes. The main objective is to infer labels for the unlabeled nodes from the labeled ones [50].

Since a small proportion of labeled nodes is sometimes sufficient to determine unknown labels, several values of the labeling rate (i.e., the proportion of nodes whose label is known) were considered. In order to assess the ability of the algorithms to determine the correct labels of the nodes, some labels were removed from the original dataset and the resulting data was used as a test set (the data not belonging to this test set constitutes 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.

Section 7 describes the application of the nine kernel-based algorithms to semisupervised classification on several partially labeled graphs (NewsGroup, Cora, WebKB, IMDb). In total, seven datasets for semisupervised classification [10, 50] were used. For each dataset, a local consistency of node labeling was assumed, i.e., nodes within a neighborhood are likely to share the same label.

5.2 Kernel-based scoring algorithms

Table 1 lists the kernel-based algorithms. The “Equation” column refers to the equation used to compute each kernel. Most algorithms need a value for some parameters (given in the “Parameter” column). Those values (see the “Tested values” column) were tuned following a nested (also called double) cross-validation procedure detailed in Section 6.3 for recommendation and in Section 7.2 for classification.

Other, more standard, algorithms are also tested for comparison; these algorithms are described in Section 6.4 for recommendation and in Section 7.3 for classification. All these algorithms, both kernel-based and standard, are referred to as the scoring algorithms.

5.3 Computational issues

Computing graph kernels relies on efficiently inverting matrices, with computation times that can be significant for large graphs. Also, inverse matrices are often dense and may not fit in main memory. Cholesky factorizations address those issues (see, e.g., [27]). Incomplete (lower-rank) Cholesky matrix factorizations (see for instance [26]) 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.

6 Testing the kernels on collaborative recommendation

6.1 Datasets

The MovieLens dataset (ML) was built from the web-based recommender system MovieLens1. Visitors of that web site are invited to rate movies that they watched and to ask for recommendations about movies that they should watch given the preferences that they express. We selected a sample of that database as suggested in [69]. Enough users (i.e., 943 users or persons) were randomly selected to obtain 100,000 ratings, retaining only users that had rated at least 20 movies on a total of 1,682 movies. The resulting user-movie matrix contains about 6.3% of “1” values and 93.7% of “0” values.

The Book-Crossing dataset (BC) used in this paper was collected in 20042 from the Book-Crossing community3. It contains 278,858 users providing 1,149,780 ratings about 271,379 books. We selected from it a sample of 2,222 books, 1,028 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-movie 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 values of ratings provided in the datasets (that is, we retained only the fact that a person did or did not rate an item). Indeed, very 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.

6.2 Direct and indirect recommendation

We used three methods to apply the kernels. We call them the direct method, the user-based indirect method, and the item-based indirect method. Both indirect methods are

---

1See http://www.movielens.umn.edu
2See http://www.informatik.uni-freiburg.de/∼cziegler/BX/
3See http://www.bookcrossing.com
based on a nearest-neighbor technique that requires a measure of closeness or similarity. Important considerations involved in choosing a similarity measure [35] include the nature of variables (discrete, continuous, binary), the scale of measurements (nominal, ordinal, interval, 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 [23]. 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 $\text{sim}(p_0, i)$ with each of the scoring algorithms (for instance provided by the element of the kernel matrix) 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 $\text{sim}(p_0, p)$ between user $p_0$ and users $p$. Then, it computes from the $k$ nearest neighbor users of $p_0$ the predicted value (or preference) of each item. The predicted value of item $i_0$ for user $p_0$ is the sum, weighted by $\text{sim}(p_0, p)$, of the values of the link weight (0 or 1) of item $i_0$ for the $k$ users $p$ closest to $p_0$:

$$\text{pred}(p_0, i_0) = \frac{\sum_{p=1}^{k} \text{sim}(p_0, p) \; w_{pi_0}}{\sum_{p=1}^{k} \text{sim}(p_0, p)} \tag{41}$$

where $w_{pi_0}$ is “1” if user $p$ rated item $i_0$ and “0” otherwise. The higher the predicted value $\text{pred}(p_0, i_0)$, the stronger the recommendation that $p_0$ should 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 [39]. It first applies scoring algorithms to compute similarities $\text{sim}(i_0, i)$ 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 $i_0$, the predicted value of $i_0$ for $p_0$ as a sum, weighted by $\text{sim}(i_0, i)$, of the values of the link weight (“0” or “1”) of $p_0$ and items $i$:

$$\text{pred}(p_0, i_0) = \frac{\sum_{i=1}^{k} \text{sim}(i_0, i) \; w_{pi_0}}{\sum_{i=1}^{k} \text{sim}(i_0, i)} \tag{42}$$
where $w_{pi0}$ is defined as above. The items are suggested to user $p_0$ in decreasing order of predicted values, among those not rated by $p_0$. Both for the user-based and item-based indirect methods, we optimized the number of neighbors thanks to an internal cross-validation loop, according to our accuracy measures described in Section 6.3.

For each of the nine kernel-based algorithms, we also computed four derived measures (see Section 2.2): (1) the centered kernel matrix, (2) the cosine kernel matrix, (3) the centered cosine kernel matrix, and (4) the distance measure. Each measure was systematically investigated exactly like the original kernel. Notice that, for indirect methods, there are two ways of working with the original kernel matrix: (1) apply the transformation to the whole kernel matrix (centering, etc.), 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 (centering, etc.) on this submatrix, and proceed from there.

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 $k$ (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 $i^{th}$ 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 Equations (41) and (42)). The parameter $k$ is tuned by using an internal cross-validation (see Section 6.3 below).

6.3 Performance evaluation

The performance of the various scoring algorithms was compared by applying a standard nested (or double) cross-validation. Tuning parameters (number of neighbors – see Section 6.2, kernels-related parameters – see Section 5.2) was performed in an internal nine-fold cross-validation and performance was averaged on an external ten-fold cross-validation.

More precisely, each dataset was divided in ten subsets. Each subset was in turn used for performance estimation in the external cross-validation (it will be referred to as the external test set). The nine remaining subsets, referred to as external training sets, were 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 averaged 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.

Thus, each scoring algorithm provides, for each person, a set of proximity measures indicating preferences about the items. 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.

To evaluate the accuracy of the scoring algorithms, we compared their performance
by applying a recall measure (see [32]). For each user, the test set (either external or internal) contains a set of items that the user actually rated and that were removed from the training set, i.e., these links were deleted from the original graph. 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 – and thus bought – by the user. The recall accuracy measure (called “recall score” in the sequel) was used to compare the ranking provided by a particular scoring 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 proportion (in percents) 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 1, 682 items in the ML dataset and 2, 222 books in the BC dataset).

Notice that 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.

6.4 Standard (non kernel-based) scoring algorithms

The kernel-based scoring algorithms investigated in this paper were compared to three more standard techniques: the baseline (Basic) algorithm, the standard binary algorithm (Bin), and the cosine algorithm (Cos). The last two algorithms were chosen for comparison since they provide the best results among a number of standard methods [27]: Katz’ method, shortest-path algorithm, maximum flow algorithm, correspondence analysis, and latent-class algorithm (the last one, although having provided good results in previous work, was not included in this paper because of severe computational issues). The details about these methods as well as their results obtained on the MovieLens dataset are provided in [27].

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

1. In its direct form (see Section 6.2), 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” \( WW^T \) of bibliometry [34, 41] where \( W \) is the user-item adjacency matrix. Two users are similar if the corresponding element of the \( WW^T \) matrix is high, i.e., if there are many items that have been rated by both users. From these similarity measures \( \text{sim}(p_i, p_j) = \text{element } i,j \text{ of matrix } WW^T \), a ranked list of items is then computed by applying Equation (41).
3. Symmetrically, in its item-based indirect form, the item-item similarity matrix to recommend items is the “co-citation matrix” \( W^T W \) where \( W \) is, again, the user-item adjacency matrix. Two items are similar if the corresponding element of the \( W^T W \) matrix is high, i.e., if there are many users who have rated both items. From these similarity measures \( \text{sim}(i_k, i_l) = \text{element } k,l \) of matrix \( W^T W \), a ranked list of items is then computed by applying Equation (42).

**Binary algorithm (Bin).** Pairs of items are compared on the basis of the presence or absence of certain features (e.g., watching a particular movie or buying 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 6.2); 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 \( v_i \) (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 \( v_j \) according to some binary similarity measure \( \text{sim}(i, j) = s(v_i, v_j) \). In [27], we performed systematic comparisons between eight such measures \( s \) (listed in [35], p. 674) for different values of \( k \) (= 10, 20, ..., 100). The best recall scores were obtained with the “a 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.

**Cosine algorithm (Cos).** The cosine coefficient between variables \( i \) and \( j \), which measures the strength and the direction of a linear relationship between two variables, is defined as \( \text{sim}(i, j) = \cos(i, j) = (v_i^T v_j) / (\|v_i\| \|v_j\|) \) [24]. In our case, 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 \( v_i \)), 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 cannot be used in the direct method and that can only be used with bipartite graphs.

### 6.5 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 the proposed rankings with the test set by using the recall performance measure described in Section 6.3.

All 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 twelve scoring algorithms (nine kernels, + Basic, Bin, and Cos). The standard deviation of the results (STD) across the 10 emph external cross-validation sets is also reported. Notice that we used a paired \( t \)-test to determine if there is a significant difference (with a \( p \)-value smaller than 5.10\(^{-2} \)) between the results of the various scoring algorithms. The best results overall, for each measure of performance, are displayed in bold, based on the \( t \)-test.
Table 2: Average results obtained on the MovieLens dataset, by performing a nested cross-validation, for the 12 scoring algorithms and the three methods defined to use them (direct, user-based indirect, and item-based indirect).

<table>
<thead>
<tr>
<th>KRD</th>
<th>KCED</th>
<th>KCMD</th>
<th>KRL</th>
<th>KRC</th>
<th>KCEC</th>
<th>KCE</th>
<th>KRCM</th>
<th>KCEC</th>
<th>Bin</th>
<th>Cos</th>
<th>Basic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.78</td>
<td>0.89</td>
<td>0.90</td>
<td>0.89</td>
<td>0.78</td>
<td>0.77</td>
<td>0.77</td>
<td>0.78</td>
<td>0.77</td>
<td>0.77</td>
<td>0.78</td>
</tr>
<tr>
<td>20</td>
<td>37.84</td>
<td>28.35</td>
<td>29.35</td>
<td>30.22</td>
<td>28.35</td>
<td>28.35</td>
<td>30.22</td>
<td>29.35</td>
<td>28.35</td>
<td>28.35</td>
<td>29.35</td>
</tr>
</tbody>
</table>

MovieLens dataset. Table 2 shows that, when using the direct method to rank the movies for each user, the best results were obtained by K_{MD}. For the user-based indirect method, the best results were obtained by K_{MD} for both recalls (21.40 and 31.98 for recall 10 and 20, respectively), and by K_{CT} (31.72), K_{RL} (31.70), and K_{RCT} (31.60) for recall 20 (differences not significant according to the paired t-tests). For the item-based indirect method, the best results were provided by K_{ED} for recall 10 (21.69) and recall 20 (31.58), followed by K_{MD} and K_{CED}.

The global performance (regardless of whether the direct or indirect method was applied to compute similarities) of the various scoring algorithms, shows (i.e., results in bold) that the best recall 10 was provided by K_{ED} (21.69) and K_{MD} (21.40) while the best recall 20 was provided by five kernel-based scoring algorithms, namely K_{MD} (31.98), K_{CT} (31.72), K_{RL} (31.70), K_{RCT} (31.60), and K_{ED} (31.58). The best recall scores for a standard scoring algorithm were provided by Cos in the user-based indirect method (20.70 for recall 10 and 31.18 for recall 20).

Book-Crossing dataset. Table 3 shows that, when using the direct method for ranking books, the best results were provided by K_{CT} and K_{RL} for both recalls (7.26 and 7.36 respectively for recall 10, and 11.73 and 11.58 respectively for recall 20). For the user-based indirect method, the best results were provided by three kernels (i.e., K_{CT}, K_{RL}, and K_{MD}) for both recalls (7.07, 7.09, and 7.00 respectively for recall 10; 11.39, 11.23, and 11.31 respectively for recall 20). For the item-based indirect method, the best results were provided by two kernels (i.e., K_{CT} and K_{MD}) and a standard scoring algorithm (i.e., Bin) for both recalls (8.84, 8.70, and 8.79 respectively for recall 10; 13.58, 13.35, and 13.50 respectively for recall 20).

The global performance of the various scoring algorithms shows (i.e., results in bold) that the best recall scores were provided by applying the item-based indirect method, for two kernels (i.e., K_{CT} and K_{MD}) and a standard scoring algorithm (i.e., Bin).

Global discussion. Table 4 shows, for each scoring algorithm and each accuracy measure, the best result and the method applied to provide it (i.e., “dir” for the direct method, “user” for the user-based indirect method, and “item” for the item-based in-
### Table 3: Average results obtained on the Book-Crossing dataset, by performing a nested cross-validation, for the 12 scoring algorithms and the three methods defined to use them (direct, user-based indirect, and item-based indirect).

<table>
<thead>
<tr>
<th></th>
<th>$K_{ED}$</th>
<th>$K_{LED}$</th>
<th>$K_{VND}$</th>
<th>$K_{RL}$</th>
<th>$K_{CT}$</th>
<th>$K_{RCT}$</th>
<th>$K_{MD}$</th>
<th>$K_{VDE}$</th>
<th>$K_{Mod}$</th>
<th>$Bin$</th>
<th>$Cos$</th>
<th>Basic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall 10</td>
<td>4.72</td>
<td>9.98</td>
<td>7.51</td>
<td>7.36</td>
<td>7.67</td>
<td>7.46</td>
<td>7.41</td>
<td>6.22</td>
<td>5.74</td>
<td>/</td>
<td>/</td>
<td>5.75</td>
</tr>
<tr>
<td>KLED</td>
<td>12.34</td>
<td>0.16</td>
<td>0.10</td>
<td>0.10</td>
<td>0.10</td>
<td>0.10</td>
<td>0.10</td>
<td>0.10</td>
<td>0.10</td>
<td>/</td>
<td>/</td>
<td>0.10</td>
</tr>
<tr>
<td>Recall 20</td>
<td>10.34</td>
<td>10.33</td>
<td>10.28</td>
<td>11.02</td>
<td>10.92</td>
<td>11.35</td>
<td>11.31</td>
<td>10.74</td>
<td>9.95</td>
<td>10.64</td>
<td>10.24</td>
<td>9.95</td>
</tr>
<tr>
<td>User-based indirect method (in %)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recall 10</td>
<td>6.84</td>
<td>9.95</td>
<td>5.42</td>
<td>7.09</td>
<td>5.79</td>
<td>7.09</td>
<td>5.79</td>
<td>6.36</td>
<td>6.36</td>
<td>6.36</td>
<td>6.36</td>
<td></td>
</tr>
<tr>
<td>Recall 20</td>
<td>10.34</td>
<td>10.33</td>
<td>10.28</td>
<td>11.02</td>
<td>10.92</td>
<td>11.35</td>
<td>11.31</td>
<td>10.74</td>
<td>9.95</td>
<td>10.64</td>
<td>10.24</td>
<td>9.95</td>
</tr>
<tr>
<td>Item-based indirect method (in %)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recall 10</td>
<td>6.84</td>
<td>9.95</td>
<td>5.42</td>
<td>7.09</td>
<td>5.79</td>
<td>7.09</td>
<td>5.79</td>
<td>6.36</td>
<td>6.36</td>
<td>6.36</td>
<td>6.36</td>
<td></td>
</tr>
<tr>
<td>Recall 20</td>
<td>10.34</td>
<td>10.33</td>
<td>10.28</td>
<td>11.02</td>
<td>10.92</td>
<td>11.35</td>
<td>11.31</td>
<td>10.74</td>
<td>9.95</td>
<td>10.64</td>
<td>10.24</td>
<td>9.95</td>
</tr>
</tbody>
</table>

Table 4: Best accuracy results for each scoring algorithm and each method.

<table>
<thead>
<tr>
<th></th>
<th>$K_{ED}$</th>
<th>$K_{LED}$</th>
<th>$K_{VND}$</th>
<th>$K_{RL}$</th>
<th>$K_{CT}$</th>
<th>$K_{RCT}$</th>
<th>$K_{MD}$</th>
<th>$K_{VDE}$</th>
<th>$K_{Mod}$</th>
<th>$Bin$</th>
<th>$Cos$</th>
<th>Basic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall 10</td>
<td>7.96</td>
<td>7.96</td>
<td>7.96</td>
<td>7.96</td>
<td>7.96</td>
<td>7.96</td>
<td>7.96</td>
<td>7.96</td>
<td>7.96</td>
<td>/</td>
<td>/</td>
<td>7.96</td>
</tr>
<tr>
<td>Recall 20</td>
<td>8.96</td>
<td>8.96</td>
<td>8.96</td>
<td>8.96</td>
<td>8.96</td>
<td>8.96</td>
<td>8.96</td>
<td>8.96</td>
<td>8.96</td>
<td>/</td>
<td>/</td>
<td>8.96</td>
</tr>
<tr>
<td>MovieLens dataset</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recall 20</td>
<td>31.58</td>
<td>31.09</td>
<td>29.83</td>
<td>31.70</td>
<td>31.56</td>
<td>31.72</td>
<td>31.98</td>
<td>30.70</td>
<td>30.68</td>
<td>31.16</td>
<td>31.14</td>
<td>30.34</td>
</tr>
<tr>
<td>Book-Crossing dataset</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recall 10</td>
<td>6.45</td>
<td>6.33</td>
<td>6.42</td>
<td>6.40</td>
<td>7.14</td>
<td>8.84</td>
<td>8.70</td>
<td>8.63</td>
<td>8.70</td>
<td>8.03</td>
<td>8.32</td>
<td></td>
</tr>
</tbody>
</table>

**direct method**. Remember that Bin and Cos can only be used in one of the indirect methods.

For the ML dataset, we observe that, with the exception of $K_{ED}$ (for both recalls), $K_{CED}$ (for both recalls), and Bin (for recall 20), the best results (in 19 out of 24 cases) were obtained with the user-based indirect method. For the BC dataset, we observe that, with the exception of $K_{ED}$, $K_{LED}$, $K_{VND}$ (all 3 for both recalls), and $K_{RCT}$ (for recall 20), the best results (in 17 out of 24 cases) were obtained with the item-based indirect method.

This interesting observation suggests that the degree of sparsity of the dataset and/or the structure of the graph has an influence on which indirect method is best suited. Considering sparsity, remember that 6.3% of the potential links are effective in the ML dataset while this rate drops to 4.8% in the BC dataset. Still, admittedly, the evidence is weak to argue for a relationship between sparsity and which indirect method to apply.

Considering the four cases (i.e., best recall 10 for ML dataset, best recall 20 for ML dataset, best recall 10 for BC dataset, and best recall 20 for BC dataset), we observe that $K_{MD}$ provided the best recall scores in the 4 cases, $K_{CT}$ in 3 cases, $K_{ED}$ and Bin in 2 cases, and $K_{RL}$ in 1 case.
### Table 5: Composition of the IMDb dataset.

<table>
<thead>
<tr>
<th>Class</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-revenue</td>
<td>572</td>
</tr>
<tr>
<td>Low-revenue</td>
<td>597</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>1,169</td>
</tr>
<tr>
<td><strong>Majority class proportion</strong></td>
<td>51.1%</td>
</tr>
<tr>
<td><strong>Number of Edges</strong></td>
<td>40,564</td>
</tr>
</tbody>
</table>

### Table 6: Composition of the CORA dataset.

<table>
<thead>
<tr>
<th>Class</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case-based</td>
<td>402</td>
</tr>
<tr>
<td>Genetic algorithms</td>
<td>551</td>
</tr>
<tr>
<td>Neural networks</td>
<td>1,064</td>
</tr>
<tr>
<td>Probabilistic methods</td>
<td>529</td>
</tr>
<tr>
<td>Reinforcement learning</td>
<td>335</td>
</tr>
<tr>
<td>Rule learning</td>
<td>230</td>
</tr>
<tr>
<td>Theory</td>
<td>472</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>3,583</td>
</tr>
<tr>
<td><strong>Majority class proportion</strong></td>
<td>29.7%</td>
</tr>
<tr>
<td><strong>Number of Edges</strong></td>
<td>22,516</td>
</tr>
</tbody>
</table>

7 Testing the kernels on semisupervised classification

7.1 Datasets

The case studies used in this work are real-world representations of networked data. They come from different domains that have been the subject of prior study in machine learning [50].

The collaborative Internet Movie Database **IMDb** has several applications such as making movie recommendations or classifying movies into categories. The classification problem deals with predicting movie popularity (whether a movie is a high-revenue hit or a low-revenue movie). It exploits a graph of movies linked together when they share a production company. The weight of an edge is the number of production companies that two movies have in common. The composition of the dataset is summarized in Table 5.

The **Cora** dataset is a graph of 3,583 nodes collected from research papers on machine learning labeled with 1 of 7 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 6.

The **WebKB** data consists of four sets of web pages gathered from 4 university departments, with each page manually labeled as belonging to 1 of 6 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 7.

The **Newsgroup** dataset\(^4\) comprises about 20,000 unstructured documents, taken from 20 Usenet newsgroups. News articles are labeled as belonging to 1 of 5 categories. The graph of documents was built by sampling about 200 documents at random from 5 different topics (see Table 8). The links between the documents are extracted from the

\(^4\) Available from http://people.csail.mit.edu/jrennie/20Newsgroups/
Table 7: Composition of the WebKB dataset.

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

Table 8: Composition of the Newsgroup dataset.

<table>
<thead>
<tr>
<th>Class</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer/graphics</td>
<td>200</td>
</tr>
<tr>
<td>Computer/pchardware</td>
<td>200</td>
</tr>
<tr>
<td>Motor/auto</td>
<td>200</td>
</tr>
<tr>
<td>Religion/atheism</td>
<td>200</td>
</tr>
<tr>
<td>Politics/mideast</td>
<td>200</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>1,000</td>
</tr>
<tr>
<td><strong>Majority class proportion</strong></td>
<td>20.0%</td>
</tr>
<tr>
<td><strong>Number of Edges</strong></td>
<td>328,904</td>
</tr>
</tbody>
</table>

five samples by eliminating stopwords, stemming, using a TF-IDF (term frequency-inverse document frequency) technique, computing mutual information, and finally computing the document-document matrix, as described in [82]. The composition of the dataset is summarized in Table 8.

7.2 Performance evaluation

All the kernels studied in this paper were compared on the seven datasets using the following methodology. The classification of unlabeled nodes was performed according to a simple kernel alignment with the labeled nodes (as described, e.g., in [86]). More precisely, consider an \( n \)-dimensional indicator vector \( \mathbf{y}_i \) containing a “1” entry when the corresponding node belongs to class \( i \) and a “0” entry otherwise, that is, when the node is unlabeled or belongs to another class. For each node, its similarity with the set of nodes belonging to class \( i \) is the value of the corresponding entry of the column vector \( \mathbf{Ky}_i \) where \( \mathbf{K} \) is the graph kernel matrix. Then, each node is assigned to the class with largest similarity; the column vector containing the predicted class index for all nodes is thus provided by \( \hat{\mathbf{y}} = \text{argmax}_i (\mathbf{Ky}_i) \).

The task consists in classifying unlabeled nodes. Classifiers assign the most suitable 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 whose label is known. 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 performance was averaged on those 50 runs.

For each run, a 10-fold internal cross-validation is performed in order to tune the parameters of each classifier (typically, parameter \( \alpha \) of kernels). The performance on
each run was assessed on the remaining unlabeled nodes with the parameter tuned during the internal cross-validation. The classification rate averaged on 50 runs is reported for each method and each labeling rate.

### 7.3 Standard scoring algorithm: Netkit

NetKit [50] is based on a relational learning approach. 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. This flexibility comes however with a time-consuming tuning process to optimize performance. Concerning our experiments, testing all the module configurations would have been prohibitive. For the baseline, we chose here the parameters that generally provide good results according to [50]. 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.

### 7.4 Results and discussion

Classification results for each kernel compared on every dataset are shown in Figures 1 to 4. $K_{RCT}$ and $K_{RL}$ clearly outperform the other kernels on every dataset, with a small advantage for $K_{RCT}$. More precisely, $K_{RCT}$ obtains the best results on almost every dataset except for CORA, while $K_{RL}$ obtains the best results on the CORA dataset and is in second position (after $K_{RCT}$) on the other ones. Moreover, they appear to be very stable across datasets. Notice that $K_{RWR}$ and $K_{MD}$ are competitive but never get better results than the two leading kernels.

Some other kernels perform less well. $K_{CT}$ lies far behind on almost every dataset, except 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. Other kernels, such as $K_{ED}$, $K_{VND}$, and $K_{CED}$ give poor and unstable results on almost every dataset.

### 8 Global Discussion

Table 9 shows a summary of the scores obtained on all datasets (both collaborative-recommendation and semisupervised classification experiments) by the various kernel-based algorithms. The collaborative-recommendation results (i.e., the first six rows) show the score obtained by the 9 kernel-based algorithms for the recall 20, the 2 datasets (i.e., the MovieLens and the Book-Crossing datasets), and the 3 methods (i.e., “$dir$” for the direct method, “$user$” for the user-based indirect method, and “$item$” for the item-based indirect method). For the results on semisupervised classification, each of the seven rows corresponds to one of the datasets, for a fixed labeling rate of 30%.

A two-step procedure was used for computing the scores of an algorithm for each row $i$: (1) the initial score of the algorithm was weighed against the baseline score by computing a new score $= (\text{initial score} - \text{baseline})/\text{baseline}$, and (2) the new score was normalized for the results of row $i$ (final score $= (\text{new score} - \text{average}(i))/\text{std}(i)$, where
Figure 1: Classification rate provided by the kernels and by NetKit. Error bars report standard deviations over 50 independent runs. Partially labeled graphs are considered, from 10% to 90% of labeled nodes. The figures show the average results obtained on the CORA and IMDb datasets. Results of the $K_{ED}$ and $K_{CED}$ ($K_{ED}$ and $K_{VND}$) kernels are out of classification-rate range for the CORA (IMDb) dataset (they perform poorly).
Figure 2: Classification rate provided by the kernels and by NetKit. Error bars report standard deviations over 50 independent runs. Partially labeled graphs are considered, from 10% to 90% of labeled nodes. The figures show the average results obtained on the WebKB Cornell and WebKB Texas datasets. Results of the $K_{CED}$ kernel are out of classification-rate range for both datasets ($K_{CED}$ performs poorly).
Classification rate on the WebKB-washington-graph dataset

Classification rate on the WebKB-wisconsin-graph dataset

Figure 3: Classification rate provided by the kernels and by NetKit. Error bars report standard deviations over 50 independent runs. Partially labeled graphs are considered, from 10% to 90% of labeled nodes. The figures show the average results obtained on the WebKB Washington and WebKB Wisconsin datasets. Results of $K_{CED}$ ($K_{CED}$, $K_{ED}$, and $K_{LED}$) are out of classification-rate range for the Washington (WebKB Wisconsin) dataset.
Figure 4: Classification rate provided by the kernels and by NetKit. Error bars report standard deviations over 50 independent runs. Partially labeled graph are considered, from 10% to 90% of labeled nodes. The figures show the results obtained on the NewsGroup dataset. Results of the KED are out of classification-rate range.

average(i) is the mean score on row i and std(i) is the standard deviation on row i). The "baseline score" is the score provided by Basic for collaborative recommendation and by NetKit for semisupervised classification. The “Average score” row computes, for each scoring algorithm, the average of its 13 results, with the same weight given for all the criteria (i.e., the rows). The “Final ranking” row ranks the algorithms according to their average score.

Table 9 suggests that KMD is the best kernel-based algorithm, with an average score of 0.83, followed by KRL (with a score of 0.59), KRCT (with a score of 0.50), and KCT (with a score of 0.45). Still, (1) we cannot conclude that KMD provides significantly better results than KRL (using a paired t-test to determine if there is a significant difference between the algorithms, $p < 5 \times 10^{-2}$) and (2) the results of Table 9 confirm that

<table>
<thead>
<tr>
<th></th>
<th>ML-dir20</th>
<th>ML-user20</th>
<th>ML-item20</th>
<th>BC-dir20</th>
<th>BC-user20</th>
<th>BC-item20</th>
<th>IMDb</th>
<th>Cora</th>
<th>WebKB-Cornell</th>
<th>WebKB-Texas</th>
<th>WebKB-Wisconsin</th>
<th>WebKB-Washington</th>
<th>NewsGroup</th>
<th>Average average</th>
<th>Average score</th>
<th>Final Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>KED</td>
<td>0.39</td>
<td>-0.20</td>
<td>-1.35</td>
<td>0.43</td>
<td>1.02</td>
<td>0.67</td>
<td>1.28</td>
<td>1.25</td>
<td>-0.63</td>
<td>-0.71</td>
<td>-0.45</td>
<td>-0.85</td>
<td>0.68</td>
<td>30</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>KRWR</td>
<td>0.59</td>
<td>0.70</td>
<td>1.40</td>
<td>0.41</td>
<td>1.00</td>
<td>0.55</td>
<td>1.14</td>
<td>0.06</td>
<td>0.24</td>
<td>1.11</td>
<td>0.68</td>
<td>0.91</td>
<td>0.67</td>
<td>9</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>KRL</td>
<td>0.70</td>
<td>0.06</td>
<td>1.41</td>
<td>1.03</td>
<td>0.35</td>
<td>0.46</td>
<td>1.64</td>
<td>0.56</td>
<td>0.24</td>
<td>0.94</td>
<td>0.77</td>
<td>0.55</td>
<td>0.53</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>KRCT</td>
<td>0.81</td>
<td>0.85</td>
<td>-0.75</td>
<td>0.77</td>
<td>1.13</td>
<td>1.05</td>
<td>0.11</td>
<td>0.10</td>
<td>-1.21</td>
<td>0.01</td>
<td>0.22</td>
<td>0.36</td>
<td>0.35</td>
<td>1</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>KCT</td>
<td>1.21</td>
<td>1.35</td>
<td>-2.10</td>
<td>0.58</td>
<td>0.58</td>
<td>0.58</td>
<td>0.58</td>
<td>0.26</td>
<td>0.58</td>
<td>0.58</td>
<td>0.58</td>
<td>0.58</td>
<td>0.58</td>
<td>1</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>NetKit</td>
<td>1.29</td>
<td>0.46</td>
<td>-0.35</td>
<td>0.34</td>
<td>0.36</td>
<td>0.34</td>
<td>0.34</td>
<td>0.34</td>
<td>0.34</td>
<td>0.34</td>
<td>0.34</td>
<td>0.34</td>
<td>0.34</td>
<td>1</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>Basic</td>
<td>-0.13</td>
<td>-0.13</td>
<td>-0.55</td>
<td>0.73</td>
<td>0.73</td>
<td>0.73</td>
<td>0.73</td>
<td>0.73</td>
<td>-2.90</td>
<td>0.73</td>
<td>0.73</td>
<td>0.73</td>
<td>0.73</td>
<td>9</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>WebKB Texas</td>
<td>-0.56</td>
<td>-0.56</td>
<td>-0.52</td>
<td>0.60</td>
<td>0.60</td>
<td>0.60</td>
<td>0.60</td>
<td>0.60</td>
<td>-2.52</td>
<td>0.60</td>
<td>0.60</td>
<td>0.60</td>
<td>0.60</td>
<td>1</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>WebKB Washington</td>
<td>0.48</td>
<td>0.48</td>
<td>0.11</td>
<td>-0.07</td>
<td>-0.07</td>
<td>-0.07</td>
<td>-0.07</td>
<td>-0.07</td>
<td>0.77</td>
<td>0.77</td>
<td>0.77</td>
<td>0.77</td>
<td>0.77</td>
<td>1</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>WebKB Wisconsin</td>
<td>1.72</td>
<td>0.86</td>
<td>-0.58</td>
<td>0.73</td>
<td>0.73</td>
<td>0.73</td>
<td>0.73</td>
<td>0.73</td>
<td>-1.99</td>
<td>0.73</td>
<td>0.73</td>
<td>0.73</td>
<td>0.73</td>
<td>1</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>Average</td>
<td>0.50</td>
<td>0.50</td>
<td>0.14</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
<td>0.30</td>
<td>1</td>
<td>7</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 9: Overall scores for ranking the algorithms (the higher the value, the better the algorithm).
the choice of the best kernel is application-dependent ($K_{MD}$ for collaborative recommendation and $K_{RCT}$ for semisupervised classification).

However, three kernels on a graph provide quite stable and good overall performance on all the datasets, independently of the application, namely the $K_{MD}$, the $K_{RL}$ and the $K_{RCT}$ kernels. All the other kernels perform badly on at least one dataset.

9 Conclusion and further work

This paper investigated various kernel-based procedures to compute similarities between elements of a database or, more generally, nodes of an undirected graph. These similarity measures can be used to compare items belonging to database tables that are not necessarily directly connected. They rely on the degree and the importance of the connectivity between these items. Our experiments showed that some of these quantities perform well in comparison with standard algorithms, namely the Markov diffusion kernel, the regularized Laplacian kernel and the regularized commute-time kernel. These three kernels on a graph perform constantly well and have a nice, 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 algorithms is that they induce a data matrix on which standard multivariate statistical analysis methods can be applied for mining graphs or databases. This allows to generalize useful techniques, like clustering, principal-component analysis, discriminant analysis, or canonical correlation analysis to the study of graphs or databases.

Further work will investigate the use of user or item features as well as the incorporation of ratings on the edges of the adjacency matrix. Other kernel-based multivariate statistical techniques, such as the already mentioned discriminant analysis or canonical correlation analysis will also be studied in the context of graph mining.

References


