Journal of Machine Learning Research 15 (2014) 3871-3914 Submitted 4/13; Revised 4/14; Published 11/14

Semi-Supervised Eigenvectors for Large-Scale Locally-Biased Learning

Toke J. Hansen TJHA@DTU.DK
Department of Applied Mathematics and Computer Science
Technical University of Denmark
Richard Petersens Plads, 2800 Lyngby, Denmark

Michael W. Mahoney MMAHONEY@STAT.BERKELEY.EDU
International Computer Science Institute and Dept. of Statistics
University of California
Berkeley, CA 94720-1776, USA

Editor: Mikhail Belkin

Abstract

In many applications, one has side information, e.g., labels that are provided in a semi-supervised manner, about a specific target region of a large data set, and one wants to perform machine learning and data analysis tasks "nearby" that prespecified target region. For example, one might be interested in the clustering structure of a data graph near a prespecified "seed set" of nodes, or one might be interested in finding partitions in an image that are near a prespecified "ground truth" set of pixels. Locally-biased problems of this sort are particularly challenging for popular eigenvector-based machine learning and data analysis tools. At root, the reason is that eigenvectors are inherently global quantities, thus limiting the applicability of eigenvector-based methods in situations where one is interested in very local properties of the data.

In this paper, we address this issue by providing a methodology to construct *semi-supervised eigenvectors* of a graph Laplacian, and we illustrate how these locally-biased eigenvectors can be used to perform *locally-biased machine learning*. These semi-supervised eigenvectors capture successively-orthogonalized directions of maximum variance, conditioned on being well-correlated with an input seed set of nodes that is assumed to be provided in a semi-supervised manner. We show that these semi-supervised eigenvectors can be computed quickly as the solution to a system of linear equations; and we also describe several variants of our basic method that have improved scaling properties. We provide several empirical examples demonstrating how these semi-supervised eigenvectors can be used to perform locally-biased learning; and we discuss the relationship between our results and recent machine learning algorithms that use global eigenvectors of the graph Laplacian.

Keywords: semi-supervised learning, spectral clustering, kernel methods, large-scale machine learning, local spectral methods, locally-biased learning

1. Introduction

In many applications, one has information about a specific target region of a large data set, and one wants to perform common machine learning and data analysis tasks "nearby" the pre-specified target region. In such situations, eigenvector-based methods such as those

that have been popular in machine learning in recent years tend to have serious difficulties. At root, the reason is that eigenvectors, e.g., of Laplacian matrices of data graphs, are inherently *global* quantities, and thus they might not be sensitive to very *local* information. Motivated by this, we consider the problem of finding a set of locally-biased vectors—we will call them *semi-supervised eigenvectors*—that inherit many of the "nice" properties that the leading nontrivial global eigenvectors of a graph Laplacian have—for example, that capture "slowly varying" modes in the data, that are fairly-efficiently computable, that can be used for common machine learning and data analysis tasks such as kernel-based and semi-supervised learning, etc.—so that we can perform what we will call *locally-biased machine learning* in a principled manner.

1.1 Locally-Biased Learning

By *locally-biased machine learning*, we mean that we have a data set, e.g., represented as a graph, and that we have information, e.g., given in a semi-supervised manner, that certain "regions" of the data graph are of particular interest. In this case, we may want to focus predominantly on those regions and perform data analysis and machine learning, e.g., classification, clustering, ranking, etc., that is "biased toward" those pre-specified regions. Examples of this include the following.

- *Locally-biased community identification.* In social and information network analysis, one might have a small "seed set" of nodes that belong to a cluster or community of interest (Andersen and Lang, 2006; Leskovec et al., 2008); in this case, one might want to perform link or edge prediction, or one might want to "refine" the seed set in order to find other nearby members.

- *Locally-biased image segmentation.* In computer vision, one might have a large corpus of images along with a "ground truth" set of pixels as provided by a face detection algorithm (Eriksson et al., 2007; Mahoney et al., 2012; Maji et al., 2011); in this case, one might want to segment entire heads from the background for all the images in the corpus in an automated manner.

- *Locally-biased neural connectivity analysis.* In functional magnetic resonance imaging applications, one might have small sets of neurons that "fire" in response to some external experimental stimulus (Norman et al., 2006); in this case, one might want to analyze the subsequent temporal dynamics of stimulation of neurons that are "nearby," either in terms of connectivity topology or functional response, members of the original set.

In each of these examples, the data are modeled by a graph—which is either "given" from the application domain or is "constructed" from feature vectors obtained from the application domain—and one has information that can be viewed as semi-supervised in the sense that it consists of exogeneously-specified "labels" for the nodes of the graph. In addition, there are typically a relatively-small number of labels and one is interested in obtaining insight about the data graph nearby those labels.

These examples present considerable challenges for standard global spectral techniques and other traditional eigenvector-based methods. (Such eigenvector-based methods have received attention in a wide range of machine learning and data analysis applications in recent

years. They have been useful, for example, in non-linear dimensionality reduction Belkin and Niyogi 2003; Coifman et al. 2005; in kernel-based machine learning Schölkopf and Smola 2001; in Nyström-based learning methods Williams and Seeger 2001; Talwalkar and Rostamizadeh 2010; spectral partitioning Pothen et al. 1990; Shi and Malik 2000; Ng et al. 2001, and so on.) At root, the reason is that eigenvectors are inherently global quantities, thus limiting their applicability in situations where one is interested in very local properties of the data. That is, very local information can be "washed out" and essentially invisible to these globally-optimal vectors. For example, a sparse cut in a graph may be poorly correlated with the second eigenvector and thus invisible to a method based only on eigenvector analysis. Similarly, if one has semi-supervised information about a specific target region in the graph, as in the above examples, one might be interested in finding clusters near this prespecified local region in a semi-supervised manner; but this local region might be essentially invisible to a method that uses only global eigenvectors. Finally, one might be interested in using kernel-based methods to find "local correlations" or to regularize with respect to a "local dimensionality" in the data, but this local information might be destroyed in the process of constructing kernels with traditional kernel-based methods.

1.2 Semi-Supervised Eigenvectors

In this paper, we provide a methodology to construct what we will call *semi-supervised eigenvectors* of a graph Laplacian; and we illustrate how these locally-biased eigenvectors (locally-biased in the sense that they will be well-correlated with the input seed set of nodes or that most of their "mass" will be on nodes that are "near" that seed set) inherit many of the properties that make the leading nontrivial global eigenvectors of the graph Laplacian so useful in applications. In order to make this method useful, there should ideally be a "knob" that allows us to interpolate between very local and the usual global eigenvectors, depending on the application at hand; we should be able to use these vectors in common machine learning pipelines to perform common machine learning tasks; and the intuitions that make the leading k nontrivial global eigenvectors of the graph Laplacian useful should, to the extent possible, extend to the locally-biased setting. To achieve this, we will formulate an optimization ansatz that is a variant of the usual global spectral graph partitioning optimization problem that includes a natural locality constraint as well as an orthogonality constraint, and we will iteratively solve this problem.

In more detail, assume that we are given as input a (possibly weighted) data graph $G = (V, E)$, an indicator vector s of a small "seed set" of nodes, a *correlation parameter* $\kappa \in [0, 1]$, and a positive integer k. Then, informally, we would like to construct k vectors that satisfy the following bicriteria: first, each of these k vectors is well-correlated with the input seed set; and second, those k vectors describe successively-orthogonalized directions of maximum variance, in a manner analogous to the leading k nontrivial global eigenvectors of the graph Laplacian. (We emphasize that the seed set s of nodes, the integer k, and the correlation parameter κ are part of the input; and thus they should be thought of as being available in a semi-supervised manner.) Somewhat more formally, our main algorithm, Algorithm 1 in Section 3, returns as output k semi-supervised eigenvectors; each of these is the solution to an optimization problem of the form of Generalized LocalSpectral in Figure 1, and thus each "captures" (say) κ/k of the correlation with the seed set. Our

main theoretical result, described in Section 3, states that these vectors define successively-orthogonalized directions of maximum variance, conditioned on being κ/k-well-correlated with an input seed set s; and that each of these k semi-supervised eigenvectors can be computed quickly as the solution to a system of linear equations. To extend the practical applicability of this basic result, we will in Section 4 describe several heuristic extensions of our basic framework that will make it easier to apply the method of semi-supervised eigenvectors at larger size scales. These extensions involve using the so-called Nyström method, computing one locally-biased eigenvector and iteratively "peeling off" successive components of interest, as well as performing random walks that are "local" in a stronger sense than our basic method considers.

Finally, in order to illustrate how the method of semi-supervised eigenvectors performs in practice, we also provide a detailed empirical evaluation using a wide range of both small-scale as well as larger-scale data.

1.3 Related Work

From a technical perspective, the work most closely related to ours is the recently-developed "local spectral method" of Mahoney et al. (2012). The original algorithm of Mahoney et al. (2012) introduced a methodology to construct a locally-biased version of the *leading* non-trivial eigenvector of a graph Laplacian and also showed (theoretically and empirically in a social network analysis application) that that the resulting vector could be used to partition a graph in a locally-biased manner. From this perspective, our extension incorporates a natural orthogonality constraint that successive vectors need to be orthogonal to previous vectors. Subsequent to the work of Mahoney et al. (2012), Maji et al. (2011) applied the algorithm of Mahoney et al. (2012) to the problem of finding locally-biased cuts in a computer vision application. Similar ideas have also been applied somewhat differently. For example, Andersen and Lang (2006) use locally-biased random walks, e.g., short random walks starting from a small seed set of nodes, to find clusters and communities in graphs arising in Internet advertising applications; Leskovec et al. (2008) used locally-biased random walks to characterize the local and global clustering structure of a wide range of social and information networks; and Joachims (2003) developed the Spectral Graph Transducer, which performs transductive learning via spectral graph partitioning.

The objectives in both (Joachims, 2003) and (Mahoney et al., 2012) are constrained eigenvalue problems that can be solved by finding the smallest eigenvalue of an asymmetric generalized eigenvalue problem; but in practice this procedure can be highly unstable (Gander et al., 1989). The algorithm of Joachims (2003) reduces the instabilities by performing all calculations in a subspace spanned by the d smallest eigenvectors of the graph Laplacian; whereas the algorithm of Mahoney et al. (2012) performs a binary search, exploiting the monotonic relationship between a control parameter and the corresponding Lagrange multiplier. The form of our optimization problem also has similarities to other work in computer vision applications: e.g., (Yu and Shi, 2002) and (Eriksson et al., 2007) find good conductance clusters subject to a set of linear constraints.

In parallel, Belkin and Niyogi (2003) and a large body of subsequent work including (Coifman et al., 2005) used (the usual global) eigenvectors of the graph Laplacian to perform dimensionality reduction and data representation, in unsupervised and semi-supervised

settings (Tenenbaum et al., 2000; Roweis and Saul, 2000; Zhou et al., 2004). Typically, these methods construct some sort of local neighborhood structure around each data point, and they optimize some sort of global objective function to go "from local to global" (Saul et al., 2006). In some cases, these methods can be understood in terms of data drawn from an hypothesized manifold (Belkin and Niyogi, 2008), and more generally they have proven useful for denoising and learning in semi-supervised settings (Belkin and Niyogi, 2004; Belkin et al., 2006). These methods are based on spectral graph theory (Chung, 1997); and thus many of these methods have a natural interpretation in terms of diffusions and kernel-based learning (Schölkopf and Smola, 2001; Kondor and Lafferty, 2002; Szummer and Jaakkola, 2002; Chapelle et al., 2003; Ham et al., 2004). These interpretations are important for the usefulness of these global eigenvector methods in a wide range of applications. As we will see, many (but not all) of these interpretations can be ported to the "local" setting, an observation that was made previously in a different context (Cucuringu and Mahoney, 2011).

Many of these diffusion-based spectral methods also have a natural interpretation in terms of spectral ranking (Vigna, 2009). "Topic sensitive" and "personalized" versions of these spectral ranking methods have also been studied (Haveliwala, 2003; Jeh and Widom, 2003); and these were the motivation for diffusion-based methods to find locally-biased clusters in large graphs (Spielman and Teng, 2004; Andersen et al., 2006; Mahoney et al., 2012). Our optimization ansatz is a generalization of the linear equation formulation of the PageRank procedure (Page et al., 1999; Mahoney et al., 2012; Vigna, 2009); and its solution involves Laplacian-based linear equation solving, which has been suggested as a primitive is of more general interest in large-scale data analysis (Teng, 2010).

1.4 Outline of the Paper

In the next section, Section 2, we will provide notation and some background and discuss related work. Then, in Section 3 we will present our main algorithm and our main theoretical result justifying the algorithm; and in Section 4 we will present several extensions of our basic method that will help for certain larger-scale applications of the method of semi-supervised eigenvectors. In Section 5, we present an empirical analysis, including both toy data to illustrate how the "knobs" of our method work, as well as applications to realistic machine learning and data analysis problems.

2. Background and Notation

Let $G = (V, E, w)$ be a connected undirected graph with $n = |V|$ vertices and $m = |E|$ edges, in which edge $\{i, j\}$ has weight w_{ij}. For a set of vertices $S \subseteq V$ in a graph, the *volume of* S is $\mathrm{vol}(S) \stackrel{\mathrm{def}}{=} \sum_{i \in S} d_i$, in which case the *volume of the graph* G is $\mathrm{vol}(G) \stackrel{\mathrm{def}}{=} \mathrm{vol}(V) = 2m$. In the following, $A_G \in \mathbb{R}^{V \times V}$ will denote the adjacency matrix of G, while $D_G \in \mathbb{R}^{V \times V}$ will denote the diagonal degree matrix of G, i.e., $D_G(i,i) = d_i = \sum_{\{i,j\} \in E} w_{ij}$, the weighted degree of vertex i. The Laplacian of G is defined as $L_G \stackrel{\mathrm{def}}{=} D_G - A_G$. (This is also called the combinatorial Laplacian, in which case the normalized Laplacian of G is $\mathcal{L}_G \stackrel{\mathrm{def}}{=} D_G^{-1/2} L_G D_G^{-1/2}$.)

The Laplacian is the symmetric matrix having quadratic form $x^T L_G x = \sum_{ij \in E} w_{ij}(x_i - x_j)^2$, for $x \in \mathbb{R}^V$. This implies that L_G is positive semidefinite and that the all-one vector $1 \in \mathbb{R}^V$ is the eigenvector corresponding to the smallest eigenvalue 0. The generalized eigenvalues of $L_G x = \lambda_i D_G x$ are $0 = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_N$. We will use v_2 to denote smallest non-trivial eigenvector, i.e., the eigenvector corresponding to λ_2; v_3 to denote the next eigenvector; and so on. We will overload notation to use $\lambda_2(A)$ to denote the smallest non-zero generalized eigenvalue of A with respect to D_G. Finally, for a matrix A, let A^+ denote its (uniquely defined) Moore-Penrose pseudoinverse. For two vectors $x, y \in \mathbb{R}^n$, and the degree matrix D_G for a graph G, we define the *degree-weighted inner product* as $x^T D_G y \stackrel{\text{def}}{=} \sum_{i=1}^n x_i y_i d_i$. In particular, if a vector x has unit norm, then $x^T D_G x = 1$. Given a subset of vertices $S \subseteq V$, we denote by 1_S the indicator vector of S in $\mathbb{R}^V$ and by 1 the vector in $\mathbb{R}^V$ having all entries set equal to 1.

3. Optimization Approach to Semi-Supervised Eigenvectors

In this section, we provide our main technical results: a motivation and statement of our optimization ansatz; our main algorithm for computing semi-supervised eigenvectors; and an analysis that our algorithm computes solutions of our optimization ansatz.

3.1 Motivation for the Program

Recall the optimization perspective on how one computes the leading nontrivial global eigenvectors of the normalized Laplacian $\mathcal{L}_G$ or, equivalently, of the leading nontrivial generalized eigenvectors of L_G. The first nontrivial eigenvector v_2 is the solution to the problem GlobalSpectral that is presented on the left of Figure 1. Equivalently, although GlobalSpectral is a non-convex optimization problem, strong duality holds for it and it's solution may be computed as v_2, the leading nontrivial generalized eigenvector of L_G. (In this case, the value of the objective is λ_2, and global spectral partitioning involves then doing a "sweep cut" over this vector and appealing to Cheeger's inequality.) The next eigenvector v_3 is the solution to GlobalSpectral, augmented with the constraint that $x^T D_G v_2 = 0$; and in general the t^{th} generalized eigenvector of L_G is the solution to GlobalSpectral, augmented with the constraints that $x^T D_G v_i = 0$, for $i \in \{2, \ldots, t-1\}$. Clearly, this set of constraints and the constraint $x^T D_G 1 = 0$ can be written as $x^T D_G X = 0$, where 0 is a $(t-1)$-dimensional all-zeros vector, and where X is an $n \times (t-1)$ orthogonal matrix whose i^{th} column equals v_i (where $v_1 = 1$, the all-ones vector, is the first column of X).

Also presented in Figure 1 is LocalSpectral, which includes a constraint that the solution be well-correlated with an input seed set. This LocalSpectral optimization problem was introduced in Mahoney et al. (2012), where it was shown that the solution to LocalSpectral may be interpreted as a locally-biased version of the second eigenvector of the Laplacian.[1] In particular, although LocalSpectral is not convex, it's solution can be computed efficiently as the solution to a set of linear equations that generalize the popular

1. In Mahoney et al. (2012), the locality constraint was actually a quadratic constraint, and thus a somewhat involved analysis was required. In retrospect, given the form of the solution, and in light of the discussion below, it is clear that the quadratic part was not "real," and thus we present this simpler form of LocalSpectral here. This should make the connections with our Generalized LocalSpectral objective more immediate.

GlobalSpectral	LocalSpectral	Generalized LocalSpectral
$\begin{aligned} \text{minimize} \quad & x^T L_G x \\ \text{s.t} \quad & x^T D_G x = 1 \\ & x^T D_G 1 = 0 \end{aligned}$	$\begin{aligned} \text{minimize} \quad & x^T L_G x \\ \text{s.t} \quad & x^T D_G x = 1 \\ & x^T D_G 1 = 0 \\ & x^T D_G s \geq \sqrt{\kappa} \end{aligned}$	$\begin{aligned} \text{minimize} \quad & x^T L_G x \\ \text{s.t} \quad & x^T D_G x = 1 \\ & x^T D_G X = 0 \\ & x^T D_G s \geq \sqrt{\kappa} \end{aligned}$

Figure 1: Left: The usual GlobalSpectral partitioning optimization problem; the vector achieving the optimal solution is v_2, the leading nontrivial generalized eigenvector of L_G with respect to D_G. Middle: The LocalSpectral optimization problem, which was originally introduced in Mahoney et al. (2012); for $\kappa = 0$, this coincides with the usual global spectral objective, while for $\kappa > 0$, this produces solutions that are biased toward the seed vector s. Right: The Generalized LocalSpectral optimization problem we introduce that includes both the locality constraint and a more general orthogonality constraint. Our main algorithm for computing semi-supervised eigenvectors will iteratively compute the solution to Generalized LocalSpectral for a sequence of X matrices. In all three cases, the optimization variable is $x \in \mathbb{R}^n$.

Personalized PageRank procedure; in addition, by performing a sweep cut and appealing to a variant of Cheeger's inequality, this locally-biased eigenvector can be used to perform locally-biased spectral graph partitioning (Mahoney et al., 2012).

3.2 Our Main Algorithm

We will formulate the problem of computing semi-supervised vectors in terms of a primitive optimization problem of independent interest. Consider the Generalized LocalSpectral optimization problem, as shown in Figure 1. For this problem, we are given a graph $G = (V, E)$, with associated Laplacian matrix L_G and diagonal degree matrix D_G; an indicator vector s of a small "seed set" of nodes; a *correlation parameter* $\kappa \in [0, 1]$; and an $n \times \nu$ constraint matrix X that may be assumed to be an orthogonal matrix. We will assume (without loss of generality) that s is properly normalized and orthogonalized so that $s^T D_G s = 1$ and $s^T D_G 1 = 0$. While s can be a general unit vector orthogonal to 1, it may be helpful to think of s as the indicator vector of one or more vertices in V, corresponding to the target region of the graph.

In words, the problem Generalized LocalSpectral asks us to find a vector $x \in \mathbb{R}^n$ that minimizes the variance $x^T L_G x$ subject to several constraints: that x is unit length; that x is orthogonal to the span of X; and that x is $\sqrt{\kappa}$-well-correlated with the input seed set vector s. In our application of Generalized LocalSpectral to the computation of semi-supervised eigenvectors, we will iteratively compute the solution to Generalized LocalSpectral, updating X to contain the already-computed semi-supervised eigenvectors. That is, to compute the first semi-supervised eigenvector, we let $X = 1$, i.e., the

n-dimensional all-ones vector, which is the trivial eigenvector L_G, in which case X is an $n \times 1$ matrix; and to compute each subsequent semi-supervised eigenvector, we let the columns of X consist of 1 and the other semi-supervised eigenvectors found in each of the previous iterations.

To show that Generalized LocalSpectral is efficiently-solvable, note that it is a quadratic program with only one quadratic constraint and one linear equality constraint.[2] In order to remove the equality constraint, which will simplify the problem, let's change variables by defining the $n \times (n-\nu)$ matrix F as

$$\{x : X^T D_G x = 0\} = \{x : x = F\hat{x}\}.$$

That is, F is a span for the null space of X^T; and we will take F to be an orthogonal matrix. In particular, this implies that $F^T F$ is an $(n-\nu) \times (n-\nu)$ Identity and FF^T is an $n \times n$ Projection. Then, with respect to the $\hat{x}$ variable, Generalized LocalSpectral becomes

$$\begin{aligned} \underset{y}{\text{minimize}} \quad & \hat{x}^T F^T L_G F y \\ \text{subject to} \quad & \hat{x}^T F^T D_G F \hat{x} = 1, \\ & \hat{x}^T F^T D_G s \geq \sqrt{\kappa}. \end{aligned} \tag{1}$$

In terms of the variable x, the solution to this optimization problem is of the form

$$\begin{aligned} x^* &= cF \left(F^T \left(L_G - \gamma D_G\right) F\right)^+ F^T D_G s \\ &= c \left(FF^T \left(L_G - \gamma D_G\right) FF^T\right)^+ D_G s, \end{aligned} \tag{2}$$

for a normalization constant $c \in (0, \infty)$ and for some γ that depends on $\sqrt{\kappa}$. The second line follows from the first since F is an $n \times (n-\nu)$ orthogonal matrix. This so-called "S-procedure" is described in greater detail in Chapter 5 and Appendix B of (Boyd and Vandenberghe, 2004). The significance of this is that, although it is a non-convex optimization problem, the Generalized LocalSpectral problem can be solved by solving a linear equation, in the form given in Eqn. (2).

Returning to our problem of computing semi-supervised eigenvectors, recall that, in addition to the input for the Generalized LocalSpectral problem, we need to specify a positive integer k that indicates the number of vectors to be computed. In the simplest case, we would assume that we would like the correlation to be "evenly distributed" across all k vectors, in which case we will require that each vector is $\sqrt{\kappa/k}$-well-correlated with the input seed set vector s; but this assumption can easily be relaxed, and thus Algorithm 1 is formulated more generally as taking a k-dimensional vector $\kappa = [\kappa_1, \ldots, \kappa_k]^T$ of correlation coefficients as input.

To compute the first semi-supervised eigenvector, we will let $X = 1$, the all-ones vector, in which case the first nontrivial semi-supervised eigenvector is

$$x_1^* = c \left(L_G - \gamma_1 D_G\right)^+ D_G s, \tag{3}$$

2. Alternatively, note that it is an example of an constrained eigenvalue problem (Gander et al., 1989). We thank the numerous individuals who pointed this out to us subsequent to our dissemination of the original version of this paper.

where γ_1 is chosen to saturate the part of the correlation constraint along the first direction. (Note that the projections FF^T from Eqn. 2 are not present in Eqn. 3 since by design $s^T D_G 1 = 0$.) That is, to find the correct setting of γ_1, it suffices to perform a binary search over the possible values of γ_1 in the interval $(-\mathrm{vol}(G), \lambda_2(G))$ until the correlation constraint is satisfied, that is, until $(s^T D_G x_1)^2$ is sufficiently close to κ_1.

To compute subsequent semi-supervised eigenvectors, i.e., at steps $t = 2, \ldots, k$ if one ultimately wants a total of k semi-supervised eigenvectors, then one lets X be the $n \times t$ matrix of the form

$$X = [1, x_1^*, \ldots, x_{t-1}^*], \tag{4}$$

where $x_1^*, \ldots, x_{t-1}^*$ are successive semi-supervised eigenvectors; and the projection matrix FF^T is of the form

$$FF^T = I - D_G X (X^T D_G D_G X)^{-1} X^T D_G,$$

due to the the degree-weighted inner norm.

Then, by Eqn. (2), the t^{th} semi-supervised eigenvector takes the form

$$x_t^* \quad = \quad c \left(FF^T (L_G - \gamma_t D_G) FF^T \right)^+ D_G s.$$

Algorithm 1 Main algorithm to compute semi-supervised eigenvectors

Require: $L_G, D_G, s, \kappa = [\kappa_1, \ldots, \kappa_k]^T, \epsilon$ such that $s^T D_G 1 = 0$, $s^T D_G s = 1$, $\kappa^T 1 \leq 1$
1: $X = [1]$
2: **for** $t = 1$ to k **do**
3: $\quad FF^T \leftarrow I - D_G X (X^T D_G D_G X)^{-1} X^T D_G$
4: $\quad \top \leftarrow \lambda_2$ where $FF^T L_G FF^T v_2 = \lambda_2 FF^T D_G FF^T v_2$
5: $\quad \perp \leftarrow -\mathrm{vol}(G)$
6: $\quad$ **repeat**
7: $\quad\quad \gamma_t \leftarrow (\perp + \top)/2$ (Binary search over γ_t)
8: $\quad\quad x_t \leftarrow (FF^T (L_G - \gamma_t D_G) FF^T)^+ FF^T D_G s$
9: $\quad\quad$ Normalize x_t such that $x_t^T D_G x_t = 1$
10: $\quad\quad$ **if** $(x_t^T D_G s)^2 > \kappa_t$ **then** $\perp \leftarrow \gamma_t$ **else** $\top \leftarrow \gamma_t$ **end if**
11: $\quad$ **until** $\|(x_t^T D_G s)^2 - \kappa_t\| \leq \epsilon$ **or** $\|(\perp + \top)/2 - \gamma_t\| \leq \epsilon$
12: $\quad$ Augment X with x_t^* by letting $X = [X, x_t^*]$.
13: **end for**

In more detail, Algorithm 1 presents pseudo-code for our main algorithm for computing semi-supervised eigenvectors. The algorithm takes as input a graph $G = (V, E)$, a seed set s (which could be a general vector $s \in \mathbb{R}^n$, subject for simplicity to the normalization constraints $s^T D_G 1 = 0$ and $s^T D_G s = 1$, but which is most easily thought of as an indicator vector for the local "seed set" of nodes), a number k of vectors we want to compute, and a vector of locality parameters $(\kappa_1, \ldots, \kappa_k)$, where $\kappa_i \in [0, 1]$ and $\sum_{i=1}^k \kappa_i = 1$ (where, in the simplest case, one could choose $\kappa_i = \kappa/k$, $\forall i$, for some $\kappa \in [0, 1]$). Several things should be noted about our implementation of our main algorithm. First, as we will discuss in more detail below, we compute the projection matrix FF^T only *implicitly*. Second, a naïve approach to Eqn. (2) does not immediately lead to an efficient solution, since $D_G s$ will not be in the span of $(FF^T (L_G - \gamma D_G) FF^T)$, thus leading to a large residual. By changing variables so that $x = FF^T y$, the solution becomes

$$x_t^* \propto FF^T (FF^T (L_G - \gamma_t D_G) FF^T)^+ FF^T D_G s.$$

Since FF^T is a projection matrix, this expression is equivalent to

$$x_t^* \propto \left(FF^T(L_G - \gamma_t D_G)FF^T\right)^+ FF^T D_G s. \tag{5}$$

Third, regarding the solution x_i, we exploit that $FF^T(L_G - \gamma_i D_G)FF^T$ is an SPSD matrix, and we apply the conjugate gradient method, rather than computing the explicit pseudoinverse. That is, in the implementation we never explicitly represent the dense matrix FF^T, but instead we treat it as an operator and we simply evaluate the result of applying a vector to it on either side. Fourth, we use that λ_2 can never decrease (here we refer to λ_2 as the smallest non-zero eigenvalue of the modified matrix), so we only recalculate the upper bound for the binary search when an iteration saturates without satisfying $\|(x_t^T D_G s)^2 - \kappa_t\| \leq \epsilon$. Estimating the bound is critical for the semi-supervised eigenvectors to be able to interpolate all the way to the global eigenvectors of the graph, so in Section 3.4 we return to a discussion on efficient strategies for computing the leading nontrivial eigenvalue of L_G projected down onto the space perpendicular to the previously computed solutions.

From this discussion, it should be clear that Algorithm 1 solves the semi-supervised eigenvector problem by solving in an iterative manner optimization problems of the form of Generalized LocalSpectral; and that the running time of Algorithm 1 boils down to solving a sequence of linear equations.

3.3 Discussion of Our Main Algorithm

There is a natural "regularization" interpretation underlying our construction of semi-supervised eigenvectors. To see this, recall that the first step of our algorithm can be computed as the solution of a set of linear equations

$$x^* = c\,(L_G - \gamma D_G)^+ D_G s, \tag{6}$$

for some normalization constant c and some γ that can be determined by a binary search over $(-\mathrm{vol}(G), \lambda_2(G))$; and that subsequent steps compute the analogous quantity, subject to additional constraints that the solution be orthogonal to the previously-computed vectors. The quantity $(L_G - \gamma D_G)^+$ can be interpreted as a "regularized" version of the pseudoinverse of L, where $\gamma \in (-\infty, \lambda_2(G))$ serves as the regularization parameter. This interpretation has recently been made precise: Mahoney and Orecchia (2011) show that running a PageRank computation—as well as running other diffusion-based procedures—*exactly* optimizes a regularized version of the GlobalSpectral (or LocalSpectral, depending on the input seed vector) problem; and (Perry and Mahoney, 2011) provide a precise statistical framework justifying this.

The usual interpretation of PageRank involves "random walkers" who uniformly (or non-uniformly, in the case of Personalized PageRank) "teleport" with a probability $\alpha \in (0, 1)$. As described in (Mahoney et al., 2012), choosing $\alpha \in (0, 1)$ corresponds to choosing $\gamma \in (-\infty, 0)$. By rearranging Eqn. (6) as

$$\begin{aligned} x^* &= c\,((D_G - A_G) - \gamma D_G)^+ D_G s \\ &= \frac{c}{1-\gamma}\left(D_G - \frac{1}{1-\gamma}A_G\right)^+ D_G s \\ &= \frac{c}{1-\gamma} D_G^{-1}\left(I - \frac{1}{1-\gamma}A_G D_G^{-1}\right)^+ D_G s, \end{aligned}$$

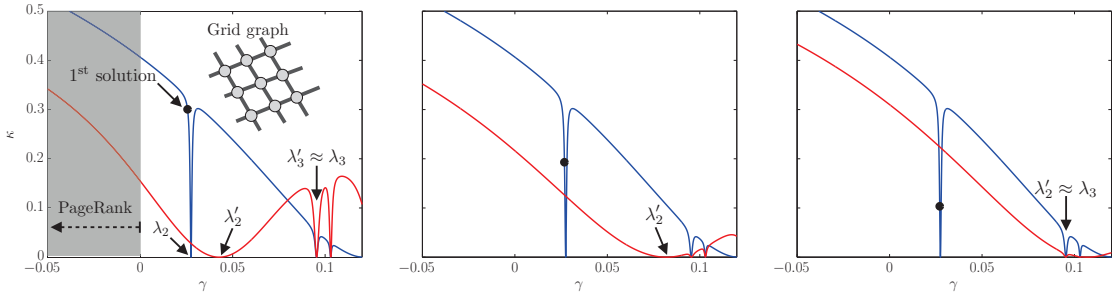
we recognize $A_G D_G^{-1}$ as the standard random walk matrix, and it becomes immediate that the solution based on random walkers,

$$x^* = \frac{c}{1-\gamma} D_G^{-1} \left(I + \sum_{i=1}^{\infty} \left(\frac{1}{1-\gamma} D_G^{-1} A_G \right)^i \right) D_G s,$$

is divergent for $\gamma > 0$. Since $\gamma = \lambda_2(G)$ corresponds to no regularization and $\gamma \to -\infty$ corresponds to heavy regularization, viewing this problem in terms of solving a linear equation is formally more powerful than viewing it in terms of random walkers. That is, while all possible values of the regularization parameter—and in particular the (positive) value $\lambda_2(\cdot)$—are achievable algorithmically by solving a linear equation, only values in $(-\infty, 0)$ are achievable by running a PageRank diffusion. In particular, if the optimal value of γ that saturates the κ-dependent locality constraint is negative, then running the PageRank diffusion could find it; otherwise, the "best" one could do will still not saturate the locality constraint, in which case some of the intended correlation is "unused."

An important technical and practical point has to do with the precise manner in which the i^{th} vector is well-correlated with the seed set s. In our formulation, this is captured by a *locality parameter* γ_i that is chosen (via a binary search) to "saturate" the correlation condition, i.e., so that the i^{th} vector is κ/k-well-correlated with the input seed set. As a general rule, successive γ_is need to be chosen that successive vectors are *less* well-localized around the input seed set. (Alternatively, depending on the application, one could choose this parameter so that successive γ_is are equal; but this will involve "sacrificing" some amount of the κ/k correlation, which will lead to the last or last few eigenvectors being very poorly-correlated with the input seed set. These tradeoffs will be described in more detail below.) Informally, if s is a seed set consisting of a small number of nodes that are "nearby" each other, then to maintain a given amount of correlation, we must "view" the graph over larger and larger size scales as we compute more and more semi-supervised eigenvectors. More formally, we need to let the value of the regularization parameter γ at the i^{th} round, we call it γ_i, vary for each $i \in \{1, \ldots, k\}$. That is, γ_i is not pre-specified, but it is chosen via a binary search over the region $(-\mathrm{vol}(G), \lambda_2(\cdot))$, where $\lambda_2(\cdot)$ is the leading nontrivial eigenvalue of L_G projected down onto the space perpendicular to the previously-computed vectors (which is in general larger than $\lambda_2(G)$). In this sense, our semi-supervised eigenvectors are both "locally-biased", in a manner that depends on the input seed vector and correlation parameter, and "regularized", in a manner that depends on the local graph structure.

To illustrate the previous discussion, Figure 2 considers the two-dimensional grid. In each subfigure, the blue curve shows the correlation with a single seed node as a function of γ for the leading semi-supervised eigenvector, and the black dot illustrates the value of γ for three different values of the locality parameter κ. This relationship between κ and γ is in general non-convex, but it is monotonic for $\gamma \in (-\mathrm{vol}(G), \lambda_2(G))$. The red curve in each subfigure shows the decay for the second semi-supervised eigenvector. Recall that it is perpendicular to the first semi-supervised eigenvector, that the decay is monotonic for $\gamma \in (-\mathrm{vol}(G), \lambda_2'(G))$, and that $\lambda_2 < \lambda_2' \leq \lambda_3$. In Figure 2(a), the first semi-supervised eigenvector is not "too" close to λ_2, and so λ_2' (i.e., the second eigenvalue of the next semi-supervised eigenvector) increases just slightly. In Figure 2(b), we consider a locality



(a) $(\boldsymbol{x}_1^T \boldsymbol{D}_G \boldsymbol{s})^2 = 0.3$ (b) $(\boldsymbol{x}_1^T \boldsymbol{D}_G \boldsymbol{s})^2 = 0.2$ (c) $(\boldsymbol{x}_1^T \boldsymbol{D}_G \boldsymbol{s})^2 = 0.1$

Figure 2: Interplay between the γ parameter and the correlation κ that a semi-supervised eigenvector has with a seed $\boldsymbol{s}$ on a two-dimensional grid. In Figure 2(a)-2(c), we vary the locality parameter for the leading semi-supervised eigenvector, which in each case leads to a value of γ which is marked by the black dot on the blue curve. This allows us to illustrate the influence on the relationship between γ and κ on the next semi-supervised eigenvector. Figure 2(a) also highlights the range ($\gamma < 0$) in which Personalized PageRank can be used for computing solutions to semi-supervised eigenvectors.

parameter that leads to a value of γ that is closer to λ_2, thereby increasing the value of λ_2'. Finally, in Figure 2(c), the locality parameter is such that the leading semi-supervised eigenvector almost coincides with $\boldsymbol{v}_2$; this results in $\lambda_2' \approx \lambda_3$, as required if we were to compute the global eigenvectors.

3.4 Bounding the Binary Search

For the following derivations it is more convenient to consider the normalized graph Laplacian, in which case we define the first solution as

$$y_1 = c\,(\mathcal{L}_G - \gamma_1 I)^+ D_G^{1/2} s, \tag{7}$$

where $x_1^* = D_G^{-1/2} y_1$. This approach is convenient since the projection operator with null space defined by previous solutions can be expressed as $FF^T = I - YY^T$, assuming that $Y^T Y = 1$. That is, Y is of the form

$$Y = [D_G^{1/2}, y_1^*, \ldots, y_{t-1}^*],$$

where y_i^* are successive solutions to Eqn. (7). In the following the type of projection operator will be implicit from the context, i.e., when working with the combinatorial graph Laplacian $FF^T = I - D_G X (X^T D_G D_G X)^{-1} X^T D_G$, whereas for the normalized graph Laplacian $FF^T = I - YY^T$.

For the normalized graph Laplacian $\mathcal{L}_G$, the eigenvalues of $\mathcal{L}_G v = \lambda v$ equal the eigenvalues of the generalized eigenvalue problem $L_G v = \lambda D_G v$. The binary search employed in Algorithm 1 uses a monotonic relationship between the $\gamma \in (-\mathrm{vol}(G), \lambda_2(\cdot))$ parameter and the

correlation with the seed $x^T D_G s$, that can be deduced from the KKT-conditions (Mahoney et al., 2012). Note, that if the upper bound for the binary search $\top = \lambda_2(FF^T\mathcal{L}_G FF^T)$ is not determined with sufficient precision, the search will (if we underestimate $\top$) fail to satisfy the constraint, or (if we overestimate $\top$) fail to converge because the monotonic relationship no longer hold.

By Lemma 1 in Appendix A it follows that $\lambda_2(FF^T\mathcal{L}_G FF^T) = \lambda_2(\mathcal{L}_G + \omega YY^T)$ when $\omega \to \infty$. Since the latter term is a sum of two PSD matrices, the value of the upper bound can only increase as stated by Lemma 2 in Appendix A. This is an important property, because if we do not recalculate $\top$, the previous value is guaranteed to be an underestimate, meaning that the objective will remain convex. Thus, it may be more efficient to first recompute $\top$ when the binary search fails to satisfy $(x^T D_G s)^2 = \kappa$, meaning that $\top$ must be recomputed to increase the search range.

We compute the value for the upper bound of the binary search by transforming the problem in such a way that we can determine the greatest eigenvalue of a new system (fast and robust), and from that, deduce the new value of $\top = \lambda_2(FF^T\mathcal{L}_G FF^T)$. We do so by expanding the expression as

$$\begin{aligned} FF^T\mathcal{L}_G FF^T &= FF^T\left(I - D_G^{-1/2}A_G D_G^{-1/2}\right)FF^T \\ &= FF^T - FF^T D_G^{-1/2}A_G D_G^{-1/2}FF^T \\ &= I - \left(FF^T D_G^{-1/2}A_G D_G^{-1/2}FF^T + YY^T\right). \end{aligned}$$

Since all columns of Y will be eigenvectors of $FF^T\mathcal{L}_G FF^T$ with zero eigenvalue, these will all be eigenvectors of $FF^T D_G^{-1/2}A_G D_G^{-1/2}FF^T + YY^T$ with eigenvalue 1. Hence, the largest algebraic eigenvalue $\lambda_{\mathrm{LA}}(FF^T D_G^{-1/2}A_G D_G^{-1/2}FF^T)$ can be used to compute the upper bound for the binary search as

$$\top = \lambda_2(FF^T\mathcal{L}_G FF^T) = 1 - \lambda_{\mathrm{LA}}(FF^T D_G^{-1/2}A_G D_G^{-1/2}FF^T). \tag{8}$$

The reason for not considering the largest magnitude eigenvalue, is that A_G may be indefinite. Finally, with respect to our implementation we emphasize that FF^T is used as a projection operator, and not represented explicitly.

4. Extension of Our Main Algorithm And Implementation Details

In this section, we present two variants of our main algorithm that are more well-suited for very large-scale applications; the first uses a column-based low-rank approximation, and the second uses random walk ideas. In Section 4.1, we describe how to use the Nyström method, which constructs a low-rank approximation to the kernel matrix by sampling columns, to construct a general solution for semi-supervised eigenvectors, where the low-rankness is exploited for very efficient computation. Then, in Section 4.2, we describe a "Push-peeling heuristic," based on the efficient Push algorithm by Andersen et al. (2006). The basic idea is that if, rather than iteratively computing locally-biased semi-supervised eigenvectors using the procedure described in Algorithm 1, we instead compute solutions to LocalSpectral and then construct the semi-supervised eigenvectors by "projecting away" pieces of these solutions, then we can take advantage of local random walks that have improved algorithmic properties.

4.1 A Nyström-Based Low-rank Approach

Here we describe the use of the recently-popular Nyström method to speed up the computation of semi-supervised eigenvectors. We do so by considering how a low-rank decomposition can be exploited to yield solutions to the Generalized LocalSpectral objective in Figure 1, where the running time largely depends on a matrix-vector product. These methods are most appropriate when the kernel matrix is reasonably well-approximated by a low-rank matrix (Drineas and Mahoney, 2005; Gittens and Mahoney, 2012; Williams and Seeger, 2000).

Given some low-rank approximation $\mathcal{L}_G \approx I - V\Lambda V^T$, we apply the Woodbury matrix identity, and we derive an explicit solution for the leading semi-supervised eigenvector

$$\begin{aligned} y_1 &\approx c\left((1-\gamma)I - V\Lambda V^T\right)^+ D_G^{1/2}s \\ &\approx c\left(\frac{1}{1-\gamma}I + \frac{1}{(1-\gamma)^2}V\left(\Lambda^{-1} - \frac{1}{1-\gamma}I\right)^{-1}V^T\right)D_G^{1/2}s \\ &\approx \frac{c}{1-\gamma}\left(I + V\Sigma V^T\right)D_G^{1/2}s, \end{aligned}$$

where $\Sigma_{ii} = \frac{1}{\frac{1-\gamma}{\lambda_i}-1}$. In order to compute efficiently the subsequent semi-supervised eigenvectors we must accommodate for the projection operator $FF^T = I - YY^T$, while yet exploiting the explicit closed-form inverse $(\mathcal{L}_G - \gamma I)^+ \approx \frac{1}{1-\gamma}\left(I + V\Sigma V^T\right)$. However, the projection operator complicates the expression, since the previous solution can be spanned by multiple global eigenvectors, so leveraging from the low-rank decomposition is more difficult for the inverse $(FF^T(\mathcal{L}_G - \gamma I)FF^T)^+$.

Conveniently, we can decouple the projection operator by treating the orthogonality constraint using a Lagrangian approach, such that the solution can be expressed as

$$y_t = c\left(\mathcal{L}_G - \gamma I + \omega YY^T\right)^+ D_G^{1/2}s,$$

where $\omega \geq 0$ denotes the associated Lagrange multiplier, and where the sign is deduced from the KKT conditions. Applying the Woodbury matrix identity is now straightforward

$$\left(P_\gamma + \omega YY^T\right)^+ = {P_\gamma}^+ - \omega {P_\gamma}^+ Y\left(I + \omega Y^T {P_\gamma}^+ Y\right)^+ Y^T {P_\gamma}^+,$$

where for notational convenience we have introduced $P_\gamma = \mathcal{L}_G - \gamma I$. By decomposing $Y^T {P_\gamma}^+ Y$ with an eigendecomposition USU^T the equation simplifies as follows

$$\begin{aligned} \left(P_\gamma + \omega YY^T\right)^+ &= {P_\gamma}^+ - \omega {P_\gamma}^+ Y\left(I + \omega USU^T\right)^+ Y^T {P_\gamma}^+ \\ &= {P_\gamma}^+ - {P_\gamma}^+ YU\Omega U^T Y^T {P_\gamma}^+, \end{aligned}$$

where $\Omega_{ii} = \frac{1}{\frac{1}{\omega}+S_{ii}}$. Note how this result gives a well defined way of controlling the amount of "orthogonality", and by Lemma 1 in Appendix A, we get exact orthogonality in the limit of $\omega \to \infty$, in which case the expression simplifies to

$$\left(P_\gamma + \omega YY^T\right)^+ = {P_\gamma}^+ - {P_\gamma}^+ Y(Y^T {P_\gamma}^+ Y)^+ Y^T {P_\gamma}^+.$$

Using the explicit expression for ${P_\gamma}^+$, the solution now only involves matrix-vector products and the inverse of a small matrix

$$y_t = c\left({P_\gamma}^+ - {P_\gamma}^+ Y (Y^T {P_\gamma}^+ Y)^+ Y^T {P_\gamma}^+\right) D_G^{1/2} s. \tag{9}$$

To conclude this section, let us also consider how we can optimize the efficiency of the calculation of $\lambda_2(FF^T\mathcal{L}_G FF^T)$ used for bounding the binary search in Algorithm 1. According to Eqn. (8) the bound can be calculated efficiently as $\top = 1-\lambda_{\mathrm{LA}}(FF^T D_G^{-1/2} A_G D_G^{-1/2} FF^T)$. However, by substituting with $D_G^{-1/2} A_G D_G^{-1/2} \approx V\Lambda V^T$, we can exploit low-rankness since

$$\top = 1 - \lambda_{\mathrm{LA}}(FF^T V\Lambda V^T FF^T) = 1 - \lambda_{\mathrm{LA}}(\Lambda^{1/2} V^T FF^T V \Lambda^{1/2}),$$

where the latter is a much smaller system.

4.2 A Push-Peeling Heuristic

Here we present a variant of our main algorithm that exploits the connections between diffusion-based procedures and eigenvectors, allowing semi-supervised eigenvectors to be efficiently computed for large networks. This is most well-known for the leading nontrivial eigenvectors of the graph Laplacian (Chung, 1997); but recent work has exploited these connections in the context of performing locally-biased spectral graph partitioning (Spielman and Teng, 2004; Andersen et al., 2006; Mahoney et al., 2012). In particular, we can compute the locally-biased vector using the first step of Algorithm 1, or alternatively we can compute it using a locally-biased random walk of the form used in (Spielman and Teng, 2004; Andersen et al., 2006). Here we present a heuristic that works by peeling off components from a solution to the PageRank problem, and by exploiting the regularization interpretation of γ, we can from these components obtain the subsequent semi-supervised eigenvectors.

Specifically, we focus on the Push algorithm by Andersen et al. (2006). This algorithm approximates the solution to PageRank very efficiently, by exploiting the local modifications that occur when the seed is highly concentrated. This makes our algorithm very scalable and applicable for large-scale data, since only the local neighborhood near the seed set will be touched by the algorithm. In comparison, by solving the linear system of equations we explicitly touch all nodes in the graph, even though most spectral rankings will be below the computational precision (Boldi and Vigna, 2011).

We adapt a similar notation as in Andersen et al. (2006) and start by defining the usual PageRank vector $\mathrm{pr}(\alpha, s_{\mathrm{pr}})$ as the unique solution of the linear system

$$\mathrm{pr}(\alpha, s_{\mathrm{pr}}) = \alpha s_{\mathrm{pr}} + (1-\alpha) A_G D_G^{-1} \mathrm{pr}(\alpha, s_{\mathrm{pr}}), \tag{10}$$

where α is the teleportation parameter, and s_{pr} is the sparse starting vector. For comparison, the push algorithm by Andersen et al. (2006) computes an approximate PageRank vector $\mathrm{pr}_\epsilon(\alpha', s_{\mathrm{pr}})$ for a slightly different system

$$\mathrm{pr}_\epsilon(\alpha', s_{\mathrm{pr}}) = \alpha' s_{\mathrm{pr}} + (1-\alpha') W \mathrm{pr}_\epsilon(\alpha', s_{\mathrm{pr}}),$$

where $W = \frac{1}{2}(I + A_G D_G^{-1})$ and not the usual random walk matrix AD_G^{-1} as used in Eqn. (10). However, these equations are only superficially different, and equivalent up to a change

of the respective teleportation parameter. Thus, it is straightforward to verify that these teleportation parameters and the γ parameter of Eqn. (6) are related as

$$\alpha = \frac{2\alpha'}{1+\alpha'} \Leftrightarrow \alpha' = \frac{\alpha}{2-\alpha} \Leftrightarrow \alpha' = \frac{\gamma}{\gamma-2},$$

and that the leading semi-supervised eigenvector for $\gamma \in (-\infty, 0)$ can be approximated as

$$x_1^* \approx \frac{c}{-\gamma} D_G^{-1} \mathrm{pr}_\epsilon \left(\frac{\gamma}{\gamma-2}, D_G s \right).$$

To generalize subsequent semi-supervised eigenvectors to this diffusion based framework, we need to accommodate for the projection operator such that subsequent solutions can be expressed in terms of graph diffusions. By requiring distinct values of γ for all semi-supervised eigenvectors, we may use the solution for the leading semi-supervised eigenvector and then systematically "peel off" components, thereby obtaining the solution of one of the consecutive semi-supervised eigenvectors. By Lemma 5, in Appendix A the general solution in Eqn. (5) can be approximated by

$$x_t^* \approx c \left(I - XX^T D_G\right) \left(L_G - \gamma_t D_G\right)^+ D_G s, \tag{11}$$

under the assumption that all γ_k for $1 < k \leq t$ are sufficiently apart. If we think about γ_k as being distinct eigenvalues of the generalized eigenvalue problem $L_G x_k = \gamma_k D_G x_k$, then it is clear that Eqn. (11), correctly computes the sequence of generalized eigenvectors. This is explained by the fact that $(L_G - \gamma_t D_G)^+ D_G s$ can be interpreted as the first step of the Rayleigh quotient iteration, where γ_t is the estimate of the eigenvalue, and $D_G s$ is the estimate of the eigenvector. Given that the estimate of the eigenvalue is right, this algorithm will in the initial step compute the corresponding eigenvector, and the operator $\left(I - XX^T D_G\right)$ will be superfluous, as the global eigenvectors are already orthogonal in the degree-weighted norm. To quantify the failure modes of the approximation, let us consider what happens when γ_2 starts to approach γ_1. What constitutes the second solution for a particular value of γ_2 is the perpendicular component with respect to the projection onto the solution given by γ_1. As γ_2 approaches γ_1, this perpendicular part diminishes and the solution becomes ill-posed. Fortunately, we can easily detect such issues during the binary search in Algorithm 1, and in general the approximation has turned out to work very well in practice as our experimental results in Section 5 show.

In terms of the approximate PageRank vector $\mathrm{pr}_\epsilon(\alpha', s_{\mathrm{pr}})$, the general approximate solution takes the following form

$$x_t^* \approx c \left(I - XX^T D_G\right) D_G^{-1} \mathrm{pr}_\epsilon \left(\frac{\gamma_t}{\gamma_t - 2}, D_G s \right). \tag{12}$$

As already stated in Section 3.3, the impact of using a diffusion based procedure is that we cannot interpolate all the way to the global eigenvectors, and that the main challenge is that the solutions do not become too localized. The ϵ parameter of the Push algorithm controls the threshold for propagating mass away from the seed set and into the adjacent nodes in the graph. If the threshold is too high, the solution will be very localized and make it difficult to find more than a few semi-supervised eigenvectors, as characterized by

Lemma 3 in Appendix A, because the leading ones will then span the entire space of the seed set. As the choice of ϵ is important for the applicability of our algorithm, we will in Section 5 investigate the influence of this parameter on large data graphs.

To conclude this section, we consider an important implementation detail that have been omitted so far. In the work of Mahoney et al. (2012) the seed vector was defined to be perpendicular to the all-ones vector, and for the sake of consistency we have chosen to define it in the same way. The impact of projecting the seed set to a space that is perpendicular to the all-ones vector is that the resulting seed vector is no longer sparse, making the use of the Push algorithm in Eqn. (12) inefficient. The seed vector can, however, without loss of generality, be defined as $s \propto D_G^{-1/2}\left(I - v_0 v_0^T\right) s_0$ where s_0 is the sparse seed, and $v_0 \propto \operatorname{diag}\left(D_G^{1/2}\right)$ is the leading eigenvector of the normalized graph Laplacian (corresponding to the all-ones vector of the combinatorial graph Laplacian). If we substitute with this expression for the seed in Eqn. (12), it follows by plain algebra (see Appendix B) that

$$x_t^* \approx c\left(I - XX^T D_G\right)\left(D_G^{-1}\mathrm{pr}_\epsilon\left(\frac{\gamma_t}{\gamma_t - 2}, D_G^{1/2} s_0\right) - D_G^{-1/2} v_0 v_0^T s_0\right). \tag{13}$$

Now the Push algorithm is only defined on the sparse seed set making the the expression very scalable. Finally, the Push algorithm maintains a queue of high residual nodes that are yet to be processed. The order in which nodes are processed influences the overall running time, and in Boldi and Vigna (2011) preliminary experiments showed that a FIFO queue resulted in the best performance for large values of γ, as compared to a priority queue that scales logarithmically. For this reason we have chosen to use a FIFO queue in our implementation.

5. Empirical Results

In this section, we provide a detailed empirical evaluation of the method of semi-supervised eigenvectors and how it can be used for locally-biased machine learning. Our goal is two-fold: first, to illustrate how the "knobs" of the method work; and second, to illustrate the usefulness of the method in real applications. To do this, we consider several classes of data.

- **Toy data.** In Section 5.1, we consider one-dimensional examples of the popular "small world" model (Watts and Strogatz, 1998). This is a parameterized family of models that interpolates between low-dimensional grids and random graphs; and, as such, it allows us to illustrate the behavior of the method and its various parameters in a controlled setting.

- **Congressional voting data.** In Section 5.2, we consider roll call voting data from the United States Congress that are based on (Poole and Rosenthal, 1991). This is an example of realistic data set that has relatively-simple global structure but nontrivial local structure that varies with time (Cucuringu and Mahoney, 2011); and thus it allows us to illustrate the method in a realistic but relatively-clean setting.

- **Handwritten image data.** In Section 5.3, we consider data from the MNIST digit data set (Lecun and Cortes). These data have been widely-studied in machine learning and related areas and they have substantial "local heterogeneity." Thus, these

data allow us to illustrate how the method may be used to perform locally-biased versions of common machine learning tasks such as smoothing, clustering, and kernel construction.

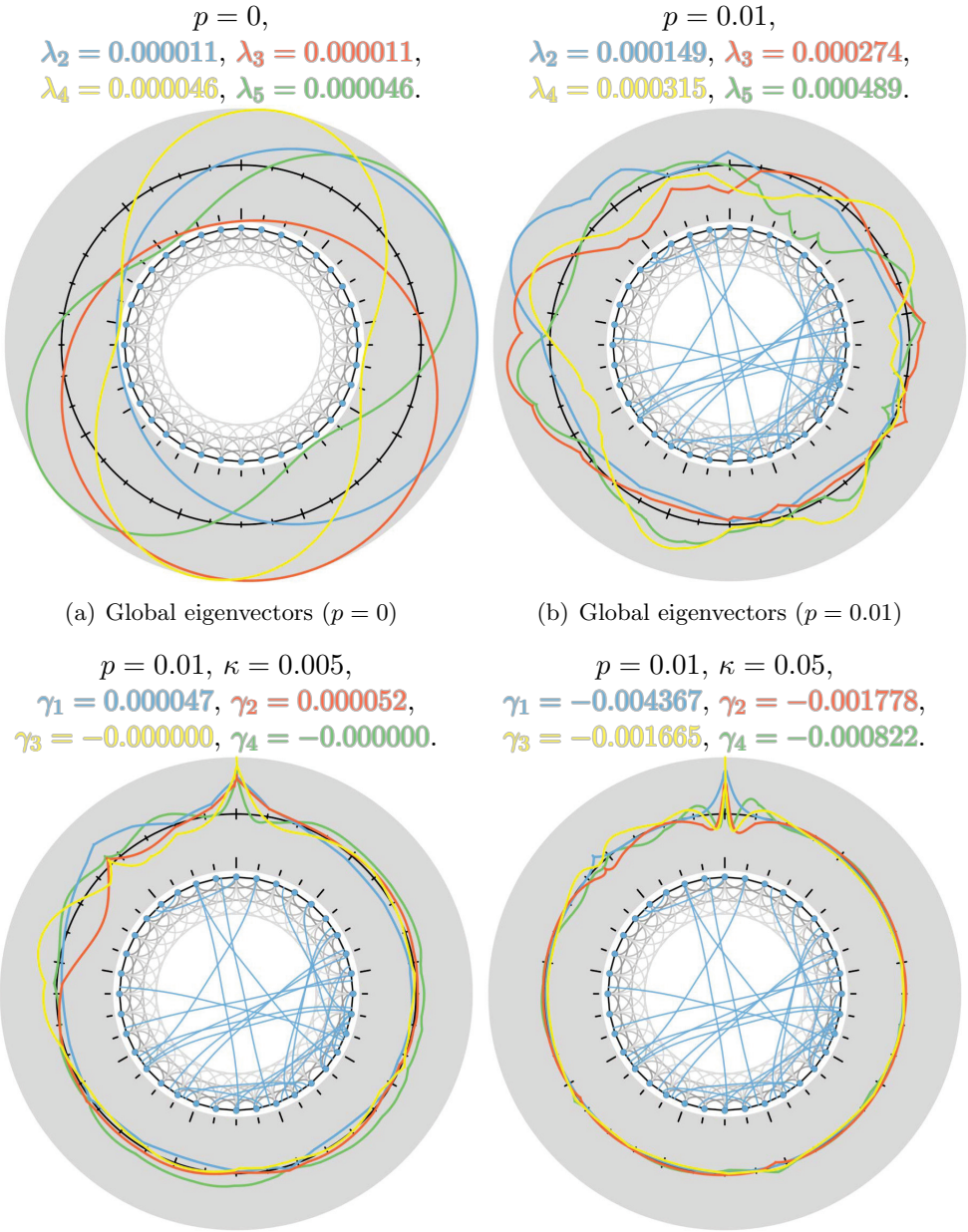
- **Large-scale network data.** In Section 5.4, we consider large-scale network data, and demonstrate significant performance improvements of the push-peeling heuristic compared to solving the same equations using a conjugate gradient solver. These improvements are demonstrated on data sets from the DIMACS implementation challenge, as well as on large web-crawls with more then 3 billion non-zeros in the adjacency matrix (Paolo et al., 2004, 2011; Paolo and Sebastiano, 2004).

5.1 Small-World Data

The first data sets we consider are networks constructed from the so-called small-world model. This model can be used to demonstrate how semi-supervised eigenvectors focus on specific target regions of a large data graph to capture slowest modes of local variation; and it can also be used to illustrate how the "knobs" of the method work, e.g., how κ and γ interplay, in a practical setting. In Figure 3, we plot the usual global eigenvectors, as well as locally-biased semi-supervised eigenvectors, around illustrations of non-rewired and rewired realizations of the small-world graph, i.e., for different values of the rewiring probability p and for different values of the locality parameter κ.

To start, in Figure 3(a) that we show a graph with no randomly-rewired edges ($p = 0$) and a parameter κ such that the global eigenvectors are obtained. This yields a symmetric graph with eigenvectors corresponding to orthogonal sinusoids, i.e., the first two capture the slowest mode of variation and correspond to a sine and cosine with equal random phase-shift (up to a rotational ambiguity). In Figure 3(b), random edges have been added with probability $p = 0.01$ and the parameter κ is still chosen such that the global eigenvectors—now of the rewired graph—are obtained. Note the many small kinks in the eigenvectors at the location of the randomly added edges. Note also the slow mode of variation in the interval on the top left; a normalized-cut based on the leading global eigenvector would extract this region, since the remainder of the ring is more well-connected due to the random rewiring.

In Figure 3(c), we see the same graph realization as in Figure 3(b), except that the semi-supervised eigenvectors have a seed node at the top of the circle, i.e., at "12 o-clock," and the locality parameter $\kappa_t = 0.005$, which corresponds to moderately well-localized eigenvectors. As with the global eigenvectors, the locally-biased semi-supervised eigenvectors are of successively-increasing (but still localized) variation. Note also that the neighborhood around "11 o-clock" contains more mass, e.g., when compared with the same parts of the circle in Figure 3(b) or with other parts of the circle in Figure 3(c), even though it is not very near the seed node in the original graph geometry. The reason for this is that this region is well-connected with the seed via a randomly added edge, and thus it is close in the modified graph topology. Above this visualization, we also show the value of γ_t that saturates κ_t, i.e., γ_t is the Lagrange multiplier that defines the effective locality κ_t. Not shown is that if we kept reducing κ_t, then γ_t would tend towards λ_{t+1}, and the respective semi-supervised eigenvectors would tend towards the global eigenvectors that are illustrated in Figure 3(b). Finally, in Figure 3(d), the desired locality is increased to $\kappa = 0.05$ (which



(a) Global eigenvectors ($p = 0$)

(b) Global eigenvectors ($p = 0.01$)

(c) Semi-supervised eigenvectors

(d) Semi-supervised eigenvectors

Figure 3: Illustration of small-world graphs with rewiring probability of $p = 0$ or $p = 0.01$ and with different values of the κ parameter. For each subfigure, the data consist of 3600 nodes, each connected to it's 8 nearest-neighbors. In the center of each subfigure, we show the nodes (blue) and edges (black and light gray are the local edges, and blue are the randomly-rewired edges). We wrap around the plots (black x-axis and gray background), visualizing the 4 smallest semi-supervised eigenvectors. Eigenvectors are color coded as **blue**, **red**, **yellow**, and **green**, starting with the one having the smallest eigenvalue.

has the effect of decreasing the value of γ_t), making the semi-supervised eigenvectors more localized in the neighborhood of the seed. It should be clear that, in addition to being determined by the locality parameter, we can think of γ as a regularizer biasing the global eigenvectors towards the region near the seed set. That is, variation in eigenvectors that are near the initial seed (in the modified graph topology) are most important, while variation that is far away from the initial seed matters much less.

5.2 Congressional Voting Data

The next data set we consider is a network constructed from a time series of roll call voting patterns from the United States Congress that are based on Poole and Rosenthal (1991). This is a particularly well-structured social network for which there is a great deal of meta-information, and it has been studied recently with graph-based methods (Mucha et al., 2010; Waugh et al., 2009; Cucuringu and Mahoney, 2011). Thus, it permits a good illustration of the method of semi-supervised eigenvectors in a real application (Poole, Fall 2005). This data set is known to have nontrivial time-varying structure at different time steps, and we will illustrate how the method of semi-supervised eigenvectors can perform locally-biased classification with a traditional kernel-based algorithm.

In more detail, we evaluate our method by considering the known Congress data-set containing the roll call voting patterns in the U.S Senate across time. We considered Senates in the 70^{th} Congress through the 110^{th} Congress, thus covering the years 1927 to 2008. During this time, the U.S went from 48 to 50 states, hence the number of senators in each of these 41 Congresses was roughly the same. We constructed an $N \times N$ adjacency matrix, with $N = 4196$ (41 Congresses each with ≈ 100 Senators) where $A_{ij} \in [0, 1]$ represents the extent of voting agreement between legislators i and j, and where identical senators in adjacent Congresses are connected with an inter-Congress connection strength. We then considered the Laplacian matrix of this graph, constructed in the usual way (Cucuringu and Mahoney, 2011).

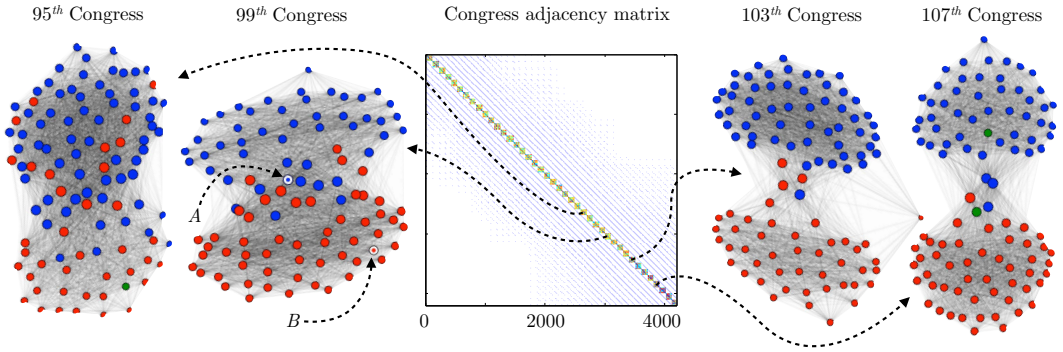


Figure 4: Shows the Congress adjacency matrix, along with four of the individual Congresses. Nodes are scaled according to their degree, blue nodes correspond to Democrats, red to Republicans, and green to Independents.

Figure 4 visualizes the adjacency matrix, along with four of the individual Congresses, color coded by party. This illustrates that these data should be viewed—informally—as a structure (depending on the specific voting patterns of each Congress) evolving along a one-dimensional temporal axis, confirming the results of Cucuringu and Mahoney (2011). Note that the latter two Congresses are significantly better described by a simple two-clustering than the former two Congresses, and an examination of the clustering properties of each of the 40 Congresses reveals significant variation in the local structure of individual Congresses, in a manner broadly consistent with Poole (Fall 2005) and Poole and Rosenthal (1991). In particular, the more recent Congresses are significantly more polarized.

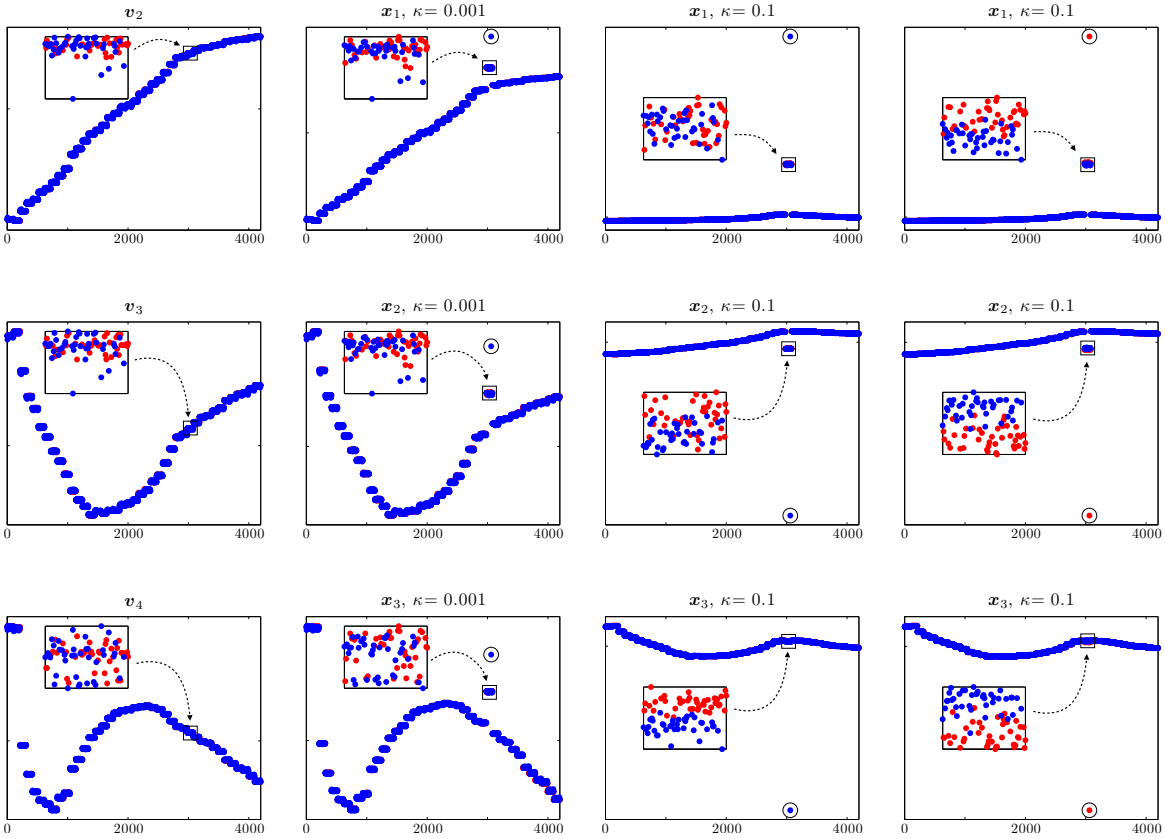


Figure 5: First column: The leading three nontrivial global eigenvectors. Second column: The leading three semi-supervised eigenvectors seeded (circled node) in an articulation point between the two parties in the 99^{th} Congress (see Figure 4), for correlation $\kappa = 0.001$. Third column: Same seed as previous column, but for a correlation of $\kappa = 0.1$. Notice the localization on the third semi-supervised eigenvector. Fourth column: Same correlation as the previous column, but for another seed node well within the cluster of Republicans. Notice the localization on all three semi-supervised eigenvectors.

The first vertical column of Figure 5 illustrates the first three global eigenvectors of the full data set, illustrating fluctuations that are sinusoidal and consistent with the one-

dimensional temporal scaffolding. Also shown in the first column are the values of that eigenfunction for the members of the 99^{th} Congress, illustrating that there is *not* a good separation based on party affiliations. The next three vertical columns of Figure 5 illustrate various localized eigenvectors computed by starting with a seed node in the 99^{th} Congress. For the second column, we visualize the semi-supervised eigenvectors for a very low correlation ($\kappa = 0.001$), which corresponds to only a weak localization—in this case one sees eigenvectors that look very similar to the global eigenvectors, and the elements of the eigenvector on that Congress do not reveal partitions based on the party cuts.

The third and fourth column of Figure 5 illustrate the semi-supervised eigenvectors for a much higher correlation ($\kappa = 0.1$), meaning a much stronger amount of locality. In particular, the third column starts with the seed node marked A in Figure 4, which is at the articulation point between the two parties, while the fourth column starts with the seed node marked B, which is located well within the cluster of Republicans. In both cases the eigenvectors are much more strongly localized on the 99^{th} Congress near the seed node, and in both cases one observes the partition into two parties based on the elements of the localized eigenvectors. Note, however, that when the initial seed is at the articulation point between two parties then the situation is much noisier: in this case, this "partitionability" is seen only on the third semi-supervised eigenvector, while when the initial seed is well within one party then this is seen on all three eigenvectors. Intuitively, when the seed set is strongly within a good cluster, then that cluster tends to be found with semi-supervised eigenvectors (and we will observe this again below). This is consistent with the diffusion interpretation of eigenvectors. This is also consistent with Cucuringu and Mahoney (2011), who observed that the properties of eigenvector localization depended on the local structure of the data around the seed node, as well as the larger scale structure around that local cluster.

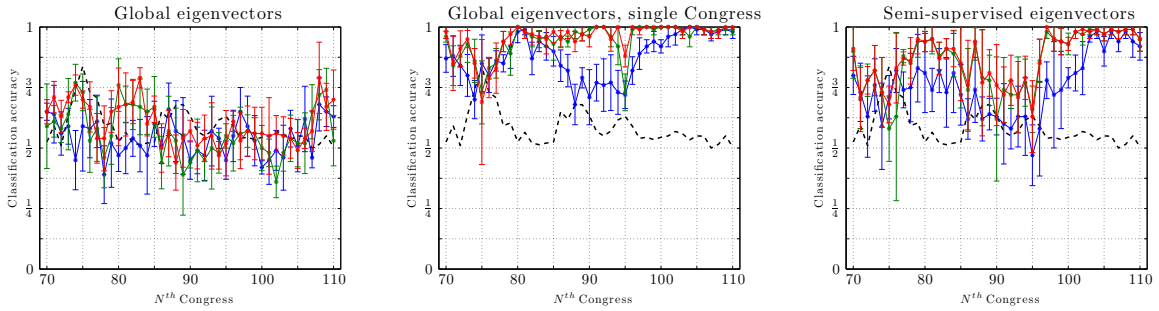


Figure 6: Classification accuracy measured in individual Congresses. For each Congress we perform 5-fold cross validation based on ≈ 80 samples and leave out the remaining 20 samples to estimate an unbiased test error. Error bars are obtained by resampling and they correspond to 1 standard deviation. For each approach we consider features based on the 1^{st} (blue), 2^{nd} (green), and 3^{rd} (red) smallest eigenvector(s), excluding the all-one vector. We also plot the probability of the most probable class as a baseline measure (black) as some Congresses are very imbalanced.

To illustrate how these structural properties manifest themselves in a more traditional machine learning task, we also consider the classification task of discriminating between Democrats and Republicans in single Congresses, i.e., we measure to what extent we can extract local discriminative features. To do so, we apply L_2-regularized L_2-loss support vector classification with a linear kernel, where features are extracted using the global eigenvectors of the entire data set, global eigenvectors from a single Congress (best case measure), and our semi-supervised eigenvectors. Figure 6 illustrates the classification accuracy for 1, 2, and 3 eigenvectors. As reported by Cucuringu and Mahoney (2011), locations that exhibit discriminative information are best found on low-order eigenvectors of this data, explaining why the classifier based global eigenvectors performs poorly. In the classifier based on global eigenvectors in the single Congress we exploit *a priori* knowledge to extract the relevant data, that in a usual situation would be impossible. Hence, this is simply to define a baseline point of reference for the best case classification accuracy. The classifier based on semi-supervised eigenvectors is seeded using a few training samples and performs in-between the two other approaches. Compared to our point of reference, Congresses in the range 88 to 96 do worse with the semi-supervised eigenvectors; whereas for Congresses after 100 the semi-supervised approach almost performs on par, even for a single single eigenvector. This is consistent with the visualization in Figure 4 illustrating that earlier Congresses are less cleanly separable, as well as with empirical evidence indicating heterogeneity due to Southern Democrats in earlier Congresses and the recent increase in party polarization in more recent Congresses, as described in Poole (Fall 2005) and Poole and Rosenthal (1991).

5.3 MNIST Digit Data

The next data set we consider is the well-studied MNIST data set containing $60,000$ training digits and $10,000$ test digits ranging from 0 to 9; and, with these data, we demonstrate the use of semi-supervised eigenvectors as a feature extraction preprocessing step in a traditional machine learning setting. We construct the full $70,000 \times 70,000$ k-NN graph, with $k = 10$ and with edge weights given by $w_{ij} = \exp(-\frac{4}{\sigma_i^2}\|x_i - x_j\|^2)$, where σ_i^2 is the Euclidean distance of the i^{th} node to it's nearest neighbor; and from this we define the graph Laplacian in the usual way. We then evaluate the semi-supervised eigenvectors in a transductive learning setting by disregarding the majority of labels in the entire training data. We use a few samples from each class to seed our semi-supervised eigenvectors as well as a few others to train a downstream classification algorithm. For this evaluation, we use the Spectral Graph Transducer (SGT) of Joachims (2003); and we choose to use it for two main reasons. First, the transductive classifier is inherently designed to work on a subset of global eigenvectors of the graph Laplacian, making it ideal for validating that the localized basis constructed by the semi-supervised eigenvectors can be more informative when we are solely interested in the "local heterogeneity" near a seed set. Second, using the SGT based on global eigenvectors is a good point of comparison, because we are only interested in the effect of our subspace representation. (If we used one type of classifier in the local setting, and another in the global, the classification accuracy that we measure would obviously be confounded.) As in Joachims (2003), we normalize the spectrum of both global and semi-supervised eigenvectors by replacing the eigenvalues with some monotonically increasing function. We use $\lambda_i = \frac{i^2}{k^2}$, i.e., focusing on ranking among smallest cuts; see (Chapelle

et al., 2003). Furthermore, we fix the regularization parameter of the SGT to $c = 3200$, and for simplicity we fix $\gamma = 0$ for all semi-supervised eigenvectors, implicitly defining the effective $\kappa = [\kappa_1, \ldots, \kappa_k]^T$. Clearly, other correlation distributions κ and other values of γ parameter may yield subspaces with even better discriminative properties (which is an issue to which we will return in Section 5.3.2 in greater detail).

	#Semi-supervised eigenvectors for SGT						#Global eigenvectors for SGT					
Labeled points	1	2	4	6	8	10	1	5	10	15	20	25
1 : 1	0.39	0.39	0.38	0.38	0.38	0.36	0.50	0.48	0.36	0.27	0.27	0.19
1 : 10	0.30	0.31	0.25	0.23	0.19	0.15	0.49	0.36	0.09	0.08	0.06	0.06
5 : 50	0.12	0.15	0.09	0.08	0.07	0.06	0.49	0.09	0.08	0.07	0.05	0.04
10 : 100	0.09	0.10	0.07	0.06	0.05	0.05	0.49	0.08	0.07	0.06	0.04	0.04
50 : 500	0.03	0.03	0.03	0.03	0.03	0.03	0.49	0.10	0.07	0.06	0.04	0.04

Table 1: Classification error for discriminating between 4s and 9s for the SGT based on, respectively, semi-supervised eigenvectors and global eigenvectors. The first column from the left encodes the configuration, e.g., 1:10 interprets as 1 seed and 10 training samples from each class (total of 22 samples—for the global approach these are all used for training). When the seed is well-determined and the number of training samples moderate (50:500), then a single semi-supervised eigenvector is sufficient; whereas for less data, we benefit from using multiple semi-supervised eigenvectors. All experiments have been repeated 10 times.

5.3.1 Discriminating Between Pairs of Digits

Here, we consider the task of discriminating between two digits; and, in order to address a particularly challenging task, we work with $4s$ and $9s$. (This is particularly challenging since these two classes tend to overlap more than other combinations since, e.g., a closed 4 can resemble a 9 more than an open 4.) Hence, we expect that the class separation axis will not be evident in the leading global eigenvector, but instead it will be "buried" further down the spectrum; and we hope to find a "locally-biased class separation axis" with locally-biased semi-supervised eigenvectors. Thus, this example will illustrate how semi-supervised eigenvectors can represent relevant heterogeneities in a local subspace of low dimensionality. See Table 1, which summarizes our classification results based on, respectively, semi-supervised eigenvectors and global eigenvectors, when we use the SGT. See also Figure 7 and Figure 8, which illustrate two realizations for the 1:10 configuration. In these two figures, the training samples are fixed; and, to demonstrate the influence of the seed, we have varied the seed nodes. In particular, in Figure 7 the seed nodes s_+ and s_- are located well-within the respective classes; while in Figure 8, they are located much closer to the boundary between the two classes. As intuitively expected, when the seed nodes fall well within the classes to be differentiated, the classification is much better than when the seed nodes are located closer to the boundary between the two classes. See the caption in these figures for further details.

5.3.2 Effect of Choosing The κ Correlation/Locality Parameter

Here, we discuss the effect of the choice of the correlation/locality parameter κ at different steps of Algorithm 1, e.g., how $\{\kappa_t\}_{t=1}^k$ should be distributed among the k components.

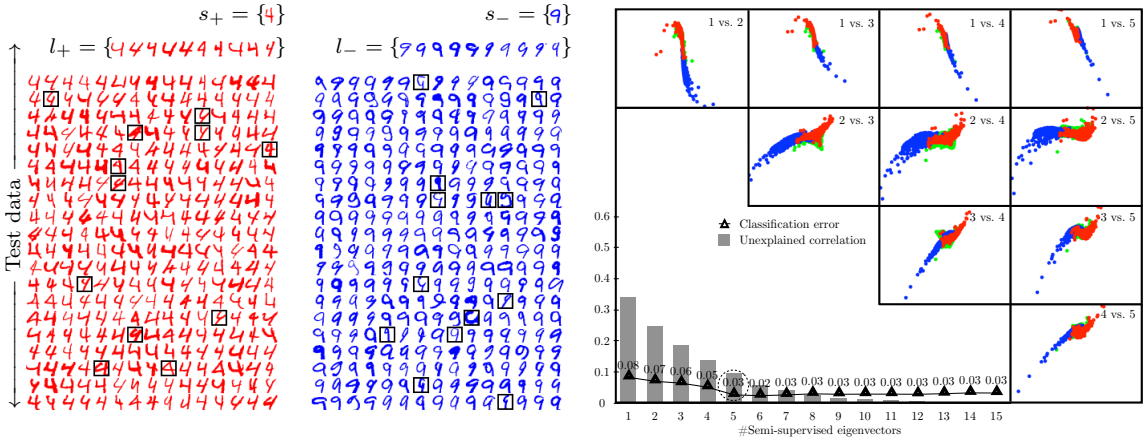


Figure 7: Discrimination between 4s and 9s. Left: Shows a subset of the classification results for the SGT based on 5 semi-supervised eigenvectors seeded in s_+ and s_-, and trained using samples l_+ and l_-. Misclassifications are marked with black frames. Right: Visualizes all test data spanned by the first 5 semi-supervised eigenvectors, by plotting each component as a function of the others. Red (blue) points correspond to 4 (9), whereas green points correspond to remaining digits. As the seed nodes are good representatives, we note that the eigenvectors provide a good class separation. We also plot the error as a function of local dimensionality, as well as the unexplained correlation, i.e., initial components explain the majority of the correlation with the seed (effect of $\gamma = 0$). The particular realization based on the leading 5 semi-supervised eigenvectors yields an error of ≈ 0.03 (dashed circle).

For example, will the downstream classifier benefit the most from a uniform distribution or will there exist some other nonuniform distribution that is better? Although this will be highly problem specific, one might hope that in realistic applications the classification performance is not too sensitive to the actual choice of distribution. To investigate the effect in our example of discriminating between 4s and 9s, we consider 3 semi-supervised eigenvectors for various κ distributions. Our results are summarized in Figure 9.

Figures 9(a), 9(b), and 9(c) show, for the global eigenvectors and for semi-supervised eigenvectors, where the κ vector has been chosen to be very nonuniform and very uniform, the top three (global or semi-supervised) eigenvectors plotted against each other as well as the ROC curve for the SGT classifier discriminating between 4s and 9s; and Figure 9(d) shows the test error as the κ vector is varied over the unit simplex. In more detail, red (respectively, blue) corresponds to 4s (respectively, 9s), and green points are the remaining digits; and, for Figures 9(b) and 9(c), the semi-supervised eigenvectors are seeded using 50 samples from each target class (4s vs. 9s) and having a non-uniform distribution of κ, as specified. As seen from the visualization of the semi-supervised eigenvectors in Figures 9(b) and 9(c), the classes are much better separated than by using the global eigenvectors, which are shown in Figure 9(a). For example, this is supported by the Area Under the Curve

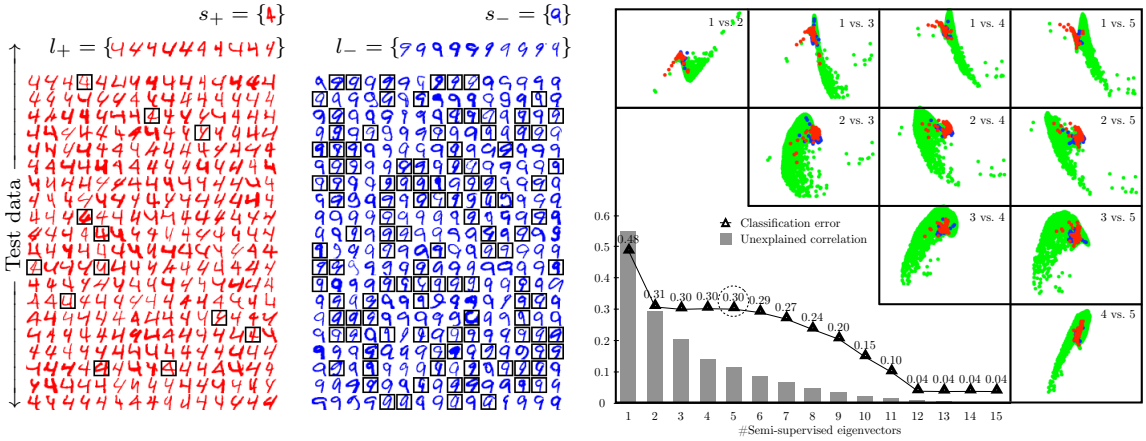
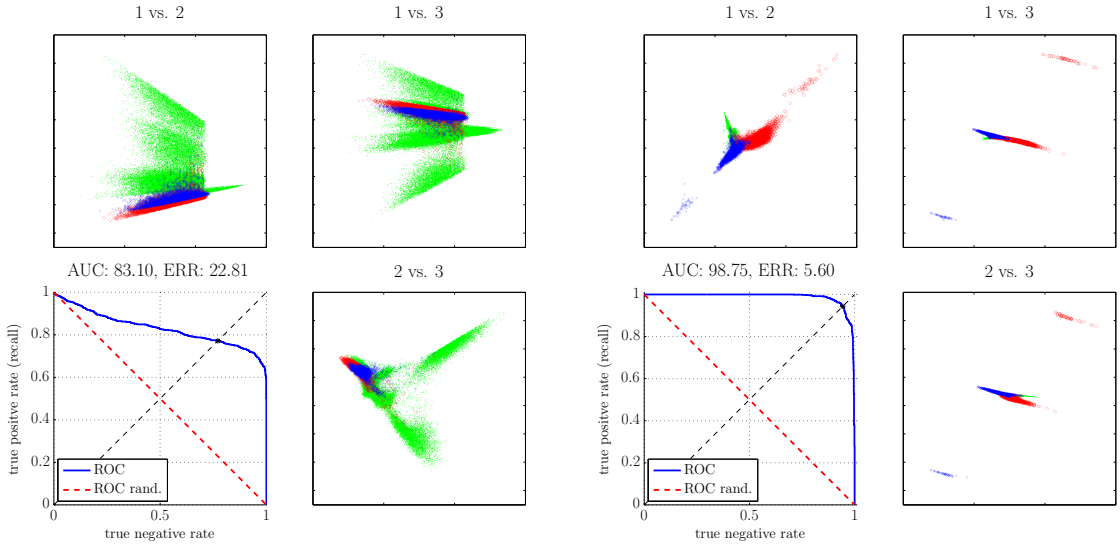


Figure 8: Discrimination between 4s and 9s. See the general description in Figure 7. Here we illustrate an instance where the s_+ shares many similarities with s_-, i.e., s_+ is on the boundary of the two classes. This particular realization achieves a classification error of ≈ 0.30 (dashed circle). In this constellation we first discover localization on low order semi-supervised eigenvectors (≈ 12 eigenvectors), which is comparable to the error based on global eigenvectors (see Table 1), i.e., further down the spectrum we recover from the bad seed and pickup the relevant mode of variation.

(AUC) and Error Rate (ERR), being the point on the Receiver Operating Characteristic (ROC) curve that corresponds to having an equal probability of miss-classifying a positive or negative sample, which is a fair estimate as the classes in the MNIST data set is fairly balanced. For Figure 9(c), where we use a uniform distribution of κ, the classifier performs slightly better than in Figure 9(b), which uses the non-uniform κ distribution (but both semi-supervised approaches are significantly better than the using the global eigenvectors). For Figure 9(d), we see the test error on the simplex defined by κ. To obtain this plot we sampled 500 different κ distributions according to a uniform Dirichlet distribution. With the exception of one extreme very nonuniform corner, the classification accuracy is not too sensitive to the choice of κ distribution. Thus, if we think of the semi-supervised eigenvectors as a locally-regularized version of the global eigenvectors, the desired discriminative properties are not too sensitive to the details of the locally-biased regularization.

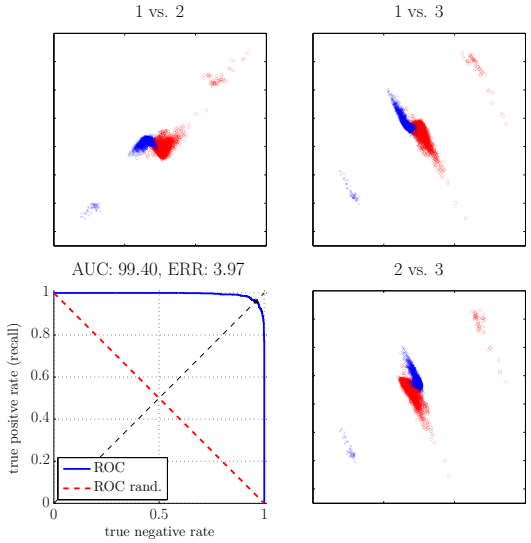
5.3.3 Effect of Approximately Computing Semi-Supervised Eigenvectors

Here, we discuss of the push-peeling procedure from Section 4 that is designed to compute efficient approximations to the semi-supervised eigenvectors by using local random walks to compute an approximation to personalized PageRank vectors. Consider Figure 10, which shows results for two values of the ϵ parameter (i.e., the parameter in the push algorithm that implicitly determines how many nodes will be touched). Using the same graph as defined previously, we compute 3 semi-supervised eigenvectors seeding using 50 samples from each class (4s vs. 9s). However, in this case, we fix the regularization parameter

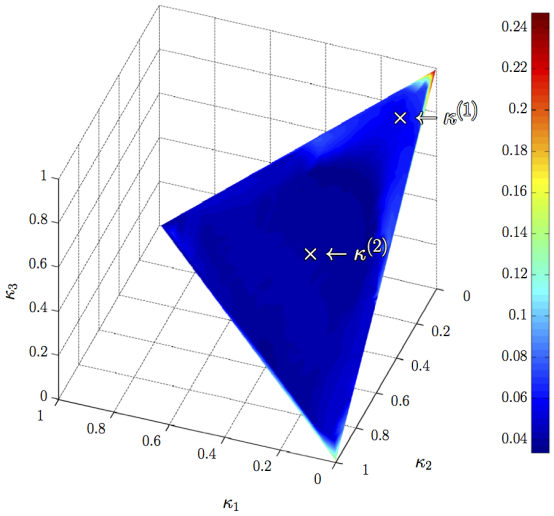


(a) Global eigenvectors

(b) Semi-supervised, with $\kappa^{(1)} = (\frac{1}{10}, \frac{1}{10}, \frac{8}{10})$



(c) Semi-supervised, with $\kappa^{(2)} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$



(d) Test error on the κ simplex

Figure 9: The effect of varying the correlation/locality parameter κ on the classification accuracy. 9(a), 9(b), 9(c) show the top three (global or semi-supervised) eigenvectors plotted against each other as well as the ROC curve for the SGT classifier discriminating between 4s and 9s; and 9(d) shows the test error as the κ vector is varied over the unit simplex.

vector as $\gamma = [-0.0150, -0.0093, -\text{vol}(G)]$; and note that choosing these specific values correspond to the solutions visualized in Figure 9(c) when the equations are solved exactly. Figure 10(a) shows the results for $\epsilon = 0.001$. This approximation gives us sparse solutions,

and the histogram in the second row illustrates the digits that are assigned a nonzero value in the respective semi-supervised eigenvector. In particular, note that most of the mass of the eigenvector is distributed on 4s and 9s; but, for this choose of ϵ, only few digits of interest ($\approx 2.8243\%$, meaning, in particular, that not all of the 4s and 9s) have been touched by the algorithm. This results in the lack of a clean separation between the two classes as one sweeps along the leading semi-supervised eigenvector, as illustrated in the first row; the very uniform correlation distribution $\kappa = [0.8789, 0.0118, 0.1093]$; and the high classification error, as shown in the ROC curve in the bottom panel.

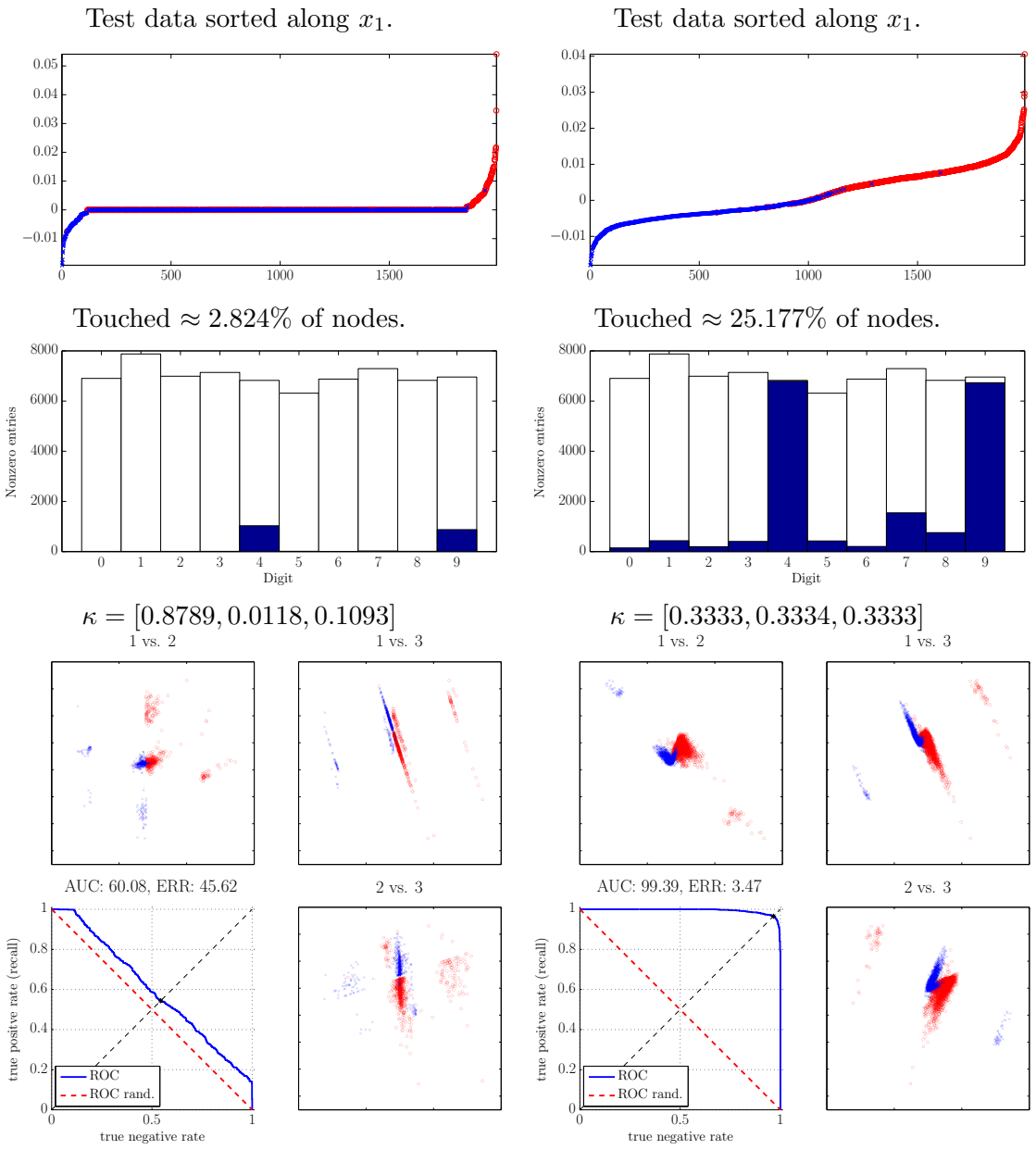
Consider, next, Figure 10(b), which shows the results for $\epsilon = 0.0001$. In this case, the algorithm reproduces the solution by touching only $\approx 25.177\%$ of the nodes in the graph, i.e., basically all of the 4s and 9s and only a few other digits. This leads to a much cleaner separation between the two classes as one sweeps over the leading semi-supervised eigenvector; a much more uniform distribution over κ; and a classification accuracy that is much better and is similar to what we saw in Figure 9(c). This example illustrates that this push-peeling approximation provides a principled manner to generalize the concept of semi-supervised eigenvectors to large-scale settings, where it will be infeasible to touch all nodes of the graph.

5.3.4 Effect of Low-Rank Nyström Approximation

Here we discuss the use of the low-rank Nyström approximation which is commonly used in large-scale kernel-based machine learning. The memory requirements for representing the explicit kernel matrix, that we here take to be our graph, scales with $\mathcal{O}(N^2)$, whereas inverting the matrix scales with $\mathcal{O}(N^3)$, which, in large-scale settings, is infeasible. The Nyström technique subsamples the data set to approximate the kernel matrix, and the memory requirements scales with $\mathcal{O}(nN)$ and runs in $\mathcal{O}(n^2N)$, where n is size of the subsample. For completeness we include the derivation of the Nyström approximation for the normalized graph Laplacian in Appendix C.

In the beginning of Section 5.3 we constructed the $70,000 \times 70,000$ k-nearest neighbor graph, with $k = 10$ and with edge weights given by $w_{ij} = \exp(-\frac{4}{\sigma_i^2}\|x_i - x_j\|^2)$. Such a sparse construction reduces the effect of "hubs", as well as being fairly insensitive to the choice of kernel parameter, as the 10 nearest neighbors are likely to be very close in the Euclidean norm. Because the Nyström method will approximate the dense kernel matrix, the choice of kernel parameter is more important, so in the following we will consider the interplay between this parameter, as well as the rank parameter n of the Nyström approximation. Moreover, to allow us to compare a rank-n Nyström approximation with the full rank-N kernel matrix, we choose to subsample the data set for all of the following experiments, due to the $\mathcal{O}(N^2)$ memory requirements. Thus, to provide a baseline, Figure 11 shows results based on a k-nearest neighbor graph constructed from 5% and 10% percent of the training data, where in both cases we used 10% for the test data. For both cases, when compared with the results of Figure 9(c), the classification quality is degraded, and so we emphasize that the goal of the following results are not to outperform the results reported in Figure 9(c), but to be comparable with this baseline.

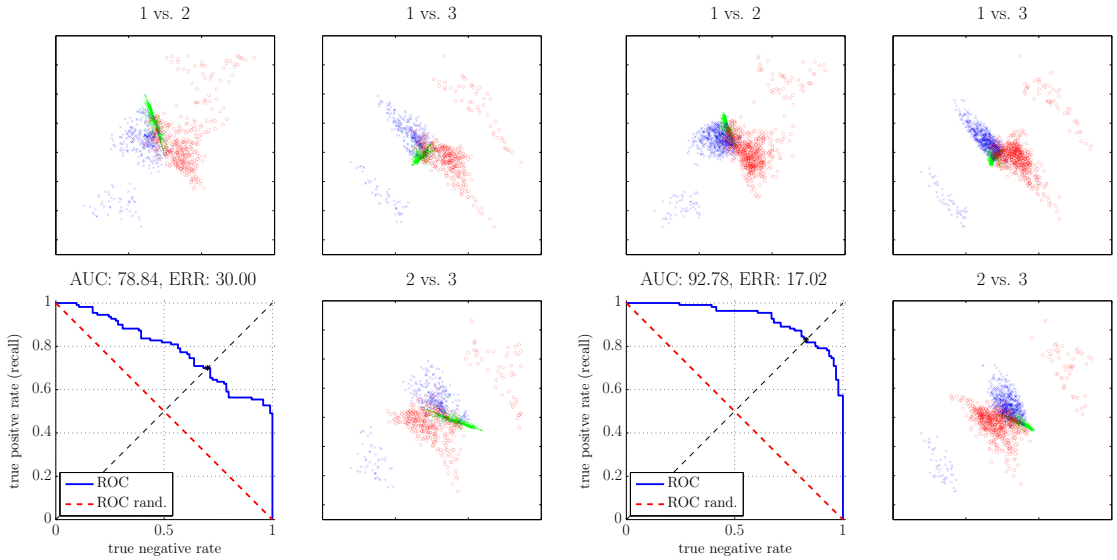
In light of this baseline, Figure 12 provides a thorough analysis for the choices of σ_i^2 that we used. Figures 12(a) and 12(b) show the classification error when using the global



(a) Locality parameter $\epsilon = 0.001$

(b) Locality parameter $\epsilon = 0.0001$

Figure 10: Illustration of the push-peeling procedure to compute 3 semi-supervised eigenvectors for $\gamma = [-0.0150, -0.0093, -\text{vol}(G)]$. 10(a) shows results for $\epsilon = 0.001$; and 10(b) shows results for $\epsilon = 0.0001$. First row shows the entries in the leading semi-supervised eigenvector corresponding to test points, color-coded and sorted according to magnitude; second row shows the distribution of digits touched in the full graph when executing the push algorithm; and bottom panels provide visualizations similar to the ones in Figure 9 (and shown above these is the correlation vector κ obtained for the fixed choice of γ.



(a) 5% of MNIST

(b) 10% of MNIST

Figure 11: Example of the impact of subsampling the data set down to 5% (in 11(a)) and 10% (in 11(b)) of the original size. Remaining parameters are the same as in Figure 9(c), which shows the result to which these two plots should be compared.

eigenvectors, for various rank approximations based on the Nyström method as well as the exact method (corresponding to rank $= n$). Interestingly, these two plots are very dissimilar in terms of their behavior as a function of the number of components. In particular, the plot in Figure 12(b) shows that the low rank approximations for a given set of components outperform the high rank approximations, and the exact representation fails to reduce the error beyond 0.4 for any of the considered set of components. This may seem counterintuitive, but the reason for this type of behavior is that the relevant global eigenvectors, for $\sigma_i^2 = 200$, are located far from the end of the spectrum (if we visualized more components for rank $= n$ the classification error would eventually drop). For the same reason, the low rank approximations improve more rapidly than the high rank approximations, as the latter approximate the lower part of the spectrum better, and these turn out to have poor discriminative properties. In contrast, the results shown in Figure 12(a) provide good class separation in the lower part of the spectrum, resulting in the high rank approximations to reduce the error most rapidly.

Finally, Figures 12(c) and 12(d) show the classification error for the SGT trained using the semi-supervised eigenvectors. (Note that the scale of the x-axis is much smaller in these subfigures.) For both kernel widths in Figures 12(c) and 12(d), the ordering of the approximations are similar, i.e., the semi-supervised eigenvectors constructed from the rank $= n$ approximation performs the best. Moreover, the gap between the rank $= 400$ and rank $= n$ is largest for $\sigma_i^2 = 200$, again suggesting this approximation is of insufficient rank to model the relevant local heterogeneities deep down in the spectrum; whereas for $\sigma_i^2 = 80$,

the rank = 400 the approximation comes very close to the exact representation, suggesting that local structures are well modeled near the end of the spectrum.

To summarize these results, the method of semi-supervised eigenvectors successfully extracts relevant local structures to perform locally-biased classification, even when they are located far from the end of the spectrum. Moreover, in both cases we considered, the classification error is reduced significantly by using only a few locally-biased components. This contrasts with the global eigenvectors, where for $\sigma_i^2 = 80$ at least 20 eigenvectors are needed in order to obtain similar performance; and for $\sigma_i^2 = 200$, the classification error remains high even for 200 eigenvectors in case of rank = n.

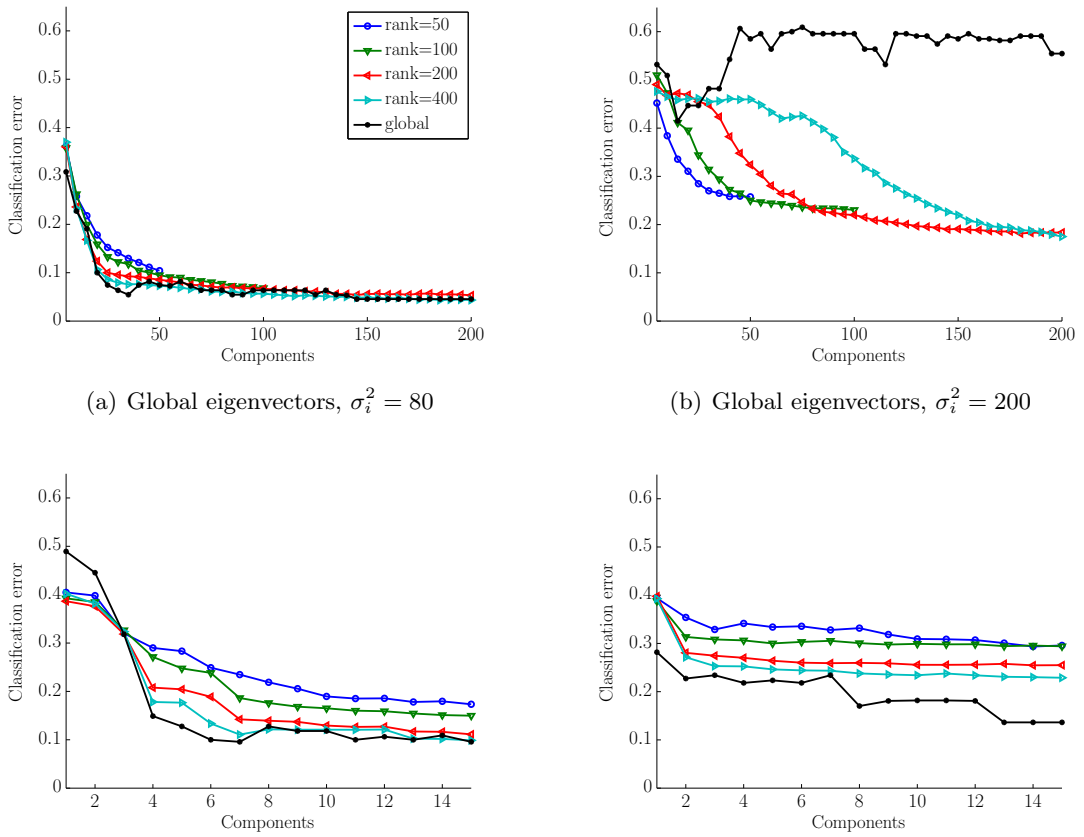
5.4 Large-scale Network Data

The final data sets we consider are from a collection of large sparse networks (Paolo et al., 2004, 2011; Paolo and Sebastiano, 2004). On these data, we demonstrate that the Push-peeling Heuristic introduced in Section 4.2 is attractive due to an improved running time, as compared to solving a system of linear equations. Moreover, we also show that the ability to obtain multiple semi-supervised eigenvectors depends on the degree heterogeneity near the seed. Finally, we empirically evaluate the influence of the ϵ parameter of the Push algorithm that implicitly determines how many nodes the algorithm will touch. This parameter can be interpreted as a regularization parameter (different from γ parameter), and setting it too large means we fail to distribute mass in the network, so that a few semi-supervised eigenvectors will consume all of the correlation. In particular, this behavior was investigated on the MNIST digits in Section 5.3.3. The basic properties for the networks considered in this section are shown in Table 2.

We start by considering the moderately sized networks from the DIMACS implementation challenge, as these networks are commonly used for the purpose of measuring realistic algorithm performance. Figure 13 shows analysis results for 6 networks from this collection, where we evaluate the performance and feasibility of the Push algorithm for approximating the leading semi-supervised eigenvector.

Network name	Number of nodes	Number of edges
DIMACS10/de2010	24,115	116,056
DIMACS10/ct2010	67,578	336,352
DIMACS10/il2010	451,554	2,164,464
DIMACS10/smallworld	100,000	999,996
DIMACS10/333SP	3,712,815	22,217,266
DIMACS10/AS365	3,799,275	22,736,152
LAW/arabic-2005	22,744,080	1,107,806,146
LAW/indochina-2004	7,414,866	301,969,638
LAW/it-2004	41,291,594	2,054,949,894
LAW/sk-2005	50,636,154	3,620,126,660
LAW/uk-2002	18,520,486	523,574,516
LAW/uk-2005	39,459,925	1,566,054,250

Table 2: Summary of the networks considered in this section. Some of these networks are directed and have been symmetrized for the purpose of this analysis, i.e., the number edges in this table refer to the number of edges in the undirected graph.



(a) Global eigenvectors, $\sigma_i^2 = 80$

(b) Global eigenvectors, $\sigma_i^2 = 200$

(c) Semi-supervised eigenvectors, $\sigma_i^2 = 80$

(d) Semi-supervised eigenvectors, $\sigma_i^2 = 200$

Figure 12: We consider 10% of the MNIST training and test data and investigate the classification accuracy of a downstream SGT classifier for various approximations of the dense similarity matrix. 12(a) and 12(b): Classification error for the SGT evaluated directly on global eigenvectors, based on various Nyström approximations and the two choices of the kernel width parameter (respectively, $\sigma_i^2 = 80$ and $\sigma_i^2 = 200$). 12(c) and 12(d): Classification error we have used the Nyström approximations as basis for computing semi-supervised eigenvectors that are then used in the downstream SGT classifier. All plots show the mean over 30 repetitions.

As stated in Section 3.3, diffusion based procedures such as the Push algorithm can be used to solve our objective for $\gamma < 0$. The impact of the reduced search range is that such procedures may not be able to produce a uniform correlation distribution for a set of semi-supervised eigenvectors. Hence, the leading solution(s) will instead pickup too much correlation, because sufficient mass cannot to diffuse away from the seed set. However, the effect of a non-uniform correlation distribution was analyzed on the MNIST data in Section 5.3, where we found that the performance of a downstream classifier is fairly robust to such

non-uniformities, as seen by the simplex in Figure 9. Consequently, we emphasize that in a large-scale setting such side effects of diffusion based procedures is offset by the advantage of a greatly improved time complexity as compared to solving the system of linear equations, that implicitly touch every node.

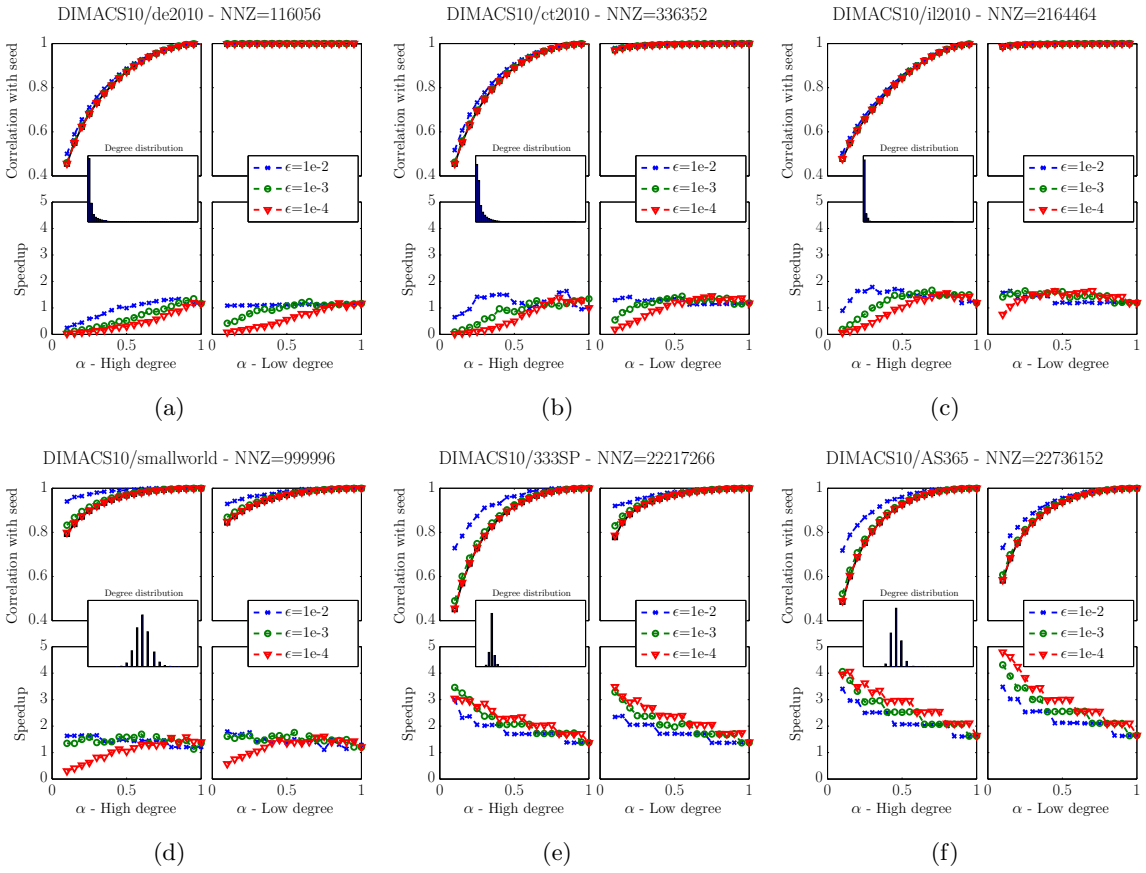


Figure 13: For each network the first row depicts how the correlation decays as α tends towards 0, whereas the bottom row shows the speedup relative to the standard approach using conjugate gradient with a tolerance of 1e-6, that is the default approach in our software distribution. Besides the three considered values of ϵ the correlation plots also illustrate the decay based on conjugate gradient (black curve), however this may be difficult to see, as the Push algorithm for $\epsilon = 1\text{e-}4$ coincides with that solution. Finally, seeds based on a high degree and low degree node are presented in respectively the first and last column, and the degree distribution for the network is visualized in a minor overlapping plot.

For each of the 6 analyzed networks in Figure 13, we run two experiments considering different seeds, using respectively a high degree and low degree single seed node. Figure 13(a)-13(c) considers census block networks characterized by heavy-tailed degree distributions, whereas Figure 13(d)-13(f) considers more densely connected synthetic networks. For

each of these 6 networks the speedup is measured by comparing with a standard conjugate gradient implementation using a tolerance of 1e-6, and we stress that this tolerance cannot be directly compared with ϵ in the Push algorithm. Moreover, we test three different settings of the ϵ parameter, and we emphasize that for $\epsilon =$ 1e-4, the Push algorithm produces a similar result as the conjugate gradient algorithm. In Figure 13 this can be seen by the red curve ($\epsilon =$ 1e-4) in the correlation decay plots (see the figure caption) being on top of the black curve (conjugate gradient).

Common for Figure 13(a)-13(c) are that low degree seed nodes yield very localized solutions for the entire range of α, opposed to the high degree nodes that all succeed in gradually reducing the correlation when α is reduced. Also, the choice of ϵ is obviously very important, i.e., choosing it too large results in a solution that correlates too much with the seed, whereas choosing it too small means that we will be touching more nodes than necessary, resulting in a performance penalty. In general the networks analyzed in Figure 13(a)-13(c) are too small to yield significant performance improvements over the conjugate gradient algorithm, and the Push algorithm is only competitive for large values of α.

For the network in Figure 13(d), we see similar performance characteristics as the networks analyzed in Figure 13(a)-13(c) due to its small size. However, the two final networks analyzed in Figure 13(e)-13(f) share similar characteristics in terms of the degree distribution, but due to a much larger size they show significant performance improvements over the conjugate gradient algorithm. Interestingly, the Push algorithm instantiated with $\epsilon =$ 1e-4 yields a greater speedup in some settings, which may be explained by faster convergence, caused by a reduced threshold for distributing mass. Hence, the running time of the Push algorithm may not always decrease monotonically as ϵ increases.

In general it seems that seeding in a sparsely connected region of a network results in a solution having a large correlation with the seed for most values of α. This is obviously a limiting factor if we are interested in using the peeling procedure to find multiple semi-supervised eigenvectors in that particular region. However, for large networks and more densely connected regions the benefit of the Push algorithm is immediate.

Finally, we scale up to demonstrate that we can adapt the notion of semi-supervised eigenvectors to large data sets, and we do so by analyzing 6 large web-crawl networks. These networks are large enough that touching all nodes is infeasible, i.e., conjugate gradient is not a feasible option, so in Figure 14 we resort to absolute timings. For the analysis results shown in Figure 14, we are solely interested in giving the reader some intuition about the running time in a large-scale setting, as well as an idea on how the parameters interplay. Hence, we only consider experiments where we seed in a high degree node, as these are likely yield the worst running times, but also succeed in reducing the correlation the most. This will make the peeling procedure described in Section 4.2 applicable, allowing us to obtain multiple semi-supervised eigenvectors. As seen for all networks analyzed in Figure 14(a)-14(f) the solution is highly sensitive to the choice of ϵ, but for all networks we are able to reduce the correlation when α tends towards 0 in case of $\epsilon =$ 1e-6. We emphasize that the reason for ϵ being smaller for these experiments, as compared to the previous is that the seed is normalized to have unit norm, implicitly requiring a lower ϵ when the network increases in size.

For diffusion based procedures to be useful with respect to the computation of semi-supervised eigenvectors, mass must be able "bleed" away from the seed set and into the

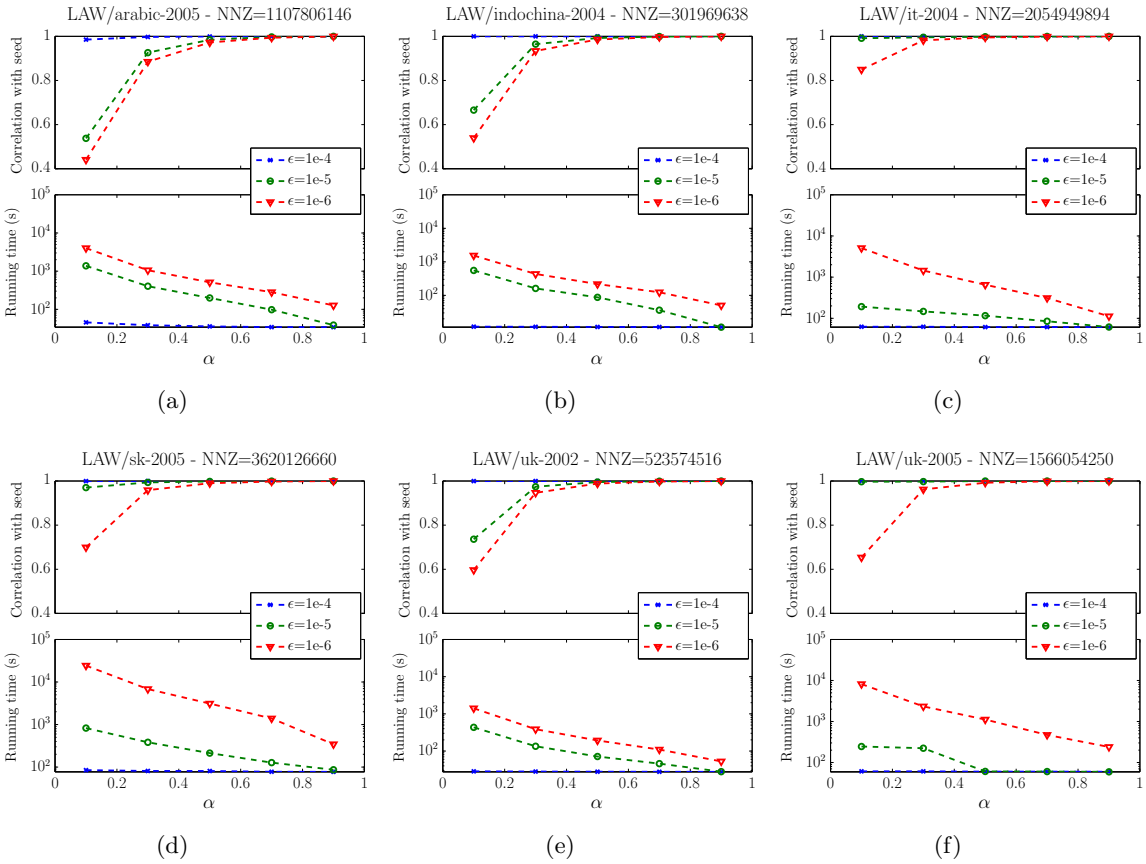


Figure 14: Visualizes results for applying the Push algorithm to 6 very large web-crawl networks. For all networks we seed in the node with the highest degree. The top plot in each subfigure shows the correlation decay as a function of α, whereas in the bottom plot we resort to absolute timings as the conjugate gradient algorithm is not feasible in this setting, as opposed to showing speedups as in Figure 13.

surrounding network. Otherwise only few semi-supervised eigenvectors can be found as the leading solution(s) become too correlated with the seed set. For moderately sized problems conjugate gradient performs very well, but in a large-scale setting, as considered here, the presented approach proves very efficient, allowing us to compute approximations to semi-supervised eigenvectors in networks consuming more than 30GB of working memory. Obtaining an improved understanding of how the method of semi-supervised eigenvectors can be used to perform common machine learning tasks on graphs of that size is an obvious direction raised by our work.

6. Conclusion

We have introduced the concept of semi-supervised eigenvectors as local analogues of the global eigenvectors of a graph Laplacian that have proven so useful in a wide range of machine learning and data analysis applications. These vectors are biased toward prespecified local regions of interest in a large data graph; and we have shown that since they inherit many of the nice properties of the usual global eigenvectors, except in a locally-biased context, they can be used to perform locally-biased machine learning. The basic method is conceptually simple and involves solving a sequence of linear equation problems; we have also presented several extensions of the basic method that have improved scaling properties; and we have illustrated the behavior of the method. Due to the speed, simplicity, stability, and intuitive appeal of the method, as well as the range of applications in which local regions of a large data set are of interest, we expect that the method of semi-supervised eigenvectors can prove useful in a wide range of machine learning and data analysis applications.

Acknowledgments

Partial support of this work has been provided by the ARO, DARPA, and NSF.

Appendix A. Supplementary Proofs

Lemma 1 *Given an SPSD matrix M and some vector x where $x^\top x = 1$, it holds that*

$$\lim_{\omega\to\infty} \left(M + \omega xx^\top\right)^+ = \left(\left(I - xx^\top\right) M \left(I - xx^\top\right)\right)^+. \tag{14}$$

Proof: Prior to applying the pseudo inverse, x is clearly an eigenvector with eigenvalue $\lambda = 0$ on the right hand side, and for left hand side x is an eigenvector with eigenvalue $\lambda = \infty$. Hence, without loss of generality we can decompose $M = \alpha xx^\top + X_\perp \Lambda X_\perp^\top$, where Λ is a diagonal matrix, such that $M^+ = \frac{1}{\alpha} xx^\top + X_\perp \Lambda^+ X_\perp^\top$. First we consider the expansion of the left hand side of Eqn. (14)

$$\lim_{\omega\to\infty} \left((\alpha+\omega)\, xx^\top + X_\perp \Lambda X_\perp^\top\right)^+ = \lim_{\omega\to\infty} \frac{1}{\alpha+\omega} xx^\top + X_\perp \Lambda^+ X_\perp^\top = X_\perp \Lambda^+ X_\perp^\top.$$

Similar, by expanding the right hand side we get

$$\left(\left(I - xx^\top\right)\left(\alpha xx^\top + X_\perp \Lambda X_\perp^\top\right)\left(I - xx^\top\right)\right)^+ = \left(X_\perp \Lambda X_\perp^\top\right)^+ = X_\perp \Lambda^+ X_\perp^\top.$$

■

Lemma 2 *For $M' = M + \omega \sum_i x_i x_i^\top$ where $\omega \geq 0$ it holds that $\lambda_k(M') \geq \lambda_k(M)$.*

Proof: All eigenvalues of the sum of rank-1 perturbations are non-negative

$$\omega \sum_i x_i x_i^\top \succeq 0 \Rightarrow M' \succeq M.$$

■

Lemma 3 *Given an orthonormal basis, $X = [x_1, \ldots, x_{n-1}]$, i.e., $X^\top D_G X = I$, and unit length seed $s^\top D_G s = 1$. Then, any unit length vector $x_n^\top D_G x_n = 1$, perpendicular to the subspace $X^\top D_G x_n = 0$, will have a correlation with the seed bounded by*

$$0 \leq (x_n^\top D_G s)^2 \leq 1 - \sum_{i=1}^{n-1} (x_i^\top D_G s)^2.$$

Proof: The proof follows directly from the Pythagorean theorem. Let $X = [x_1, \ldots, x_N]$ be the orthonormal basis of $\mathbb{R}^N$, i.e., spanning s. Then

$$\sum_{i=1}^{N} (x_i^\top D_G s)^2 = (s^\top D_G s)^2 = 1.$$

■

Lemma 4 *For the matrix $P_\gamma = \mathcal{L}_G - \gamma I$ it holds that*

$$P_\gamma^+ - P_{\hat{\gamma}}^+ = (\gamma - \hat{\gamma}) P_{\hat{\gamma}}^+ P_\gamma^+, \tag{15}$$

given that neither γ nor $\hat{\gamma}$ coincides with an eigenvalue of $\mathcal{L}_G$.

Proof: The proof follows directly by plain algebra. Simply substitute the SVD $P_\gamma = V \Lambda_\gamma V^T$, where Λ_γ is a diagonal matrix with the eigenvalues shifted by γ, into Eqn. (15)

$$\begin{aligned} V\Lambda_\gamma^+ V^T - V\Lambda_{\hat{\gamma}}^+ V^T &= (\gamma - \hat{\gamma}) V \Lambda_{\hat{\gamma}}^+ V^T V \Lambda_\gamma^+ V^T \\ V\Lambda_\gamma^+ V^T - V\Lambda_{\hat{\gamma}}^+ V^T &= (\gamma - \hat{\gamma}) V \Lambda_{\hat{\gamma}}^+ \Lambda_\gamma^+ V^T \\ \Rightarrow \Lambda_\gamma^+ - \Lambda_{\hat{\gamma}}^+ &= (\gamma - \hat{\gamma}) \Lambda_{\hat{\gamma}}^+ \Lambda_\gamma^+. \end{aligned}$$

The system is decoupled so it will be sufficient to consider a single eigenvalue

$$\begin{aligned} \frac{1}{\lambda_i - \gamma} - \frac{1}{\lambda_i - \hat{\gamma}} &= \frac{\gamma - \hat{\gamma}}{(\lambda_i - \hat{\gamma})(\lambda_i - \gamma)} \\ \frac{\lambda_i - \hat{\gamma}}{(\lambda_i - \hat{\gamma})(\lambda_i - \gamma)} - \frac{\lambda_i - \gamma}{(\lambda_i - \hat{\gamma})(\lambda_i - \gamma)} &= \frac{\gamma - \hat{\gamma}}{(\lambda_i - \hat{\gamma})(\lambda_i - \gamma)} \\ \frac{\gamma - \hat{\gamma}}{(\lambda_i - \hat{\gamma})(\lambda_i - \gamma)} &= \frac{\gamma - \hat{\gamma}}{(\lambda_i - \hat{\gamma})(\lambda_i - \gamma)}. \end{aligned}$$

Also, this trivially holds for the rank deficient case, i.e., $0 = 0$.

■

Lemma 5 *As pointed out in Section 3.3, it is already immediate that the initial semi-supervised eigenvector can be computed using a diffusion-based procedure, such as the Push algorithm. However, from that discussion it remains unclear how the approach can be generalized for the consecutive $k-1$ semi-supervised eigenvectors. It turns out that the k^{th} solution is approximated by*

$$x_k \approx c(I - XX^T D_G)(L_G - \gamma_k D_G)^+ D_G s, \tag{16}$$

given that $(L_G - \gamma_k D_G)^+ D_G s$ is linearly independent with respect to the previous $k-1$ solutions contained in X.

Proof: By Eqn. (9) the solution for the second semi-supervised eigenvector can be expressed as

$$y_2 = c\left(P_{\gamma_2}{}^{+} - P_{\gamma_2}{}^{+} y_1 (y_1^T P_{\gamma_2}{}^{+} y_1)^{+} y_1^T P_{\gamma_2}{}^{+}\right) D_G^{1/2} s,$$

where $(y_1^T P_{\gamma_2}{}^{+} y_1)^{+}$ is a constant. For notational convenience we start by substituting $b = D_G^{1/2} s$ together with the explicit solution $y_1 \propto P_{\gamma_1}{}^{+} b$

$$y_2 = c P_{\gamma_2}{}^{+} b - \frac{c P_{\gamma_2}{}^{+} P_{\gamma_1}{}^{+} b b^T P_{\gamma_1}{}^{+} P_{\gamma_2}{}^{+} b}{b^T P_{\gamma_1}{}^{+} P_{\gamma_2}{}^{+} P_{\gamma_1}{}^{+} b},$$

and for the same reason we also introduce $\rho_{\gamma_1\gamma_2} = b^\top P_{\gamma_1}{}^{+} P_{\gamma_2}{}^{+} b$

$$y_2 = c P_{\gamma_2}{}^{+} b - \frac{c \rho_{\gamma_1\gamma_2} P_{\gamma_2}{}^{+} P_{\gamma_1}{}^{+} b}{b^T P_{\gamma_1}{}^{+} P_{\gamma_2}{}^{+} P_{\gamma_1}{}^{+} b}.$$

We can approximate this expression by exploiting the structural result of Lemma 4, namely that $P_{\gamma_1}{}^{+} - P_{\gamma_2}{}^{+} = (\gamma_1 - \gamma_2) P_{\gamma_2}{}^{+} P_{\gamma_1}{}^{+}$

$$\begin{aligned} y_2 &\approx c P_{\gamma_2}{}^{+} b - \frac{c \rho_{\gamma_1\gamma_2} (P_{\gamma_1}{}^{+} - P_{\gamma_2}{}^{+}) b}{b^T P_{\gamma_1}{}^{+} (P_{\gamma_1}{}^{+} - P_{\gamma_2}{}^{+}) b} \\ &= c P_{\gamma_2}{}^{+} b - \frac{c \rho_{\gamma_1\gamma_2} (P_{\gamma_1}{}^{+} - P_{\gamma_2}{}^{+}) b}{\rho_{\gamma_1\gamma_1} - \rho_{\gamma_1\gamma_2}}. \end{aligned}$$

We emphasize that this approximation is exact whenever $P_{\gamma_1}{}^{+} - P_{\gamma_2}{}^{+}$ is well-conditioned, and singular for $\gamma_1 = \gamma_2$. Then, substitute $c = \frac{\rho_{\gamma_1\gamma_1} - \rho_{\gamma_1\gamma_2}}{\rho_{\gamma_1\gamma_1}}$

$$\begin{aligned} y_2 &\approx \frac{\rho_{\gamma_1\gamma_1} P_{\gamma_2}{}^{+} b - \rho_{\gamma_1\gamma_2} P_{\gamma_2}{}^{+} b}{\rho_{\gamma_1\gamma_1}} - \frac{\rho_{\gamma_1\gamma_2} (P_{\gamma_1}{}^{+} - P_{\gamma_2}{}^{+}) b}{\rho_{\gamma_1\gamma_1}} \\ &= \frac{\rho_{\gamma_1\gamma_1} P_{\gamma_2}{}^{+} b - \rho_{\gamma_1\gamma_2} P_{\gamma_2}{}^{+} b - \rho_{\gamma_1\gamma_2} P_{\gamma_1}{}^{+} b + \rho_{\gamma_1\gamma_2} P_{\gamma_2}{}^{+} b}{\rho_{\gamma_1\gamma_1}} \\ &= \frac{\rho_{\gamma_1\gamma_1} P_{\gamma_2}{}^{+} b - \rho_{\gamma_1\gamma_2} P_{\gamma_1}{}^{+} b}{\rho_{\gamma_1\gamma_1}} \\ &= P_{\gamma_2}{}^{+} b - \frac{\rho_{\gamma_1\gamma_2} P_{\gamma_1}{}^{+} b}{\rho_{\gamma_1\gamma_1}}. \end{aligned}$$

By resubstituting for the auxiliary variables we obtain the desired result

$$y_2 \approx c(I - y_1 y_1^T)(\mathcal{L}_G - \gamma I)^{+} D_G^{1/2} s,$$

and by applying this procedure recursively it follows that

$$y_k \approx c(I - Y Y^T)(\mathcal{L}_G - \gamma_k I)^{+} D_G^{1/2} s.$$

Finally, we can relate this result to the combinatorial graph Laplacian as follows

$$\begin{aligned} y_k &\approx c(I - D_G^{1/2} X X^T D_G^{1/2}) D_G^{1/2} (L_G - \gamma_k D_G)^+ D_G s \\ &= c(D_G^{1/2} - D_G^{1/2} X X^T D_G)(L_G - \gamma_k D_G)^+ D_G s \\ &= c D_G^{1/2} (I - X X^T D_G)(L_G - \gamma_k D_G)^+ D_G s, \end{aligned}$$

and due to the relationship $x_k = D_G^{-1/2} y_k$ it follows that

$$x_k \approx c(I - X X^T D_G)(L_G - \gamma_k D_G)^+ D_G s.$$

■

Appendix B. Derivation of Sparse Graph Diffusions

To allow efficient computation of semi-supervised eigenvectors by graph diffusions, we must make the relationship with the sparse seed vector explicit. Here we specifically consider the derivation of Eqn. (13). Given a sparse seed indicator s_0, we can write the seed vector s as $s \propto D_G^{-1/2}(I - v_0 v_0^T) s_0$, where $v_0 \propto \mathrm{diag}(D^{1/2})$ is the leading eigenvector of the normalized graph Laplacian (corresponding to the all-one vector of the combinatorial graph Laplacian). Using this explicit form of s we can rewrite the leading solution as

$$\begin{aligned} x_1 &= c(L_G - \gamma D_G)^+ D_G s \\ &= c D_G^{-1/2} (\mathcal{L}_G - \gamma I)^+ D_G^{1/2} s \\ &= c D_G^{-1/2} (\mathcal{L}_G - \gamma I)^+ D_G^{1/2} D_G^{-1/2} (I - v_0 v_0^T) s_0 \\ &= c D_G^{-1/2} \left((\mathcal{L}_G - \gamma I)^+ s_0 - (\mathcal{L}_G - \gamma I)^+ v_0 v_0^T s_0 \right). \end{aligned}$$

Since $\mathcal{L}_G - \gamma I$ simply shifts the eigenvalues of $\mathcal{L}_G$ by $-\gamma$, the latter term simplifies to

$$\begin{aligned} x_1 &= c D_G^{-1/2} \left((\mathcal{L}_G - \gamma I)^+ s_0 - \left(\frac{1}{-\gamma} v_0 v_0^T \right) v_0 v_0^T s_0 \right) \\ &= c D_G^{-1/2} \left((\mathcal{L}_G - \gamma I)^+ s_0 + \frac{1}{\gamma} v_0 v_0^T s_0 \right) \\ &= c D_G^{-1/2} \left(\frac{1}{-\gamma} D_G^{-1/2} \mathrm{pr}_\epsilon \left(\frac{\gamma}{\gamma - 2}, D_G^{1/2} s_0 \right) + \frac{1}{\gamma} v_0 v_0^T s_0 \right). \end{aligned}$$

Finally, by exploiting the peeling result in Eqn. (16), we can use the Push algorithm to approximate the sequence of semi-supervised eigenvectors in an extremely efficient manner

$$x_t^* \approx c \left(I - X X^T D_G \right) \left(D_G^{-1} \mathrm{pr}_\epsilon \left(\frac{\gamma_t}{\gamma_t - 2}, D_G^{1/2} s_0 \right) - D_G^{-1/2} v_0 v_0^T s_0 \right),$$

as the Push algorithm is only applied on the sparse seed set.

Appendix C. Nyström Approximation For The Normalized Graph Laplacian

The vanilla procedure is as follows; we choose m samples at random from the full data set, and for notational simplicity we reorder the samples so that these m samples are followed by the remaining $n = N - m$ samples, i.e., we can partition the adjacency matrix as

$$A_G = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix},$$

where $A \in \mathbb{R}^{m \times m}$, $B \in \mathbb{R}^{m \times n}$, and $C \in \mathbb{R}^{n \times n}$, with $N = m + n$ and $m \ll n$. The Nyström extension then approximates the huge C matrix in terms of A and B, so the resulting approximation to weight matrix becomes

$$A_G \approx \hat{A}_G = \begin{pmatrix} A & B \\ B^T & B^T A^{-1} B \end{pmatrix}.$$

Hence, rather than encoding only each nodes k-nearest-neighbors into the weight matrix, the Nyström methods provides a low-rank approximation to the entire dense weight matrix. Since the leading eigenvectors of $D_G^{-1/2} A_G D_G^{-1/2}$ correspond to the smallest of $\mathcal{L}_G$, our goal is to diagonalize $D_G^{-1/2} A_G D_G^{-1/2}$. At the risk of washing out the local heterogeneities the Nyström procedure approximates the largest eigenvectors of $D_G^{-1/2} A_G D_G^{-1/2}$ using the normalized matrices $\tilde{A}$ and $\tilde{B}$

$$\tilde{A}_{ij} = \frac{A_{ij}}{\sqrt{\hat{d}_i \hat{d}_j}}, \quad i, j = 1, \ldots m$$
$$\tilde{B}_{ij} = \frac{B_{ij}}{\sqrt{\hat{d}_i \hat{d}_{j+m}}}, \quad i = 1, \ldots m, \quad j = 1, \ldots, n.$$

Finally, let $U \Lambda U^T$ be the SVD of $\tilde{A} + \tilde{A}^{-1/2} \tilde{B} \tilde{B}^T \tilde{A}^{-1/2}$, then the m leading eigenvectors are approximated by

$$V = \begin{pmatrix} \tilde{A} \\ \tilde{B}^T \end{pmatrix} \tilde{A}^{-1/2} U \Lambda^{-1/2},$$

and the normalized graph Laplacian by $\mathcal{L}_G \approx I - V \Lambda V^T$.

References

R. Andersen and K. Lang. Communities from seed sets. In *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, pages 223–232, 2006.

R. Andersen, F.R.K. Chung, and K. Lang. Local graph partitioning using PageRank vectors. In *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 475–486, 2006.

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

M. Belkin and P. Niyogi. Semi-supervised learning on Riemannian manifolds. *Machine Learning*, 56:209–239, 2004.

M. Belkin and P. Niyogi. Towards a theoretical foundation for Laplacian-based manifold methods. *Journal of Computer and System Sciences*, 74(8):1289–1308, 2008.

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

P. Boldi and S. Vigna. The push algorithm for spectral ranking. Technical report, 2011. Preprint: arXiv:1109.4680 (2011).

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems 15: Proceedings of the 2002 Conference*, pages 585–592, 2003.

F.R.K. Chung. *Spectral graph theory*, volume 92 of *CBMS Regional Conference Series in Mathematics*. American Mathematical Society, 1997.

R.R. Coifman, S. Lafon, A.B. Lee, M. Maggioni, B. Nadler, F. Warner, and S.W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition in data: Diffusion maps. *Proceedings of the National Academy of Sciences of the United States of America*, 102(21):7426–7431, 2005.

M. Cucuringu and M. W. Mahoney. Localization on low-order eigenvectors of data matrices. Technical report, 2011. Preprint: arXiv:1109.1355 (2011).

P. Drineas and M. W. Mahoney. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

A. P. Eriksson, C. Olsson, and F. Kahl. Normalized cuts revisited: A reformulation for segmentation with linear grouping constraints. In *Proceedings of the 11th International Conference on Computer Vision*, pages 1–8, 2007.

W. Gander, G. H. Golub, and U. von Matt. A constrained eigenvalue problem. *Linear Algebra and its Applications*, 114/115:815–839, 1989.

A. Gittens and M. W. Mahoney. Revisiting the Nyström method for improved large-scale machine learning. Technical report, 2012. Preprint arXiv:1303.1849 (2012).

J. Ham, D.D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *Proceedings of the 21st International Conference on Machine Learning*, pages 000–000, 2004.

T.H. Haveliwala. Topic-sensitive PageRank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):784–796, 2003.

G. Jeh and J. Widom. Scaling personalized web search. In *WWW '03: Proceedings of the 12th International Conference on World Wide Web*, pages 271–279, 2003.

T. Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the 20th International Conference on Machine Learning*, pages 290–297, 2003.

R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th International Conference on Machine Learning*, pages 315–322, 2002.

Y. Lecun and C. Cortes. The MNIST database of handwritten digits. URL `http://yann.lecun.com/exdb/mnist/`. `http://yann.lecun.com/exdb/mnist/`.

J. Leskovec, K.J. Lang, A. Dasgupta, and M.W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW '08: Proceedings of the 17th International Conference on World Wide Web*, pages 695–704, 2008.

M. W. Mahoney and L. Orecchia. Implementing regularization implicitly via approximate eigenvector computation. In *Proceedings of the 28th International Conference on Machine Learning*, pages 121–128, 2011.

M. W. Mahoney, L. Orecchia, and N. K. Vishnoi. A local spectral method for graphs: with applications to improving graph partitions and exploring data graphs locally. *Journal of Machine Learning Research*, 13:2339–2365, 2012.

S. Maji, N. K. Vishnoi, and J. Malik. Biased normalized cuts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2057–2064, 2011.

P.J. Mucha, T. Richardson, K. Macon, M.A. Porter, and J.P. Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328(5980):876–878, 2010.

A.Y. Ng, M.I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 15: Proceedings of the 2001 Conference*, 2001.

K. A. Norman, S. M. Polyn, G. J. Detre, and J. V. Haxby. Beyond mind-reading: multi-voxel pattern analysis of fMRI data. *Trends in Cognitive Sciences*, 10(9):424–430, 2006.

L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

B. Paolo and V. Sebastiano. The WebGraph framework I: Compression techniques. In *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*, pages 595–601, Manhattan, USA, 2004. ACM Press.

B. Paolo, C. Bruno, S. Massimo, and V. Sebastiano. Ubicrawler: A scalable fully distributed web crawler. *Software: Practice & Experience*, 34(8):711–726, 2004.

B. Paolo, R. Marco, S. Massimo, and V. Sebastiano. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In *Proceedings of the 20th International Conference on World Wide Web*. ACM Press, 2011.

P. O. Perry and M. W. Mahoney. Regularized Laplacian estimation and fast eigenvector approximation. In *Advances in Neural Information Processing Systems 25: Proceedings of the 2011 Conference*, 2011.

K.T. Poole. The decline and rise of party polarization in congress during the twentieth century. *Extensions: A Journal of the Carl Albert Congressional Research and Studies Center*, Fall 2005.

K.T. Poole and H. Rosenthal. Patterns of congressional voting. *American Journal of Political Science*, 35:228–278, 1991.

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

S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by local linear embedding. *Science*, 290:2323–2326, 2000.

L. K. Saul, K. Q. Weinberger, J. H. Ham, F. Sha, and D. D. Lee. Spectral methods for dimensionality reduction. In O. Chapelle, B. Schoelkopf, and A. Zien, editors, *Semisupervised Learning*, pages 293–308. MIT Press, 2006.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transcations of Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

D.A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC '04: Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 81–90, 2004.

M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. In *Advances in Neural Information Processing Systems 14: Proceedings of the 2001 Conference*, pages 945–952, 2002.

A. Talwalkar and A. Rostamizadeh. Matrix coherence and the Nyström method. In *Proceedings of the 26th Conference in Uncertainty in Artificial Intelligence*, 2010.

J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

S.-H. Teng. The Laplacian paradigm: Emerging algorithms for massive graphs. In *Proceedings of the 7th Annual Conference on Theory and Applications of Models of Computation*, pages 2–14, 2010.

S. Vigna. Spectral ranking. Technical report, 2009. Preprint: arXiv:0912.0238 (2009).

D.J. Watts and S.H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393: 440–442, 1998.

A. S. Waugh, L. Pei, J. H. Fowler, P. J. Mucha, and M. A. Porter. Party polarization in congress: A network science approach. Technical report, 2009. Preprint: arXiv:0907.3509 (2009).

C.K.I. Williams and M. Seeger. The effect of the input density distribution on kernel-based classifiers. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1159–1166, 2000.

C.K.I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, pages 682–688, 2001.

S. X. Yu and J. Shi. Grouping with bias. In *Advances in Neural Information Processing Systems 14: Proceedings of the 2001 Conference*, pages 1327–1334, 2002.

D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*, pages 321–328, 2004.