How to learn a graph from smooth signals

Vassilis Kalofolias
Signal Processing Laboratory 2 (LTS2)
Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland

Abstract

We propose a framework that learns the graph structure underlying a set of smooth signals. Given $X \in \mathbb{R}^{m \times n}$ whose rows reside on the vertices of an unknown graph, we learn the edge weights $w \in \mathbb{R}_+^{m(m-1)/2}$ under the smoothness assumption that $\mathrm{tr}\left(X^\top L X\right)$ is small. We show that the problem is a weighted ℓ-1 minimization that leads to naturally sparse solutions. We point out how known graph learning or construction techniques fall within our framework and propose a new model that performs better than the state of the art in many settings. We present efficient, scalable primal-dual based algorithms for both our model and the previous state of the art, and evaluate their performance on artificial and real data.

1 INTRODUCTION

We consider a matrix $X \in \mathbb{R}^{m \times n} = [x_1, \ldots, x_m]^\top$, where each row $x_i \in \mathbb{R}^n$ resides on one of m nodes of a graph G. Then each of the n columns of X can be seen as a signal on the same graph. A simple assumption about data residing on graphs, but also the most widely used one is that it changes smoothly between connected nodes. An easy way to quantify how smooth is a set of vectors $x_1, \ldots, x_m \in \mathbb{R}^n$ on a given weighted undirected graph is through the function

$$\frac{1}{2} \sum_{i,j} W_{ij} \|x_i - x_j\|^2,$$

where W_{ij} denotes the weight of the edge between nodes i and j. In words, if two vectors x_i and x_j from <u>a smooth set</u> reside on two well connected nodes (so

W_{ij} is big), they are expected to have a small distance. Using the graph Laplacian matrix $L = D - W$, where D is the diagonal degree matrix with $D_{ii} = \sum_j W{ij}$, this function can be written in matrix form as

$$\mathrm{tr}\left(X^\top L X\right).$$

The importance of the graph Laplacian has long been known as a tool for embedding, manifold learning, clustering and semisupervised learning, see e.g. Belkin and Niyogi (2001), Belkin, Niyogi, and Sindhwani (2006), and Zhu, Ghahramani, Lafferty, et al. (2003). More recently we find an abundance of methods that exploit this notion of smoothness to regularize various machine learning tasks, solving problems of the form

$$\underset{X}{\text{minimize}} \quad g(X) + \mathrm{tr}\left(X^\top L X\right). \tag{1}$$

Zhang, Popescul, and Dom (2006) use it to enhance web page categorization with graph information, Zheng et al. (2011) for graph regularized sparse coding. Cai et al. (2011) use the same term to regularize NMF, Jiang et al. (2013) for PCA and Kalofolias et al. (2014) for matrix completion. Having good quality graphs is key to the success of the above methods.

The goal of this paper is to solve the complementary problem of learning a good graph:

$$\underset{L \in \mathcal{L}}{\text{minimize}} \quad \mathrm{tr}\left(X^\top L X\right) + f(L), \tag{2}$$

where $\mathcal{L}$ denotes the set of valid graph Laplacians.

Why is this problem important? Firstly because it enables us to directly learn the hidden graph structure behind our data. Secondly because in most problems that can be written in the form of eq. (1), we are often given a noisy graph, or no graph at all. Therefore, starting from the initial graph and alternating between solving problems (1) and (2) we can at the same time get a better quality graph and solve the task of the initial problem.

Related Work. Dempster (1972) was one of the first to propose the problem of finding connectivity from

measurements, under the name "covariance selection". Years later, Banerjee, El Ghaoui, and d'Aspremont (2008) proposed solving an ℓ-1 penalized log-likelihood problem to estimate a sparse inverse covariance with unknown pattern of zeros. However, while a lot of work has been done on inverse covariance estimation, the latter differs substantially from a graph Laplacian. For instance, the off-diagonal elements of a Laplacian must be non-positive, while it is not invertible like the inverse covariance.

Wang and Zhang (2008) learn a graph with normalized degrees by minimizing the objective $\sum_i \|x_i - \sum_j w_{ij} x_j\|^2$, but they assume a fixed k-NN edge pattern. Daitch, Kelner, and Spielman (2009) considered the similar objective $\|LX\|_F^2$ and they approximately minimized it with a greedy algorithm and a relaxation.

Zhang et al. (2010) alternate between problems (1) and a variation of (2). However, while they start from an initial graph Laplacian L, they finally learn a s.p.s.d. matrix that is not necessarily a valid Laplacian.

The works most relevant to ours are the ones by Lake and Tenenbaum (2010) and by Dong et al. (2015a,b). In the first one, the authors consider a problem similar to the one of the inverse covariance estimation, but impose additional constraints in order to obtain a valid Laplacian. However, their final objective function contains many constraints and a computationally demanding log-determinant term that makes it difficult to solve. To the best of our knowledge, there is no scalable algorithm in the literature to solve their model. Dong et al. (2015b) propose a model that outperforms the one by Lake and Tenenbaum, but still do not provide a scalable algorithm. This work is complementary to theirs, as we not only compare against their model, but also provide an analysis and a scalable algorithm to solve it.

Contributions. In this paper we make the link between smoothness and sparsity. We show that the smoothness term can be equivalently seen as a weighted ℓ-1 norm of the adjacency matrix, and minimizing it leads to naturally sparse graphs (Section 2). Based on this, we formulate our objective as a weighted ℓ-1 problem that we propose as a general framework for solving problem (2). Using this framework we propose a new model for learning a graph. We prove that our model has effectively one parameter that controls how sparse is the learnt graph (Section 4).

We show how our framework includes the standard Gaussian kernel weight construction, but also the model by Dong et al. (2015b). We simplify their model and prove fundamental properties (Section 4).

We provide a fast, scalable and convergent primal-dual algorithm to solve our proposed model, but also the one by Dong et al. To the best of our knowledge, these are the first scalable solutions in the literature to learn a graph under smoothness assumption (2) (Section 5).

To evaluate our model, we first review different definitions of smooth signals in the literature. We show how they can be unified under the notion of graph filtering (Section 3). We compare the models under artificial and real data settings. We conclude that our model is superior in many cases and achieves better connectivity when sparse graphs are sought (Section 6).

2 PROPERTIES OF THE LAPLACIAN

Throughout this paper we use the combinatorial graph Laplacian defined as $L = D - W$, where $D = \operatorname{diag}(W\mathbf{1})$ and $\mathbf{1} = [1, \ldots, 1]^\top$. The space of all valid combinatorial graph Laplacians, is by definition

$$\mathcal{L} = \Big\{ L \in \mathbb{R}^{m \times m} :$$
$$(\forall i \neq j) \quad L_{ij} = L_{ji} \leq 0, \quad L_{ii} = -\sum_{j \neq i} L_{ij} \Big\}.$$

In order to learn a valid graph Laplacian, we might be tempted to search in the above space, as is done e.g. by Dong et al. (2015b) and Lake and Tenenbaum (2010). We argue that it is more intuitive to search for a valid weighted adjacency matrix W from the space

$$\mathcal{W}_m = \left\{ W \in \mathbb{R}_+^{m \times m} : \quad W = W^\top, \quad \operatorname{diag}(W) = 0 \right\},$$

leading to simplified problems. Even more, when it comes to actually solving the problem by optimization techniques, we should consider the space of all valid edge weights for a graph

$$\mathcal{W}_v = \left\{ w \in \mathbb{R}_+^{m(m-1)/2} \right\},$$

so that we do not have to deal with the symmetricity of W explicitly. The spaces $\mathcal{L}$, $\mathcal{W}_m$ and $\mathcal{W}_v$ are equivalent, and connected by bijective linear mappings. In this paper we use $\mathcal{W}_m$ to analyze the problem in hand and $\mathcal{W}_v$ when we solve the problem. Table 1 exhibits some of the equivalent forms in the three spaces.

2.1 Smooth manifold means graph sparsity

Let us define the *pairwise distances matrix* $Z \in \mathbb{R}_+^{m \times m}$:

$$Z_{i,j} = \|x_i - x_j\|^2.$$

Using this, we can rewrite the trace term as

$$\operatorname{tr}\left(X^\top L X\right) = \frac{1}{2} \operatorname{tr}\left(WZ\right) = \frac{1}{2} \|W \circ Z\|_{1,1}, \qquad (3)$$

Table 1: Equivalent terms for representations from sets $\mathcal{L}, \mathcal{W}_m, \mathcal{W}_v$. We use $z = \text{vectorform}(Z)$, and linear operator S that performs summation in the vector form.

$L \in \mathcal{L}$	$W \in \mathcal{W}_m$	$w \in \mathcal{W}_v$
$2\operatorname{tr}\left(X^\top L X\right)$	$\|W \circ Z\|_{1,1}$	$2w^\top z$
$\operatorname{tr}(L)$	$\|W\|_{1,1}$	$2w^\top \mathbf{1} = 2\|w\|_1$
$-$	$\|W\|_F^2$	$2\|w\|_2^2$
$\operatorname{diag}(L)$	$W\mathbf{1}$	Sw
$\mathbf{1}^\top \log(\operatorname{diag}(L))$	$\mathbf{1}^\top \log(W\mathbf{1})$	$\mathbf{1}^\top \log(Sw)$
$\|L\|_F^2$	$\|W\|_F^2 + \|W\mathbf{1}\|_2^2$	$2\|w\|_2^2 + \|Sw\|_2^2$

where $\|A\|_{1,1}$ is the elementwise norm-1 of A and $\circ$ is the Hadamard product (see Appendix). In words, *the smoothness term is a weighted ℓ-1 norm of W*, encoding *weighted sparsity*, that penalizes edges connecting distant rows of X. The interpretation is that when the given distances come from a smooth manifold, the corresponding graph has a sparse set of edges, preferring only the ones associated to small distances in Z.

Explicitly adding a sparsity term $\gamma\|W\|_{1,1}$ to the objective function is a common tactic for inverse covariance estimation. However, it brings little to our problem, as here it can be translated as merely adding a constant to the squared distances in Z:

$$\operatorname{tr}\left(X^\top L X\right) + \gamma\|W\|_{1,1} = \frac{1}{2}\|W \circ (2\gamma + Z)\|_{1,1}. \quad (4)$$

Note that all information of X conveyed by the trace term is contained in the pairwise distances matrix Z, so that the original could be omitted. Moreover, using the last term of eq. (3) instead of the trace enables us to define other kinds of distances instead of Euclidean.

Note finally that the separate rows of X do not have to be smooth signals in some sense. Two non-smooth signals x_i, x_j can have a small distance between them, and therefore a small entry $Z_{i,j}$.

3 WHAT IS A SMOOTH SIGNAL?

Given a graph, different definitions of what is a smooth signal have been used in different contexts. In this section we unify these different definitions using the notion of filtering on graphs. For more information about signal processing on graphs we refer to the work of Shuman et al. (2013). Filtering of a graph signal $x \in \mathbb{R}^m$ by a filter $h(\lambda)$ is defined as the operation[1]

$$y = h(L)x = \sum_i u_i h(\lambda_i) u_i^\top x = \sum_i u_i h(\lambda_i)\hat{x}_i, \quad (5)$$

where $\{u_i, \lambda_i\}$ are eigenvector-eigenvalue pairs of L, and $\hat{x} \in \mathbb{R}^m$ is the graph Fourier representation of x

---

[1]We denote by h both the function $h : \mathbb{R} \to \mathbb{R}$ and its matrix counterpart $h : \mathbb{R}^{m \times m} \to \mathbb{R}^{m \times m}$ acting on the matrix's eigenvalues.

containing its *graph frequencies* $\hat{x}_i \in \mathbb{R}$. Low frequencies correspond to small eigenvalues, and low-pass or smooth filters correspond to decaying functions h.

In the sequel we show how different models for smooth signals in the literature can be written as smoothing problems of an initial non-smooth signal. We give an example of three different filters applied on the same signal in Figure 4 (Appendix).

Smooth signals by Tikhonov regularization.
Solving problem (1) leads to smooth signals. By setting $g(x) = \frac{1}{\alpha}\|x - x_0\|^2$ we have a Tikhonov regularization problem, that given an arbitrary x_0 as input gives its graph-smooth version $x = (\alpha L + I)^{-1}x_0$. Equivalently, we can see this as filtering x_0 by

$$h(\lambda) = \frac{1}{1 + \alpha\lambda}, \quad (6)$$

where big α values result in smoother signals.

Smooth signals from a probabilistic generative model.
Dong et al. (2015a) proposed that smooth signals can be generated from a colored Gaussian distribution as $x = \bar{x} + \sum_i u_i\hat{x}_i$, where $\hat{x}_i \sim \mathcal{N}\left(0, \lambda_i^\dagger\right)$ and $\dagger$ denotes the pseudoinverse. Therefore x follows the distribution

$$x \sim \mathcal{N}\left(\bar{x}, L^\dagger\right).$$

To sample from the above, it suffices to draw an initial non-smooth signal $x_0 \sim \mathcal{N}(0, I)$ and then compute

$$x = \bar{x} + h(L)x_0, \quad (7)$$

with $h(L) = \sqrt{L^\dagger}$, or equivalently filter it by

$$h(\lambda) = \begin{cases} \sqrt{\lambda^{-1}} & , \lambda > 0 \\ 0 & , \lambda = 0 \end{cases} \quad (8)$$

and add the mean $\bar{x}$. We point out here that using eq. (7) on any $x_0 \sim \mathcal{N}(0, I)$ and for any filter $h(\lambda)$ would yield samples from

$$x \sim \mathcal{N}\left(\bar{x}, h(L)^2\right),$$

therefore the probabilistic generative model can be used for any filter h. However, it does not cover cases where the initial x_0 is not white Gaussian.

Smooth signals by heat diffusion on graphs
Another type of smooth signals in the literature results from the process of heat diffusion on graphs. See for example the work by Zhang and Hancock (2008) for an application on image denoising by heat diffusion smoothing on the pixels graph. Given an initial signal x_0, the result of the heat diffusion on a graph after time t is $x = \exp(-Lt)x_0$, therefore the corresponding filter is

$$h(\lambda) = \exp(-t\lambda), \quad (9)$$

where bigger values of t result in smoother signals.

4 LEARNING A GRAPH FROM SMOOTH SIGNALS

In order to learn a graph from smooth signals, we propose, as explained in Section 2, to rewrite problem (2) using the weighted adjacency matrix W and the pairwise distance matrix Z instead of X:

$$\underset{W \in \mathcal{W}_m}{\text{minimize}} \quad \|W \circ Z\|_{1,1} \; + \; f(W). \qquad (10)$$

Since W is positive we could replace the first term by $\operatorname{tr}(WZ)$, but we prefer this notation to keep in mind that our problem already has a sparsity term on W. *This means that $f(W)$ has to play two important roles: (1) prevent W from going to the trivial solution $W = 0$ and (2) impose further structure using prior information on W.* This said, depending on f the solution is expected to be sparse, that is important for large scale applications.

In order to motivate this general graph learning framework, we show that the most standard weight construction, as well as the state of the art graph learning model are special cases thereof.

4.1 Classic Laplacian computations

In the literature one of the most common practices is to construct edge weights given X from the Gaussian function

$$w_{ij} = \exp\left(-\frac{\|x_i - x_j\|_2^2}{2\sigma^2}\right). \qquad (11)$$

It turns out that this choice of weights can be seen as the result of solving problem (10) with a specific prior on the weights W:

Proposition 1. *The solution of the problem*

$$\underset{W \in \mathcal{W}_m}{\text{minimize}} \quad \|W \circ Z\|_{1,1} \; + \; 2\sigma^2 \sum_{ij} W_{ij} \left(\log(W_{ij}) - 1\right)$$

is given by eq. (11).

Proof. The problem is edge separable and the objective can be written as $\sum_{i,j} W_{ij}Z_{ij} + 2\sigma^2 W_{ij}(\log(W_{ij} - 1))$. Deriving w.r.t. W_{ij} we obtain the optimality condition $Z_{ij} + 2\sigma^2 \log(W_{ij}) = 0$, or $W_{ij} = \exp(-Z_{ij}/(2\sigma^2))$, that proves the proposition. $\square$

Note that here, the logarithm in f prevents the weights from going to 0, leading to full matrices, and sparsification has to be imposed explicitly afterwards.

4.2 Our proposed model

Based on our framework (10) our goal is to give a general purpose model for learning graphs, when no prior information is available. In order to obtain meaningful

graphs, we want to make sure that *each node has at least one edge with another node.* It is also desirable to *have control of how sparse is the resulting graph.* To meet these expectations, we propose the following model with parameters $\alpha > 0$ and $\beta \geq 0$ controlling the shape of the edges:

$$\underset{W \in \mathcal{W}_m}{\text{minimize}} \; \|W \circ Z\|_{1,1} - \alpha \mathbf{1}^\top \log(W\mathbf{1}) + \beta \|W\|_F^2. \quad (12)$$

The logarithmic barrier acts on the node degree vector $W\mathbf{1}$, unlike the model of Proposition 1 that has a similar barrier on the edges. This means that it forces the degrees to be positive, but does not prevent edges from becoming zero. This improves the overall connectivity of the graph, without compromising sparsity. Note however, that adding solely a logarithmic term ($\beta = 0$) leads to very sparse graphs, and changing α only changes the scale of the solution and not the sparsity pattern (Proposition 2 for $\beta = 0$). For this reason, we add the third term.

We showed in eq. (4) that adding an ℓ-1 norm to control sparsity is not very useful. On the other hand, adding a Frobenius norm we penalize the formation of big edges but do not penalize smaller ones. This leads to more dense edge patterns for bigger values of β. An interesting property of our model is that even if it has two terms shaping the weights, if we fix the scale we then need to search for only one parameter:

Proposition 2. *Let $F(Z, \alpha, \beta)$ denote the solution of our model (12) for input distances Z and parameters α, β. Then the following property holds for any $\gamma > 0$:*

$$F(Z, \alpha, \beta) = \gamma F\left(Z, \frac{\alpha}{\gamma}, \beta\gamma\right) = \alpha F(Z, 1, \alpha\beta). \quad (13)$$

Proof. See appendix. $\square$

This means that for example if we want to obtain a W with a fixed scale $\|W\| = s$ (for any norm), we can solve the problem with $\alpha = 1$, search only for a parameter β that gives the desired edge density and then normalize the graph by the norm we have chosen.

The main advantage of our model over the method by Dong et al. (2015a), is that it promotes connectivity by putting a log barrier directly on the node degrees. Even for $\beta = 0$, we obtain the sparsest solution possible, that assigns at least one edge to each node. In this case, the distant nodes will have smaller degrees (because of the first term), but still be connected to their closest neighbour similarly to a 1-NN graph.

4.3 Fitting the state of the art in our framework

Dong et al. (2015a) proposed the following model for learning a graph:

$$\underset{L \in \mathcal{L}}{\text{minimize}} \quad \text{tr}\left(X^\top L X\right) + \alpha \|L\|_F^2,$$

$$\text{s.t.,} \qquad \text{tr}\left(L\right) = s.$$

Parameter $s > 0$ controls the scale (Dong et al. set it to m), and parameter $\alpha \geq 0$ controls the density of the solution. This formulation has two weaknesses. First, using a Frobenius norm on the Laplacian has a reduced interpretability: the elements of L are not only of different scales, but also linearly dependent. Secondly, optimizing it is difficult as it has 4 constraints on L: 3 in order to constrain L in space $\mathcal{L}$, and one to keep the trace constant. We propose to solve their model using our framework: Using transformations of Table 1, we obtain the equivalent simplified model

$$\underset{W \in \mathcal{W}_m}{\text{minimize}} \quad \|W \circ Z\|_{1,1} + \alpha \|W\mathbf{1}\|^2 + \alpha \|W\|_F^2,$$

$$\text{s.t.,} \qquad \|W\|_{1,1} = s. \tag{14}$$

Using this parametrization, solving the problem becomes much simpler, as we show in Section 5. Note that for $\alpha = 0$ we have a linear program that assigns weight s to the edge corresponding to the smallest pairwise distance in Z, and zero everywhere else. On the other hand, setting α to big values, we penalize big degrees (through the second term), and in the limit $\alpha \to \infty$ we obtain a dense graph with constant degrees across nodes. We can also prove some interesting properties of (14):

Proposition 3. *Let $H(Z, \alpha, s)$ denote the solution of model (14) for input distances Z and parameters α and s. Then for $\gamma > 0$ the following properties hold:*

$$H(Z + \gamma, \alpha, s) = H(Z, \alpha, s) \tag{15}$$

$$H(Z, \alpha, s) = \gamma H\left(Z, \alpha\gamma, \frac{s}{\gamma}\right) = s H\left(Z, \alpha s, 1\right) \tag{16}$$

Proof. See appendix. □

In other words, model (14) is invariant to adding any constant to the squared distances. The second property means that similarly to our model, the scale of the solution does not change the shape of the connectivity. If we fix the scale to s, we obtain the whole range of edge shapes given by H only by changing parameter α.

5 OPTIMIZATION

An advantage of using the formulation of problem (10) is that it can be solved efficiently for a wide range of choices of $f(W)$. We use primal dual techniques that scale, like the ones reviewed by Komodakis and Pesquet (2014) to solve the two state of the art models: the one we propose and the one by Dong et al. (2015b). Using these as examples, it is easy to solve many interesting models from the general framework (10).

In order to make optimization easier, we use the vector form representation from space $\mathcal{W}_v$ (see Table 1), so that the symmetricity does not have to be imposed as a constraint. We write the problem as a sum of three functions in order to fit it to primal dual algorithms reviewed by Komodakis and Pesquet (2014). The general form of our objective is

$$\underset{w \in \mathcal{W}_v}{\text{minimize}} \quad f_1(w) + f_2(Kw) + f_3(w), \tag{17}$$

where f_1 and f_2 are functions for which we can efficiently compute proximal operators, and f_3 is differentiable with gradient that has Lipschitz constant $\zeta \in (0, \infty)$. K is a linear operator, so f_2 is defined on the dual variable Kw. In the sequel we explain how this general optimization framework can be applied to the two models of interest, leaving the details in the Appendix. For a better understanding of primal dual optimization or proximal splitting methods we refer the reader to the works of Combettes and Pesquet (2011) and Komodakis and Pesquet (2014).

In our model, the second term acts on the degrees of the nodes, that are a linear function of the edge weights. Therefore we use $K = S$, where S is the linear operator that satisfies $W\mathbf{1} = Sw$ if w is the vectorform of W. In the first term we group the positivity constraint of $\mathcal{W}_v$ and the weighted ℓ-1, and the second and third terms are the priors for the degrees and the edges respectively. In order to solve our model we define

$$f_1(w) = \mathbb{1}\{w \geq 0\} + 2w^\top z,$$
$$f_2(d) = -\alpha \mathbf{1}^\top \log(d),$$
$$f_3(w) = \beta \|w\|^2, \text{ with } \zeta = 2\beta,$$

where $\mathbb{1}\{\}$ is the indicator function that becomes zero when the condition in the brackets is satisfied, infinite otherwise. Note that the second function f_2 is defined on the dual variable $d = Sw \in \mathbb{R}^m$, that here is very conveniently the vector of the node degrees.

For model (14) we can define in a similar way

$$f_1(w) = \mathbb{1}\{w \geq 0\} + 2w^\top z,$$
$$f_2(c) = \mathbb{1}\{c = s\},$$
$$f_3(w) = \alpha \left(2\|w\|^2 + \|Sw\|^2\right), \text{ with } \zeta = 2\alpha(m+1),$$

and use $K = 2\mathbf{1}^\top$ so that the dual variable is $c = Kw = \|W\|_{1,1}$, constrained by f_2 to be equal to s.

Algorithm 1 Primal dual algorithm for model (12).

1: **Input:** $z, \alpha, \beta, w^0 \in \mathcal{W}_v, d^0 \in \mathbb{R}_+^m, \gamma$, tolerance ϵ
2: **for** $i = 1, \ldots, i_{max}$ **do**
3:      $y^i = w^i - \gamma(2\beta w^i + S^\top d^i)$
4:      $\bar{y}^i = d^i + \gamma(S w^i)$
5:      $p^i = \max(0, y^i - 2\gamma z)$
6:      $\bar{p}^i = (\bar{y}^i - \sqrt{(\bar{y}^i)^2 + 4\alpha\gamma})/2$ ▷ elementwise
7:      $q^i = p^i - \gamma(2\beta p^i + S^\top p^i)$
8:      $\bar{q}^i = \bar{p}^i + \gamma(S p^i)$
9:      $w^i = w^i - y^i + p^i$;
10:     $d^i = d^i - \bar{y}^i + \bar{q}^i$;
11:     **if** $\|w^i - w^{i-1}\|/\|w^{i-1}\| < \epsilon$ **and**
12:        $\|d^i - d^{i-1}\|/\|d^{i-1}\| < \epsilon$ **then**
13:        **break**
14:     **end if**
15: **end for**

Using these functions, the final algorithm for our model is given as Algorithm 1, and for the model by Dong et al. as Algorithm 2 in the Appendix. Vector $z \in \mathbb{R}_+^{m \times (m-1)/2}$ is the vector form of Z, and parameter $\gamma \in (0, \ 1+\zeta+\|K\|)$ is the stepsize.

5.1 Complexity and Convergence

Both algorithms that we propose have a complexity of $\mathcal{O}(m^2)$ per iteration, for m nodes graphs, and they can easily be parallelized. As the objective functions of both models are proper, convex, and lower-semicontinuous, our algorithms are guaranteed to converge to the minimum (Komodakis and Pesquet 2014).

6 EXPERIMENTS

We compare our model against the state of the art model by Dong et al. (2015a) solved by our Algorithm 2 for both artificial and real data. Comparing to the model by Lake and Tenenbaum (2010) was not possible even for the small graphs of our artificial experiments, as there is no scalable algorithm in the literature and the use of CVX with the log-determinant term is prohibitive. Other models based on the log-det term, for which scalable algorithms exist, are irrelevant to our problem as a sparse inverse covariance is not a valid Laplacian and are known to not perform well for our setting (see Dong et al. (2015a) for a comparison).

6.1 Artificial data

The difficulty of solving problem (10) depends both on the quality of the graph behind the data and on the type of smoothness of the signals. We test 4 different types of graphs using 3 different types of signals.

Graph Types. We use two 2-D manifold based graphs, one uniformly and one non-uniformly sampled, and two graphs that are not manifold structured:

1. Random Geometric Graph (RGG): We sample x uniformly from $[0,1]^2$ and connect nodes using eq. (11) with $\sigma = 0.2$, then threshold weights < 0.6.
2. Non-uniform: We sample x in $[0,1] \times [0,5]$ from a non-uniform distribution $p_{x_1,x_2} \propto 1/(1 + \alpha x_2)$ and connect nodes using eq. (11) with $\sigma = 0.2$. We threshold weights smaller than the best connection of the most distant node (≈ 0.01).
3. Erdős Rényi: Random graph as proposed by Gilbert (1959) ($p=3/m$).
4. Barabási-Albert: Random scale-free graph with preferential attachment as proposed by Barabási and Albert (1999) ($m_0=1$, $m=2$).

Signal Types. To create a smooth signal we filter a Gaussian i.i.d. x_0 by eq. (5), using one of the three filter types of Section 3. We normalize the Laplacian ($\|L\|_2 = 1$) so that the filters $g(\lambda)$ are defined for $\lambda \in [0,1]$. See Table 3 (Appendix) for a summary.

1. Tikhonov: $g(\lambda) = \frac{1}{1+10\lambda}$ as in eq. (6).
2. Generative Model: $g(\lambda) = 1/\sqrt{\lambda}$ if $\lambda > 0$, $g(0) = 0$ from model of eq. (8) ($\bar{x} = 0$).
3. Heat Diffusion: $g(\lambda) = \exp(-10\lambda)$ as eq. (9).

For all cases we use $m = 100$ nodes, smooth signals of length $n = 1000$, and add 10% (ℓ-2 sense) noise before computing pairwise distances. We perform grid search to find the best parameters for each model. We repeat the experiment 20 times for each case and report the average result of the parameter value that performs best for each of the different metrics.

Metrics. Since we have the ground truth graphs for each case, we can measure directly the relative edge error in the ℓ-1 and ℓ-2 sense. We also report the relative error of the weighted degrees $d_i = \sum_j W_{ij}$. This is important because both models are based on priors on the degrees as we show in section 4. We also report the F-measure (harmonic mean of edge precision and recall), that only takes into account the binary pattern of existing edges and not the weights.

Baselines. The baseline for the relative errors is a classic graph construction using equation (11) with a grid search for the best σ. Note that this exact equation was used to create the two first artificial datasets. However, using a fully connected graph with the F-measure does not make sense. For this metric the baseline is set to the best edge pattern found by thresholding (11) with different thresholds.

Table 2 summarizes all the results for different combinations of graphs/signals. In most of them, our model performs better for all metrics. We can see that the signals constructed following the generative model (7) do not yield better results in terms of graph reconstruction. Using smoother "Tikhonov" signals from eq. (6) or "Heat Diffusion" signals from (9) by set-

Table 2: Performance of Different Models on Artificial Data.

	Tikhonov			Generative Model			Heat Diffusion		
	base	Dong etal	Ours	base	Dong etal	Ours	base	Dong etal	Ours
Rand. Geometric									
F-measure	0.685	0.885	**0.913**	0.686	0.877	**0.909**	0.758	0.837	**0.849**
edge ℓ-1	0.866	0.357	**0.298**	0.798	0.371	**0.348**	0.609	0.524	**0.447**
edge ℓ-2	0.676	0.376	**0.336**	0.658	0.397	**0.390**	0.576	0.531	**0.468**
degree ℓ-1	0.142	0.146	**0.065**	0.261	0.147	**0.112**	0.209	0.227	**0.142**
degree ℓ-2	0.708	0.172	**0.079**	0.689	0.174	**0.128**	0.474	0.264	**0.176**
Non Uniform									
F-measure	0.686	**0.863**	0.858	0.633	**0.840**	0.832	0.766	**0.839**	0.830
edge ℓ-1	0.821	0.423	**0.349**	0.864	0.487	**0.472**	0.594	0.565	**0.473**
edge ℓ-2	0.706	0.434	**0.344**	0.735	0.480	**0.474**	0.550	0.587	**0.451**
degree ℓ-1	0.160	0.184	**0.055**	0.235	0.185	**0.100**	0.233	0.255	**0.128**
degree ℓ-2	0.612	0.209	**0.073**	0.632	0.215	**0.161**	0.427	0.324	**0.157**
Erdős Rényi									
F-measure	0.288	0.766	**0.893**	0.199	0.755	**0.896**	0.377	0.629	**0.655**
edge ℓ-1	1.465	0.448	**0.391**	1.566	0.478	**0.427**	1.379	**0.832**	0.841
edge ℓ-2	1.060	0.442	**0.402**	1.105	0.457	**0.440**	1.033	0.735	**0.726**
degree ℓ-1	0.094	0.107	**0.046**	0.099	0.105	**0.066**	0.182	**0.179**	0.183
degree ℓ-2	0.986	0.161	**0.066**	1.312	0.181	**0.151**	0.892	**0.236**	0.273
Barabási-Albert									
F-measure	0.345	0.710	**0.868**	0.382	0.739	**0.838**	0.352	0.690	**0.765**
edge ℓ-1	1.531	0.614	**0.533**	1.496	0.652	**0.624**	1.468	0.740	**0.675**
edge ℓ-2	1.061	0.568	**0.506**	1.036	0.611	**0.571**	1.041	0.662	**0.590**
degree ℓ-1	0.175	0.264	**0.111**	**0.199**	0.264	0.207	0.254	0.317	**0.148**
degree ℓ-2	0.554	0.340	**0.201**	0.556	0.333	**0.287**	0.568	0.414	**0.283**

ting $\lambda = 20$ yielded slightly worse results in both cases (not reported here). It also seems that the results are slightly better for the manifold related graphs than for the Erdős Rényi and Barabási-Albert models, an effect that is more prevalent when we use signals of length $n = 100$ smooth signals instead of 1000 (c.f. Table 4 of Appendix). This would be interesting to investigate theoretically.

6.2 Real data

We also evaluate the performance of our model on real data. In this case, the actual ground truth graph is not known. We therefore measure the performance of different models on spectral clustering and label propagation, two algorithms that depend solely on the graph. Note that an explicit Laplacian normalization is not needed for the learned models (it is even harmful as found experimentally), since this role is already played by the regularization.

Learning the graph of USPS digits

We first learn the graph connecting 1001 different images of the USPS dataset, that are images of digits from 0 to 9 (10 classes). We follow Zhu, Ghahramani, Lafferty, et al. (2003) and sample the class sizes non-uniformly. For each class $i \in \{1 \ldots 10\}$ we take round($2.6i^2$) images, resulting to classes with sizes from 3 to 260 images each. We learn graphs of different densities using both models. As baseline we use a k-Nearest Neighbors (k-NN) graph for different k.

For each of the graphs, we run standard spectral clustering (as in the work of Ng, Jordan, Weiss, et al. 2002 but without normalizing the Laplacian) with k-means 100 times. We also run label propagation we choose 100 times a different subset of 10% known labels.

In Fig. 1 we plot the behavior of different models for different density levels. The horizontal axis is the average number of non-zero edges per node. In the **left plot** we see the clustering quality. Even though the best result of both algorithms is almost the same (0.24 vs 0.25), our model is more robust in terms of the graph density choice. A similar behavior is exhibited for label propagation plotted in the **middle**. The classification quality is better for our model in the sparser graph density levels.

The robustness of our model for small graph densities can be explained by the connectivity quality plotted in the **right**. The continuous lines are the number of different connected components in the learned graphs, that is a measure of connectivity: the less components there are, the better connected is the graph. The dashed blue line is the number of disconnected nodes of model Dong et al. 2015a. The latter fails to assign connections to the most distant nodes, unless the density of the graph reaches a fairly high level. If we want a graph with 6 edges per node, our model returns a graph with 3 components and no disconnected nodes. The model by Dong et al. returns a graph with 35 components out of which 22 are disconnected nodes.
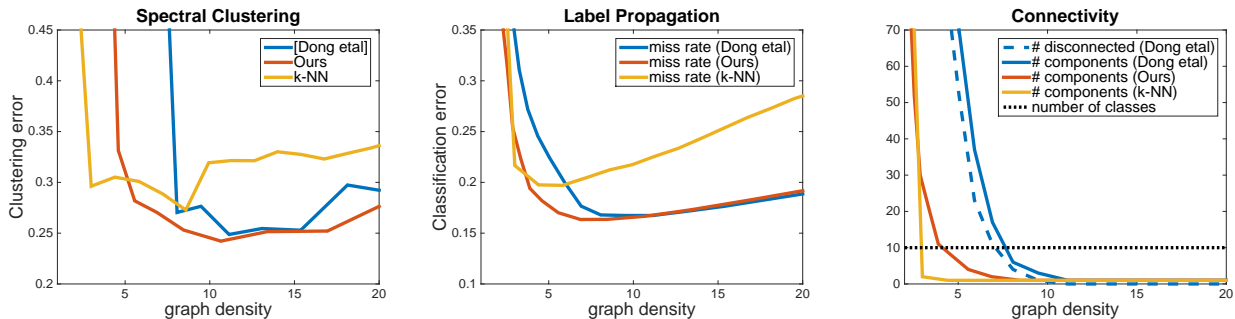
Figure 1: Graph learned from 1001 USPS images. **Left**: Clustering quality. **Middle**: Label propagation quality. **Right**: Number of completely disconnected components (continuous lines) and number of disconnected nodes for model by Dong etal. (blue dashed line). Our model and k-NN have no disconnected nodes.
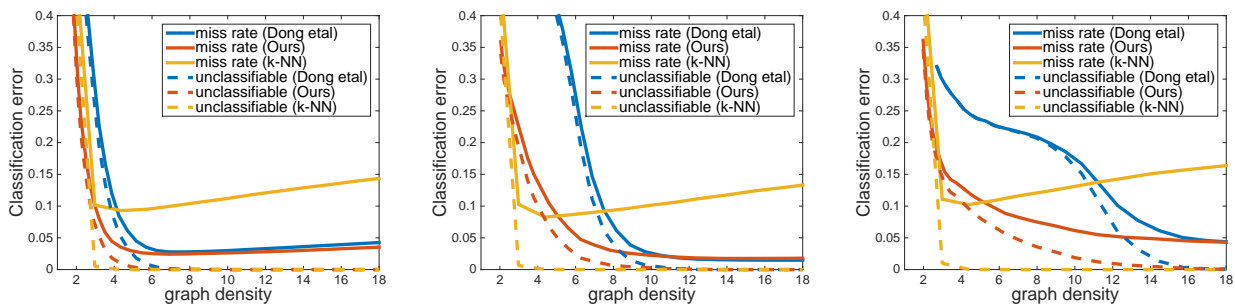

Figure 2: Label propagation for the problem "1" vs. "2" of MNIST with different class size proportions: 1 to 4 (**left**), 1 to 1 (**middle**) or 4 to 1 (**right**). Missclassification rate for different number of edges per node.

Note that in real applications where the best density level is not known a priori, it is important for a graph learning model to perform well for sparse levels. This is especially the case for large scale applications, where more edges mean more computations.

Time: Algorithm 1 implemented in Matlab[2] learned a 10-edge/node graph of 1001 USPS images in 5 seconds (218 iterations) and Algorithm 2 in 1 minute (2043 iterations) on a standard PC for tolerance $\epsilon = 1e\text{-}4$.

Learning the graph of MNIST 1 vs 2

To demonstrate the different behaviour of the two models for non-uniform sampling cases, we use the problem of classification between digits 1 and 2 of the MNIST dataset. This problem is particular because digits "1" are close to each other (average square distance of 45), while digits "2" differ more from each other (average square distance of 102). In Figure 2 we report the average miss-classification rate for different class size proportions, with 40 1's and 160 2's (**left**), 100 1's and 100 2's (**middle**) or 160 1's and 40 2's (**right**). Results are averaged over 40 random draws. The dashed lines denote the number of nodes contained in components without labeled nodes, that can not be classified. In this case, the model of Dong

et al. 2015a fails to recover edges between different digits "2" unless the returned graph is fairly dense, unlike our model that even for very sparse graph levels treats the different classes more fairly. The effect is stronger when the set of 2's is also the smallest of the two.

7 CONCLUSION

We introduce a new way of addressing the problem of learning a graph under the assumption that $\operatorname{tr}\left(X^{\top}LX\right)$ is small. We show how the problem can be simplified into a weighted sparsity problem, that implies a general framework for learning a graph. We show how the standard Gaussian weight construction from distances is a special case of this framework. We propose a new model for learning a graph, and provide an analysis of the state of the art model of Dong et al. (2015a) that also fits our framework. The new formulation enables us to propose a fast and scalable primal dual algorithm for our model, but also for the one of Dong et al. 2015a that was missing from the literature. Our experiments suggest that when sparse graphs are to be learned, but connectivity is crucial, our model is expected to outperform the current state of the art.

We hope not only that our solution will be used for many applications that require good quality graphs, but also that our framework will trigger defining new graph learning models targeting specific applications.

---

[2]Code for both models is available as part of the open-source toolbox GSPBox by Perraudin et al. (2014b) using code from UNLocBoX, Perraudin et al. (2014a).
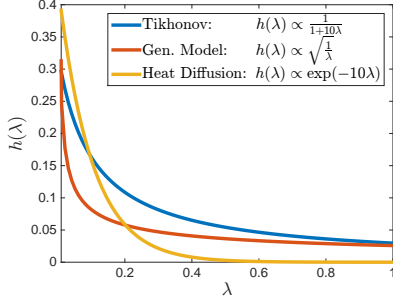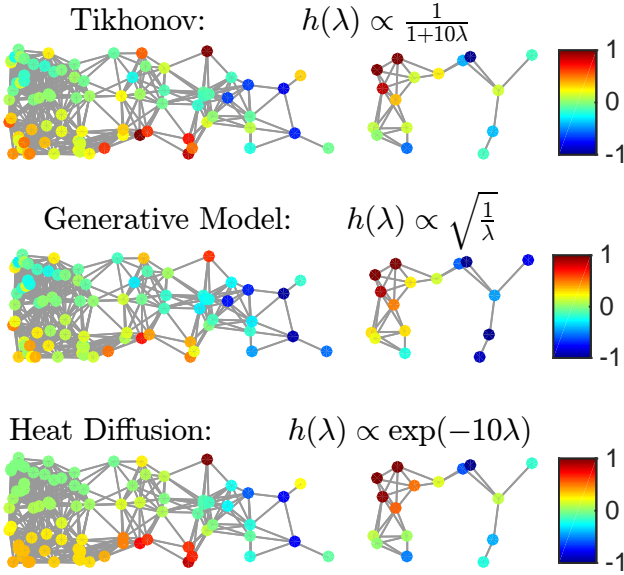
References

Banerjee, Onureena, Laurent El Ghaoui, and Alexandre d'Aspremont (2008). "Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data". In: *The Journal of Machine Learning Research* 9, pp. 485–516.

Barabási, Albert-László and Réka Albert (1999). "Emergence of scaling in random networks". In: *Science* 286.5439, pp. 509–512.

Belkin, Mikhail and Partha Niyogi (2001). "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering." In: *NIPS*. Vol. 14, pp. 585–591.

Belkin, Mikhail, Partha Niyogi, and Vikas Sindhwani (2006). "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples". In: *The Journal of Machine Learning Research* 7, pp. 2399–2434.

Cai, Deng et al. (2011). "Graph regularized nonnegative matrix factorization for data representation". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 33.8, pp. 1548–1560.

Combettes, Patrick L and Jean-Christophe Pesquet (2011). "Proximal splitting methods in signal processing". In: *Fixed-point algorithms for inverse problems in science and engineering*. Springer, pp. 185–212.

Daitch, Samuel I, Jonathan A Kelner, and Daniel A Spielman (2009). "Fitting a graph to vector data". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, pp. 201–208.

Dempster, Arthur P (1972). "Covariance selection". In: *Biometrics*, pp. 157–175.

Dong, Xiaowen et al. (2015a). "Laplacian Matrix Learning for Smooth Graph Signal Representation". In: *Proceedings of IEEE ICASSP*.

– (2015b). "Learning Laplacian Matrix in Smooth Graph Signal Representations". In: *arXiv preprint arXiv:1406.7842v2*.

Gilbert, Edgar N (1959). "Random graphs". In: *The Annals of Mathematical Statistics*, pp. 1141–1144.

Jiang, Bo et al. (2013). "Graph-Laplacian PCA: Closed-form solution and robustness". In: *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, pp. 3492–3498.

Kalofolias, Vassilis et al. (2014). "Matrix completion on graphs". In: *arXiv preprint arXiv:1408.1717*.

Komodakis, Nikos and Jean-Christophe Pesquet (2014). "Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems". In: *arXiv preprint arXiv:1406.5429*.

Lake, Brenden and Joshua Tenenbaum (2010). "Discovering structure by learning sparse graph". In: *Proceedings of the 33rd Annual Cognitive Science Conference*. Citeseer.

Ng, Andrew Y, Michael I Jordan, Yair Weiss, et al. (2002). "On spectral clustering: Analysis and an algorithm". In: *Advances in neural information processing systems* 2, pp. 849–856.

Perraudin, N. et al. (2014a). "UNLocBoX A matlab convex optimization toolbox using proximal splitting methods". In: *ArXiv e-prints*. arXiv: `1402.0779`.

Perraudin, Nathanaël et al. (2014b). "GSPBOX: A toolbox for signal processing on graphs". In: *ArXiv e-prints*. arXiv: `1408.5781 [cs.IT]`.

Shuman, David et al. (2013). "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *Signal Processing Magazine, IEEE* 30.3, pp. 83–98.

Wang, Fei and Changshui Zhang (2008). "Label propagation through linear neighborhoods". In: *Knowledge and Data Engineering, IEEE Transactions on* 20.1, pp. 55–67.

Zhang, Fan and Edwin R Hancock (2008). "Graph spectral image smoothing using the heat kernel". In: *Pattern Recognition* 41.11, pp. 3328–3342.

Zhang, Tong, Alexandrin Popescul, and Byron Dom (2006). "Linear prediction models with graph regularization for web-page categorization". In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 821–826.

Zhang, Yan-Ming et al. (2010). "Transductive Learning on Adaptive Graphs." In: *AAAI*.

Zheng, Miao et al. (2011). "Graph regularized sparse coding for image representation". In: *Image Processing, IEEE Transactions on* 20.5, pp. 1327–1336.

Zhu, Xiaojin, Zoubin Ghahramani, John Lafferty, et al. (2003). "Semi-supervised learning using gaussian fields and harmonic functions". In: *ICML*. Vol. 3, pp. 912–919.

Figure 3: The filters of Table 3 for $\alpha = 10$.

Tikhonov: $\qquad h(\lambda) \propto \frac{1}{1+10\lambda}$

Generative Model: $\qquad h(\lambda) \propto \sqrt{\frac{1}{\lambda}}$

Heat Diffusion: $\qquad h(\lambda) \propto \exp(-10\lambda)$

Figure 4: Different smooth signals on the Non Uniform graph used for our artificial data experiments. All signals are obtained by smoothing the same initial $x_0 \sim \mathcal{N}(0, I)$ with three different filters. This instance of the graph is disconnected with 2 components.

A Derivations and proofs

A.1 Detailed explanation of eq. (3)

$$
\begin{aligned}
\|W \circ Z\|_1 &= \sum_{i=1}^{m}\sum_{j=1}^{m} W_{ij}\|x_i - x_j\|_2^2 \\
&= \sum_{i=1}^{m}\sum_{j=1}^{m}(x_i - x_j)^\top W_{ij}(x_i - x_j) \\
&= 2\sum_{i=1}^{m}\sum_{j=1}^{m} x_i^\top W_{ij} x_i - 2\sum_{i=1}^{m}\sum_{j=1}^{m} x_i^\top W_{ij} x_j \\
&= 2\sum_{i=1}^{m} x_i^\top x_i \sum_{j=1}^{m} W_{ij} - 2\operatorname{tr}\left(X^\top W X\right) \\
&= 2\operatorname{tr}\left(X^\top D X\right) - 2\operatorname{tr}\left(X^\top W X\right) \\
&= 2\operatorname{tr}\left(X^\top L X\right),
\end{aligned}
$$

where D is the diagonal matrix with elements $D_{ii} = \sum_i W_{ij}$.

A.2 Proof of proposition 2

Proof. We change variable $\tilde{W} = W/\gamma$ to obtain

$$
\begin{aligned}
F(Z, \alpha, \beta) &= \\
&= \gamma\operatorname*{argmin}_{\tilde{W}} \|\gamma\tilde{W} \circ Z\|_{1,1} - \alpha \mathbf{1}^\top \log(\gamma\tilde{W}\mathbf{1}) + \beta\|\gamma\tilde{W}\|_F^2 \\
&= \gamma\operatorname*{argmin}_{\tilde{W}} \gamma\|\tilde{W} \circ Z\|_{1,1} - \alpha \mathbf{1}^\top \log(\tilde{W}\mathbf{1}) + \beta\gamma^2\|\tilde{W}\|_F^2 \\
&= \gamma\operatorname*{argmin}_{\tilde{W}} \|\tilde{W} \circ Z\|_{1,1} - \frac{\alpha}{\gamma} \mathbf{1}^\top \log(\tilde{W}\mathbf{1}) + \beta\gamma\|\tilde{W}\|_F^2 \\
&= \gamma F\left(Z, \frac{\alpha}{\gamma}, \beta\gamma\right),
\end{aligned}
$$

where we used the fact that $\log(\gamma\tilde{W}\mathbf{1}) = \log(\tilde{W}\mathbf{1}) + \mathrm{const.}(W)$. The second equality is obtained from the first one for $\gamma = \alpha$. $\qquad\square$

A.3 Proof of proposition 3

Proof. For equation (15)

$$
\begin{aligned}
H(Z + \gamma, \alpha, s) &= \\
&= \operatorname*{argmin}_{W \in \mathcal{W}_m} \|W \circ Z + \gamma W\|_{1,1} + \alpha\|W\|_F^2 + \alpha\|W\mathbf{1}\|^2 \\
&\qquad \text{s. t.,} \qquad \|W\|_{1,1} = s \\
&= \operatorname*{argmin}_{W \in \mathcal{W}_m} \|W \circ Z\|_{1,1} + \gamma\|W\|_{1,1} + \alpha\|W\|_F^2 + \alpha\|W\mathbf{1}\|^2 \\
&\qquad \text{s. t.,} \qquad \|W\|_{1,1} = s \\
&= \operatorname*{argmin}_{W \in \mathcal{W}_m} \|W \circ Z\|_{1,1} + \gamma s + \alpha\|W\|_F^2 + \alpha\|W\mathbf{1}\|^2 \\
&\qquad \text{s. t.,} \qquad \|W\|_{1,1} = s \\
&= H(Z, \alpha, s),
\end{aligned}
$$

because $\|a + b\|_1 = \|a\|_1 + \|b\|_1$ for positive a, b.

For equation (16) we change variable in the optimization and use $\tilde{W} = W/\gamma$ to obtain

$$
\begin{aligned}
H(Z, \alpha, s) &= \\
&= \gamma\operatorname*{argmin}_{\tilde{W} \in \mathcal{W}_m} \|\gamma\tilde{W} \circ Z\|_{1,1} + \alpha\|\gamma\tilde{W}\|_F^2 + \alpha\|\gamma\tilde{W}\mathbf{1}\|^2 \\
&\qquad \text{s. t.,} \qquad \|\gamma\tilde{W}\|_{1,1} = s \\
&= \gamma\operatorname*{argmin}_{\tilde{W} \in \mathcal{W}_m} \gamma\|\tilde{W} \circ Z\|_{1,1} + \gamma^2\alpha\|\tilde{W}\|_F^2 + \gamma^2\alpha\|\tilde{W}\mathbf{1}\|^2 \\
&\qquad \text{s. t.,} \qquad \|\tilde{W}\|_{1,1} = \frac{s}{\gamma} \\
&= \gamma\operatorname*{argmin}_{\tilde{W} \in \mathcal{W}_m} \|\tilde{W} \circ Z\|_{1,1} + \gamma\alpha\|\tilde{W}\|_F^2 + \gamma\alpha\|\tilde{W}\mathbf{1}\|^2 \\
&\qquad \text{s. t.,} \qquad \|\tilde{W}\|_{1,1} = \frac{s}{\gamma} \\
&= \gamma H\left(Z, \alpha\gamma, \frac{s}{\gamma}\right).
\end{aligned}
$$

Table 3: Different Types of Smooth Signals.

Concept	Model	Graph filter
Tikhonov	$X = \arg\min_X \frac{1}{2}\|X - X_0\|_F^2 + \frac{1}{\alpha}\operatorname{tr}\left(X^\top L X\right)$	$g(\lambda) = \frac{1}{1+\alpha\lambda}$
Generative model	$X \sim \mathcal{N}\left(0, L^\dagger\right)$	$g(\lambda) = \begin{cases} \frac{1}{\sqrt{\lambda}} & \text{if } \lambda > 0 \\ 0 & \text{if } \lambda = 0 \end{cases}$
Heat diffusion	$X = \exp\left(-\alpha L\right) X_0$	$g(\lambda) = \exp(-\alpha\lambda)$

The second equality follows trivially for $\gamma = s$.

$\square$

B Optimization details and algorithm for model of Dong et al. 2015a

To obtain Algorithm 1 (for our model), we need the following:

$$K = S \quad (\|S\|_2 = \sqrt{2(m-1)})$$
$$\operatorname{prox}_{\lambda f_1}(y) = \max(0, y - \lambda z),$$
$$\operatorname{prox}_{\lambda f_2}(y) = \frac{y_i + \sqrt{y_i^2 + 4\alpha\lambda}}{2},$$
$$\nabla f_3(w) = 2\beta w,$$
$$\zeta = 2\beta \qquad \text{(Lipschitz constant of gradient of } f_3\text{),}$$

where m is the number of nodes of the graph.

To obtain Algorithm 2 (for model by Dong et al. 2015a), we need the following:

$$K = \mathbf{21} \quad (\|\mathbf{21}\|_2 = 2\sqrt{m(m-1)/2})$$
$$\operatorname{prox}_{\lambda f_1}(y) = \max(0, y - \lambda z),$$
$$\operatorname{prox}_{\lambda f_2}(y) = s,$$
$$\nabla f_3(w) = \alpha(4w + 2S^\top S w),$$
$$\zeta = 2\alpha(m+1) \text{ (Lipschitz constant of gradient of } f_3\text{).}$$

C More real data experiments

C.1 Learning the graph of COIL 20 images

We randomly sample the classes so that the average size increases non-linearly from around 3 to around 60 samples per class. The distribution for one of the instances of this experiment is plotted in fig. 5. We sample from the same distribution 20 times and measure the average performance of the models for different graph densities. For each of the graphs, we run standard spectral clustering (as in the work of Ng, Jordan, Weiss, et al. 2002 but without normalizing the Laplacian) with k-means 100 times. For label propagation

---

Algorithm 2 Primal dual algorithm for model Dong et al. 2015a.

1: **Input:** $z, \alpha, s, w^0 \in \mathcal{W}_v, c^0 \in \mathbb{R}_+, \gamma$, tolerance ϵ
2: **for** $i = 1, \ldots, i_{max}$ **do**
3: $\quad y^i = w^i - \gamma(2\alpha(2w^i + S^\top S w^i) + 2c^i)$
4: $\quad \bar{y}^i = c^i + \gamma(2\sum_j w_j^i)$
5: $\quad p^i = \max(0, y^i - 2\gamma z)$
6: $\quad \bar{p}^i = \bar{y}^i - \gamma s$
7: $\quad q^i = p^i - \gamma(2\alpha(2p^i + S^\top S p^i) + 2p^i)$
8: $\quad \bar{q}^i = \bar{p}^i + \gamma(2\sum_j p_j^i)$
9: $\quad w^i = w^i - y^i + q^i;$
10: $\quad c^i = c^i - \bar{y}^i + \bar{q}^i;$
11: $\quad$ **if** $\|w^i - w^{i-1}\|/\|w^{i-1}\| < \epsilon$ **and**
12: $\quad\quad |c^i - c^{i-1}|/|c^{i-1}| < \epsilon$ **then**
13: $\quad\quad$ **break**
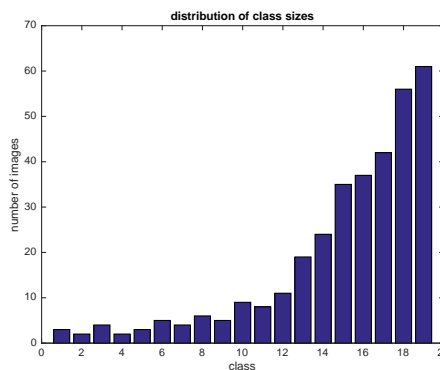14: $\quad$ **end if**
15: **end for**

Figure 5: Distribution of class sizes for one of the random instances of the COIL 20 experiments.

Table 4: Performance of Different Algorithms on Artificial Data. Each setting has a random graph with 100 nodes and **100** smooth signals from 3 different smoothness models and added 10% noise. Results averaged over 20 random graphs for each setting. F-measure: the bigger the better (weights ignored). Edge and degree distances: the lower the better. For relative $\ell-1$ distances we normalize s.t. $\|w\|_1 = \|w_0\|_1$. For relative $\ell-2$ distances we normalize s.t. $\|w\|_2 = \|w_0\|_2$. Baseline: for F-measure, the best result by thresholding $\exp(-d^2)$. For edge and degree distances we use $\exp(-d^2/2\sigma^2)$ without thresholding.

	Tikhonov			Generative Model			Heat Diffusion		
	base	Dong etal	Ours	base	Dong etal	Ours	base	Dong etal	Ours
Rand. Geometric									
F-measure	0.667	0.860	**0.886**	0.671	0.836	**0.858**	0.752	0.837	**0.848**
edge ℓ-1	0.896	0.414	**0.364**	0.851	0.487	**0.468**	0.620	0.526	**0.451**
edge ℓ-2	0.700	0.430	**0.390**	0.692	0.494	**0.477**	0.582	0.535	**0.471**
degree ℓ-1	0.158	0.151	**0.080**	0.268	0.159	**0.128**	0.216	0.225	**0.143**
degree ℓ-2	0.707	0.179	**0.095**	0.679	0.193	**0.145**	0.479	0.264	**0.177**
Non Uniform									
F-measure	0.674	**0.821**	0.817	0.650	**0.779**	0.774	0.763	**0.835**	0.827
edge ℓ-1	0.847	0.547	**0.480**	0.931	0.711	**0.673**	0.612	0.583	**0.491**
edge ℓ-2	0.724	0.545	**0.462**	0.784	0.673	**0.624**	0.565	0.598	**0.464**
degree ℓ-1	0.167	0.190	**0.075**	0.241	0.204	**0.139**	0.235	0.257	**0.132**
degree ℓ-2	0.605	0.228	**0.099**	0.614	0.261	**0.187**	0.433	0.325	**0.164**
Erdős Rényi									
F-measure	0.293	0.595	**0.676**	0.207	0.473	**0.512**	0.358	0.595	**0.619**
edge ℓ-1	1.513	0.837	**0.798**	1.623	1.113	**1.090**	1.401	**0.896**	0.899
edge ℓ-2	1.086	0.712	**0.697**	1.129	0.896	**0.888**	1.045	0.767	**0.759**
degree ℓ-1	0.114	0.129	**0.084**	0.135	0.146	**0.114**	0.185	**0.182**	0.184
degree ℓ-2	0.932	0.202	**0.116**	1.053	0.227	**0.185**	0.875	**0.241**	0.276
Barabási-Albert									
F-measure	0.325	0.564	**0.636**	0.357	0.588	**0.632**	0.349	0.631	**0.711**
edge ℓ-1	1.541	0.939	**0.885**	1.513	0.940	**0.914**	1.473	0.843	**0.774**
edge ℓ-2	1.073	0.802	**0.761**	1.052	0.808	**0.773**	1.049	0.732	**0.672**
degree ℓ-1	0.225	0.309	**0.145**	0.243	0.311	**0.229**	0.281	0.336	**0.181**
degree ℓ-2	0.560	0.378	**0.281**	0.563	0.386	**0.350**	0.570	0.429	**0.319**

we choose 100 times a different subset of 50% known labels. We set a baseline by using the same techniques with a k-Nearest neighbors graph (k-NN) with different choices of k.

In Fig. 6 we plot the behavior of different models for different density levels. The horizontal axis is the average number of non-zero edges per node.

The dashed lines of the middle plot denote the number of nodes contained in components without labeled nodes, that can not be classified.
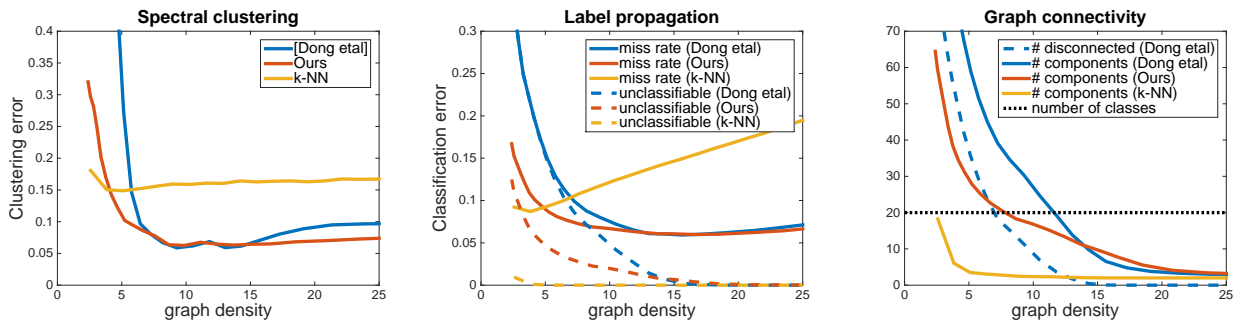
Figure 6: Graph learned from non-uniformly sampled images from COIL 20. Average over 20 different samples from the same non-uniform distribution of images. **Left**: Clustering quality. **Middle**: Label propagation quality. Dashed lines are the number of nodes in components without labeled nodes. **Right**: Number of disconnected components and number of disconnected nodes (Our model and k-NN have no disconnected nodes).