The Principal Components Analysis of a Graph, and its Relationships to Spectral Clustering

Marco Saerens[1], Francois Fouss[1], Luh Yen[1] & Pierre Dupont[2]

[1]Information Systems Research Unit, IAG, Université catholique de Louvain, Place des Doyens 1, B-1348 Louvain-la-Neuve, Belgium
{saerens, fouss, yen}@isys.ucl.ac.be
[2]Department of Computing Science and Engineering, INGI, Université catholique de Louvain, Place Sainte Barbe 2, B-1348 Louvain-la-Neuve, Belgium
pdupont@info.ucl.ac.be

Abstract. This work presents a novel procedure for computing (1) distances between nodes of a weighted, undirected, graph, called the Euclidean Commute Time Distance (ECTD), and (2) a subspace projection of the nodes of the graph that preserves as much variance as possible, in terms of the ECTD – a principal components analysis of the graph. It is based on a Markov-chain model of random walk through the graph. The model assigns transition probabilities to the links between nodes, so that a random walker can jump from node to node. A quantity, called the **average commute time**, computes the average time taken by a random walker for reaching node j when starting from node i, and coming back to node i. The square root of this quantity, the ECTD, is a distance measure between any two nodes, and has the nice property of decreasing when the number of paths connecting two nodes increases and when the "length" of any path decreases. The ECTD can be computed from the pseudoinverse of the Laplacian matrix of the graph, which is a kernel. We finally define the **Principal Components Analysis** (PCA) of a graph as the subspace projection that preserves as much variance as possible, in terms of the ECTD. This graph PCA has some interesting links with spectral graph theory, in particular spectral clustering.

1. Introduction

This work introduces a general procedure allowing (1) to compute dissimilarities between nodes of a **weighted**, **undirected, graph** and (2) to represent the nodes of the graph in an Euclidean space of reduced dimensionality. Computing dissimilarities between pairs of nodes allows to determine the item that is most relevant (that is, similar) to a given item and allows, for instance, to cluster them. We present an application of this technique to collaborative filtering, with promising results, in a related paper [31].

The procedure used to compute the dissimilarities is based on a Markov-chain model. We define a **random-walk model** through the graph by assigning a transition probability to each edge. Thus, a random walker can jump from node to node, and each node therefore represents a state of the Markov model. From this Markov-chain

Lecture Notes in Artificial Intelligence, No. 3201, 15th European Conference on Machine Learning (ECML), Pisa, Italy, pp. 371-383, October 2004. © Springer-Verlag

model, we then compute a quantity, $m(j|i)$, called the **average first-passage time** (see for instance [17]), which is the average number of steps needed by a random walker for reaching state j for the first time, when starting from state i. The symmetrized quantity, $n(i,j) = m(j|i) + m(i|j)$, called the **average commute time** (see for instance [14]), provides a distance measure between any pair of states/nodes. The fact that this quantity is indeed a distance on a graph has been proved independently by Klein & Randic [18] and Gobel & Jagers [14]. Moreover, we show that $[n(i,j)]^{1/2}$, which is also a distance between nodes, takes a remarkable form and will be referred to as the **Euclidean Commute Time Distance** (ECTD). The ECTD can easily be computed from the pseudoinverse of the Laplacian matrix of the graph, which is shown to be a valid kernel.

These quantities have the nice property of decreasing when the number of paths connecting the two nodes increases and when the "length" of any path decreases (the communication is facilitated [11]). In short, two nodes are considered similar if there are many short paths connecting them. On the contrary, the "shortest path" (also called "geodesic" or "Dijkstra") distance does not necessarily decrease when connections between nodes are added, and thus does not capture the fact that strongly connected nodes are at a smaller distance than weakly connected nodes. This fact has already been recognized in the field of mathematical chemistry where there were attempts to use the "commute time" distance instead of the "shortest path" distance [18]. To our knowledge, while being interesting alternatives to the well-known "shortest path" or "geodesic" distance on a graph [6], these quantities have not been exploited as-is in the context of pattern recognition and machine learning. They have, however, been indirectly used in the framework of spectral clustering as will be shown in Section 6. This work therefore provides a new interpretation for spectral clustering since we will show that spectral clustering can be interpreted in terms of ECTD.

We further show that we can project the nodes space of the graph into an Euclidean subspace that maximaly preserves ECTD among all linear subspace projections. This subspace is optimal in the following sense: it keeps as much variance of the projected data as possible (in terms of the ECTD). It is therefore an equivalent of principal component analysis in terms of the ECTD; we call this technique the **principal components analysis of the graph**.

In summary, this paper has five main contributions: (1) it suggests the use of the average first-passage time and the ECTD between nodes of a graph as a useful pattern recognition tool; (2) it shows that the average first-passage time and the ECTD can be computed in terms of the pseudoinverse $\mathbf{L}^{+}$ of the Laplacian matrix of the graph, from the definition of the average first-passage time; (3) it shows that $\mathbf{L}^{+}$ is a kernel and could be used as such for SVM classification; (4) it introduces the PCA of a graph which is a principal component analysis computed on the ECTD matrix; (5) it provides an elegant interpretation of both spectral clustering and spectral embedding in terms of random walks on a graph.

Section 2 introduces the random-walk model – a Markov chain model. Section 3 develops our dissimilarity measures as well as the iterative formulae to compute them. Section 4 gives details for the computation of the average first-passage time and the average commute time from the Laplacian matrix of the graph. It also derives a number of interesting properties of the Laplacian pseudoinverse. Section 5 introduces an eigenvector decomposition of the pseudoinverse of the Laplacian matrix that maximizes the variance of the projected data. It also shows that this pseudoinverse is a valid kernel. Section 6 summarizes related work and develops some interesting relationships with spectral clustering, among others. Section 7 is the conclusion.

2. A Markov-chain model of random walk on a graph

2.1. The Laplacian matrix of a weighted graph

Let us consider that we are given a weighted, undirected, graph, G, with symmetric weights $w_{ij} > 0$ between every couple of nodes, i and j, which are linked by an edge (say G has n nodes in total). The weight w_{ij} of the edge connecting node i and node j should be set to some meaningful value, with the following convention: the more important the relation between node i and node j, the larger the value of w_{ij}, and consequently the easier the communication through the edge. Notice that we require that the weights are both positive ($w_{ij} > 0$) and symmetric ($w_{ij} = w_{ji}$). The elements a_{ij} of the adjacency matrix $\mathbf{A}$ of the graph are defined in a standard way as

$$a_{ij} = \begin{cases} w_{ij} \text{ if node } i \text{ is connected to node } j \\ 0 \text{ otherwise} \end{cases} \tag{2.1}$$

where $\mathbf{A}$ is symmetric. We also introduce the Laplacian matrix $\mathbf{L}$ of the graph, defined in the usual manner: $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D} = \text{diag}(a_{i.})$ is the degree matrix, and $d_{ii} = [\mathbf{D}]_{ii} = a_{i.} = \sum_{j=1}^{n} a_{ij}$. Furthermore, the volume of the graph is defined as $V_G = vol(G) = \sum_{i=1}^{n} d_{ii} = \sum_{i,j=1}^{n} a_{ij}$.

We suppose that the graph has a single connected component; that is, any node can be reached from any other node of the graph. In this case, $\mathbf{L}$ has rank $n-1$, where n is the number of nodes [9]. If $\mathbf{e}$ is a column vector made of 1 (i.e., $\mathbf{e} = [1, 1, \ldots, 1]^{\text{T}}$, where T denotes the matrix transpose) and $\mathbf{0}$ is a column vector made of 0, $\mathbf{Le} = \mathbf{0}$ and $\mathbf{e}^{\text{T}}\mathbf{L} = \mathbf{0}$ hold: $\mathbf{L}$ is doubly centered. The null space of $\mathbf{L}$ is therefore the one-dimensional space spanned by $\mathbf{e}$. Moreover, one can easily show that $\mathbf{L}$ is symmetric and positive semidefinite (see for instance [4] or [9]).

2.2. A random walk model on the graph

The Markov chain describing the sequence of nodes visited by a random walker is called a random walk on a weighted graph (see for instance [14]). We associate a state of the Markov chain to every node; we also define a random variable, $s(t)$, representing the state of the Markov model at time step t. If the random walker is in state i at time t, we say $s(t) = i$. We define a random walk by the following single-step transition probabilities $\text{P}(s(t+1) = j | s(t) = i) = a_{ij}/d_{ii} = p_{ij}$.

In other words, to any state or node i, we associate a probability of jumping to an adjacent node, $s(t+1) = j$, which is proportional to the weight w_{ij} of the edge connecting i and j. The transition probabilities only depend on the current state and not on the past ones (first-order Markov chain). Since the graph is connected, the Markov chain is irreducible, that is, every state can be reached from any other state.

Now, if we denote the probability of being in state i at time t by $x_i(t) = \mathrm{P}(s(t) = i)$ and we define $\mathbf{P}$ as the transition matrix whose entries are $p_{ij} = \mathrm{P}(s(t+1) = j|s(t) = i)$, we have $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$ and the evolution of the Markov chain is characterized by

$$\begin{cases} \mathbf{x}(0) = \mathbf{x}^0 \\ \mathbf{x}(t+1) = \mathbf{P}^{\mathrm{T}}\mathbf{x}(t) \end{cases} \tag{2.2}$$

This provides the state probability distribution $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^{\mathrm{T}}$ at time t once the initial probability distribution, $\mathbf{x}^0$, is known (see [5], [17], [26] for more details).

3. Average first-passage time and average commute time

In this section, we review two basic quantities that can be computed from the definition of the Markov chain, that is, from its probability transition matrix: the average first-passage time and the average commute time.

The **average first-passage time**, $m(k|i)$ is defined as the average number of steps a random walker, starting in state i, will take to enter state k for the first time [26]. More precisely, we define the minimum time until hitting state k as $T_{ik} = \min\left(t \geq 0 \mid s(t) = k \text{ and } s(0) = i\right)$ for one realization of the stochastic process. The average first-passage time is the expectation of this quantity, when starting from state i: $m(k|i) = E\left[T_{ik}|s(0) = i\right] = E_i\left[T_{ik}\right]$. The $m(k|i)$ verify the following recurrence relations (see for instance [26])

$$\begin{cases} m(k|i) = 1 + \displaystyle\sum_{\substack{j=1 \\ j \neq k}}^{n} p_{ij}\, m(k|j), \text{ for } i \neq k \\ m(k|k) = 0 \end{cases} \tag{3.1}$$

These equations can be used in order to iteratively compute the average first-passage times [26]. The meaning of these formulae is quite obvious: in order to go from state i to state k, one has to go to any adjacent state j and proceed from there.

We now introduce a closely related quantity, the **average commute time**, $n(i,j)$, which is defined as the average number of steps a random walker, starting in state i, will take before entering a given state j for the first time, and go back to i. That is, $n(i,j) = m(j|i) + m(i|j)$. Notice that, while $n(i,j)$ is symmetric by definition, $m(i|j)$ is not.

As shown by several authors [14], [18], the average commute time is a distance measure, since, for any states i, j, k: $n(i,j) \geq 0$, $n(i,j) = 0$ if and only if $i = j$, $n(i,j) = n(j,i)$ and $n(i,j) \leq n(i,k) + n(k,j)$. In Section 4, we show that $[n(i,j)]^{1/2}$, which is also a distance on the graph, takes a remarkable form. Both the average first-passage time and the average commute time provide dissimilarity measures on any pairs i, j of nodes.

4. Computation of the basic quantities by means of $\mathbf{L}^+$

Methods for computing these two quantities are based on matrix pseudoinverses or on iterative procedures. If the matrices are too large, the computation by pseudoinverse becomes untractable; in this case, one may use iterative techniques based on Equation 3.1 and on the sparseness of the probability transition matrix.

In this section, we will show how the average first-passage time and the average commute time can be computed from Equation 3.1, by using the pseudoinverse of the Laplacian matrix of the graph, which plays a fundamental role and has a number of interesting properties. The developments in this section are inspired by the work of Klein & Randic [18] which showed, based on an electrical equivalence (see last paragraph of Section 6), that the effective resistance (which is equivalent to the average commute time [8]) can be computed from the Laplacian matrix. We thus extend their results by showing that the formula computing the average commute time can be directly derived from 3.1, and by providing formulae for the average first-passage time as well.

Let us denote by l_{ij} element i, j of the Laplacian matrix $\mathbf{L}$; in other words, $l_{ij} = [\mathbf{L}]_{ij}$. The Moore-Penrose pseudoinverse of $\mathbf{L}$ (see [1]) will be denoted by $\mathbf{L}^+$, with elements $l^+_{ij} = [\mathbf{L}^+]_{ij}$. In Appendix A, we prove some useful properties of $\mathbf{L}^+$.

In [31], we show[1] that the computation of the average first-passage time in terms of $\mathbf{L}^+$ can be obtained from Equation 3.1:

$$m(k|i) = \sum_{j=1}^{n} \left(l^+_{ij} - l^+_{ik} - l^+_{kj} + l^+_{kk} \right) d_{jj} \tag{4.1}$$

For $n(i, j)$, we obtain from Equation 4.1:

$$n(i, j) = V_G \left(l^+_{ii} + l^+_{jj} - 2l^+_{ij} \right) \tag{4.2}$$

This formula has already been obtained by using the electrical equivalent of commute times (the effective resistance) [18], [31]. If we further define $\mathbf{e}_i$ as the ith column of $\mathbf{I}$, $\mathbf{e}_i = [\underset{1}{0}, \ldots, \underset{i-1}{0}, \underset{i}{1}, \underset{i+1}{0}, \ldots, \underset{n}{0}]^{\mathrm{T}}$, Equation 4.2 can be rewritten in the form:

$$n(i, j) = V_G \, (\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}} \mathbf{L}^+ (\mathbf{e}_i - \mathbf{e}_j) \tag{4.3}$$

where each node i is represented by a unit basis vector, $\mathbf{e}_i$, of the node space. We easily observe that $[n(i, j)]^{1/2}$ is a distance in the Euclidean space of the nodes of the graph since $\mathbf{L}^+$ is positive semidefinite. As already mentioned, it will therefore be called the **Euclidean Commute Time Distance** (ECTD).

5. The principal components analysis of a graph

In this section, we show that, based on the eigenvector decomposition, the nodes vectors, $\mathbf{e}_i$, can be mapped into a new Euclidean space that preserves the ECTD, or a subspace keeping as much variance as possible, in terms of ECTD.

[1] We do not provide the proof here because it is both lengthy and technical.

5.1. Transformation to an Euclidean space preserving the ECTD

Let us first show that the node vectors $\mathbf{e}_i$ can be mapped into an Euclidean space that preserves the ECTD. Indeed, every positive semidefinite matrix can be transformed to a diagonal matrix, $\mathbf{\Lambda} = \mathbf{U}^{\mathrm{T}}\mathbf{L}^{+}\mathbf{U}$, where $\mathbf{U}$ is an orthonormal matrix made of the eigenvectors of $\mathbf{L}^{+}$, $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{n-1}, \mathbf{u}_n = \mathbf{0}]$: the column vectors $\mathbf{u}_k$ are the orthonormal eigenvectors of $\mathbf{L}^{+}$, $\mathbf{u}_i^{\mathrm{T}}\mathbf{u}_j = \delta_{ij}$ or $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}$ (see for instance [25]). The diagonal matrix $\mathbf{\Lambda}$ contains the eigenvalues of $\mathbf{L}^{+}$ in decreasing order of importance. Hence we have

$$\begin{aligned} n(i,j) &= V_G\,(\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}}\mathbf{L}^{+}(\mathbf{e}_i - \mathbf{e}_j) = V_G\,(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}}\mathbf{U}^{\mathrm{T}}\mathbf{L}^{+}\mathbf{U}(\mathbf{x}_i - \mathbf{x}_j) \\ &= V_G\,(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}}\mathbf{\Lambda}(\mathbf{x}_i - \mathbf{x}_j) = V_G\,(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}}(\mathbf{\Lambda}^{1/2})^{\mathrm{T}}\mathbf{\Lambda}^{1/2}(\mathbf{x}_i - \mathbf{x}_j) \\ &= V_G\,(\mathbf{x}'_i - \mathbf{x}'_j)^{\mathrm{T}}(\mathbf{x}'_i - \mathbf{x}'_j) \end{aligned}$$

where we made the transformations

$$\mathbf{x}_i = \mathbf{U}^{\mathrm{T}}\mathbf{e}_i \text{ and } \mathbf{x}'_i = \mathbf{\Lambda}^{1/2}\mathbf{x}_i \tag{5.1}$$

So, in this n-dimensional Euclidean space, the transformed node vectors, $\mathbf{x}'_i$, are exactly separated by ECTD. In Appendix B, we show that $\mathbf{L}^{+}$ is a kernel since it corresponds to the matrix of the inner products of the $\mathbf{x}'_i$. Moreover, one can easily show [31] that the $\mathbf{x}'_i$ are centered (their center of gravity is $\mathbf{0}$): $\sum_{i=1}^{n}\mathbf{x}'_i = \mathbf{0}$.

5.2. Approximate ECTD based on the projection in a subspace

However, the transformed space introduced in previous section has dimensionality n (the graph order), which is untractable for most applications. We therefore define an approximation of this transformation that preserves as much information as possible.

The so-called spectral (or eigenvector) decomposition of $\mathbf{L}^{+}$ is defined as (see any textbook on linear algebra, for instance, [20]) $\mathbf{L}^{+} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}} = \sum_{k=1}^{n-1}\lambda_k\,\mathbf{u}_k\mathbf{u}_k^{\mathrm{T}}$ where $\lambda_1 > \lambda_2 > \ldots > \lambda_{n-1} > \lambda_n = 0$ are the eigenvalues of $\mathbf{L}^{+}$. As in previous section, the column vectors $\mathbf{u}_k$ are the orthonormal eigenvectors of $\mathbf{L}^{+}$, $\mathbf{u}_i^{\mathrm{T}}\mathbf{u}_j = \delta_{ij}$.

Suppose now that we compute the eigenvector expansion of $\mathbf{L}^{+}$ up to $m < n - 1$: $\widetilde{\mathbf{L}}^{+} = \sum_{k=1}^{m}\lambda_k\,\mathbf{u}_k\mathbf{u}_k^{\mathrm{T}} = \widetilde{\mathbf{U}}\widetilde{\mathbf{\Lambda}}\widetilde{\mathbf{U}}^{\mathrm{T}}$ where $\widetilde{\mathbf{U}} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m, \mathbf{0}, \ldots, \mathbf{0}]$ and $\widetilde{\mathbf{\Lambda}} = \mathrm{diag}[\lambda_1, \lambda_2, \ldots, \lambda_m, 0, \ldots, 0]$. Let us compute the corresponding distance between nodes i and j: $\widetilde{n}(i,j) = V_G\,(\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}}\widetilde{\mathbf{L}}^{+}(\mathbf{e}_i - \mathbf{e}_j)$.

By $\widetilde{\mathbf{L}}^{+} = \widetilde{\mathbf{U}}\,\widetilde{\mathbf{\Lambda}}\widetilde{\mathbf{U}}^{\mathrm{T}}$ and using the same reasoning as in previous section, we can recompute the distance as follows: $\widetilde{n}(i,j) = V_G\,(\widetilde{\mathbf{x}}'_i - \widetilde{\mathbf{x}}'_j)^{\mathrm{T}}(\widetilde{\mathbf{x}}'_i - \widetilde{\mathbf{x}}'_j)$, where, this time, the $\widetilde{\mathbf{x}}'_i$ are column vectors containing zeroes from the $m+1$ position: $\widetilde{\mathbf{x}}' = [\widetilde{x}'_1, \widetilde{x}'_2, \ldots, \widetilde{x}'_m, 0, \ldots, 0]^{\mathrm{T}}$. The transformation is therefore defined by

$$\widetilde{\mathbf{x}}_i = \widetilde{\mathbf{U}}^{\mathrm{T}}\mathbf{e}_i \text{ and } \widetilde{\mathbf{x}}'_i = \widetilde{\mathbf{\Lambda}}^{1/2}\widetilde{\mathbf{x}}_i \tag{5.2}$$

This subspace is an m-dimensional space where the commute time distances are approximately preserved. A bound on this approximation is provided in [31]:
$||n(i,j) - \widetilde{n}(i,j)|| \leq V_G \sum_{k=m+1}^{n-1}\lambda_k$.

Now, we easily observe from (5.1) that if u_i^k is coordinate i of eigenvector $\mathbf{u}_k$ ($[\mathbf{u}_k]_i = u_i^k$) corresponding to eigenvalue λ_k of $\mathbf{L}^+$, and if x_k^i is coordinate k of vector $\mathbf{x}_i$ ($[\mathbf{x}_i]_k = x_k^i$), $x_k^i = u_i^k$ holds. We thus have $x_k'^i = \sqrt{\lambda_k} u_i^k$ where $x_k'^i$ is coordinate k of vector $\mathbf{x}_i'$ ($[\mathbf{x}_i']_k = x_k'^i$).

In other words, the first coordinate of the n node vectors, $\mathbf{x}_i, i = 1 \dots n$, corresponding to the **first axis** ($k = 1$) of the transformed space, are $x_1'^1, x_1'^2, \dots, x_1'^n$, or $\sqrt{\lambda_1} u_1^1, \sqrt{\lambda_1} u_2^1, \dots, \sqrt{\lambda_1} u_n^1$. Thus, the first coordinate of these n node vectors is simply the projection of the original node vectors, $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_n$, on the first eigenvector, $\mathbf{u}_1$, weighted by $\sqrt{\lambda_1}$. More generally, coordinate k of the node vectors in the transformed space is simply the projection of the original node vectors, $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_n$, on $\mathbf{u}_k$, weighted by $\sqrt{\lambda_k}$. The idea is thus to discard the axes corresponding to the smallest eigenvalues of $\mathbf{L}^+$.

5.3. Relations to principal components analysis

We will now show that this decomposition is similar to principal components analysis in the sense that the projection has maximal variance among all the possible candidate projections. If $\mathbf{X}'$ denotes the data matrix containing the coordinates of the nodes in the transformed space, $\mathbf{x}_i'^{\mathrm{T}}$, on each row (see Appendix B), we easily deduce from (5.1) that $\mathbf{X}' = \mathbf{U}\boldsymbol{\Lambda}^{1/2}$.

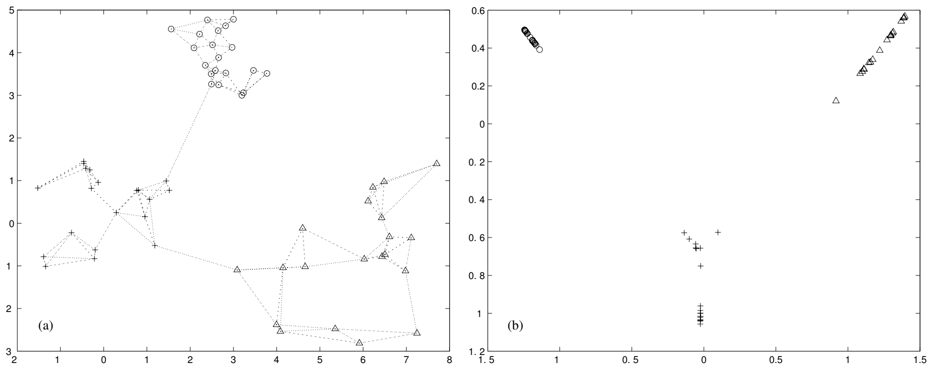

Fig. 5.1. One example of principal components analysis based on the ECTD. The original graph (the weight of each edge is inversely proportional to 1/(its length)) is shown on the left (a); its projection on the two first principal components is shown on the right (b).

Now, it is well-known that the principal components analysis of a data matrix $\mathbf{X}'$ yields, as kth principal component, the eigenvector, $\mathbf{v}_k$, of $(\mathbf{X}')^{\mathrm{T}}\mathbf{X}'$ (which is the variance-covariance matrix, since the $\mathbf{x}_i'$ are centered). But $(\mathbf{X}')^{\mathrm{T}}\ \mathbf{X}' = (\mathbf{U}\boldsymbol{\Lambda}^{1/2})^{\mathrm{T}}\mathbf{U}\boldsymbol{\Lambda}^{1/2}$ $= \boldsymbol{\Lambda}$. Since $\boldsymbol{\Lambda}$ is a diagonal matrix, we deduce that the $\mathbf{x}_i'$ *are already expressed in the principal components coordinate system* – the eigenvectors of $(\mathbf{X}')^{\mathrm{T}}\mathbf{X}'$ are the basis vectors of the transformed space. Thus, if $x_k'^i$ is coordinate k of vector $\mathbf{x}_i'$, it corresponds to the projection of node i on the kth principal component. The variance, in terms of ECTD, of the nodes cloud on each principal component k is therefore λ_k.

We thus conclude that this projection can be viewed as a principal components analysis in the Euclidean space where the nodes are exactly separated by ECTD. This decomposition therefore defines the projection of the node vectors that has maximal variance (in terms of the ECTD) among all the possible candidate projections. An exemple of PCA is provided in Figure 5.1. Notice that it can be shown that performing a multidimensional scaling on the ECTD gives exactly the same results as the principal components analysis.

Furthermore, since $\mathbf{L}$ and $\mathbf{L}^{+}$ both have rank $(n-1)$ and have the same set of eigenvectors but inverse eigenvalues (if λ_i^l are the eigenvalues of $\mathbf{L}$ and λ_i^{+} are the eigenvalues of $\mathbf{L}^{+}$, $\lambda_i^l = 1/\lambda_i^{+}$, for $i \neq n$; $\lambda_n^l = \lambda_n^{+} = 0$; see Appendix A) we do not need to explicitly compute the pseudoinverse of $\mathbf{L}$ in order to compute the projection. We only need to compute the m smallest (except $\lambda_n^l = 0 = \lambda_n^{+}$) eigenvectors (that is, with lowest eigenvalues) of $\mathbf{L}$, which become the largest of $\mathbf{L}^{+}$.

6. Related work

There is a vast literature on *spectral graph theory* (see [9] for a monograph) and several results about the *Laplacian spectrum* of (hyper-)graphs are summarized in [22, 23]. Spectral techniques have been applied in a wide variety of contexts including high performance computing [28], image segmentation [33], web page ranking [27, 19], information retrieval [10], RNA motif classification [13], data clustering [24, 37], and dimensionality reduction [2].

In particular, *spectral clustering* refers to a collection of techniques that cluster n data points using eigenvectors of a matrix derived from the $n \times n$ *affinity matrix* $\mathbf{W}$: $w_{ij} = \exp[-d^2(\mathbf{x}_i, \mathbf{x}_j)/2\sigma^2]$, where $d(\mathbf{x}_i, \mathbf{x}_j)$ denotes the dissimilarity between the points $\mathbf{x}_i$ and $\mathbf{x}_j$, and σ is a free parameter (see [36] for a review). A common choice is $d^2(\mathbf{x}_i, \mathbf{x}_j) = ||\mathbf{x}_i - \mathbf{x}_j||^2$. An automatic procedure for determining σ so as to minimize the cluster distortion is proposed in [24].

Spectral clustering is a graph-theoretic approach to clustering and the affinity matrix precisely corresponds to the (weighted) adjacency matrix $\mathbf{A}$ defined in section 2.1. In the context of image segmentation, Shi and Malik [33] introduced the *normalized cut* (NCut) criterion to define an optimal bipartitioning of a graph:

Let $\{A, \overline{A}\}$ denotes a bipartition of the set of vertices V of a graph G ($A \neq \varnothing$, $\overline{A} \neq \varnothing$, $A \cap \overline{A} = \varnothing$ and $A \cup \overline{A} = V$). The *cut* is the total weight of edges connecting the two disjoint sets A and $\overline{A}$: $cut(A, \overline{A}) = \sum_{i \in A, j \in \overline{A}} w_{ij}$. The NCut criterion aims at finding the bipartition minimizing: $NCut(A, \overline{A}) = ((1/V_A) + (1/V_{\overline{A}}))\, cut(A, \overline{A})$ where $V_A = vol(A)$ is the subgraph volume. This criterion seeks a balance between the goal of clustering (finding tight clusters) and segmentation (finding well separated clusters). A very similar notion of *conductance* $\phi(A, \overline{A})$ of a cut is presented in [29]: $\phi(A, \overline{A}) = cut(A, \overline{A}) / \min(V_A, V_{\overline{A}})$. Finding the optimal NCut is NP-complete even for a graph on a regular grid but an approximation can be found by computing the eigenvalues of the *normalized Laplacian* $\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ [33]. In particular, the eigenvector associated to the second smallest eigenvalue (also known as the *algebraic connectivity* or *Fiedler value* [12]) is used to bipartition the graph. Note that the standard spectral graph partitioning uses the Laplacian matrix $\mathbf{L}$ (instead of its normalized version). This corresponds to the minimization of the *average cut criterion*: $AverageCut(A, \overline{A}) = \big((1/|A|) + (1/|\overline{A}|)\big)\, cut(A, \overline{A})$ where $|A|$ is the graph order. This last version is exactly similar to our method since we showed, in the previous section, that computing the largest eigenvalues/eigenvectors of $\mathbf{L}^{+}$ is equivalent to computing the smallest non-trivial eigenvalues/eigenvectors of $\mathbf{L}$.

Meila and Shi present links between spectral segmentation and *Markov random walks* [21]. Their random walk model is identical to the one defined in Section 2 but different properties are stressed. It is shown in particular that $NCut(A, \overline{A}) = p_{A\overline{A}} + p_{\overline{A}A}$, where p_{AB} denotes the probability of a random walk transiting from state set A to state set B in one step, given the current state is in A and the random walk is started according to the stationary distribution of the Markov chain.

An application of the same random walk model to partially labeled classification is proposed in [35]. In this learning framework (also known as *transductive learning*), unlabeled examples provide information of the structure of the domain while the class labels of a few examples are known. As for spectral clustering, each data point is associated to a graph vertex, *i.e.* a state of the associated Markov chain. The model assumes a uniform initial distribution and includes a distribution $p(y|i)$ of class label y given state i. $p(y|i)$ can be estimated with EM or a margin-based criterion. Classification of point k is performed so as to maximize the posterior probability $p(y|k) = \sum_i p(y|i)[\mathbf{P}^r]_{ik}$, where $\mathbf{P}^r$ denotes power r of the transition matrix $\mathbf{P}$. Thus the classification of the example k depends on $[\mathbf{P}^r]_{ik}$, the probability that the Markov process started from the state i given that it ended up in k after r steps. The value of r is a parameter controlling the smoothness of the random walk representation. Another work using the Laplacian spectrum for transductive learning is described in [15].

The random walk model presented in Section 2 defines the transition matrix $\mathbf{P}$ of a Markov process from the adjacency (or affinity) matrix $\mathbf{A}$ as $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$. Let us mention that several alternative definitions of the Markov transition matrix are proposed in [16], in the context of supervised or unsupervised classification.

Laplacian eigenmaps (also called spectral embedding) is a dimensionality reduction procedure that has been proposed recently by Belkin and Niyogi [3]. The authors solve the following related eigenproblem: $\mathbf{Lu} = \lambda\mathbf{Du}$. The smallest eigenvalue is left out and the other small eigenvalues are used for the embedding. This is the same embedding that is computed with the spectral clustering algorithm from Shi and Malik [33]. As noted in [36], an equivalent result can be obtained by renormalizing the adjacency matrix: $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ and computing the eigenvectors/eigenvalues of $\mathbf{L}$ instead. It should be clear that, once more, this reduction technique is closely related to our definition of the principal components analysis of a graph.

Smola and Kondor present connections between spectral graph theory and *graph kernels* [34]. In particular, they define a graph regularization, which aims to emphasize the role of the smallest non-trivial eivenvalues of $\mathbf{L}$ and, on the contrary, discard the largest ones. Once more, this has interesting links with our definition of the PCA of a graph, since the PCA discard the smallest nontrivial eivenvalues/eigenvectors of $\mathbf{L}^{+}$ (which, as already stressed, correspond to the largest of $\mathbf{L}$). They also stress the links between these techniques and web page ranking algorithms such as PageRank [27], HITS [19] and randomized HITS [38].

Finally, there is an intriguing correspondence between random walk on a graph and electrical networks theory, as popularized by Doyle and Snell in their nice book [11] (see also [4]). Average commute time has an equivalent in terms of electrical networks. Indeed, in [8], it is shown that $n(i,j) = V_G \, r^e_{ij}$, where r^e_{ij} is the effective resistance between node i and node j. In other words, average commute time and effective resistance basically measure the same quantity (see [31] for more details).

7. Conclusion and further work

We introduced a general procedure for computing dissimilarities between nodes of a graph. It is based on a particular Markov-chain model of random walk through the graph. More precisely, we compute a quantity, called the Euclidean Commute Time Distance, that provides a distance measure between any pair of nodes. We also introduced a subspace projection method preserving as much variance (in terms of the ECTD) as possible; it therefore defines a principal components analysis on a graph. We are now exploiting this ECTD in various problems, including inexact graph matching, collaborative filtering [31], supervised classification and clustering. We are also working on the definition of a discriminant analysis of a graph, with application to graph vizualisation.

References

1. S. Barnett. *Matrices: Methods and Applications.* Oxford University Press, 1992.
2. M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 585–591. MIT Press, 2001.
3. M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15:1373–1396, 2003.
4. B. Bollobas. *Modern graph theory.* Springer, 1998.
5. P. Bremaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues.* Springer-Verlag, 1999.
6. F. Buckley and F. Harary. *Distance in graphs.* Addison-Wesley Publishing Company, 1990.
7. S. Campbell and C. Meyer. *Generalized inverses of linear transformations.* Pitman Publishing Company, 1979.
8. A. K. Chandra, P. Raghavan, W. L. Ruzzo, R. Smolensky, and P. Tiwari. The electrical resistance of a graph captures its commute and cover times. *Annual ACM Symposium on Theory of Computing*, pages 574–586, 1989.
9. F. R. Chung. *Spectral Graph Theory.* American Mathematical Society, 1997.
10. S. Deerweester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.
11. P. G. Doyle and J. L. Snell. *Random Walks and Electric Networks.* The Mathematical Association of America, 1984.
12. M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its applications to graph theory. *Czech. Math. J.*, 25(100):619–633, 1975.

13. H. Gan, D. Fera, J. Zorn, N. Shiffeldrim, M. Tang, U. Laserson, N. Kim, and T.Schlick. RAG: RNA-As-Graphs Database - Concepts, Analysis, and Features. to appear in Bioinformatics, 2004.
14. F. Gobel and A. Jagers. Random walks on graphs. *Stochastic Processes and their Applications*, 2:311–336, 1974.
15. T. Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the 20th International Conference on Machine Learning*, Washington DC, 2003.
16. S. Kamvar, D. Klein, and C. Manning. Spectral learning. In *Proceedings of the International Joint Conference of Artificial Intelligence*, April 2003.
17. J. G. Kemeny and J. L. Snell. *Finite Markov Chains.* Springer-Verlag, 1960.
18. D. J. Klein and M. Randic. Resistance distance. *Journal of Mathematical Chemistry*, 12:81–95, 1993.
19. J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
20. D. Lay. *Linear algebra and its applications, 3th ed.* Addison-Wesley, 2003.
21. M. Meila and J. Shi. A random walks view of spectral segmentation. In *Proceedings of AISTATS*, 2001.
22. B. Mohar. The Laplacian spectrum of graphs. In Y. Alavi, G. Chartrand, and A. S. O.R. Oellermann, editors, *Graph Theory, Combinatorics, and Applications*, volume 2, pages 871–898. Wiley, 1991.
23. B. Mohar. Laplace eigenvalues of graphs – a survey. *Discrete Mathematics*, 109:171–183, 1992.
24. A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 849–856, Vancouver, Canada, 2001. MIT Press.
25. B. Noble and J. Daniels. *Applied linear algebra, 3th ed.* Prentice-Hall, 1988.
26. J. Norris. *Markov Chains.* Cambridge University Press, 1997.
27. L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. *Technical Report, Computer System Laboratory, Stanford University*, 1998.
28. A. Pothen, H. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Matrix Analysis*, 11(3):430–452, 1990.
29. A. V. R. Kannan, S. Vempala. On clusterings: Good, bad and spectral. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, 2000.
30. C. Rao and S. Mitra. *Generalized inverse of matrices and its applications.* John Wiley and Sons, 1971.
31. M. Saerens, A. Pirotte, and F. Fouss. Computing dissimilarities between nodes of a graph: Application to collaborative filtering. *Technical Report, IAG, Universite catholique de Louvain (www.isys.ucl.ac.be/staff/francois/Articles)*, 2004.
32. B. Scholkopf and A. Smola. *Learning with kernels.* The MIT Press, 2002.
33. J. Shi and J. Malik. Normalised cuts and image segmentation. *IEEE Transactions on Pattern Matching and Machine Intelligence*, 22:888–905, August 2000.
34. A. J. Smola and R. Kondor. Kernels and regularization on graphs. In M. Warmuth and B. Schölkopf, editors, *Proceedings of the Conference on Learning Theory and Kernels Workshop*, 2003.
35. M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, Vancouver, Canada, 2001. MIT Press.
36. Y. Weiss. Segmentation using eigenvectors: a unifying view. In *International Conference on Computer Vision*, 1999.

37. H. Zha, C. Ding, M. Gu, X. He, and H. Simon. Spectral relaxation for K-means clustering. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 1057–1064, Vancouver, Canada, 2001. MIT Press.
38. A. X. Zheng, A. Y. Ng, and M. I. Jordan. Stable eigenvector algorithms for link analysis. *Proceedings of the 24th International Conference on Research and Development in Information Retrieval*, pages 258–296, 2001.

Acknowledgments

We thank Prof. Van Dooren and Prof. Blondel, from the "Département d'Ingénierie Mathématique" of the Université catholique de Louvain, for insightful discussions.

A. Appendix: Some useful properties of the Laplacian matrix

(1) $\mathbf{L}^{+}$ is symmetric. Since $\mathbf{L}$ is symmetric and, for any matrix $\mathbf{A}$, $(\mathbf{A}^{\mathrm{T}})^{+} = (\mathbf{A}^{+})^{\mathrm{T}}$ (see [1]), we easily obtain $\mathbf{L}^{+} = (\mathbf{L}^{\mathrm{T}})^{+} = (\mathbf{L}^{+})^{\mathrm{T}}$. Therefore, $\mathbf{L}^{+}$ is symmetric.

(2) $\mathbf{L}$ is an EP-matrix. An EP matrix $\mathbf{A}$ is a matrix which commutes with its pseudoinverse, i.e. $\mathbf{A}^{+}\mathbf{A} = \mathbf{A}\mathbf{A}^{+}$. Since $\mathbf{L}$ is real symmetric, it is automatically an EP-matrix (see [1], p.253). In particular, it is worth mentioning the following properties of EP-matrices:

1. If $(\lambda_i^l \neq 0, \mathbf{u}_i)$ are (eigenvalues, eigenvectors) of $\mathbf{L}$, then $(\lambda_i^{-1} \neq 0, \mathbf{u}_i)$ are corresponding (eigenvalues, eigenvectors) of $\mathbf{L}^{+}$. On the other hand, if $(\lambda_j^l = 0, \mathbf{u}_j)$ are (eigenvalues, eigenvectors) of $\mathbf{L}$, then they are also (eigenvalues, eigenvectors) of $\mathbf{L}^{+}$.
2. In particular, $\mathbf{L}^{+}$ has rank $n-1$ and has the same null space as $\mathbf{L}$: $\mathbf{L}^{+}\mathbf{e} = 0$.
3. The previous property implies that $\mathbf{L}^{+}$ is doubly centered (the sum of the columns and rows of $\mathbf{L}^{+}$ is zero), just as $\mathbf{L}$ (see also [30], chapter 10, for a discussion of this topic).
4. Other properties of EP-matrices are described in [1] or [7].

(3) $\mathbf{L}^{+}$ is positive semidefinite. Indeed, from the previous property, the eigenvalues of $\mathbf{L}$ and $\mathbf{L}^{+}$ have the same sign and $\mathbf{L}$ is positive semidefinite; therefore $\mathbf{L}^{+}$ is also positive semidefinite.

B. Appendix: $\mathbf{L}^{+}$ is a kernel

In this section, we show that $\mathbf{L}^{+}$ is a kernel (see for instance [32]). Indeed, $\mathbf{L}^{+}$ is the matrix containing the inner products of the transformed vectors $\mathbf{x}'_i$:

$$\mathbf{x}_i'^{\mathrm{T}}\mathbf{x}'_j = (\mathbf{\Lambda}^{1/2}\mathbf{x}_i)^{\mathrm{T}}\mathbf{\Lambda}^{1/2}\mathbf{x}_j = \mathbf{x}_i^{\mathrm{T}}\mathbf{\Lambda}\mathbf{x}_j = \mathbf{e}_i^{\mathrm{T}}\mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathrm{T}}\mathbf{e}_j = \mathbf{e}_i^{\mathrm{T}}\mathbf{L}^{+}\mathbf{e}_j = l_{ij}^{+}$$

Thus, if $\mathbf{X}'$ denotes the data matrix containing the coordinates of the nodes on each row, $\mathbf{X}' = [\mathbf{x}'_1, \mathbf{x}'_2, ..., \mathbf{x}'_n]^{\mathrm{T}}$, we have $\mathbf{L}^{+} = \mathbf{X}'(\mathbf{X}')^{\mathrm{T}}$ with elements $l_{ij}^{+} = \mathbf{x}_i'^{\mathrm{T}}\mathbf{x}'_j$.