Using Local Spectral Methods to Robustify Graph-Based Learning Algorithms

David F. Gleich
Computer Science Department
Purdue University
West Lafayette, IN
dgleich@purdue.edu

Michael W. Mahoney
ICSI and Dept. of Statistics
UC Berkeley
Berkeley, CA
mmahoney@stat.berkeley.edu

ABSTRACT

Graph-based learning methods have a variety of names including semi-supervised and transductive learning. They typically use a diffusion to propagate labels from a small set of nodes with known class labels to the remaining nodes of the graph. While popular, these algorithms, when implemented in a straightforward fashion, are extremely sensitive to the details of the graph construction. Here, we provide four procedures to help make them more robust: recognizing implicit regularization in the diffusion, using a scalable push method to evaluate the diffusion, using rank-based rounding, and densifying the graph through a matrix polynomial. We study robustness with respect to the details of graph constructions, errors in node labeling, degree variability, and a variety of other real-world heterogeneities, studying these methods through a precise relationship with mincut problems. For instance, the densification strategy explicitly adds new weighted edges to a sparse graph. We find that this simple densification creates a graph where multiple diffusion methods are robust to several types of errors. This is demonstrated by a study with predicting product categories from an Amazon co-purchasing network.

Categories and Subject Descriptors

G.2.2 [**Discrete mathematics**]: Graph theory—*Graph algorithms*

1. INTRODUCTION

Graph-based data analysis and machine learning tools are common, and they typically involve a two-step process: first, construct in some way a graph from the data; and second, run a graph data mining algorithm on that graph. While most algorithmic and statistical research focuses on the latter step, in most downstream applications the initial construction step is the bottleneck to obtaining insight from the data. In particular, an important, yet underappreciated, aspect of this process is the effect of "noise" or "perturbations" or "arbitrary decisions" in the construction of the graph on the output of the subsequent graph algorithm. For example, data graphs are typically constructed by making plausible but ultimately somewhat arbitrary design decisions, *e.g.*, the exact form of the interaction kernel; the exact means of sparsifying the graph; the limits of the computational platform to handle graphs, matrices, and weights. It is clearly of interest to understand how robust are the outputs of graph mining algorithms to the details of these design or data modeling decisions.

This paper links the impact of these two related activities: the graph creation process (and implicitly the data fidelity), and the statistical properties of algorithmic procedures run on that graph. We provide a set of computational evidence that suggests ways to make these processes robust to a variety of issues with data creation, and we show results in improved performance in downstream applications. Our codes are available to reproduce this research: `https://www.cs.purdue.edu/homes/dgleich/codes/robust-diffusions`. We begin by reviewing relevant results from these two areas.

1.1 Graph constructions

There are potentially many things that fall under this general heading. Here are perhaps the three most common and the three that most motivated this work.

Problems with edges/nonedges in explicit graphs. By an explicit graph, we mean one that is given to the data analyst to work with, where the graph was constructed by someone else via a process that is, perhaps, not-fully-described. A canonical example of this is with social networks, where the graph structure is released but details regarding its construction are not provided [21, 17, 28]. For instance, the Facebook100 graphs [28] used an induced subgraph of nodes from one day of logged in users. In this case, note that the graph structure changes if this period changed to 2 or 30 days. These quantities are clearly well-defined properties of the data at hand, but in some sense they and other somewhat arbitrary cutoffs inject "noise" into the graph, relative to the "real" processes generating the underlying original data.

Problems with edges/nonedges in constructed graphs. By a constructed graph, we mean one that is constructed by the data analyst from some other primary data. A canonical example of this is a graph that is constructed from feature vectors according to some nearest-neighbor (NN) rule, as is common in machine learning [27, 24, 3, 7]. Here, one must choose a distance function, whether one works with k-NN or ϵ-NN, the values of k and ϵ, etc. Similarly, one may construct the adjacency matrix by thresholding small values of a corre-

KDD'15, August 10-13, 2015, Sydney, NSW, Australia.

ACM 978-1-4503-3664-2/15/08 ...$15.00.
DOI: http://dx.doi.org/10.1145/2783258.2783376 .

lation matrix or similarity matrix to zero according to some rule. These are best viewed as "model selection" decisions, and the appropriate values of the parameters can be chosen according to a plausible model selection rule. Nevertheless, many of these design decisions are made for methodological or computational convenience and not because they are directly related to the processes generating the data. For instance, many algorithms run more quickly on sparsified networks with small k in k-NN constructions. Thus, they may be viewed as injecting some sort of "noise" into the graph construction process, again relative to the hypothesized "real" processes generating the underlying original data.

Problems with labels associated with the nodes or edges. In either of these cases, one is typically given some sort of labels on some of the nodes and/or edges, and one wants to use the graph structure to propagate that information to predict labels for other nodes/edges. This could be unsupervised, *e.g.*, one might want to find a cluster, or it could be supervised or semi-supervised, *e.g.*, one might want to rank nodes nearby a pre-specified node or one might want to predict the value of labels on unlabeled nodes. In these cases, it is plausible that if the graphs are very "nice" (in some sense) then the clusterings, rankings, or predictions will be robust to errors or noise in the labels, while otherwise (*e.g.*, if "noise" has implicitly been injected into the graph generation step of the data analysis pipeline with one of the mechanisms described above) then the clusterings, rankings, or predictions might not be robust.

In each of these cases, it is of interest to understand how sensitive subsequent analysis is to the details of these initial data modeling decisions—in our experience, they are often *very* sensitive—as well as to develop analysis methods for large-scale data that provide robustness to these decisions.

1.2 Scalable graph algorithms, regularization, and diffusion-based learning

A seemingly very different direction in graph data mining is to develop algorithms that scale to extremely large graphs, *e.g.*, algorithms that don't even have to look at the entire graph but that still come with strong performance guarantees. For example, recent work on local spectral methods has shown that one can find provably-good clusters in very large graphs without even looking at the entire graph [26, 1]. Most relevant for this paper is the so-called "push procedure" of Andersen, Chung, and Lang (ACL) [1], which was originally introduced as an extremely scalable and strongly-local spectral approximation algorithm for the personalized PageRank problem. (It was invented in a variety of forms by [12, 20, 5].) The push procedure in particular has proven critical in establishing very strong empirical results on the existence and non-existence of clusters and communities of different sizes in large social and information networks [18, 13].

Informally, what these local spectral methods do is start with a seed set of nodes and run a few steps of a random walk. If there is a good cluster near the seed set, then this manifests itself as a bottleneck to mixing, the random walk is stopped, and the set of nodes that has non-negligible mass is returned as the cluster. Otherwise, the algorithm detects that there is not a bottleneck and stops before it touches too many nodes. An optimization formulation of these local spectral methods (which may be interpreted as a normalized cut *biased* toward the initial seed set, and which may be solved with a Laplacian-based linear equation solver) has been provided by Mahoney, Orecchia, and Vishnoi (MOV) [19].[1] In addition, Mahoney and Orecchia have shown that these random walk based approximation algorithms implicitly but exactly optimize regularized forms of semi-definite programming variants of the usual Rayleigh quotient objective [23].

To use ACL and MOV as an example of how local spectral methods help robustify graph-based learning algorithms to details of the graph construction process, we apply and extend a framework that was developed to understand the implicit regularization properties in these scalable approximation algorithms. The framework, that of *algorithmic anti-differentiation* (which we introduced recently [10]) aims to make precise the objective functions that approximation algorithms and popular heuristics solve exactly. Among other things, we [10] showed that—for appropriate parameter settings—the MOV procedure solves a certain constrained ℓ_2-regression problem *exactly* (which should not be very surprising to readers familiar with spectral methods); and (much more surprisingly) that the ACL push procedure (which, recall, was originally introduced as an extremely scalable and strongly-local spectral *approximation* algorithm for the personalized PageRank problem) solves an ℓ_1-regularized version of that ℓ_2-regression problem *exactly*. In each of these cases, the algorithmic anti-differentiation method involves identifying an implicitly-defined objective function that is optimized and the construction of an implicitly-defined graph where the algorithm runs. As we noted then, this framework is more general and we now seek to apply it to diffusion-based learning methods [14, 31, 32].

Perhaps the best known diffusion-based learning method is semi-supervised learning on a graph. This involves taking a graph (which is either the primary data, or more typically which has been constructed from neighborhood or distance information between data points), where some of the nodes of which are labeled with various class labels [14, 31, 32], and then inferring the label on the unknown nodes. This approach has also been combined with manifold learning and related methods [4], where the graph is first projected onto an hypothesized manifold and then the points are predicted in light of that projection.

1.3 Contributions

In this paper, we consider three popular diffusion-based machine learning algorithms [31, 14, 32], which we call Zhou et al., Joachims, and ZGL after their authors.

1. To understand their behavior, we place them into a common framework based on mincut construction. This makes a series of implicit regularization properties explicit, following the ideas of algorithmic anti-differentiation we recently proposed. For instance, the diffusion behavior in [31] provides some regularization behavior for the graph edges. In contrast with most previous work, we consider noisy and problematic graphs for these methods to illustrate how they differ in the presence of noise, erroneous edges, mistaken labels.

2. We show that we can use the highly scalable PageRank push algorithm simultaneously to accelerate Zhou et al.'s diffusion to provide another type of implicit regularization.

[1]Importantly, while ACL is strongly local, in that it finds a locally-biased solution without even touching most of the nodes of the original graph, MOV is weakly local, in that it finds a locally-biased solution, but it does so by running an algorithm that touches all of the nodes of the graph.

3. The relationship with the mincut construction provides a new type of label rounding rule based on ranks.

4. Our experiments identify a key weakness of these diffusion methods: they cannot use many straightforward constructions that provide additional edges. Put more plainly, their performance is worse when the graph becomes dense. We propose a particular type of densification procedure to generate *useful* additional edges. These additional edges improve the classification performance in both the classic digits dataset as well as a product-category prediction task on the Amazon co-purchasing network.

1.4 Related work

There is a large body of existing work on diffusion-based and eigenvector-based machine learning methods, some of which we have already surveyed in our introduction. Our work is very related to both Laplacian Eigenmaps [3], which embeds the data on the eigenvectors of a Laplacian matrix, as well as to the related Diffusion Maps, which refines Laplacian Eigenmaps construction by considering more sophisticated diffusions [7]. These methods are typically of greatest interest when the data are not too noisy and when there exists some sort of "nice" underlying structure to be found—e.g., if there is a clear separation between eigenvalues of the Laplacian or when the data are drawn from an underlying manifold. In what follows, we have examined graphs which are not "nice" in these senses. We study a synthetic example to identify a series of effects that predict behavior on real-world datasets.

Some of the most related work has similar goals. The importance of the graph construction process on semi-supervised learning is already recognized [8]. There are a variety of techniques in the literature to improve edge weighting to make the diffusions perform better [25], or to choose parameters of the methods in automated ways [30]. These nicely complement our new results on graph densification, rank rounding, and implicit regularization of the methods.

2. SEMI-SUPERVISED LEARNING VIA DIFFUSIONS AND CUTS

Consider a graph G on n nodes, in which a subset of nodes are labeled with K classes. The goal is to predict the class label of all the unlabeled nodes. Let $\mathbf{S}$ be the labeled sample matrix, where $S_{i,j} = 1$ if node i has class label j, and $S_{i,j} = 0$ otherwise. This is an $n \times K$ matrix. Our results apply in the case when graph G is undirected, connected, and weighted (although they also extend with some technicalities to the case of disconnected graphs). The algorithms we study here use a *graph diffusion*—the precise diffusion is given in the following sections—to produce an $n \times K$ matrix $\mathbf{Y}$. Elements of $Y_{i,j}$ should be large if node i should be labeled as class j. This particular setup has the intuitive flavor of diffusing large values from the labeled nodes to the unlabeled nodes. These diffusions are the first key ingredient of the methods. Most of our discussion and results below will focus on aspects of this first ingredient. The second key ingredient, which we will also discuss briefly, is the scheme to translate the diffusion values $\mathbf{Y}$ into a predicted label. Based on the approximation algorithms literature, we call this second step the *rounding step*. Hence, the overarching framework for diffusion-based semi-supervised learning is *diffuse and round*.

We should note that the basic ideas of this framework, based on mincuts, are likely unsurprising to experts on semi-supervised learning on graphs. Indeed, elements of our derivations appear in the literature [14, 32], and the connection was mentioned in passing as a remark in our previous work [10]. But the details matter for the noisy situations we consider, and academic data analysts are often cavalier about such details. Thus, we describe the connections between the diffusion-based methods and s,t-min-cut problems precisely.

2.1 Technical background

Let $\mathbf{A}$ be the adjacency matrix for a graph G. The graphs we consider are connected and undirected, but they may be weighted, in which case the elements of $\mathbf{A}$ contain the weights. Let $\mathbf{e}$ be the vector of all ones, and let $\mathbf{d} = \mathbf{A}\mathbf{e}$ be the vector of degrees—some call these weighted degrees, but we will not make that distinction. The matrix $\mathbf{D}$ is the diagonal matrix with the degrees $\mathbf{d}$ on the diagonal. Then $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the Laplacian matrix of G. Let $\mathbf{B}$ be the unweighted edge-node incidence matrix, and let $\mathbf{C}$ be the corresponding diagonal matrix of weights associated with each edge. Thus, $\mathbf{L} = \mathbf{B}^T\mathbf{C}\mathbf{B}$. It is well-known that the Laplacian matrix has deep connections to random walks, electrical flows, and minimum cuts; and we are most interested in these latter connections. Here, we will follow our prior work on algorithmic anti-differentiation, in particular following the notation and setup used previously [10], and we will frame our discussion in terms of spectral graph theory and min-cut problems.

Recall that, in our setup, the s,t-minimum cut problem is:

$$\begin{array}{ll} \text{minimize} & \|\mathbf{B}\mathbf{x}\|_{C,1} = \sum_{(u,v)\in E} C_{(u,v)}|x_u - x_v| \\ \text{subject to} & x_s = 1, x_t = 0. \end{array} \tag{1}$$

The ℓ_2-minorant of (1) is a key components of our framework:

$$\begin{array}{ll} \text{minimize} & \|\mathbf{B}\mathbf{x}\|_{C,2} = \sqrt{\sum_{(u,v)\in E} C_{(u,v)}|x_u - x_v|^2} \\ \text{subject to} & x_s = 1, x_t = 0, \end{array} \tag{2}$$

or, equivalently, of this problem:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|\mathbf{B}\mathbf{x}\|_{C,2}^2 = \frac{1}{2}\sum_{(u,v)\in E} C_{(u,v)}|x_u - x_v|^2 = \frac{1}{2}\mathbf{x}^T\mathbf{L}\mathbf{x} \\ \text{subject to} & x_s = 1, x_t = 0. \end{array} \tag{3}$$

Note that (3) is not a minorant of (1); although, due to the equivalence between (2) and (3), we refer to it this way. Note also that although (1) requires a min-cut/max-flow network linear program solve, (3) requires only a linear solve with a sub-matrix of $\mathbf{L}$. (This sub-matrix is non-singular if the graph remains connected with nodes s and t disconnected.)

Existing diffusion-based learning methods, *e.g.*, those of [14, 31, 32], tend to focus on deriving the linear systems themselves. We prefer to explain these diffusions in terms of the ℓ_2 cut minorants of cut problems. In the following sections, we'll describe an s,t minimum cut construction whose ℓ_2 minorant is the diffusion-based learning method. See Figure 1 for reference throughout the following sections.

2.2 Zhou et al.'s diffusions and the Andersen-Lang weighting variation

Zhou et al. [31] propose the diffusion equation:[2]

$$\mathbf{Y} = (\mathbf{L} + \alpha\mathbf{D})^{-1}\mathbf{S}, \tag{4}$$

[2]Actually, they use $\mathbf{Y} = (\mathbf{D} - \beta\mathbf{A})^{-1}\mathbf{S}$. If $\alpha = \frac{1-\beta}{\beta}$, however, this is equivalent to our statement, up to scaling by a constant.

where $\alpha > 0$ and $\mathbf{L}$ is the Laplacian of the original graph.

This is equivalent to the minorant of an s,t-cut problem where the problem varies based on the class. To see this connection, consider the prediction of one column of $\mathbf{Y}$, say $\mathbf{y}_j$. Let $\mathbf{s} = \mathbf{s}_j$ be the jth column of the labeled samples matrix $\mathbf{S}$ for class j. Then the mincut graph for $\mathbf{s}$ is a graph where node s connects to each sample labeled with class j with weight α. Node t connects to all nodes in the graph (except s) with the connection weight for node i being $\alpha(d_i - s_i)$. If $s_i = d_i$, then the edge has weight 0, which is equivalent to that edge not existing at all. (This construction is illustrated in Figure 1(a).) If we then apply the ℓ_2-minorant of the s,t-cut problem, we seek $\mathbf{x}$ or $\mathbf{y}$ such that:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|\mathbf{B}\mathbf{x}\|_{C,2}^2 \\ \text{subject to} & x_s = 1, x_t = 0 \end{array} \quad \Leftrightarrow \quad \begin{array}{ll} \text{minimize} & \frac{1}{2}\begin{bmatrix} 1 \\ \mathbf{y} \\ 0 \end{bmatrix}^T \begin{bmatrix} \alpha\mathbf{e}^T\mathbf{s} & -\alpha\mathbf{s}^T & 0 \\ -\alpha\mathbf{s} & \alpha\mathbf{D}+\mathbf{L} & \alpha(\mathbf{d}-\mathbf{s}) \\ 0 & -\alpha(\mathbf{d}-\mathbf{s})^T & \alpha\mathbf{e}^T(\mathbf{d}-\mathbf{s}) \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{y} \\ 0 \end{bmatrix}, \end{array} \tag{5}$$

where $\mathbf{y}$ is the portion of $\mathbf{x}$ defined only on the vertices of the original graph. (Here, we have labeled s as node 1 and t as node $n+2$, recall we added nodes s and t.) By expanding the final equation, we find that $\mathbf{y}$ solves:

$$(\alpha\mathbf{D} + \mathbf{L})\mathbf{y} = \alpha\mathbf{s},$$

and this is exactly a rescaled column of $\mathbf{Y}$. Note that if any element of $\mathbf{s} > \mathbf{d}$, then we must scale the weights on the graph so that $\mathbf{s} \leq \mathbf{d}$.

We note in passing that there is a natural variant of this diffusion where s connects to the labeled sample nodes with weight αd_i. Then there will be no edge from the labeled samples to t because $\mathbf{d} - \mathbf{s}$ will be zero for those nodes (see Figure 1(b)). For graphs with large variations in degrees, this will affect the diffusion in a significant way. We call this *Andersen-Lang weighting* due to its relationship with the FlowImprove construction [2]. This modification has the secondary advantage that there is no need to worry about scaling the graph as $\mathbf{D}\mathbf{s} \leq \mathbf{d}$ for any binary vector $\mathbf{s}$.

2.3 Joachims's and ZGL's diffusions

Joachims [14] proposed a different type of diffusion. The construction we describe is inspired by the ideas discussed in that paper.[3] Namely, find a vector $\mathbf{y}$ that satisfies the following bicriteria: it minimizes $\mathbf{y}^T\mathbf{L}\mathbf{y}$, and it has values near 1 for nodes labeled with class j and values near -1 for nodes labeled with classes other than j.

Our framework can provide this type of bias if we connect s to the nodes labeled with the current class and connect t to the nodes labeled with other classes. One large difference is that we predict values near 0 for nodes labeled with classes other than j. (See Figure 1(c) below.) The two natural choices for the weights of these edges are either 1, or the degree of the node. We study the case of unit weights for simplicity, but note that the construction seamlessly extends to the degree-weighted cases.

[3]While we draw inspiration from Joachims' construction, the details actually differ markedly, as Joachims includes many additional features that, no doubt, improve its practical performance considerably.

The diffusion that results from taking the ℓ_2 minorant of Joachims' s,t-cut construction is:[4]

$$\mathbf{Y} = (\mathbf{D}_S + \mathbf{L})^{-1}\mathbf{S},$$

where $\mathbf{D}_S$ is a diagonal matrix with the row-sums of $\mathbf{S}$ on the diagonal. Because most rows in $\mathbf{S}$ are completely 0 (these are the unlabeled nodes), and the remainder only have a single 1, the matrix $\mathbf{D}_S$ is really a diagonal indicator matrix over the labeled nodes.

Finally, Zhu, Ghahramani, and Lafferty [32] proposed another type of diffusion, henceforth called the ZGL diffusion, that is similar to that of Joachims, except that it strictly enforces the labeling on the labeled samples. For the predictions on the jth class, ZGL solves:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{y}^T\mathbf{L}\mathbf{y} \\ \text{subject to} & y_i = \begin{cases} 1 & \text{if node } i \text{ labeled in class } j \\ 0 & \text{if node } i \text{ labeled in another class} \\ \text{free} & \text{otherwise.} \end{cases} \end{array} \tag{6}$$

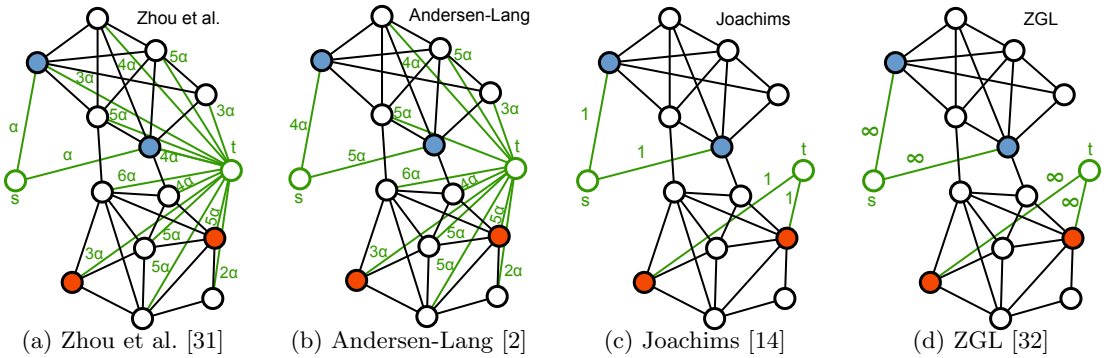
This diffusion is an ℓ_2 minorant of the s,t-cut problem where s connects to all nodes labeled with class j with infinite weight. So, in particular, they will never be cut. (See Figure 1(d) below.) Thus, $y_i = 1$ for all nodes labeled j. Likewise, all nodes labeled with another class connect to t with infinite weight, so that $y_i = 0$ for those nodes.

3. ROBUSTIFYING VANILLA DIFFUSION-BASED METHODS

We use the term "vanilla diffusion-based method" for a machine learning process that involves the basic semi-supervised strategy of performing a diffusion to obtain a vector and then using the numerical values (*i.e.*, not the ranks) of the entries of that vector to obtain class labels. In practice, these methods are enhanced with a host of corrections and modifications to improve their performance, such as class-size normalization. We wish to focus on the vanilla *diffuse and round* procedure to propose principled, simple changes that improve its robustness to data errors. That is, we hope to enhance the vanilla algorithms with features that serves multiple purposes, and these modifications should make it easier to use diffusion-based learning methods in data analysis pipelines where humans cannot intervene with small corrective fixes.

In the previous section, we showed that several popular diffusion-based propagation methods can be expressed as the solution to Laplacian-based linear equations on an implicitly-defined cut graph. In this section, we will relate these diffusion procedures to a variety of standard regularization and robustification procedures to help us understand their behavior on real-world data (that we describe in the next section). Specifically, we will show that Zhou et al.'s diffusion implicitly localizes and regularizes the solution of the diffusion in a small region of the graph through a relationship with the MOV locally biased analogue of the Fiedler vector [19]. This helps to make it robust to errors in the graph. We can also add a 1-norm, or sparsity inducing, penalty to all of the diffusion equations to render them robust to errors in the labels. When we add this penalty to Zhou's

[4]We omit the derivation of this minorant due to its similarity with the existing derivations.



(a) Zhou et al. [31] (b) Andersen-Lang [2] (c) Joachims [14] (d) ZGL [32]

Figure 1: The s,t-cut graphs associated with four different constructions for semi-supervised learning on a graph. The labeled nodes are indicated by the blue and red colors. This construction is to predict the blue-class. Both Zhou et al. and the Andersen-Lang variation only model the effect of the current class.

diffusion, we arrive at a formulation that is equivalent to a personalized PageRank problem. The Andersen-Chung-Lang push procedure is an optimized routine to efficiently estimate such diffusions. Finally, we detail a simple change to the rounding procedure that renders the methods more robust to class imbalances and is firmly rooted in the theory of ℓ_2 cut relaxations.

Note that all of the regularization we discuss is implicit in the sense that it arises as a byproduct of the model and algorithm and not by explicitly adding it as an extra regularization term that is then solved by a black-box solver.

3.1 Implicit robustness in Zhou's construction

Zhou's diffusion construction results in solving:

$$\mathbf{Y} = (\alpha\mathbf{D} + \mathbf{L})^{-1}\mathbf{S}$$

where $\mathbf{S}$ is the matrix of labels for each class and $\mathbf{Y}$ is the predicted label. We now show that each column of $\mathbf{Y}$ is equivalent to a weakly local MOV vector [19]. The MOV vector was derived as a locally-biased variation on the Fiedler vector of a graph and arose from a spectral relaxation of a minimum conductance optimization problem. A MOV vector $\mathbf{r}$ is the solution of the linear system:[5]

$$(\mathbf{L} - \gamma\mathbf{D})\mathbf{r} = \mathbf{D}\mathbf{v}$$

where $\mathbf{v}$ is a vector that such that $\mathbf{v}^T\mathbf{d} = 0$. Now let $\mathbf{y}_c$ be the solution of Zhou's diffusion for class c. This vector satisfies the linear system: $(\alpha\mathbf{D}+\mathbf{L})\mathbf{y}_c = \mathbf{s}_c$. We can convert this to a MOV problem as follows. Let $\mathbf{v} = \mathbf{D}^{-1}\mathbf{s}_c - \frac{1}{n}\mathbf{e}^T\mathbf{s}_c\mathbf{e}$, then $\mathbf{v}^T\mathbf{d} = 0$. Let $\gamma = -\alpha$. Set $\mathbf{r}$ to the MOV vector

$$(\mathbf{L} - \gamma\mathbf{D})\mathbf{r} = \mathbf{D}\mathbf{v}.$$

Then, $\mathbf{r} = \mathbf{y}_c + \mathbf{c}$ for some correction vector $\mathbf{c}$. The correction vector $\mathbf{c}$ satisfies:

$$(\mathbf{L} - \gamma\mathbf{D})\mathbf{c} = -\tfrac{\mathbf{e}^T\mathbf{s}_c}{n}\mathbf{D}\mathbf{e},$$

which has a scaled constant vector as the solution: $\mathbf{c} = \frac{\mathbf{e}^T\mathbf{s}_c}{n\gamma}\mathbf{e}$. Notice that all we have done is shifted the solution $\mathbf{r}$ by a constant amount in each component.

[5]In the MOV derivation, we considered varying γ over a regime that included singular matrices, and hence, used a pseudo-inverse solution. For this equivalence, the matrices are always non-singular in the relevant γ regime.

The importance of this result is given by the properties of the MOV vector. Using Zhou et al.'s diffusion results in a process that is robust to errors in the graph because the diffusion implicitly localizes around the labeled nodes (which is what MOV was explicitly constructed to do for a local Cheeger inequality). This prevents changes far away in the graph from having a large influence on the diffused labels. In contrast, both ZGL and Joachims's diffusions are more sensitive to the entire graph because the diffusion must touch the other class labels. Moreover, the values for class s depend on the location of the labels for the other classes.

3.2 Adding a regularizing sparsity penalty

Each of these diffusion-based learning problems corresponds to a minimization problem with a modified Laplacian matrix. One could regularize the underlying objective function, by adding a sparsity penalty, and then explicitly solve the regularized problem to obtain a more robust solution. For example, one could simply add on, say, a sparsity-inducing ℓ_1 regularization function and then call a black-box solver. *But this might be much more expensive than simply implementing the original diffusion.*

It turns out that one can solve *certain regularized versions of these original diffusion-based problems* in a much more scalable and robust way. To see this, consider (5), and observe that if one uses a degree-weighted ℓ_1-norm penalty, then one obtains the following objective:

$$\begin{aligned} &\text{minimize} \quad \tfrac{1}{2}\begin{bmatrix}1\\ \mathbf{y}\\ 0\end{bmatrix}^T \begin{bmatrix}\alpha\mathbf{e}^T\mathbf{s} & -\alpha\mathbf{s}^T & 0\\ -\alpha\mathbf{s} & \alpha\mathbf{D}+\mathbf{L} & \alpha(\mathbf{d}-\mathbf{s})\\ 0 & -\alpha(\mathbf{d}-\mathbf{s})^T & \alpha\mathbf{e}^T(\mathbf{d}-\mathbf{s})\end{bmatrix}\begin{bmatrix}1\\ \mathbf{y}\\ 0\end{bmatrix} + \kappa\|\mathbf{D}\mathbf{y}\|_1 \\ &\text{subject to} \quad \mathbf{y} \geq 0. \end{aligned} \tag{7}$$

As we pointed out in prior work, and importantly for our results below, for appropriate parameter settings, the Andersen-Chung-Lang fast "push" method for computing approximations to personalized PageRank actually solves (7) *exactly* [10]. For large values of parameter κ, systems of the form of (7) can *exactly* be solved locally (i.e., by touching a number of nodes that depends on the output and not the number of nodes in the input graphs)—making this a highly-scalable setup in practice. The push method for (7) has been applied to graphs of size up to at least tens of millions of nodes and billions of edges [18, 15], with runtimes measured in seconds and milliseconds. In this paper, we

do *not* address running time issues, since our main focus is on robustness issues and using local spectral methods to compute more robust solutions to diffusion-based machine learning problems.

Thus, let us be clear. We do not explicitly regularize the solution by solving (7). Rather, we simply call the push method to compute an approximate PageRank vector. This algorithm implicitly solves (7).

It is unclear if similar statements can be made for appropriately regularized versions of the Joachims and ZGL diffusion discussed in Section 2. The reason for this is that in each step of the push procedure for Zhou et al.'s diffusion there is a quantity that "leaks out" of the graph (this can be formalized as a probability measure). For Joachims and ZGL, this does not occur.

3.3 Rounding diffusion vectors to class labels

The result of the diffusion procedure is a real-valued number that indicates the propensity of node i to belong to class j. The standard method to round these to labels is to use the largest value among all classes [31], with many heuristic variations [32]. Our results show that these diffusions are all minorants of min-cut objective functions. The standard way to turn a real-valued cut approximation into a hard cut is to perform a sweep-cut procedure on the order of the nodes; this is the basis of the celebrated Cheeger inequality [6]. The theory of the spectral cut problems, then, suggests that the rank of the node should be important. Thus, we propose another technique to robustify diffusion methods: round to classes based on the node with the smallest rank in the ranked-list of each diffusion vector, i.e. each column of $\mathbf{Y}$.

To see the advantage of this strategy, consider a graph with the adjacency matrix that is shown in Figure 2(a). Clearly, there are three fairly well separated classes with a relatively few edges between them. The overall matrix has 60 nodes. Class 1 has 30 nodes, class 2 has 10 nodes, and class 3 has 20 nodes. Thus, each prediction matrix $\mathbf{Y}$ has three columns corresponding to each of the three classes. The remaining subfigures show the matrix $\mathbf{Y}$ from the diffusion-based learning methods we study: Zhou et al. (b, f); the Andersen-Lang weighting variation (c, g); Joachims's method (d, h); and ZGL (d, i). The top panel in each of the remaining subfigures shows the case of 3 labeled nodes (b, c, d, e), while the bottom panel in each of the remaining subfigures shows the case of 15 labeled nodes (f, g, h, i). In each figure, note that each column of $\mathbf{Y}$ is displayed as a separate line, color-coded according to one of the three class labels.

Let's consider the top row with one labeled node from each class, i.e., that corresponding to using 3 labeled nodes. The spikes in each vector represent the effect of the labeled node on each diffusion, while the values on the remaining nodes are the value of the diffusion for that node. Note that only Zhou et al.'s diffusion with three labels *and* value-based rounding will correctly classify this simple example, agreeing with the simple intuition. The Andersen-Lang variation suffers from a degree-bias (in this simple case, it just weights Zhou et al.'s diffusion with the degree of the labeled node). Both Joachims and the ZGL diffusions misclassifies class 3 as class 1 because the near-clique like structure in class 1 dominates the diffusion effects and the negative labels in class three are not enough to recover. Thus, the negative labels have the effect of changing the weight of each diffusion.

When we move to 15 labels, we label one node from each class as before and the remaining 12 labels are chosen uniformly at random. None of the methods classify the examples correctly and they all uniformly predict class 1, except for ZGL which labels the pinned points correctly along with a few nodes in classes 2 and 3. This occurs because there are more nodes randomly selected from class 1, and it receives a higher overall weight. We make two observations here. First, there are a variety of very reasonable and well-motivated heuristic corrections that would eliminate these effects and restore the performance of value based rounding, and the introduction of these heuristics is common. Second, the rank-based rounding has good performance in all these cases, regardless of the particular diffusion employed. To see this, look at the ranking (essentially, fix a color, and move along the Y axis from the top, and observe which nodes are "hit") on each color in the bottom panel of each of these subfigures. All of the diffusion show raised levels for the correct class, and these correct classes are revealed by the rank-based rounding scheme, thus yielding far more robust predictions.

4. MAIN EMPIRICAL RESULTS

Our empirical results are on the following four themes:

1. There are broad similarities between the diffusions we have discussed. We wished to study the differences among them in a synthetic setting to get a more nuanced understanding of their behavior. In particular, we focus on what happens with the methods in low and high error rate scenarios in sparse and dense graphs. These results indicate that Zhou et al.'s diffusion is best in sparse networks (most real-world networks are sparse).

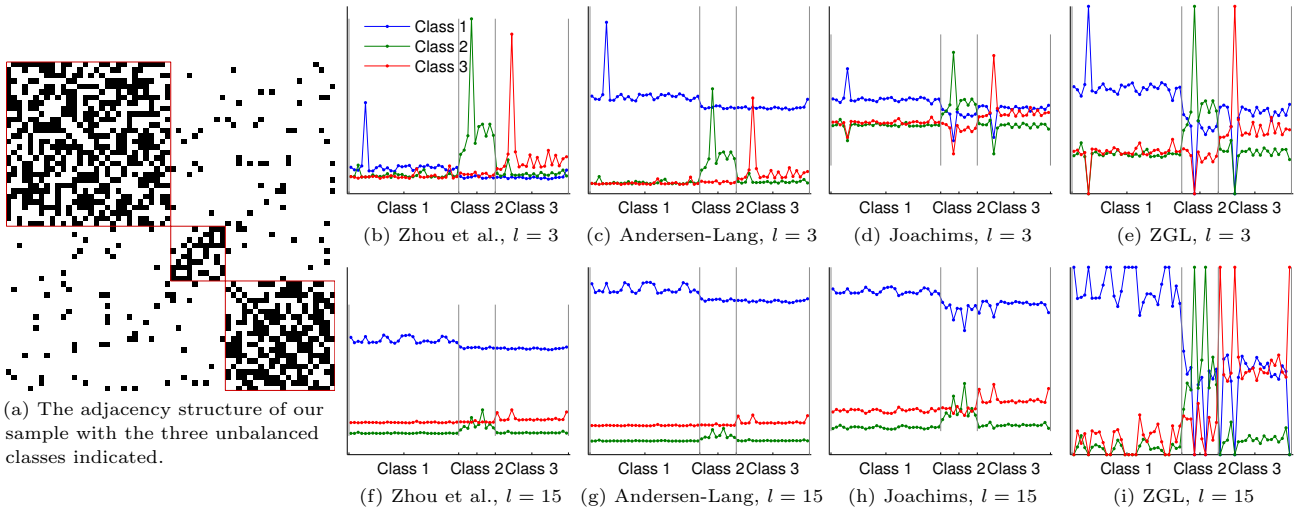
2. Next, we use the classic semi-supervised diffusion-learning task of digit prediction to investigate the impact of graph construction on the final error rate. These results show: (i) the impact of rank-based rounding on the error rate; and (ii) indicate that simple constructions of two vastly different types of dense graphs have higher error rates in the presence of labeling mistakes. In these cases, the primary advantage of using the push method is computational instead of for robustness. It still provides a mild error rate benefit.

3. The results on the digits dataset indicate, counter intuitively, that using additional nearest neighbors or kernel information increases the error rate. We return to digits and address the question of how best to use a given computational budget of edges. Our experiment shows that a densified sparse graph yields lower error rates than a denser nearest neighbor graph.

4. We continue with a study of these densifying ideas in the Amazon co-purchasing graph to illustrate how they arise in a more realistic problem. In this problem, we see the that the densified construction explored in the previous point greatly improves performance of the diffusion-learning procedure.

4.1 A comparison of the diffusions

We generate a two-class problem with 150 nodes in each class. The graph we construct is a block-model with between class probabilities of 0.02. The within class probabilities is 0.35 in the dense case and 0.06 in the sparse case. We then fix a number of nodes within each class and reveal their labels. Each class has the same number of labels revealed. We then



(a) The adjacency structure of our sample with the three unbalanced classes indicated.

(b) Zhou et al., $l = 3$ (c) Andersen-Lang, $l = 3$ (d) Joachims, $l = 3$ (e) ZGL, $l = 3$

(f) Zhou et al., $l = 15$ (g) Andersen-Lang, $l = 15$ (h) Joachims, $l = 15$ (i) ZGL, $l = 15$

Figure 2: A study of the paradoxical effects of value-based rounding on diffusion-based learning in a simple environment. With three labels, only Zhou et al.'s diffusion has correct predictions, whereas with 15 labels, none of the methods correctly predict the three classes. See the text for more about the nature of the plots.

choose a small subset of additional nodes and reveal a random label for those nodes. These random labels may be correct or incorrect and we consider two where these errors occur at a low rate and high rate. Each experiment is repeated 1000 times. The resulting mean of mistaken classifications are shown in Figure 3. *The methods should all work perfectly in these cases if there were no label errors.* We wanted to see the differences when problems arise.

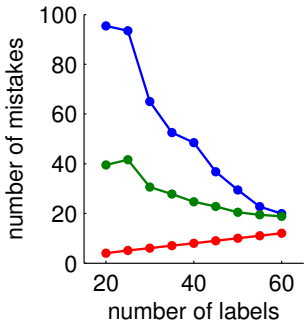
For the dense graph, the ZGL diffusion always has the best performance. It makes exactly the same number of mistakes as were given by the randomly labeled nodes. (Recall that the ZGL diffusion fixes the provided class labels and hence the number of mistakes it makes increase with the number of labels.) As the number of labels increases, all of the methods have similar performance. For the sparse graph, Zhou et al.'s diffusion does substantially better with a low error rate and also does better with a high error rate (but not too many labels). These results are consistent with the cut derivations and implicit regularization of the methods. Both ZGL and Joachims's diffusion have trouble with the sparse graph because there is no clear minimum cut between the classes to find. Zhou et al.'s diffusion propagates independently between classes and rank-based rounding enables it to find more accurate information about the class structure.

4.2 A case study with the digits dataset

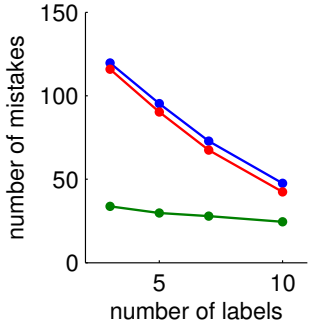
The problem setup we consider is the digit labeling task [31]. We construct a weighted graph between images of the digits that depends on a radial basis function width σ. Choosing $\sigma = 2.5$ results in a dense graph (although it is still sparse by comparison with the study in Figure 3), where each node has significant connections to many other images. Whereas $\sigma = 0.8$ yields a much sparser graph with fewer significant connections. We randomly pick labeled nodes in the input after picking one labeled node in each class. We only describe the results of Zhou et al.'s diffusion-based methods since our preliminary study showed they had better performance than the other methods. As in the prior work [31], we used $\alpha = 0.01/0.99 \approx 0.0101$. When we use the push algorithm to estimate Zhou's diffusion, we implement a simple bisection search procedure to find a value of κ that produces between 33-50% non-zeros in the final solution.

Values vs. Rank. Figure 4(a) and (b) illustrate the effect of rank vs. value rounding for this task in the case that no digits have any labeling errors. For value-based rounding, the error rates without using the implicit regularization of the push procedure are worse than random guessing and the method basically predicts just one class. This is not a bug and there is a simple illustration of this phenomenon from Figure 2. Rank-based rounding shows that there is no real difference between the regularized and non-regularized diffusions. This shows how we are able to make the method robust to differences in graph construction. In the case $\sigma \approx 0.8$, then value-based rounding is slightly better than rank-based with many training samples (this is not shown due to space). Hence, datasets can be engineered through careful construction and cross validation to perform well with value-based rounding, but one needs to be thoughtful of a myriad of perplexing effects. Using rank-based rounding avoid all of these issues.

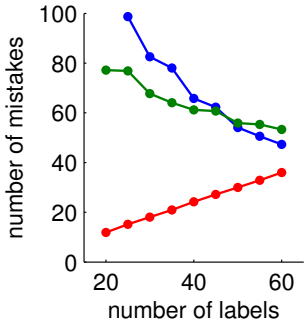
Density and error rate. In the next experiment, we consider two types of graph constructions derived from the digit dataset. First, we continue to use the standard construction and vary the kernel density width parameter σ. Second, we can convert the weighted graph into a highly sparse unweighted graph through a nearest neighbor construction. That is, for each node in the weighted graph, we form edges to its r-neighbors with the largest weights. Then we discard the weights on the edges. (Note that the value of σ is not relevant for the nearest neighbor graph as changing σ results is a non-linear, monotonic change to the values that retains the same nearest neighbors.) We use an error label rate of 20% for these experiments, and all subsequent experiments with the digits dataset. There is an average of 5 labels provided for each of the 10 classes, and at least one correct label for each class.



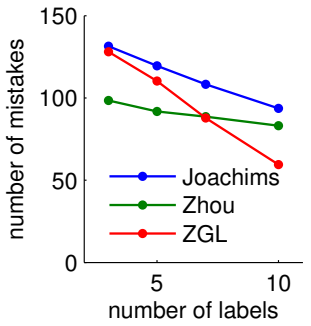
(a) A dense graph, low label error rate



(b) A sparse graph, low label error rate

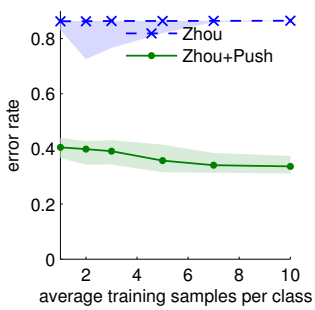


(a) A dense graph, high label error rate

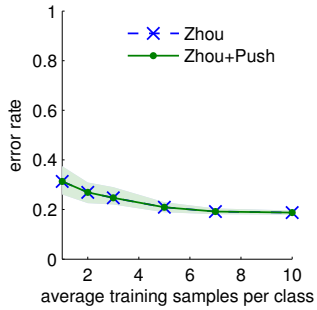


(b) A sparse graph, high label error rate

Figure 3: Comparing the diffusions in a synthetic case shows differences between sparse and dense graphs with low and high error rates.

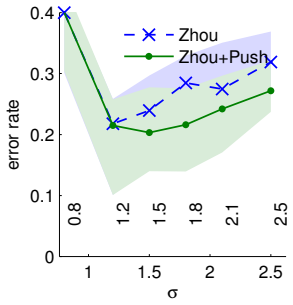


(a) $\sigma = 2.5$, no errors, value rounding

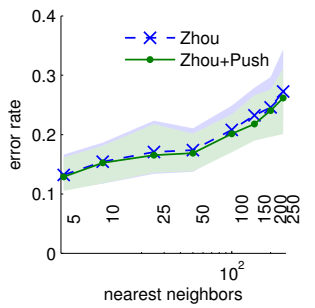


(b) $\sigma = 2.5$, no errors, rank rounding

Figure 4: Results that show the difference between value-based rounding (a) and rank-based rounding on a dense graphs, large σ. We used 50 trials for the and indicate the 20% and 80% percentiles with the shaded region.



(a) Varying σ



(b) Varying nearest neighbors

Figure 5: Results on the performance of the diffusions while varying the density through changing (a) σ or (b) varying r in the nearest neighbor construction. In both cases, making the graph "denser" results in worse performance. We used an average of 5 labels per class and 25 trials for the and indicate the 20% and 80% percentiles with the shaded region.

The results in Figure 5 show that increasing density, for both notions, only decreases the performance of the diffusion-learning algorithm. That is, we increase the error rate by incorporating additional information through the kernel width or the nearest neighbors. This is counter intuitive from the perspective that adding information should only increase performance. The results are accurate and suggest that by increasing the density of the graph, we are actually adding additional errors. If this is indeed the case, then the slight reduction in error rate that arises from using the Push procedure to compute Zhou's diffusion shows that it adds robustness against these types of error—just as a regularization method should.

4.3 Improving error rates with more edges

In the last experiment, we found that two straightforward and common ways of increasing the density (and ideally, the information) in a graph construction resulted in failure. This caused us to ask, what is the best way to use additional edges to improve performance? This question is important because many real-world scenarios involve a fixed computational budget of graph edges, dictated by storage and memory limitations. The previous experiment suggest that there is some limit where additional resources become harmful instead of helpful. We now demonstrate an idea that will allow us to use those additional edges to generate a denser graph to improve performance.

The construction we employ to densify a graph is based on neighbors at distance k. Given a graph with adjacency matrix $\mathbf{A}$, then the graph $\mathbf{A}_k$ is a weighted graph that counts the number of paths of length up to k between pairs of nodes. Formally, we can compute it through the summation:

$$\mathbf{A}_k = \sum_{\ell=1}^{k} \mathbf{A}^{\ell}.$$

These matrices become dense quickly and we suggest using small values of k. (Below we consider up to $k = 4$, and $k = 3$ for the real-world problem on Amazon).

In our current experiment, we wish to test the idea that using $\mathbf{A}_k$ results in improved error rates for a fixed budget of edges. To do so, we first compute a nearest neighbor graph with a small number of neighbors, then compute $\mathbf{A}_k$, and subsequently look for another nearest neighbor graph with about the same average degree. (The average degree of a graph constructed with r-nearest neighbors is always at least r. It is usually a little larger). Our paired sets of parameters are given in Table 1, which span the range of 13 to 97 nearest neighbors, for an average degree between 18 and 140.

Neighs.	Avg. Deg	Neighs.	k	Avg. Deg
13	19.0	3	2	18.1
28	40.5	5	2	39.2
37	53.3	3	3	52.3
73	104.4	10	2	103.8
97	138.2	3	4	127.1

Table 1: Paired sets of parameters that give us the same non-zeros in a nearest neighbor graph and a densified nearest neighbor graph $\mathbf{A_k}$.

	Zhou		Zhou w. Push	
Avg. Deg	$k=1$	$k\geq 1$	$k=1$	$k\geq 1$
19	0.163	0.114	0.156	0.117
41	0.156	0.132	0.158	0.113
53	0.183	0.142	0.179	0.136
104	0.193	0.145	0.178	0.144
138	0.216	0.102	0.204	0.101

Table 2: Median error rates over 25 trials show the benefit to making a sparse graph into a dense graph via the $\mathbf{A_k}$ construction. Note that using an average degree of 138 outperforms all of the nearest neighbor trials from Figure 5. See Table 1 for the number of nearest neighbors and values of k.

The results in Table 2 show that we can achieve our goal and decrease the error rate for Zhou's diffusion using additional edges. More specifically, we find that using a small number of nearest neighbors and a larger value of k results in the best performance. Using $r = 3$ and $k = 4$ gives an average degree of 138, and an error rate of around 0.1. *This error rate is better than any of the previous trials, and outside the confidence intervals of the nearest neighbor trials.*

4.4 A case study with Amazon

We wished to validate these densification findings in a real-world setting. Consequently, we considered the problem of predicting product categories on Amazon in the product co-purchasing network [29]. The co-purchasing network is extremely sparse with an average degree of three. It is also a sparsified version of a hidden co-purchasing network kept internally at Amazon, thus making a natural analogy with our experiments on the digits dataset. One tricky aspect of this problem is that the product categories overlap, and so we studied the single-category prediction problem. To determine the final set of labels, we used a sweep-cut procedure that converts the diffusion information into a binary cut vector by minimizing the conductance of the cut; we like to think of this as returning to the original s, t-mincut constructions as Zhou's diffusion is highly related to a mincut strategy to minimize conductance [2]. We selected categories based on the strategy from [16], i.e., categories with nearly $p_{\max}^{3/4}$ items where $p_{\max}$ is the number of products in the largest category. We create dense versions of this graph in the same way: $\mathbf{A}_k = \sum_{\ell=1}^{k} \mathbf{A}^\ell$, where $\mathbf{A}$ is the adjacency matrix of the co-purchasing network.

The F_1 results of our predictions are shown in Table 3. For $k = 2$ and $k = 3$, we see that the performance of Zhou's diffusion improves—just like in the digits dataset. The push algorithm keeps the precision high, at the expense of recall.

	mean F_1		Confidence intervals			
k	Zhou	Zhou w. Push	Zhou		Zhou w. Push	
1	0.173	0.229	[0.15	0.19]	[0.21	0.25]
2	0.197	0.231	[0.18	0.22]	[0.21	0.25]
3	0.221	0.238	[0.17	0.27]	[0.19	0.28]

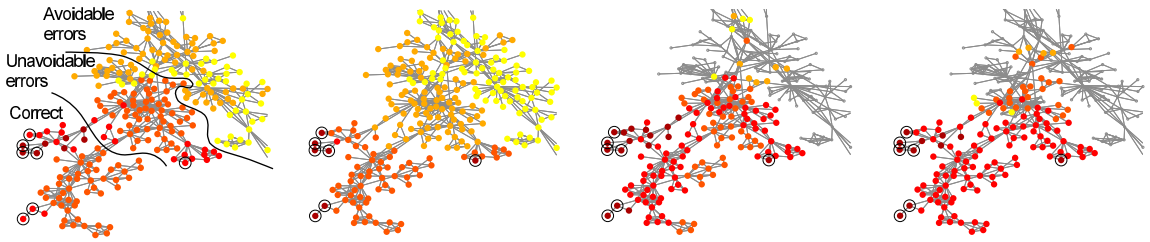
Table 3: The $\mathbf{F_1}$ results of predicting product categories on an Amazon co-purchasing network based on 20 labeled products with 2 errors show that when the graph is densified (k=3), the result of Zhou's diffusion improve more than the results of the implicitly regularized diffusion via push.

5. DISCUSSION AND CONCLUSION

We are optimistic that these schemes will enable graph-based learning to be better behaved in realistic analytics pipelines. Such pipelines are increasingly common in fields such as bioinformatics, where massive quantities of highly noisy data are mined for subtle insights about the behavior and function of new biological processes. We wish to emphasize that *errors in labels hurt the diffusions in different ways*, as in the results of Figure 3. While it may seem strange in academic machine learning to have errors on the given class labels, it is common in many applications, *e.g.*, biology. The ground truth data are often inferred from a prior computational procedure that may itself be noisy. Understanding these effects is important for future work.

We have presented this work in a simplified setting to elucidate our points easily. In terms of practical real-world usage, we note that the modifications we propose are easy to incorporate into large scale graph computation frameworks such as GraphLab or Ligra. In fact, the sparsity regularizer we consider reduces the total work in the algorithm. There are also a variety of ways to adapt these ideas to directed graphs, such as using the directed Laplacian due to Fan Chung. Also, having soft labels as the initial and final solutions are possible because our ideas seamlessly handle a *weighted set* instead of the discrete set just by using a weighted seed vector and since our methods produce a continuous valued output that must be rounded back to a label.

In terms of our result on densification, on sparse networks, diffusions propagate too quickly with this value α and thus any mistake propagates throughout the network. In this case, the regularized solutions exert a correcting weak effect themselves by restricting the "diffusion flow." If the network is dense, then regularization doesn't significantly change the solution because if the signal propagates anywhere, it propagates everywhere (think of a diffusion in a clique). We confirm this intuition in Figure 6 on the sparse network representing co-authorship in the network science community [22]. We create a *dense* version of this graph via the same procedure $\mathbf{A}_k = \sum_{\ell=0}^{k} \mathbf{A}^\ell$, for $k = 5$. The figure shows that there is hardly any difference between the regularized diffusions in either graph (compare (c) and (d)), yet the diffusion on the dense graph (b) shows relatively smaller values on the avoidable errors than the sparse graph (a). Thus, using a dense graph construction has the effect of naturally correcting mistakes, whereas these need to be explicitly regularized away in sparse graphs.



(a) Zhou on sparse (b) Zhou on dense (c) Zhou+Push on sparse (d) Zhou+Push on dense

Figure 6: We artificially densify this graph to $\mathbf{A_k}$ based to compare sparse and dense diffusions and regularization. The color indicates the magnitude of the diffusion from the circled nodes. The unavoidable errors are caused by a mislabeled node. This example illustrates how regularizing diffusions on dense graphs produces only a small effect (b vs. d), whereas it has a big effect on sparse graphs (a vs. c). Also, the regularization is more immune to density (c vs. d).

Technically, sparse graphs exhibit both localized eigenvectors [11] and other localized rank structure [9]. When we regularize in the sparse case, we explicitly regularize and bias the results towards the localization in the eigenvectors [19, 11]. Using dense versions of the graph exhibits a regularization that biases the results towards this same eigenvector localization, but implicitly instead of explicitly. There are additional connections between the type of densification we use and many of the ideas in the theory of data manifolds.

Although we have a few ideas about why densifying helps, this is an exciting area for further exploration. In particular, spectral diffusions already implicitly densify the graph through their algorithmic procedures. One question we plan to study in the future is why we see improved performance via the explicit densification. This must have a statistical explanation that parallels our algorithmic procedure.

6. REFERENCES

[1] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using PageRank vectors. In *FOCS*, 2006.

[2] R. Andersen and K. Lang. An algorithm for improving graph partitions. In *SODA*, 2008.

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

[4] M. Belkin and P. Niyogi. Semi-supervised learning on riemannian manifolds. *Mach. Learn.*, 56(1-3):209–239, 2004.

[5] P. Berkhin. Bookmark-coloring algorithm for personalized PageRank computing. *Internet Math.*, 3(1):41–62, 2007.

[6] F. R. L. Chung. *Spectral graph theory*, volume 92. American Mathematical Society, 1997.

[7] R. R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006.

[8] C. A. R. de Sousa, S. O. Rezende, and G. E. Batista. Influence of graph construction on semi-supervised learning. In *ECML/PKDD*, pages 160–175. Springer, 2013.

[9] A. Gittens and M. W. Mahoney. Revisiting the Nyström method for improved large-scale machine learning. In *ICML*, volume 28, pages 567–575, 2013.

[10] D. F. Gleich and M. M. Mahoney. Algorithmic anti-differentiation: A case study with min-cuts, spectral, and flow. In *ICML*, 2014.

[11] T. J. Hansen and M. W. Mahoney. Semi-supervised eigenvectors for locally-biased learning. In *NIPS*, 2012.

[12] G. Jeh and J. Widom. Scaling personalized web search. In *WWW*, pages 271–279. ACM, 2003.

[13] L. G. S. Jeub, P. Balachandran, M. A. Porter, P. J. Mucha, and M. W. Mahoney. Think locally, act locally: Detection of small, medium-sized, and large communities in large networks. *Physical Review E*, 91:012821, 2015.

[14] T. Joachims. Transductive learning via spectral graph partitioning. In *ICML*, pages 290–297, 2003.

[15] K. Kloster and D. F. Gleich. Heat kernel based community detection. In *KDD*, 2014.

[16] I. M. Kloumann and J. M. Kleinberg. Community membership identification from small seed sets. In *KDD*, pages 1366–1375, 2014.

[17] J. Leskovec. Supporting website. `snap.stanford.edu/data`.

[18] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, September 2009.

[19] 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. *JMLR*, 13:2339–2365, 2012.

[20] F. McSherry. A uniform approach to accelerated PageRank computation. In *WWW*, pages 575–582. ACM, 2005.

[21] M. Newman. Network data. `www-personal.umich.edu/~mejn/netdata`.

[22] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74(3):036104, September 2006.

[23] L. Orecchia and M. W. Mahoney. Implementing regularization implicitly via approximate eigenvector computation. In *ICML*, pages 121–128, 2011.

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

[25] H. Shin, N. J. Hill, A. M. Lisewski, and J.-S. Park. Graph sharpening. *Expert. Syst. Appl.*, 37(12):7870–7879, 2010.

[26] D. A. Spielman and S.-H. Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM J. Comput.*, 42:1–26, 2013.

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

[28] A. L. Traud, P. J. Mucha, and M. A. Porter. Social structure of Facebook networks. *arXiv*, cs.SI:1102.2166, 2011.

[29] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. In *ICDM*, 2012.

[30] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *NIPS*, pages 1601–1608, 2004.

[31] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, 2003.

[32] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.

Acknowledgements. Gleich is supported by NSF CAREER CCF-1149756; Mahoney is partially supported from the Army Research Office and the DARPA GRAPHS program