COMMUN. MATH. SCI.
Vol. 16, No. 5, pp. 1241–1265

A SEMI-SUPERVISED HEAT KERNEL PAGERANK MBO ALGORITHM FOR DATA CLASSIFICATION*

EKATERINA MERKURJEV†, ANDREA L. BERTOZZI‡, AND FAN CHUNG§

Abstract. We present an accurate and efficient graph-based algorithm for semi-supervised classification that is motivated by recent successful threshold dynamics approaches and derived using heat kernel pagerank. Two different techniques are proposed to compute the pagerank, one of which proceeds by simulating random walks of bounded length. The algorithm produces accurate results even when the number of labeled nodes is very small, and avoids computationally expensive linear algebra routines. Moreover, the accuracy of the procedure is comparable with or better than that of state-of-the-art methods and is demonstrated on benchmark data sets. In addition to the main algorithm, a simple novel procedure that uses heat kernel pagerank directly as a classifier is outlined. We also provide detailed analysis, including information about the time complexity, of all proposed methods.

Keywords. heat kernel pagerank; graphs; graph Laplacian; threshold dynamics; random walk.

AMS subject classifications. 35K05; 35K08; 35RO2; 60J75; 62-09; 62-07; 65S05; 94C15; 97K30.

1. Introduction

Classification is an important problem in machine learning and computer vision. The goal is to partition a data set efficiently and accurately into a number of classes. This task occurs in numerous applications, such as medical or genomics data, spam detection, face recognition and financial predictions. In this paper, we present a very accurate and efficient semi-supervised algorithm for data classification, named heat kernel pagerank MBO, formulated in a graphical setting and derived using heat kernel pagerank. This procedure is simple, performs well even for a very small number of labeled nodes, and its time complexity is linear in the number of nodes of the data set in the case of sparse graphs. The task can be done in $O(rnK)$ operations in the worst case for dense graphs, for a fixed number of classes m, where r is the number of random walks, n is the size of the data set and K is the maximum number of steps of each random walk. The heat kernel pagerank MBO method also avoids computationally expensive linear algebra routines. Moreover, in this paper, in addition to the main algorithm, we also derive a simple method that uses heat kernel pagerank directly as a classifier. To the best of our knowledge, this is the first application of heat kernel pagerank to semi-supervised data classification.

We approach the classification problem via the use of heat kernel pagerank, one of the many versions of pagerank [34]. The original pagerank [61], was designed by Google in the 1990's to rank the importance of websites in search engine results. However, partly due to its generality and simplicity, it was later applied to many other tasks in a variety of fields, such as biology and chemistry [34]. Overall, two particular uses comprise the

*Received: July 6, 2016; Accepted (in revised form): April 21, 2018. Communicated by Lexing Ying.
This work was supported by NSF grants DMS-1417674 and DMS-1118971, ONR grant N00014-16-1-2119 and AFSOR FA9550-09-1-0090. The first author was supported by the UC President's Postdoctoral Fellowship.

†Department of Mathematics and CMSE, Michigan State University, East Lansing, MI, 48864, USA. (merkurje@msu.edu).
‡Department of Mathematics, University of California at Los Angeles, Los Angeles, CA, 90095, USA. (bertozzi@math.ucla.edu).
§Department of Mathematics, University of California at San Diego, La Jolla, CA, 92093, USA. (fan@ucsd.edu).

majority of the applications of pagerank. First, pagerank measures the "significance" of each node in relation to the whole graph (i.e. in [46]); in other words, pagerank produces a ranking of the vertices. Second, it is often used for local graph clustering to find a cluster around a particular point (i.e. in [23]). In this paper, we apply heat kernel pagerank to semi-supervised data classification, using recent threshold dynamics approaches such as [31] and [56] for the motivation. In the latter work, extended to the multiclass case in the former paper, the authors propose a threshold dynamics data segmentation scheme, building upon the graph Ginzburg-Landau functional that is provably close to the graph cut measure of a partition [5], and is closely related to total variation through gamma convergence. Total variation has had a major impact in low-dimensional image processing in Euclidean space [12–14, 64] and its graphical cousin has been studied in machine learning and network science through quantities such as the Cheeger cut, Ratio cut, and modularity optimization [39, 54].

Using heat kernel pagerank provides several advantages. It can be computed very efficiently by two approaches, described in Section 4, one of which consists of simulating random walks of bounded length. The techniques avoid potentially computationally expensive linear algebra routines, such as the calculation of eigenvalues and eigenvectors of large matrices. These calculations are present in [31] and [56].

Embedding the data in a graphical setting, which is the framework used in this paper, provides several advantages. First, the setting also affords the flexibility to incorporate diverse types of data sets, i.e. set of images, hyperspectral data, text data, etc. Moreover, it provides a way to analyze data whose different clusters are *not* linearly separable. The setting also allows procedures to exploit non-local information and take into account the relationship between any two nodes in the data set. In the case of image processing, this allows one to capture repetitive structure and texture [56].

The methods we formulate are semi-supervised, where a set of known labelings, called the *fidelity set*, is to be provided a priori. One advantage of our main algorithm is that it is accurate even in the case of a very small number of labeled points. In fact, for several data sets, we obtain good accuracies even when the fidelity set is comprised of less than 0.5% of all the nodes.

We note that the number of classes needs to be specified for our algorithm. In the case when the number of desired clusters is unknown, several strategies can be applied. One strategy is to first approximate the number. The elbow method [44] plots the percentage of variance against the number of clusters to locate the "elbow criterion", which is then set to be the desired number of clusters. Another approach is X-means clustering [62], which proceeds similarly to k-means clustering and repeatedly alternates between subdividing clusters and proceeding with the best split. Once a certain stopping criterion is reached, the result will possess the desired number of clusters.

We mention a few recent papers on modularity optimization in the case when the number of clusters is unknown. The work [39] by Hu et al. interprets modularity optimization from a very new perspective as a minimization problem involving the total variation functional. In addition, Boyd et al. in [6] explore a similar connection between modularity and total variation. Moreover, [40] considers the multi-class segmentation problem using the piecewise constant Mumford-Shah model in a graph setting, and solves it very efficiently.

1.1. Main contributions and results. The main contributions of the paper are the following:

- We introduce a very accurate and efficient graph-based algorithm for semi-supervised learning (SSL) data classification, called *heat kernel pagerank MBO*,

that uses heat kernel pagerank for the main steps.

- In addition to the main algorithm, we outline a simple semi-supervised learning method that uses heat kernel pagerank *directly* as a classifier.
- This is the first application of heat kernel pagerank to semi-supervised data classification, to the best of our knowledge.
- All proposed methods avoid potentially computationally expensive linear algebra routines; for example, the calculation of eigenvalues and eigenvectors of large matrices.
- We provide a detailed analysis of our algorithms, including its time complexity. The time complexity in the case of a sparse graph is linear in the number of nodes in the data set. For a dense graph, the random walk approach requires $O(rnK)$ operations in the worst case, for fixed number of classes m, where r is the number of random walks, n is the size of the data set, and K is the bound on the number of steps of each random walk.
- Analysis and experiments show that the algorithms handle the cases of both sparse and dense graphs very efficiently.
- The *semi-supervised* heat kernel pagerank MBO algorithm performs very well even when the number of labeled nodes is very small. For example, for the MNIST data set, even when only 0.43% points are associated with a known class, the procedure is still able to classify around 94.46% of the nodes correctly! When 1.32% points are associated with a known class, the accuracy reaches 97.44%.
- We describe two different approaches for computing heat kernel pagerank, one of them is more suited to the case of sparse graphs, the second- to the case of dense graphs.
- We validate our methods by performing experiments using benchmark data sets. It produces accuracy better than or comparable with that of the state-of-the-art.

The paper is organized as follows: Section 2 provides some background on the graphical framework and heat kernel pagerank, and Section 3 presents a model using heat kernel pagerank directly as a classifier. Section 4 formulates the new algorithm as well as provides details on computing heat kernel pagerank, and Section 5 provides some detailed analysis on the algorithms presented in the paper and gives information on the time complexity of the procedures. Section 6 presents experimental results on benchmark data sets, and Section 7 concludes the paper.

1.2. Previous work. The problem of classification involves exploiting the similarities and the structure of the data to properly cluster the set into several groups. Our procedure uses the graphical framework, involving a set of vertices and edges with weights assigned on the edges, as a simple and clear way to obtain the underlying similarities in a data set.

The graphical setting has been often used in the literature for that purpose: for example, [3, 16, 72, 74–76] consider graphs for their methods. The framework has been especially useful in image processing applications [7, 27–29, 35–37, 41, 50, 52, 65]. Specific applications include the image denoising method of Buades [10] and that of Elmoataz [29], which incorporates generalizations of the graph Laplacian for the task of manifold smoothing and image denoising. In addition, methods involving spectral graph theory [19, 59], based on a graphical setting, are often formulated to solve clustering

problems.

More recently, the graphical setting has been used as a framework for some optimization tasks. In particular, a popular general minimization problem considered for machine learning and image processing is

$$\min_u \Big\{ E(u) = R(u) + F(u) \Big\},$$

where $R(u)$ is the regularization functional, $F(u)$ is the fidelity term and u is a function defined on the space of the data set that describes an appropriate characteristic. The $R(u)$ term in some sense smoothes the data, and the fidelity term allows one to incorporate labeled nodes. With regards to the regularization term, the total variation (TV) functional has been used very successfully in image processing tasks [15, 33, 73]. Due to some of its unfavorable properties, [31, 39, 56] use the Ginzburg-Landau functional as a smooth but arbitrarily close approximation to the TV semi-norm. Our method is motivated by these algorithms and the work involving heat kernel pagerank [20–23].

Heat kernel pagerank, described in detail in [20], has been applied in several recent works to a number of tasks. For example, [21] presents a local graph partitioning algorithm using the pagerank, which, for a subset with Cheeger ratio h, finds local cuts with Cheeger ratio at most $O(\sqrt{h})$. In [22], the authors describe an algorithm solving linear systems with boundary conditions using heat kernel pagerank. The method in [23] is a local clustering algorithm, which uses a novel way of computing the pagerank. An interesting application is presented in [66], which outlines a method for finding a consensus value in multi-agent networks. In addition, [24] considers the problem of computing a local cluster in a graph in the distributed setting using the pagerank.

The classification algorithm presented in the paper is semi-supervised, so that a small set of labeled data points is provided to the method in advance. Such problems have been studied in the context of diffuse interfaces; for example, [4] introduces a procedure for image processing and machine learning in which the Ginzburg-Landau functional with fidelity term is minimized using convex splitting. Other semi-supervised learning approaches involving the functional include [31, 32, 55–57]. Semi-supervised approaches derived using total variation include [2, 53].

In the case of unsupervised methods, there are no apriori labeled nodes. Such algorithms proceed by clustering similar nodes together. To prevent trivial solutions, a constraint on the size of the clusters is usually incorporated into the procedure. Two popular constraints are the normalized cut [54, 71] and the Cheeger cut [17, 54, 65]; each of them favors solutions where the clusters are of similar size. The latter constraint has been studied recently in works such as [8, 9, 38, 68].

2. Background

2.1. Graph framework. In this paper, we embed the data set into a graphical framework with an undirected graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of edges. A weight function (w) is defined on each edge, and represents a measure of similarity between the vertices it is connecting. The degree of a vertex x is formulated as $d(x) = \sum_{y \in \mathcal{V}} w(x,y)$. Denote by D the diagonal matrix containing the degrees of vertices as diagonal entries, and let W denote the similarity matrix containing the weight function values. Let the set of vertices $\mathcal{V}$ and edges $\mathcal{E}$ be Hilbert spaces defined via the following inner products:

$$\langle u, \gamma \rangle_{\mathcal{V}} = \sum_x u(x)\gamma(x)d(x)^r,$$

$$\langle \psi, \phi \rangle_{\mathcal{E}} = \frac{1}{2} \sum_{x,y} \psi(x,y)\phi(x,y)w(x,y)^{2q-1},$$
$$\|u\|_{\mathcal{V}} = \sqrt{\langle u, u \rangle_{\mathcal{V}}},$$
$$\|\phi\|_{\mathcal{E},\infty} = \max_{x,y} |\phi(x,y)|,$$

for some $r \in [0,1]$, $q \in [\frac{1}{2}, 1]$, $u, \gamma \in V$, and $\psi, \phi \in E$. The gradient operator is defined as

$$(\nabla u)_w(x,y) = w(x,y)^{1-q}(u(y) - u(x)),$$

and the divergence operator can be formulated as the adjoint of the gradient

$$(\mathrm{div}_w \phi)(x) = \frac{1}{2d(x)^r} \sum_y w(x,y)^q (\phi(x,y) - \phi(y,x)),$$

where the following adjoint equation is used: $\langle \nabla u, \phi \rangle_{\mathcal{E}} = -\langle u, \mathrm{div}_w \phi \rangle_{\mathcal{V}}$. Using the operators above, one can define a family of graph Laplacians $\triangle_w = \mathrm{div}_w \dot{\nabla} : \mathcal{V} \to \mathcal{V}$:

$$(\triangle_w u)(x) = \sum_y \frac{w(x,y)}{d(x)^r}(u(y) - u(x)).$$

If u is viewed as a vector in $\mathbb{R}^m$, we can write

$$-\triangle_w u = (D^{1-r} - D^{-r}W)u.$$

We also formulate the Dirichlet energy and total variation:

$$\frac{1}{2}\|\nabla u\|_{\mathcal{E}}^2 = \frac{1}{4} \sum_{x,y} w(x,y)(u(x) - u(y))^2,$$
$$TV_w(u) = \max\left\{ \langle \mathrm{div}_w \phi, u \rangle_{\mathcal{V}} : \phi \in \mathcal{E}, \|\phi\|_{\mathcal{E},\infty} \leq 1 \right\} = \frac{1}{2} \sum_{x.y} w(x,y)^q |u(x) - u(y)|.$$

The expression $D^{1-r} - D^{-r}W$ forms a general expression for the graph Laplacian, which is the graph version of the Laplace operator. The parameter r allows the user to control the type of Laplacian. For example, the cases of $r = 0$ and $r = 1$ result in the unnormalized Laplacian $L = D - W$ and the random walk Laplacian $L_w = D^{-1}L$.

With regards to the weight function w, the goal is to choose w so that similar vertices are weighted by a large value and dissimilar vertices are weighted by a small value. While many versions of weight functions exist, two popular ones are the Gaussian

$$w(x,y) = e^{-\frac{d(x,y)^2}{\sigma^2}}$$

and the Zelnik-Manor and Perona weight function [63], which gives a local rescaling,

$$w(x,y) = e^{-\frac{d(x,y)^2}{\sqrt{\tau(x)\tau(y)}}},$$

where $d(x,y)$ is a metric computing the distance, in some sense, between x and y. Also, $\sqrt{\tau(x)} = d(x,z)$, where z is the M^{th} closest vertex to x.

The metric $d(x,y)$ will depend on the data set. For example, in the case of points in R^2, one might consider it to be the usual Euclidean distance, since points far apart are more likely to be in separate clusters than those closer together. When working with images, one might consider comparing neighborhoods around pixels x and y using the L^2 or L^1 norm. Of course, techniques like rotation invariance can also be incorporated. For text classification, one can use term frequency inverse document frequency to form feature vectors for each document and then use cosine similarity to compare the vectors.

2.2. Heat kernel pagerank and personalized pagerank. The heat equation, formulated as

$$\frac{\partial u}{\partial t} = \Delta u,$$

occurs very frequently in many different kinds of applications. For example, in financial mathematics, it is used to solve the Black-Scholes partial differential equation. In this section, we show some ways of solving the heat equation in a graphical setting:

$$\frac{\partial u}{\partial t} = -Lu, \tag{2.1}$$

where the Laplace operator Δ is replaced by the graph Laplacian.

The *heat kernel* of a graph G is the matrix e^{-tL}, which is a solution to the heat Equation (2.1). If the random walk Laplacian $L_w = I - D^{-1}W = I - P$ is considered, the resulting heat kernel is $H_t = e^{-t(I-P)}$. Note that $P = D^{-1}W$ is just a transition probability matrix for the random walk in which one moves from vertex i to vertex j with probability given by $(D^{-1}W)_{ij}$.

The *heat kernel pagerank* is defined as $\rho_{t,f} = fH_t$ for row vector f:

$$\rho_{t,f} = fH_t = \sum_{k=0}^{\infty} e^{-t}\frac{t^k}{k!} fP^k. \tag{2.2}$$

From (2.2), one can see that, in the case that f is a probability distribution, the heat kernel pagerank is just the expected distribution of the random walk with probability matrix P, where k steps are taken with probability $e^{-t}\frac{t^k}{k!}$, and the starting point is chosen from f. Therefore, in this case, one can approximate the heat kernel pagerank vector by simulating enough random walks (with the number of steps calculated according to the Poisson distribution, and with the starting point taken from f) to obtain a close approximation to the expected distribution. This approach is the one taken by [23, 66] when computing heat kernel pagerank.

In [66], Simpson and Chung show a useful application of heat kernel pagerank. Consider the following heat equation in a graphical setting:

$$\dot{u}(t) = -L_w u(t), \qquad u(0) \in R^n. \tag{2.3}$$

According to Theorem III.2 in [66], the solution to (2.3) is given by

$$u(t) = D^{-1}\rho_{t,f}^{tr}, \qquad f = u(0)^{tr} D, \tag{2.4}$$

where M^{tr} denotes the transpose. Therefore, the heat equation in a graphical setting, defined using the random walk Laplacian can be solved using heat kernel pagerank.

Another version of pagerank is *personalized pagerank* $pr_{\alpha,f}$ [25, 70], defined by:

$$pr_{\alpha,f} = \sum_{k=0}^{\infty} \alpha(1-\alpha)^k fP^k, \tag{2.5}$$

where P is a transition probability matrix, f is a probability distribution and α is a scalar between 0 and 1. Personalized pagerank describes a surfer model in which a web surfer, with probability α, jumps to a page chosen from a probability distribution f, but with probability $1-\alpha$, moves according to the transition probability matrix. It also represents the probability distribution of the random walk arriving at each node.

3. Heat kernel pagerank as a classifier

In [51], Lin et al. introduce a simple semi-supervised learning model using personalized pagerank directly as a classifier. In fact, their method consists of calculating the personalized pagerank for different f, and assigning the class number to each node depending on which distribution gives a higher value of the pagerank for that node. Note that, very often, the sum of the heat kernel pagerank converges more rapidly than the geometric sum of personalized pagerank. Therefore, in this section, we describe a procedure based directly on heat kernel pagerank instead of personalized pagerank. The method is a direct use of heat kernel pagerank as a classifier.

For a data set of m classes, the procedure to cluster the set into m parts is:

- Calculate $\{f_1,...,f_m\}$ by setting f_i at node x to 1 if x is a labeled node assigned to class i. Otherwise, set it to 0.
- Normalize f_i to have the L^1 norm of 1.
- Calculate m different heat kernel pageranks ρ_{t,f_i} for $i=1:m$.
- The class of node x is $argmax_i\{\rho_{t,f_i}(x)\}$.

We will refer to this algorithm as $HKPR$ in Section 5, where we describe its results and compare it to the other model proposed in this paper. This simple procedure, which uses heat kernel pagerank directly as a classifier, performs well enough; in most cases, the accuracy is only about $1\%-2\%$ worse than that of the main algorithm.

To the best of our knowledge, this is the first application of heat kernel pagerank to semi-supervised classification.

4. Semi-supervised heat kernel pagerank MBO algorithm

4.1. Motivation. To formulate our method, we use a variational approach. Recall the general minimization problem for data classification mentioned in Section 1:

$$\min_u \Big\{ E(u) = R(u) + F(u) \Big\},$$

where u describes the class of the nodes, $R(u)$ is the regularization functional and $F(u)$ is the fidelity term. Regarding the $R(u)$ term, the total variation functional has been used very successfully in image processing applications as a regularization term [33]. In addition, many papers use total variation to approximate the length of the boundary of the binary segmentation function, often referred to as perimeter.

Another related functional, the Ginzburg-Landau (GL) functional, originally proposed to describe physical phenomena such as superconductivity, is formulated as

$$GL(u) = \frac{\varepsilon}{2}\int |\nabla u|^2 dx + \frac{1}{\varepsilon}\int \mathcal{W}(u)dx,$$

where $\mathcal{W}(u)$ is a double-well potential, such as $\mathcal{W}(u)=\frac{1}{4}u^2(u-1)^2$, and ε is a positive constant. It has been shown [45] that the GL functional converges, in the sense of Γ-convergence, as $\varepsilon \to 0$, to the perimeter, a quantity that is also often approximated by total variation. The relationship between the two functionals also applies in the graphical framework, and can be formulated in terms of gamma convergence. The result is shown in [5], where is it proven that for any r and $q=1$, TV_w is the Γ-limit, in the sense of gamma convergence, of a sequence of graph-based GL-type functionals. Therefore, one can consider the GL functional to be a smooth but arbitrarily close approximation to total variation.

The advantage of using the GL functional for algorithm derivations lies in the fact that it leads to a simple minimization scheme, while the minimization of total variation

results in an unwanted nonlinear curvature term. Moreover, the GL scheme contains the Laplace operator, which allows for the usage of heat kernel pagerank.

Following their ideas, the method in [56] is formulated by considering the Ginzburg-Landau functional as a regularization term. The functional plus fidelity term is minimized using gradient descent, which results in the following equation:

$$\dot{u}(t) = -2\varepsilon L u(t) - \frac{1}{\varepsilon}\mathcal{W}'(u) - \frac{\partial F}{\partial u}. \tag{4.1}$$

Here, note that the Laplace operator is now replaced by the graph Laplacian, since the graphical framework is used. One possibility to solve (4.1), explored by [56], is to split the dynamics into two steps of first propagating by the heat equation with an extra term (derived from the fidelity term) and then propagating by an equation involving the double-well potential. The last step becomes thresholding as $\varepsilon \to 0$. Therefore, this method becomes a variation (proposed in the graphical setting) of the MBO approach proposed by Merriman, Bence and Osher [58] to approximate motion by mean curvature.

Another possibility, which we follow here, is to split the dynamics into *three* steps:

(1) Propagate by the heat equation:

$$\dot{u}(t) = -Lu(t). \tag{4.2}$$

(2) Propagate by the equation involving the fidelity term:

$$\dot{u}(t) = -\frac{\partial F}{\partial u}. \tag{4.3}$$

(3) Propagate by the following equation:

$$\dot{u}(t) = -\frac{1}{\varepsilon}\mathcal{W}'(u). \tag{4.4}$$

To obtain an approximate solution of (4.1) at discrete times, one would alternate between the three steps until convergence. The advantage of this splitting is that step 1 involves solving only the heat equation, and we pursue this splitting to formulate our method. Note that, in the limit as $\varepsilon \to 0$, the last step becomes thresholding:

$$u(x) = \begin{cases} 1 & \text{if } v(x) \geq \frac{1}{2}, \\ 0 & \text{if } v(x) < \frac{1}{2}, \end{cases}$$

where v is the solution to the second step of the dynamics. Due to the binary answer in the last step, this procedure is applicable only to binary problems. The next section will adapt the procedure to an arbitrary amount of classes.

4.2. Algorithm. In the case of m classes and n data points, instead of a vector u, we use a matrix $\mathbf{U} = [\mathbf{u}_1; \ldots; \mathbf{u}_n]$, where every node is associated with a probability distribution $\mathbf{u}_i$ (row vector) over the classes. Moreover, $\mathbf{u}_i$ is an element of the Gibbs simplex Σ^m:

$$\Sigma^m := \left\{ (x_1, \ldots, x_m) \in [0,1]^m \,\middle|\, \sum_{i=1}^{m} x_i = 1 \right\}.$$

We denote the vertices of the simplex by $\boldsymbol{e}_i$ for $i = 1:m$; the vector $\boldsymbol{e}_i$ contains all zero values, except the i^{th} entry, which is equal to 1. To extend the theory of the previous section to an arbitrary amount of classes, we do not change steps (4.2) and (4.3), except that the label vector will now be a matrix $\mathbf{U}$. The third step, however, has to be modified

to pertain to any number of classes, so we first project each row of $\mathbf{U}$ to the simplex (using [18]) and then we displace towards the closest vertex in the Gibbs simplex.

The new algorithm thus consists of the following steps. First, an initial $\mathbf{U}$ is chosen. For fidelity points, we initialize the row of $\mathbf{U}$ to be the corresponding vertex of the simplex. For the rest of the points, we choose the probability distributions randomly. To obtain the next iterate of the matrix $\mathbf{U}$, three steps are taken:

(1) Propagation using the heat equation:

$$\dot{\mathbf{U}} = -L\mathbf{U}. \tag{4.5}$$

(2) Propagation using:

$$\dot{\mathbf{U}} = -\frac{\partial F}{\partial u}(\mathbf{U}) \tag{4.6}$$

(3) Thresholding:

$$\mathbf{u}_i{}^{n+1} = \boldsymbol{e}_h, \tag{4.7}$$

where vertex $\boldsymbol{e}_h$ is the vertex in the simplex closest to the projection of the i^{th} row of the result of step 2 to the simplex, using the procedure in [18].

We alternate between the three steps until:

$$\frac{\|\mathbf{u}_i{}^{n+1} - \mathbf{u}_i{}^{n}\|_2^2}{\|\mathbf{u}_i{}^{n+1}\|_2^2} < \eta,$$

where η is a small positive constant. The final classification is obtained by assigning to node i the class that holds the highest probability in its class distribution $\mathbf{u}_i$.

Step (4.5) is solved using heat kernel pagerank, which we calculate using two different approaches. To be more specific, first let c_i represent the i^{th} column of the result of step (4.5) and c_{i0} represent the i^{th} column of $\mathbf{U}$ at the start of step (4.5). Now, note that (4.5) consists of solving a system of heat equations in graph form:

$$\begin{cases} \dot{u}(t) = -L_w u(t), & u(0) = c_{00} \in R^n \\ \dot{u}(t) = -L_w u(t), & u(0) = c_{10} \in R^n \\ \quad\vdots & \quad\vdots \\ \dot{u}(t) = -L_w u(t), & u(0) = c_{m0} \in R^n, \end{cases} \tag{4.8}$$

where m is the number of classes. We can use heat kernel pagerank to solve each of those problems; details were given in Section 2.2. More precisely, it is known that column i of the result of step (4.5), which is the solution of the i^{th} equation above, is given by

$$c_i = D^{-1}\rho_{t,f}^{tr}, \qquad f = c_{i0}{}^{tr} D.$$

The c_i's are then concatenated to form the matrix $[c_1 \dots c_m]$, which is the result of step (4.5). Step (4.6) is calculated explicitly, i.e.

$$\frac{\mathbf{U}^{n+1} - \mathbf{U}^n}{dt} = -\frac{\partial F}{\partial u}(\mathbf{U}^n).$$

The heat kernel pagerank MBO algorithm is outlined in Figure 4.3. The version of the algorithm that calculates heat kernel pagerank using series will be referred to as HKPR1; the version that uses random walks for the task will be referred to as HKPR2.

Heat Kernel Pagerank Approximation (Series)

Require: a row vector f, probability matrix P of size $n \times n$, $t \in R^+$, maximum max number of terms to include in the series, tolerance tol

$\star$ Initialize $s = e^{-t} f$, $r_1 = f$, $k = 1$.
while $m > tol$ and $c < max$ **do**
$\star$ $r_1^T \leftarrow P^T r_1^T$
$\star$ $r_2 \leftarrow e^{-t} \frac{t^k}{k!} r_1$
$\star$ $s \leftarrow s + r_2$
$\star$ $m \leftarrow ||r_2||_\infty$
$\star$ $k \leftarrow k + 1$
end while
return s

Figure 4.1: *Heat Kernel Pagerank Approximation (Series)*

Heat Kernel Pagerank Approximation (Random Walk)

Require: a row vector f, probability matrix P of size $n \times n$, $t \in R^+$, an upper limit K on length of a walk, a number r of random walks

Write $f = f_+ - f_-$.
Initialize row vectors μ_+ and μ_- to zero row vectors of length n.
for $i = 1 : r$ **do**
$\star$ Choose a starting vertex v_0 using the probability distribution $\frac{f_+}{||f_+||}$.
$\star$ Simulate a random walk starting from v_0 using probability matrix P, where the number of steps k taken is a Poisson random variable (with parameter t), and $k \leq K$.
$\star$ Let v_f be the last vertex of the walk.
$\star$ $\mu_+(v_f) \leftarrow \mu_+(v_f) + 1$
end for
Repeat the *for* loop using probability distribution $\frac{f_-}{||f_-||}$ and μ_-.
return $\frac{||f_+||}{r}\mu_+ - \frac{||f_-||}{r}\mu_-$

Figure 4.2: *Heat Kernel Pagerank Approximation (Random Walk)*

4.3. Computing an approximation to heat kernel pagerank. Step (4.5) of the new algorithm is solved using heat kernel pagerank:

$$\rho_{t,f} = f H_t = \sum_{k=0}^{\infty} e^{-t} \frac{t^k}{k!} f P^k. \tag{4.9}$$

We approximate it via two different approaches: using series and using random walks.

The first approach is to efficiently sum each term of the series (4.9) until the maximum value of the last term is small enough. In other words, each term of the series (4.9) is efficiently calculated until

$$||e^{-t} \frac{t^k}{k!} f P^k||_\infty < \gamma,$$

for some k and γ. This shortened sum approximates heat kernel pagerank. The approach is detailed in Figure 4.1. In practice, we calculate at most j terms of the series, where

Heat Kernel Pagerank MBO Algorithm for Classification

Require: a graph G of size n, degree matrix D, probability matrix P, matrix U holding the probability distributions over classes, number of classes m, class matrix C ($n \times 1$ matrix) holding the class assignments, $t \in R^+$, tolerance tol

Initialize U to U_0.
$\star$ $U = U_0$, $U_{old} = U_0$.
while $d < tol$ **do**
  $\star$ $U_{old} \leftarrow U$
  for $i = 1 : m$ **do**
    $\star$ Let c_i be the i^{th} column of U.
    $\star$ Calculate the heat kernel pagerank $\rho_{t,f}$ using P with the row vector $c_i^{tr} D$, using series for version one (HKPR1) and using random walks for version two (HKPR2).
    $\star$ $c_i \leftarrow D^{-1} \rho_{t,f}^{tr}$
  end for
  $\star$ $U \leftarrow [c_1 \dots c_m]$
  $\star$ Propagate U using

$$\dot{U} = -\frac{\partial F}{\partial u}(U)$$

  $\star$ Project each row of U to the simplex using [18].
  $\star$ Replace each row of U by the closest vertex in the Gibbs simplex
  $\star$ $d \leftarrow \frac{\|U - U_{old}\|_2^2}{\|U\|_2^2}$
end while
for $i = 1 : n$ **do**
  $C_i = argmax(U_i)$, where U_i is the i^{th} row of U.
end for
return C

Figure 4.3: *Heat Kernel Pagerank MBO Algorithm*

j is a chosen variable; for the experiments, we use up to 12 terms in the series. For a sparse matrix P, the sum can be computed efficiently.

To give the motivation for the second approach, we first assume that f is a probability distribution. In this case, fP^k is the distribution on the vertices after k random walk steps (with probability matrix P), starting with a vertex chosen from the probability distribution f. Therefore, heat kernel pagerank can be considered as the expected distribution of the random walk, if k steps are taken with probability $\frac{e^{-t}t^k}{k!}$ (Poisson distribution), with the starting vertex chosen from f.

This motivates a random walk procedure to calculate heat kernel pagerank. We first compute r random walks, where each walk proceeds using the probability transition matrix P for k steps, where k is a Poisson random variable, starting with a vertex chosen from f. However, since long walks occur with low probability, we limit the random walk length to K steps. Then, the distribution over the final vertices of the r random walks can approximate heat kernel pagerank. This is also the approach taken by [23].

Since f might not be a probability distribution, we first decompose it into positive and negative portions, f_+ and f_-, and then normalize each part to sum to one. Then,

$$\rho_{t,f} = fH_t = (f_+ - f_-)H_t = ||f_+|| \Big\{ \frac{f_+}{||f_+||} H_t \Big\} - ||f_-|| \Big\{ \frac{f_-}{||f_-||} H_t \Big\}, \tag{4.10}$$

where, of course, $\frac{f_+}{||f_+||}$ and $\frac{f_-}{||f_-||}$ are probability distributions. The random walk procedure is applied to both of the probability distributions, and the pagerank is obtained using the linear combination in (4.10). This approach are summarized in Figure 4.2.

REMARK 4.1. One of the differences between our algorithms and those of [56] is the fact that we avoid eigenvector and eigenvalue computations. To achieve that goal, the authors of [56] present the Nyström extension procedure. Note that the Nyström extension method can be used to approximate the eigenvectors and eigenvalues of a graph Laplacian involving a dense graph, not a sparse graph. Therefore, in the case of sparse graphs, another algorithm, such as the Rayleigh-Chebyshev procedure of Chris Anderson [1], must be used to compute the eigenvalues and eigenvectors. Of course, any complexity estimate for computing the eigenvalues and eigenvectors will be specific to the class of matrices one is applying the method to. The advantage of the algorithms introduced in this paper is that they avoid potentially computationally expensive linear algebra routines, including the calculation of eigenvectors and eigenvalues of large matrices. As an aside, the Nyström extension procedure is indeed a very efficient method to compute the eigenvectors and eigenvalues of the graph Laplacian of dense graphs; for a graph with n vertices, the complexity of the algorithm is $O(nL)$, where L is the number of samples.

5. Analysis of the algorithms

In this section, we present detailed analysis of the algorithms introduced in the paper. In Section 5.1, we derive error bounds for the approximation of heat kernel pagerank, derived using two different ways: random walks and series. We show that one can construct the approximation as close as needed to the true value of heat kernel pagerank, and specify how it can be done. In Section 5.2, we present the time complexity analysis of our methods, for both sparse and dense graphs.

5.1. Error bounds for approximating heat kernel pagerank. To derive error bounds for the approximation of heat kernel pagerank and its true value, we use a theorem from [23], detailed below as Theorem 5.1. Theorem 5.2 involves the case of computing the approximation using random walks. Theorems 5.3 and 5.4 describe the case of calculating the approximation using series.

5.1.1. Approximation of heat kernel pagerank using random walks.

THEOREM 5.1. *(from [23]) Let X_i be i.d.d. Bernoulli random variables with $X = \sum_{i=1}^{r} X_i$. Then,*

(1) For $0<\varepsilon<1$, $P(X<(1-\varepsilon)\mathbb{E}(X)) < exp(-\frac{\varepsilon^2}{2}\mathbb{E}(X))$.

(2) For $0<\varepsilon<1$, $P(X>(1+\varepsilon)\mathbb{E}(X)) < exp(-\frac{\varepsilon^2}{4}\mathbb{E}(X))$.

THEOREM 5.2. *Let $G=(\mathcal{V},\mathcal{E})$ be a graph with n vertices, with similarity and degree matrices W and D, respectively, $f\in\mathbb{R}^n$ and $P=D^{-1}W$. Let $\hat{\rho}_{t,f}$ be the approximation to the heat kernel pagerank $\rho_{t,f}$ computed using r random walks on the graph, each with at most K steps. For any small positive constants $0<\varepsilon,\delta<<1$, there exists a sufficiently large $r=r(\delta,\varepsilon)$, such that the following bound holds:*

$$||\hat{\rho}_{t,f}-\rho_{t,f}||_\infty \leq \varepsilon \tag{5.1}$$

with probability $>1-\delta$.

Proof. First, let variables ε_o and δ_o be derived from ε and δ as follows:

$$\varepsilon_o = \min(\frac{\varepsilon}{2||f||_1},\varepsilon), \qquad \delta_o = 1-\sqrt[n]{1-\delta}. \tag{5.2}$$

We start the proof for the special case when f is a probability distribution and extend this to the general case at the end of the proof. In the former case, the formula for heat kernel pagerank

$$\rho_{t,f} = fH_t = \sum_{k=0}^{\infty} e^{-t}\frac{t^k}{k!} fP^k \tag{5.3}$$

can be considered as the expected distribution of the random walk constructed using the probability matrix P, using the number of steps taken from the Poisson distribution and the starting point chosen from f. Moreover, we note that a large amount of steps are taken with a small probability, by Markov's inequality. In particular, the number of steps k taken by the random walk follows:

$$P(k > a) \leq t/a. \tag{5.4}$$

Therefore, we limit the number of steps of the random walk to K. We set K to be

$$K = \frac{\log(1/\varepsilon_o)}{\log(\log(1/\varepsilon_o))}. \tag{5.5}$$

Our algorithm for computing heat kernel pagerank consists of simulating r random walks, each with at most K steps. Let Z_k^u be an indicator random variable which equals to 1 if a random walk beginning from a vertex drawn from the probability distribution f ends at a vertex u and if it consists of k steps. The variable k is chosen for each random walk according to a Poisson distribution (with K as the upper bound). Also, let Z^u be the random variable that describes the random walk process that ends at vertex u in *at most* k steps; $Z^u = 1$ if a random walk has at most k steps and ends at vertex u, and 0 otherwise. Thus, $Z^u = Z_1^u + ... + Z_k^u$.

Let $\rho(k)_{t,f}$ be the contribution to the heat kernel pagerank vector $\rho_{t,f}$ of random walks of length at most k. By construction, the expectation of Z^u is $\rho(k)_{t,f}(u)$, or the entry of $\rho(k)_{t,f}$ corresponding of vertex u:

$$\mathbb{E}(Z^u) = \rho(k)_{t,f}(u). \tag{5.6}$$

Also, let the number r of random walks be chosen as

$$r = \frac{C}{\varepsilon_o^3}\log(1/\delta_o), \tag{5.7}$$

where $C > 0$ is a constant.

Let $\hat{\rho}_{t,f}$ be the approximation to heat kernel pagerank $\rho_{t,f}$ computed using r random walks, each with length chosen from the Poisson distribution and at most K. Let $Y_1^u, ..., Y_r^u$ be independent and identically distributed copies of Z^u when $k = K$. Let

$$Y^u = Y_1^u + ... + Y_r^u.$$

By construction,

$$\hat{\rho}_{t,f}(u) = \frac{Y^u}{r} = \frac{1}{r}(Y_1^u + ... + Y_r^u). \tag{5.8}$$

Note that if $\rho_{t,f}(u) > \varepsilon_o$, then $\rho(k)_{t,f}(u) > \varepsilon_o/2$, by definition. Then, in the case when $\rho_{t,f}(u) > \varepsilon_o$, using Theorem 5.1,

$$P(\frac{Y^u}{r} < (1-\varepsilon_o)\rho(k)_{t,f}(u)) = P(Y^u < (1-\varepsilon_o)r\rho(k)_{t,f}(u))$$

$$\begin{aligned}
&< \exp(-(\varepsilon_o^2/2) r p(k)_{t,f}(u)) \\
&< \exp(-(\varepsilon_o^2/2) \frac{C}{\varepsilon_o^3} \log(1/\delta_o)(\varepsilon_o/2)) \\
&< \exp(-(C/4)\log(1/\delta_o)) \\
&< \delta_o^{C/4} < \delta_o/4
\end{aligned} \tag{5.9}$$

for C large enough, since δ_o is a small positive number $<<1$. Similarly, using Theorem 5.1,

$$\begin{aligned}
P(\frac{Y^u}{r} > (1+\varepsilon_o)\rho(k)_{t,f}(u)) &= P(Y^u > (1+\varepsilon_o) r \rho(k)_{t,f}(u)) \\
&< \exp(-(\varepsilon_o^2/4) r p(k)_{t,f}(u)) \\
&< \exp(-(\varepsilon_o^2/4) \frac{C}{\varepsilon_o^3} \log(1/\delta_o)(\varepsilon_o/2)) \\
&< \exp(-(C/8)\log(1/\delta_o)) \\
&< \delta_o^{C/8} < \delta_o/4
\end{aligned} \tag{5.10}$$

for C large enough, since δ_o is a small positive number $<<1$.

Therefore, in the case that $\rho_{t,f}(u) > \varepsilon_o$,

$$P(|\frac{Y^u}{r} - \rho_{t,f}(u)| \le \varepsilon_o \rho_{t,f}(u)) > 1 - \delta_o/2. \tag{5.11}$$

In the case when $\rho_{t,f}(u) \le \varepsilon_o$, we see that $\rho(k)_{t,f}(u) \le \varepsilon_o$. By Chebyshev's inequality,

$$\begin{aligned}
P(|\frac{Y^u}{r} - \rho_{t,f}(u)| \ge \varepsilon_o) &\le \frac{\mathrm{var}(\frac{Y^u}{r})}{\varepsilon_o^2} \\
&\le \frac{\rho(k)_{t,f}(u) - \rho(k)_{t,f}(u)^2}{\varepsilon_o^2 r} \\
&\le \frac{\rho(k)_{t,f}(u)}{\varepsilon_o^2 r} \\
&\le \frac{\varepsilon_o^2}{C\log(1/\delta_o)} < \delta_o/2
\end{aligned} \tag{5.12}$$

for C large enough (in this case, $C > 4\varepsilon_o^2/(\delta_o \log(1/\delta_o))$ works).

Therefore, in the case that $\rho_{t,f}(u) \le \varepsilon_o$,

$$P(|\frac{Y^u}{r} - \rho_{t,f}(u)| \le \varepsilon_o) > 1 - \delta_o/2. \tag{5.13}$$

Consider the probability that a vertex u satisfies

$$|\hat{\rho}_{t,f}(u) - \rho_{t,f}(u)| \le \varepsilon_o. \tag{5.14}$$

This can be calculated from

$$\begin{aligned}
P(|\hat{\rho}_{t,f}(u) - \rho_{t,f}(u)| \le \varepsilon_o) = &P(\hat{\rho}_{t,f}(u) > \varepsilon_o) P(|\hat{\rho}_{t,f}(u) - \rho_{t,f}(u)| \le \varepsilon_o \,|\, P(\hat{\rho}_{t,f}(u) > \varepsilon_o)) \\
&+ P(\hat{\rho}_{t,f}(u) \le \varepsilon_o) P(|\hat{\rho}_{t,f}(u) - \rho_{t,f}(u)| \le \varepsilon_o \,|\, P(\hat{\rho}_{t,f}(u) \le \varepsilon_o)).
\end{aligned} \tag{5.15}$$

Note that if $|\hat{\rho}_{t,f}(u)-\rho_{t,f}(u)|\leq\varepsilon_o\rho_{t,f}(u)$, then we have that $|\hat{\rho}_{t,f}(u)-\rho_{t,f}(u)|\leq\varepsilon_o$, since $0\leq\rho_{t,f}(u)\leq 1$.

From (5.11), (5.13) and (5.15), we see that

$$\begin{aligned}P(|\hat{\rho}_{t,f}(u)-\rho_{t,f}(u)|\leq\varepsilon_o)&\geq P(\hat{\rho}_{t,f}(u)>\varepsilon_o)(1-\delta_o/2)+P(\hat{\rho}_{t,f}(u)\leq\varepsilon_o)(1-\delta_o/2)\\&\geq 1-\delta_o\end{aligned}\tag{5.16}$$

for C large enough.

Therefore, $P(|\hat{\rho}_{t,f}(u)-\rho_{t,f}(u)|\leq\varepsilon_o)\geq 1-\delta_o$ for a vertex u in the graph. It follows that $P(|\hat{\rho}_{t,f}(u)-\rho_{t,f}(u)|\leq\varepsilon_o\ \ \forall\ \ u\in\mathcal{V}\ )\geq(1-\delta_o)^n=1-\delta$. The full error bound, in the case f is a probability distribution, becomes

$$||\hat{\rho}_{t,f}-\rho_{t,f}||_\infty\leq\varepsilon_o<\varepsilon\tag{5.17}$$

with probability at least $1-\delta=(1-\delta_o)^n$.

In the case when f is not a probability distribution, the analysis above is valid for $f_+/||f||_1$ and $f_-/||f||_1$. Then, we have

$$\begin{aligned}||\hat{\rho}_{t,f}-\rho_{t,f}||_\infty&\leq||||f||_1\hat{\rho}_{t,f_+/||f||_1}+||f||_1\hat{\rho}_{t,f_-/||f||_1}-||f||_1\rho_{t,f_+/||f||_1}-||f||_1\rho_{t,f_-/||f||_1}||_\infty\\&\leq||f||_1||\hat{\rho}_{t,f_+/||f||_1}+\hat{\rho}_{t,f_-/||f||_1}-\rho_{t,f_+/||f||_1}-\rho_{t,f_-/||f||_1}||_\infty\\&\leq||f||_1||\hat{\rho}_{t,f_+/||f||_1}-\rho_{t,f_+/||f||_1}+\hat{\rho}_{t,f_-/||f||_1}-\rho_{t,f_-/||f||_1}||_\infty\\&\leq 2\varepsilon_o||f||_1\\&\leq\varepsilon\end{aligned}\tag{5.18}$$

with probability at least $1-\delta=(1-\delta_o)^n$, since $f=u(0)^{tr}D$, and $||u(0)||_\infty=1$. ∎

5.1.2. Approximation of heat kernel pagerank using series.

THEOREM 5.3. *Let $G=(\mathcal{V},\mathcal{E})$ be a graph with n vertices, with similarity and degree matrices W and D, $f\in\mathbb{R}^n$ and $P=D^{-1}W$. Let $\hat{\rho}_{t,f}$ be the approximation to heat kernel pagerank $\rho_{t,f}$ computed using the summation of the first s terms of the sum*

$$\rho_{t,f}=fH_t=\sum_{k=0}^{\infty}e^{-t}\frac{t^k}{k!}fP^k.\tag{5.19}$$

Then we have the following error estimate for $t\leq 1$ (which we use in our procedure):

$$||\hat{\rho}_{t,f}-\rho_{t,f}||_\infty=||\sum_{k=s}^{\infty}e^{-t}\frac{t^k}{k!}fP^k||\leq\frac{d_{max}}{s!},\tag{5.20}$$

where d_{max} is the maximal degree of the graph. If the graph is a sparse graph, where each node has at most p neighbors, then

$$||\hat{\rho_{t,f}}-\rho_{t,f}||_\infty=||\sum_{k=s}^{\infty}e^{-t}\frac{t^k}{k!}fP^k||\leq\frac{p}{s!}.\tag{5.21}$$

Proof. Consider the induced matrix norm

$$||P||_\infty=\max_{x\in R^n,||x||=1}||Px||_\infty.\tag{5.22}$$

Then we have that

$$||\hat{\rho}_{t,f}-\rho_{t,f}||_\infty=||\sum_{k=s}^{\infty}e^{-t}\frac{t^k}{k!}fP^k||_\infty$$

$$\leq \sum_{k=s}^{\infty} e^{-t}\frac{t^k}{k!}||f||_\infty ||P^k||_\infty$$
$$\leq \sum_{k=s}^{\infty} e^{-t}\frac{t^k}{k!}||f||_\infty ||P||_\infty^k$$
$$\leq ||f||_\infty e^{-t}\sum_{k=s}^{\infty}\frac{t^k}{k!} \tag{5.23}$$

since $||P||_\infty = 1$. Since $e^t = \sum_{k=0}^{\infty} t^k/k!$, the Taylor polynomial error bound gives

$$||\hat{\rho}_{t,f} - \rho_{t,f}||_\infty \leq ||f||_\infty e^{-t}\frac{1}{s!}t^s e^t$$
$$\leq ||f||_\infty \frac{1}{s!}t^s. \tag{5.24}$$

The variable t in the algorithm is the time step taken to propagate the equation

$$\dot{u} = -L_w x, \qquad u(0) \in R^n. \tag{5.25}$$

by timestep t. As mentioned earlier, the solution to the propagation of (5.25) is:

$$u(t) = D^{-1}\rho_{t,f}^{tr}, \qquad f = u(0)^{tr}D. \tag{5.26}$$

We can propagate by smaller timesteps by choosing $t < 1$, so

$$||\hat{\rho}_{t,f} - \rho_{t,f}||_\infty \leq \frac{||f||_\infty}{s!}. \tag{5.27}$$

Since $f = u(0)^{tr}D$ and $||u(0)||_\infty = 1$ according to the procedure of the algorithm,

$$||\hat{\rho}_{t,f} - \rho_{t,f}||_\infty \leq \frac{d_{max}}{s!}, \tag{5.28}$$

where d_{max} is the maximal degree of the graph. If the graph is sparse, where each node has at most p neighbors, then the second part of the theorem follows directly. ∎

THEOREM 5.4. *Let $G = (\mathcal{V}, \mathcal{E})$ be a graph with n vertices, with similarity and degree matrices W and D, respectively, $f \in \mathbb{R}^n$ and $P = D^{-1}W$. Let $\hat{\rho}_{t,f}$ be the approximation to heat kernel pagerank $\rho_{t,f}$ computed using the first s terms of* (5.19)*. Then, for every small positive constant $0 < \varepsilon << 1$, there exists an $R = R(\varepsilon)$ such that, for $s > R$,*

$$||\hat{\rho}_{t,f} - \rho_{t,f}||_\infty < \varepsilon. \tag{5.29}$$

Proof. The result follows for s that satisfies $s! > d_{max}/\varepsilon$. ∎

5.2. Time complexity analysis. To show the efficiency of our algorithms, we present detailed time complexity analysis, for both sparse and dense graphs. We show that the algorithms scale well with the number of nodes in the graph.

THEOREM 5.5. *Let $G = (\mathcal{V}, \mathcal{E})$ be a graph with n vertices, with similarity and degree matrices W and D, respectively, $f \in \mathbb{R}^n$ and $P = D^{-1}W$. Computing an approximation $\hat{\rho}_{t,f}$ to heat kernel pagerank $\rho_{t,f}$ using series involves $O(np)$ operations in the case of a sparse graph, where there are at most p non-zero entries in each row of W.*

Proof. We assume that the graph is sparse with each node in the data set connected to at most p neighbors. We also assume the weight matrix W is a symmetric matrix, and that, due to sparsity, at most p entries in each row of W are nonzero. To compute an approximation $\hat{\rho}_{t,f}$ to heat kernel pagerank $\rho_{t,f}$, we calculate s terms:

$$\hat{\rho}_{t,f} = \sum_{k=0}^{s-1} e^{-t} \frac{t^k}{k!} f P^k. \tag{5.30}$$

Note that

$$\hat{\rho}^T_{t,f} = \sum_{k=0}^{s-1} e^{-t} \frac{t^k}{k!} (W D^{-1})^k f^T. \tag{5.31}$$

since $P = D^{-1}W$. Since W has at most p nonzero entries in each row, it takes $O(np)$ operations to multiply WD^{-1} and an $n \times 1$ vector. Since we compute only the first few terms of the series, it takes $O(n)$ operations to perform the scalar multiplication by $e^{-t}\frac{t^k}{k!}$, as well as to perform the addition. Therefore, computing the approximation to heat kernel pagerank involves $O(np)$ computations.

We end the proof by noting that, in the case when the graph is not sparse, the computation is dominated by the matrix multiplication of WD^{-1} by an $n \times 1$ vector, which takes $O(n^2)$ operations. Thus, the heat kernel pagerank computation consists of $O(n^2)$ calculations in the non-sparse case. ∎

THEOREM 5.6. *Let $G = (\mathcal{V}, \mathcal{E})$ be a graph with n vertices, with similarity and degree matrices W and D, respectively, $f \in \mathbb{R}^n$ and $P = D^{-1}W$. If the number of neighbors of each vertex is bounded by p, where $p << n$, computing an approximation $\hat{\rho}_{t,f}$ to heat kernel pagerank $\rho_{t,f}$ using random walks involves $O(r(n+K))$ operations in the worst case, where r is the number of random walks, and K is the maximum amount of steps taken by each walk. In the non-sparse case, there are $O(rnK)$ operations in the worst case.*

Proof. First, we assume that the graph is sparse with each node in the data set connected to at most p neighbors, where $p << n$. We also assume the weight matrix W is a symmetric matrix, and that, due to sparsity, at most p entries in each row of W are nonzero. Choosing a starting vertex from a probability distribution of n elements takes, in the worst case scenario, $O(n)$ operations. We limit the number of random walk steps to K since long walks are performed with very small probability (because the number of steps is a Poisson random variable). After choosing the initial vertex, the next random walk step is more quickly performed, since the distribution for the subsequent steps occurs with bounded support. In particular, once the initial vertex is chosen, the distribution for the next step has at most p nonzero entries. We also assume that drawing from a distribution of bounded support requires constant time. Therefore, since there are r random walks to be performed, and this calculation dominates the computation, there are $O(r(n+K))$ computations in the worst case.

We end the proof by noting that, in the non-sparse case, choosing an initial vertex of a random talk takes $O(n)$ operations in the worst case. Choosing the subsequent vertex still takes $O(n)$ operations in the worst case as the graph is not sparse. Since there are at most K steps in each of the r random walks, the procedure takes $O(rnK)$ operations in the worst case in the non-sparse scenario. ∎

THEOREM 5.7. *Let $G = (\mathcal{V}, \mathcal{E})$ be a graph with n vertices, with similarity and degree matrices W and D, respectively, $f \in \mathbb{R}^n$ and $P = D^{-1}W$. In the sparse case, the time*

complexity of the Heat Kernel Pagerank MBO Algorithm is linear in the number of nodes, if the approximation to heat kernel pagerank is computed using the series, for a fixed number of classes m and a set scalar p, where each node in the graph is connected to at most p neighbors.

Proof. We see that the *for loop* in the Heat Kernel Pagerank MBO algorithm involves $O(np)$ operations to approximate heat kernel pagerank using series and $O(n)$ operations for the scaling by the degree matrix. The *for loop* is executed m times. The propagation step requires $O(nm)$ operations and the projection step requires $O(nm\log m)$ computations. The thresholding step involves $O(nm)$ calculations. Therefore, for fixed m and p, this algorithm is linear in complexity.

In the case the graph is *not sparse*, Theorem 5.5 shows that the procedure consists of $O(n^2m)$ computations. ∎

THEOREM 5.8. *Let $G=(\mathcal{V},\mathcal{E})$ be a graph with n vertices, with similarity and degree matrices W and D, respectively, $f\in\mathbb{R}^n$ and $P=D^{-1}W$. If the number of neighbors of each vertex is bounded by a number $<<n$, the time complexity of the Heat Kernel Pagerank MBO Algorithm is $O(r(n+K))$ in the worst case, for a fixed amount of classes, if the approximation to heat kernel pagerank is computed using random walks. Here, r is the number of random walks to be computed and K is the maximum number of steps in each random walk. In the non-sparse scenario, the method consists of $O(rnK)$ calculations in the worst case.*

Proof. The *for loop* in the algorithm involves $O(r(n+K))$ operations in the worst case to approximate the heat kernel pagerank via the random walk approach and $O(n)$ operations for the scaling by the degree matrix. The *for loop* is executed m times. The propagation step requires $O(mn)$ operations and the projection step requires $O(nm\log m)$ computations. The thresholding step involves $O(mn)$ calculations. Therefore, for a fixed number of classes m, the algorithm requires $O(r(n+K)$ calculations in the worst case. In the case the graph is *not sparse*, Theorem 5.6 shows that the procedure consists of $O(rnK)$ operations in the worst case. ∎

THEOREM 5.9. *Let $G=(\mathcal{V},\mathcal{E})$ be a graph with n vertices, with similarity and degree matrices W and D, respectively, $f\in\mathbb{R}^n$ and $P=D^{-1}W$. If the number of neighbors of each vertex is bounded by a number $<<n$, for a fixed number of classes m, the time complexity of the HKPR Algorithm from Section 3 is $O(n)$ if the pagerank is computed using series, and $O(r(n+K))$ in the worst case, if the pagerank is computed using random walks. Here, r is the number of random walks to be computed and K is the maximum number of steps in each random walk.*

Proof. Let the number of classes m be fixed. The first two steps take $O(n)$ computations. Moreover, for the third step, we see that $O(n)$ operations are involved to approximate heat kernel pagerank with series, while $O(r(n+K))$ operations are needed in the worst case to approximate the heat kernel pagerank via the random walk approach. The thresholding step involves $O(n)$ calculations. The result follows directly. As a side note, in the case of a dense graph, the HKPR Algorithm takes $O(rnK)$ operations in the worst case, if the pagerank is calculated using random walks. ∎

REMARK 5.1. Note that each of the versions is more advantageously applied in different scenarios. In the case of a sparse graph, our computations show that computing the pagerank with truncated series is faster. However, in the case of a large dense graph, it is better to proceed with the random walk version of the algorithm, which requires only $O(rnK)$ computations.

6. Experimental results

MNIST (10 classes- 3.57% labeled points)

Method	Accuracy
boosted stumps* [43, 49]	**92.3-98.74%**
transductive classification [69]	92.6%
tree GL [30]	93.0%
k-nearest neighbors* [48, 49]	95.0-97.17%
neural/conv. nets* [26, 48, 49]	**95.3-99.65%**
nonlinear classifiers* [48, 49]	96.4-96.7%
GL [31] (3.57% labeled pts.)	96.8%
MBO [31] (3.57% labeled pts.)	96.91%
HKPR (Section 3)	96.43%
HKPR1 MBO	**97.52%**
HKPR2 MBO	97.48%

- Note that some of the comparable algorithms, marked by *, use substantially more data for training (85.7% at the most and the 21.4% at smallest) than the proposed algorithm, see the main text for more information.

COIL (10% labeled points)

Method	Accuracy
k-nearest neighbors [67]	83.5%
LapRLS [3, 67]	87.8%
sGT [42, 67]	89.9%
SQ-Loss-I [67]	90.9%
MP [67]	91.1%
GL [31]	91.2%
MBO [31]	91.46%
HKPR (Section 3)	**92.12%**
HKPR1 MBO	91.09%
HKPR2 MBO	91.23%

MNIST (2 classes - 3.57% labeled points)

Method	Accuracy
GL [31]	98.37%
MBO [31]	98.29%
HKPR (Section 3)	94.94%
HKPR1 MBO	98.35%
HKPR2 MBO	**98.43%**

Two Moons (2.5% labeled points)

Method	Accuracy
spectral clustering [30]	80%
p-Laplacian [11]	94%
Cheeger cuts [68]	95.4%
HKPR (Section 3)	95.73%
HKPR1 MBO	**96.26%**
HKPR2 MBO	96.02%

LFR1

% of points belonging to fidelity / Mixing parameter	4%	8%	12%	16%	20%
0.1	99.38%	100%	100%	100%	100%
0.2	97.62%	100%	100%	100%	100%
0.3	86.20%	99.08%	100%	100%	100%
0.4	64.13%	99.08%	99.96%	100%	100%
0.5	40.50%	80.70%	98.36%	99.54%	99.95%
0.6	29.36%	58.34%	77.84%	85.77%	91.70%
0.7	21.14%	35.08%	45.92%	56.52%	65.10%
0.8	13.6%	23.21%	29.80%	37.64%	43.00%

LFR50

% of points belonging to fidelity / Mixing parameter	4%	8%	12%	16%	20%
0.1	100%	100%	100%	100%	100%
0.2	100%	100%	100%	100%	100%
0.3	99.95%	99.99%	99.99%	99.99%	99.99%
0.4	96.87%	99.31%	99.55%	99.70%	99.81%
0.5	70.38%	93.57%	97.83%	99.10%	99.37%
0.6	39.84%	60.14%	75.80%	81.99%	89.71%
0.7	26.26%	44.75%	53.69%	60.20%	65.56%
0.8	16.68%	28.14%	35.58%	41.76%	46.63%

Table 6.1: *Results for benchmark data sets: Moons, MNIST, LFR and COIL.*

In this section, we validate our method by presenting results of experiments on benchmark data sets. The accuracy of our algorithm is better than or comparable with that of the state-of-the-art methods. The algorithm still performs very well in the case

of a very low number of labeled points! Even when heat kernel pagerank is used directly as a classifier (HKPR method), the performance is still good. For the experiments, we use $t=0.1$, $max=12$, $tol=10^{-9}$, and calculate r and K using (5.5) and (5.7). The accuracy results of the proposed methods are shown in Tables 6.2 and 6.1, where the algorithms proposed in this paper are written in italics. The best results are written in bold. We also provide those of other methods for comparison.

6.1. Results for a small number of labeled points. Our algorithm performed very well even when the number of labeled points was very small. For example, as shown in Table 6.2, for the full MNIST data set, even when only 0.43% points were associated with a known class, the algorithm was still able to classify around 94.46% of the nodes correctly. When 1.32% points were associated with a known class, the accuracy reached around 97.44%, a remarkable achievement for such a low fidelity percentage. In the case of the Two Moons data set, the algorithm was able to segment around 91.53% points correctly when only 0.5% of the nodes were labeled. For the 0.9% fidelity set, still very small, the algorithm obtained an accuracy of 94.25%. In the case of 2.5% labeled points, the accuracy improved to around 96.26%.

Therefore, one advantage of our method is that it still performs very well when only a small number of nodes are labeled, a very realistic scenario!

MNIST (10 classes) Accuracy

% of labeled points	1.32%	1.29%	1.14%	1.00%	0.86%	0.71%	0.57%	0.5%	0.43%
HKPR1	97.44%	97.41%	97.38%	97.33%	97.29%	97.05%	96.81%	96.17%	94.46%
HKPR2	97.41%	97.40%	97.35%	97.32%	97.24%	97.01%	96.81%	96.15%	94.44%

Two Moons Accuracy

% of labeled points	2.5%	2.2%	1.9%	1.6%	1.3%	1.1%	0.9%	0.7%	0.5%
HKPR1	96.26%	95.85%	95.64%	95.48%	95.07%	94.78%	94.25%	93.15%	91.53%
HKPR2	96.02%	95.73%	95.56%	95.42%	95.02%	94.78%	94.20%	93.13%	91.49%

Table 6.2: *Accuracy results for a very small number of labeled points.*

6.2. Two moons. This data set is constructed from two half circles in $\mathbb{R}^2$ with a radius of one, with centers at (0,0) and (1,0.5). A thousand uniformly chosen points are sampled from each circle, embedded in $\mathbb{R}^{100}$ and i.d.d. Gaussian noise with standard deviation 0.02 is added to each coordinate. Therefore, the set consists of two thousand points. The graph is constructed using the 100-dimensional points as feature vectors, and is sparsified based on 10 nearest neighbors with local scaling using the 10^{th} closest neighbor.

We performed 100 experiments, each with a different fidelity set. For HKPR1 and HKPR2, for 2.5% fidelity, the average accuracy was 96.26% and 96.02%, respectively. The timing was around a second and 17 seconds, respectively. The average number of iterations was around 3.29 and 8.29, respectively. When heat kernel pagerank is used directly as a classifier (HKPR algorithm), the average accuracy was around 95.73%.

6.3. MNIST. The MNIST data set [49] is composed of 70,000 28×28 images of handwritten digits 0 through 9. The task is to correctly assign each image to a digit; this is a 10 class segmentation problem. We used feature vecctors consisting of 784 pixel intensity values present in each image. To construct the weight matrix, we use a graph sparsified using 8 nearest neighbors with local scaling based on the 8^{th} closest neighbor. Note that we do not perform any preprocessing, i.e. the graph is constructed from the raw data directly.

First, we tested the full MNIST dataset, consisting of 70,000 images of 10 digits, performing 100 experiments over different fidelity sets. For HKPR1 and HKPR2, for 3.57% fidelity, the average accuracy was 97.52% and 97.48%, respectively. The average number of iterations was 8.69 and 16, respectively. The timing was around 22.25 seconds for the first version. Using heat kernel pagerank directly as a classifier (HKPR method) gives an accuracy of around 96.43%.

We also tested a binary subset of MNIST, consisting of only digits 4 and 9, resulting in a data set of 13,782 nodes. This data set was chosen because these digits are hard to distinguish when handwritten. The graph and the fidelity set was constructed in the same way as for the 70K MNIST data set. We performed experiments with 100 fidelity sets. For HKPR1 and HKPR2, for 3.57% fidelity, the average accuracy was 98.35% and 98.43%, respectively. The timing was around a second and around 8 minutes, respectively. The average number of iterations was around 14.78 and 34, respectively. The HKPR method's average accuracy was around 94.94%.

From Table 6.2, one can see that our method performs very competitively with these procedures, and in most cases outperforms them, even with a small number of labeled points. Moreover, we note that we do not process the data, unlike some supervised approaches we compare with.

6.4. LFR network. In their paper [47], the authors introduce a class of graphs that are built with heterogeneous distributions of class size and node degree, since many real networks have this property. The degrees and class sizes are assigned from a power-law distribution with power ξ and β, respectively. In addition, this class of graphs has a mixing parameter μ associated to them, which is the fraction of edges each node shares with other classes. The maximum degree d_{max}, mean degree d_{av}, largest class size s_{max}, smallest class size s_{min}, μ, ξ, β are parameters that are entered by the user. The code is found in [47].

To construct the $LFR1$ network, each with 1000 nodes, we set the parameters d_{max}, d_{av}, s_{max}, s_{min}, ξ, β to 50, 20, 50, 10, 2, 1, respectively. We vary the μ parameter from 0.1 to 0.8 in 0.1 increments. A higher μ makes it more difficult to classify each node. The number of classes in each $LFR1$ network was about 40.

To construct the $LFR50$ network, each with 50,000 nodes, we first consider 50 LFR1 networks $L_1,...L_{50}$, each with a mixing parameter μ, and then connect each node in L_s to 20μ nodes in L_{s+1}, chosen uniformly at random, and $L_{51}=L_1$. Since each $LFR1$ network contains around 40 classes, each $LFR50$ graph consists of around 2000 classes. As in the $LFR1$ case, we vary the mixing parameter μ of the $LFR1$ networks used to construct $LFR50$, from 0.1 to 0.8 in 0.1 increments. The mixing parameter of the resulting network is approximately $\frac{2\mu}{1+\mu}$. In addition, because of the way the network is constructed, the graph has very similar structure to an $LFR1$ graph. Moreover, the strength of each node in the network is around $1+2\mu$ times that of it in the $LFR1$ network, and the total number of edges in the network is scaled by around $50(1+2\mu)$.

The accuracy for this data set depends on the mixing parameter and the size of the fidelity set. For HKPR1, the procedure took around 1 second and 19 minutes, respectively, for $LFR1$ and $LFR50$.

6.5. COIL. The COIL data set [16,60] from the Columbia University Image Library is a set of color 128×128 images of objects, taken at different angles. There are six classes and 1500 images. Discarding 38 images from each class leaves 250 per class. To construct the weight matrix, we used 4 nearest neighbors with local scaling based on the 4^{th} closest neighbor. We conduct experiments for over 100 different fidelity sets. For HKPR1 and HKPR2, for 10% fidelity, we obtained an average accuracy of 91.09% and

91.23%, respectively. The average number of iterations was 8.26 and 16, respectively. The timing was around 1 second and 92 seconds on average, respectively. When heat kernel pagerank is used directly as a classifier (HKPR), the average accuracy is 92.12%.

6.6. Timing. Details about the timing of the heat kernel pagerank MBO method are displayed in Table 6.3; we see that the algorithm is very efficient. The timing is also very comparable to (or better than) that of the state-of-the-art. To name some comparison, for the MNIST data set, the training for neural or convolution nets can take 2-3 days [26, 48, 49] due to the large size of the training set! A recent algorithm in [30] lists its timing as 132 seconds, several times slower than our proposed method. In the case of the two moons data set, [30] reports competitive times in the range of 0.5 to 50 seconds. In particular, [68] notes its time for the two moons data set as 5.6 seconds.

We also emphasize one advantage of the heat kernel pagerank MBO algorithm over graphical methods, like [31], that involve computing the eigenvectors and eigenvalues of the graph Laplacian (or any other large matrix), which is not part of the procedure for our proposed method. Computing the eigendecomposition can be computationally expensive; for example, Table 6.3 shows that the calculation for the MNIST data set, which consists of 70,000 nodes, takes around 1700 seconds using the Raleigh-Chebyshev procedure [1]. Our algorithm avoids this computation.

Timing of Heat Kernel Pagerank MBO Method (in seconds)

data set/method	2 moons	COIL	MNIST (2 digits)	MNIST (10 digits)
heat kernel pagerank MBO method [a]	1.708	1.461	1.374	24.257

[a]This is the timing of the method using pre-computed weights of the graph.

Method in [30]: Timing for the Computation of Eigenvalues and Eigenvectors of the graph Laplacian (in seconds)

data set/method	2 moons	COIL	MNIST (2 digits)	MNIST (10 digits)
computation of eigenvectors and eigenvalues in [31] [a]	0.55	0.19	45.43	1683.57

[a]Eigenvectors and eigenvalues are computed using the Rayleigh-Chebyshev procedure [1].

Table 6.3: *Timing*

7. Conclusion

In summary, we have introduced an accurate, efficient and simple graph-based semi-supervised algorithm for classification, called heat kernel pagerank MBO, with results that are competitive with or better than those of the state-of-the-art procedures, and accuracy that is high even in the case of a small number of labeled points. This method, motivated by recent threshold dynamics approaches, introduces the usage of heat kernel pagerank for its main steps. In addition to its aforementioned advantages, it avoids potentially computationally expensive linear algebra routines. The computational complexity is linear in the number of nodes for sparse graphs; the algorithm requires $O(rnK)$ operations in the worst case for a dense graph, for a fixed number of classes m, where n is the size of the data set, and r and K are variables that describe the number of and the maximum length of random walks, respectively.

Moreover, in this paper, we have also outlined a procedure that uses heat kernel pagerank directly as a classifier for semi-supervised learning. Overall, the new ideas

form a powerful approach to the classification problem.

Acknowledgements. We would like to thank Zach Boyd for helpful discussions.

REFERENCES

[1] C. Anderson, *A Rayleigh-Chebyshev procedure for finding the smallest eigenvalues and associated eigenvectors of large sparse Hermitian matrices*, J. Comput. Phys., 229:7477–7487, 2010.
[2] E. Bae and E. Merkurjev, *Convex variational methods on graphs for multiclass segmentation of high-dimensional data and point clouds*, Journal of Mathematical Imaging and Vision, 58(3):468–493, 2017.
[3] M. Belkin, P. Niyogi, and V. Sindhwani, *Manifold regularization: A geometric framework for learning from labeled and unlabeled examples*, J. Mach. Learn. Res., 7:2399–2434, 2006.
[4] A.L. Bertozzi and A. Flenner, *Diffuse interface models on graphs for classification of high dimensional data*, Multiscale Modeling and Simulations, 10(3):1090–1118, 2012.
[5] A.L. Bertozzi and Y. van Gennip, *Gamma-convergence of graph Ginzburg-Landau functionals*, Adv. Diff. Eqs., 17(11–12):1115–1180, 2012.
[6] Z. Boyd, E. Bae, X.-C. Tai, and A.L. Bertozzi, *Simplified energy landscape for modularity using total variation*, SIAM J. Appl. Math., 78(5):2439—2464, 2018.
[7] Y. Boykov, O. Veksler, and R. Zabih, *Fast approximate energy minimization via graph cuts*, IEEE Trans. Pattern Anal. Mach. Intell., 23(11):1222–1239, 2001.
[8] X. Bresson, T. Laurent, D. Uminsky, and J.H. von Brecht, *Convergence and energy landscape for Cheeger cut clustering*, Adv. Neural Inf. Process. Syst., 25:1394–1402, 2012.
[9] X. Bresson, X.-C. Tai, T.F. Chan, and A. Szlam, *Multi-class transductive learning based on ℓ^1 relaxations of Cheeger cut and Mumford-Shah-Potts model*, J. Math. Imaging Vision, 49(1):191–201, 2014.
[10] A. Buades, B. Coll, and J.-M. Morel, *A review of image denoising algorithms, with a new one*, Multiscale Model. Simul., 4(2):490–530, 2005.
[11] T. Bühler and M. Hein, *Spectral clustering based on the graph p-Laplacian*, in Proceedings of the 26th Annual International Conference on Machine Learning, 81–88, 2009.
[12] T.F. Chan, G.H. Golub, and P. Mulet, *A nonlinear primal-dual method for total variation-based image restoration*, SIAM J. Sci. Comput., 20(6):1964–1977, 1999.
[13] T.F. Chan, A. Marquina, and P. Mulet, *High-order total variation-based image restoration*, SIAM J. Sci. Comput., 22(2):503–516, 2000.
[14] T.F. Chan and C.-K. Wong, *Total variation blind deconvolution*, IEEE Trans. Image Process., 7(3):370–375, 1998.
[15] H. Chang, X. Zhang, X.-C. Tai, and D. Yang, *Domain decomposition methods for nonlocal total variation image restoration*, J. Sci. Comput., 60(1):79–100, 2014.
[16] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*, MIT Press, 2, 2006.
[17] J. Cheeger, *A lower bound for the smallest eigenvalue of the Laplacian*, Princeton Univ. Press, Princeton, N. J., 195–199, 1970.
[18] Y. Chen and X. Ye, *Projection onto a simplex*, arXiv:1101.6081, 2011.
[19] F. Chung, *Spectral Graph Theory*, American Mathematical Society, 92, 1997.
[20] F. Chung, *The heat kernel as the pagerank of a graph*, Proc. Nat. Acad. Sci., 104(50):19735–19740, 2007.
[21] F. Chung, *A local graph partitioning algorithm using heat kernel pagerank*, Internet Math., 6(3):315–330, 2009.
[22] F. Chung and O. Simpson, *Solving linear systems with boundary conditions using heat kernel pagerank*, in Algorithms and Models for the Web Graph, Springer, 2:203–219, 2013.
[23] F. Chung and O. Simpson, *Computing heat kernel pagerank and a local clustering algorithm*, in Combinatorial Algorithms, Springer, 110–121, 2014.
[24] F. Chung and O. Simpson, *Distributed algorithms for finding local clusters using heat kernel pagerank*, in Algorithms and Models for the Web Graph, Springer, 177–189, 2015.
[25] F. Chung and A. Tsiatas, *Finding and visualizing graph clusters using pagerank optimization*, Internet Mathematics, 8(1-2):46–72, 2012.
[26] D.C. Cireşan, U. Meier, J. Masci, L.M. Gambardella, and J. Schmidhuber, *Flexible, high performance convolutional neural networks for image classification*, in Proceedings of the 22nd International Joint Conference on Artificial Intelligence, 1237–1242, 2011.
[27] C. Couprie, L. Grady, L. Najman, and H. Talbot, *Power watershed: A unifying graph-based optimization framework*, IEEE Trans. Pattern Anal. Mach. Intell., 33(7):1384–1399, 2011.

[28] C. Couprie, L. Grady, H. Talbot, and L. Najman, *Combinatorial continuous maximum flow*, SIAM J. Imaging Sci., 4(3):905–930, 2011.
[29] A. Elmoataz, O. Lezoray, and S. Bougleux, *Nonlocal discrete regularization on weighted graphs: a framework for image and manifold processing*, IEEE Trans. Image Process, 17(7):1047–1060, 2008.
[30] C. Garcia-Cardona, A. Flenner, and A.G. Percus, *Multiclass diffuse interface models for semi-supervised learning on graphs*, in Proceedings of the 2th International Conference on Pattern Recognition Applications and Methods, 2013.
[31] C. Garcia-Cardona, E. Merkurjev, A.L. Bertozzi, A. Flenner, and A. Percus, *Fast multiclass segmentation using diffuse interface methods on graphs*, IEEE Trans. Pattern Anal. Mach. Intell., 36(8):1600–1613, 2014.
[32] T. Gerhart, J. Sunu, L. Lieu, E. Merkurjev, J.-M. Chang, J. Gilles, and A.L. Bertozzi, *Detection and tracking of gas plumes in LWIR hyperspectral video sequence data*, in SPIE Conference on Defense Security and Sensing, 87430, 2013.
[33] G. Gilboa and S. Osher, *Nonlocal operators with applications to image processing*, Multiscale Model. Simul., 7(3):1005–1028, 2008.
[34] D.F. Gleich, *Pagerank beyond the web*, SIAM Review, 57(3):321–363, 2015.
[35] L. Grady, *Multilabel random walker image segmentation using prior models*, in Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 763–770, 2005.
[36] L. Grady, *Random walks for image segmentation*, IEEE Trans. Pattern Anal. Mach. Intell., 28(11):1768–1783, 2006.
[37] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann, *Random walks for interactive alpha-matting*, in Proceedings of VIIP, 423–429, 2005.
[38] M. Hein and T. Bühler, *An inverse power method for nonlinear eigenproblems with applications in 1-spectral clustering and sparse PCA*, Adv. Neural Inf. Process. Syst., 23:847–855, 2010.
[39] H. Hu, T. Laurent, M. Porter, and A.L. Bertozzi, *A method based on total variation for network modularity optimization using the MBO scheme*, SIAM J. Appl. Math., 73(6):2224–2246, 2013.
[40] H. Hu, J. Sunu, and A.L. Bertozzi, *Multi-class graph Mumford-Shah model for plume detection using the MBO scheme*, in Energy Minimization Methods in Computer Vision and Pattern Recognition, 209–222, 2015.
[41] M. Jacobs, E. Merkurjev, and S. Esedoḡlu, *Auction dynamics: A volume constrained mbo scheme*, J. Comput. Phys., 354(1):288–310, 2018.
[42] T. Joachims, *Transductive learning via spectral graph partitioning*, in International Conference on Machine Learning, 20:290–297, 2003.
[43] B. Kégl and R. Busa-Fekete, *Boosting products of base classifiers*, in Proceedings of the 26th Annual International Conference on Machine Learning, 497–504, 2009.
[44] D.J. Ketchen and C.L. Shook, *The application of cluster analysis in strategic management research: an analysis and critique*, Strategic Management Journal, 17(6):441–458, 1996.
[45] R.V. Kohn and P. Sternberg, *Local minimizers and singular perturbations*, Proc. Roy. Soc. Edinburgh Sect. A, 111(1-2):69–84, 1989.
[46] D. Koschützki, K.A. Lehmann, L. Peeters, S. Richter, D. Tenfelde-Podehl, and O. Zlotowski, *Centrality indices*, in Network Analysis, Springer, 16–61, 2005.
[47] A. Lancichinetti, S. Fortunato, and F. Radicchi, *Benchmark graphs for testing community detection algorithms*, Phys. Rev. E, 78(4):046110, 2008.
[48] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proc. of IEEE, 86(11):2278–2324, 1998.
[49] Y. LeCun and C. Cortes, *The MNIST database of handwritten digits*, The Courant Institute of Mathematical Sciences, 1–10, 1998.
[50] A. Levin, A. Rav-Acha, and D. Lischinski, *Spectral matting*, IEEE Trans. Pattern Anal. Mach. Intell., 30(10):1699–1712, 2008.
[51] F. Lin and W.W. Cohen, *Semi-Supervised Classification of Network Data Using Very Few Labels*, in 2010 International Conference on Advances in Social Networks Analysis and Mining, 192–199, 2010.
[52] Z. Meng, E. Merkurjev, A. Koniges, and A.L. Bertozzi, *Hyperspectral image classification using graph clustering methods*, Image Processing Online (IPOL), 7:218–245, 2017.
[53] E. Merkurjev, E. Bae, A.L. Bertozzi, and X.-C. Tai, *Global binary optimization on graphs for classification of high-dimensional data*, J. Math Imaging Vis., 52(3):414–435, 2015.
[54] E. Merkurjev, A.L. Bertozzi, K. Lerman, and X. Yan, *Modified Cheeger and Ratio cut methods using the Ginzburg-Landau functional for classification of high-dimensional data*, Inverse Problems, 33(7):074003, 2017.

[55] E. Merkurjev, C. Garcia-Cardona, A.L. Bertozzi, A. Flenner, and A.G. Percus, *Diffuse interface methods for multiclass segmentation of high-dimensional data*, Appl. Math. Lett., 33:29–34, 2014.

[56] E. Merkurjev, T. Kostic, and A.L. Bertozzi, *An MBO scheme on graphs for classification and image processing*, SIAM J. Imaging Sci., 6(4):1903–1930, 2013.

[57] E. Merkurjev, J. Sunu, and A.L. Bertozzi, *Graph MBO method for multiclass segmentation of hyperspectral stand-off detection video*, in IEEE International Conference on Image Processing, 689–693, 2014.

[58] B. Merriman, J.K. Bence, and S. Osher, *Diffusion generated motion by mean curvature*, AMS Selected Lectures in Mathematics Series: Computational Crystal Growers Workshop, 8966:73–83, 1992.

[59] B. Mohar, *The Laplacian spectrum of graphs*, Graph Theory, Combinatorics, and Applications, 2:871–898, 1991.

[60] S.A. Nene, S.K. Nayar, and H. Murase, *Columbia Object Image Library (COIL-100)*, Technical Report CUCS-006-96, 1996.

[61] L. Page, S. Brin, R. Motwani, and T. Winograd, *The pagerank citation ranking: bringing order to the web*, 1998.

[62] Dan Pelleg, Andrew W Moore, et al., *X-means: Extending k-means with efficient estimation of the number of clusters*, in ICML, 1, 2000.

[63] P. Perona and L. Zelnik-Manor, *Self-tuning spectral clustering*, Adv. Neural Inf. Process. Syst., 17:1601–1608, 2004.

[64] L.I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Phys. D, 60(1):259–268, 1992.

[65] J. Shi and J. Malik, *Normalized cuts and image segmentation*, IEEE Trans. Pattern Anal. Mach. Intell., 22(8):888–905, 2000.

[66] O. Simpson and F. Chung, *Finding consensus in multi-agent networks using heat kernel pagerank*, eprint arXiv:1507.08968, 2015.

[67] A. Subramanya and J. Bilmes, *Semi-supervised learning with measure propagation*, J. Mach. Learn. Res., 12:3311–3370, 2011.

[68] A. Szlam and X. Bresson, *A total variation-based graph clustering algorithm for Cheeger ratio cuts*, in Proceedings of the 27th International Conference on Machine Learning, 1039–1046, 2010.

[69] A. Szlam, M. Maggioni, and R.R. Coifman, *Regularization on graphs with function-adapted diffusion processes*, J. Mach. Learn. Res., 9:1711–1739, 2008.

[70] S.A. Tabrizi, A. Shakery, M. Asadpour, M. Abbasi, and M.A. Tavallaie, *Personalized pagerank clustering: a graph clustering algorithm based on random walks*, Physica A: Statistical Mechanics and its Applications, 392(22):5772–5785, 2013.

[71] U. Von Luxburg, *A tutorial on spectral clustering*, Statistics and Computing, 17(4):395–416, 2007.

[72] J. Wang, T. Jebara, and S.F. Chang, *Graph transduction via alternating minimization*, in Proceedings of the 25th International Conference on Machine Learning, 1144–1151, 2008.

[73] X. Zhang and T.F. Chan, *Wavelet inpainting by nonlocal total variation*, Inverse Probl. Imaging, 4(1):191–210, 2010.

[74] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Schölkopf, *Learning with local and global consistency*, Adv. Neural Inf. Process. Syst., 16:321–328, 2004.

[75] D. Zhou and B. Schölkopf, *A regularization framework for learning from graph data*, International Conference on Machine Learning, Banff, Canada, 2004.

[76] X. Zhu, *Semi-supervised learning literature survey*, Technical Report 1530, University of Wisconsin-Madison, 2005.