From data to structures: graph learning under smoothness assumptions and applications in data science

THÈSE N^{O} 7302 (2016)

PRÉSENTÉE LE 3 NOVEMBRE 2016

À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR

LABORATOIRE DE TRAITEMENT DES SIGNAUX 2

PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Vasilis KALOFOLIAS

acceptée sur proposition du jury:

Prof. J.-Ph. Thiran, président du jury
Prof. P. Vandergheynst, directeur de thèse
Prof. M. Maggioni, rapporteur
Dr P. Borgnat, rapporteur
Prof. P. Thiran, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2016

To my family.

Acknowledgements

The road towards a doctoral thesis is a long one, and it is impossible to carry out alone. I am grateful to many people that directly or not, helped me during this journey.

I want to, first of all, thank my professor Pierre Vandergheynst. I thank him for accepting me in the group and for giving me the opportunity to work in a great environment. I am grateful for his trust in me, his generosity, and the freedom he gave me to eventually pursue the answers to my own questions.

I also want to especially thank Nathanael Perraudin, that more than a colleague and office-mate, became soon a great friend. I thank Nauman Shahid for being a great colleague and for all the funny conversations we shared and Kirell Benzi for always having a different point of view to talk about. I thank Johan, Mahdad, Andreas, and all the old and new members of LTS2 that made it feel like a second family to me. I finally want to thank the people in the corridor during my two years in the lab of probabilistic machine learning, and particularly Andrei Giurgiu for being a great officemate.

I want to thank my friends in Lausanne for their support during on- and off-working hours. First of all, Nikos Arvanitopoulos for being a great friend, housemate, and colleague, and Yannis ("the Panda") Klonatos for all the laughs and support. I want to also thank Vlado, Orestis, Katia, Sami, Dimitra, all the "Lausanne folks" people, and my former housemates and friends Angelos, Themis, and Kostas.

I would also like to thank my former professor Efstratios Gallopoulos, that was a great source of inspiration during my studies in Greece. I am grateful for his positive influence, that followed me also during my Ph.D. years.

Finally, I want to thank my family for their love and invaluable support during all ups and downs of my Ph.D. years. My parents are the ones who always supported my achievements and my brother the one who inspired me in my first experimentations with anything (long, long time before my Ph.D.). Without them, I would not be who I am today.

Lausanne, October 2016 Vassilis Kalofolias

Abstract

Weighted undirected graphs are a simple, yet powerful way to encode structure in data. A first question we need to address regarding such graphs is how to use them effectively to enhance machine learning problems. A second but more important question is how to obtain such a graph directly from the data. These two questions lead the direction of this thesis.

We first explore the use of graphs in matrix completion, with direct implications in recommendation systems. In such systems side information graphs arise naturally, by comparing user behavior and interactions, and by product profiling. Given incomplete object ratings by users, and known graphs for user and object similarities, we propose a new matrix completion model that exploits the graphs to achieve better recommendations. Our model is convex and is solved efficiently with an Alternating Direction Multipliers method (ADMM). It achieves better recommendations than standard convex matrix completion, especially when very few ratings are known a priori, while it is robust to the graph quality.

Continuing with the second question, we are interested in learning a graph from data. Data is usually assumed to be smooth on a graph, an assumption associated with graph sparsity and connectivity. Using this assumption, we provide a general graph learning framework that is convex and easy to optimize. We classify graph learning according to the prior structure information in edge-local, node-local or global.

Edge-local models are the cheapest computationally, learning each edge independently. Exponential decay graphs and ϵ-neighborhood graphs, the most prevalent weighting schemes in the literature, are such models of our framework. We provide a new model that interpolates between these two, and achieves better performance in many settings.

Node-local models take into account the neighborhood of each edge. While they require iterative algorithms, they are stronger models achieving better results. We provide a new node-local model for general graph learning under smoothness assumptions, that redefines the state of the art. We discuss the previous state of the art, two global models of our framework. We provide efficient primal-dual based algorithms for efficiently solving our model and the most prevalent of the previous ones. Our model performs best in many settings.

Large scale applications require graphs that are easy to compute, with cost that scales gracefully with respect to the number of nodes, and do not need parameter tuning. We provide the first large scale graph learning solution, based on approximate nearest neighbors with cost of

$\mathcal{O}\left(n\log(n)\right)$ for n nodes. Our solution automatically finds the needed parameters given the sparsity level needed.

We finally explore learning graphs that change in time. In many applications graphs are not static, and data comes from different versions of them. In such cases there is no one graph that correctly summarizes connectivity, and we need to learn several graphs, one for each selected time window. Assuming that graphs change slowly, we provide two new models based on a known global and our own node-local one. Our convex time-varying-graph models can use any convex differentiable loss function for time-change penalization, have a cost linear to the number of time windows, and are easy to parallelize. The obtained graphs fit the data better and provide a good compromise between detail in time and computational cost.

Key words: graph, graph learning, matrix completion, recommendation systems, smooth signals, signal processing on graphs, cumulative energy, total cumulative energy residual, TCER, large scale graph learning, time varying graphs, network inference, graph regression, Laplacian estimation, convex optimization, ADMM, primal-dual optimization, proximal methods.

Résumé

Les graphes pondérés non dirigés présentent un moyen simple, mais puissant d'encoder la structure des données. Cette thèse explore deux questions importantes relatives aux graphes. Premièrement comment les utiliser de façon efficace pour améliorer la résolution de problèmes d'apprentissage automatique? Deuxièmement, comment obtenir ces graphes directement à partir des données?

Pour commencer, nous appliquons les graphes au problème de complétion de matrices, un problème qui a une implication directe dans les systèmes de recommandations. Dans cette application, les graphes émergent naturellement par exemple en comparant le comportement et les interactions des utilisateurs. En se fondant sur quelques évaluations ainsi que deux graphes connectant les utilisateurs et les produits, nous proposons un nouveau modèle de complétion de matrices. Notre modèle est convexe et se résout facilement via une méthode parallélisable (ADMM). Il surpasse le modèle standard de complétion de matrice et se révèle particulièrement efficace lorsque peu d'évaluations sont connues.

Ensuite, nous nous intéressons à l'apprentissage d'un graphe directement à partir des données. En général, les données sont supposées être régulières sur un graphe, une hypothèse associée à la parcimonie et à la connectivité. Grâce à cette hypothèse, nous proposons une classe de problèmes convexes facilement optimisables qui permettent d'apprendre le graphe. Nous classifions ces problèmes en fonction de leur structure: «arête locale», «nœud local» ou «global».

Les modèles «arête locale» sont les moins gourmands en calcul car chaque arête est apprise indépendamment. Par exemple, les graphes à décroissance exponentielle et les graphes de voisinage ϵ-proche, les stratégies les plus utilisées dans la littérature, sont de tels modèles dans notre contexte. Nous proposons un nouveau modèle qui interpole entre ces deux cas et qui atteint de meilleures performances dans de nombreux scénarios.

Les modèles «nœud local» prennent en compte le voisinage de chaque arête. Ils nécessitent des algorithmes itératifs mais sont plus performants. Nous proposons un nouveau modèle «nœud local» sous l'hypothèse de la régularité qui redéfinit l'état de l'art. Nous discutons des résultats antérieurs de l'état de l'art, deux modèles généraux de notre classe. Nous proposons aussi des implémentations efficaces fondées sur des algorithmes proximaux pour notre modèle ainsi que pour l'un des précédents, montrant que notre modèle est le meilleur dans de nombreuses situations.

Par ailleurs, les applications à grande échelle nécessitent des graphes facilement calculables, c'est à dire avec un coût qui augment lentement par rapport au nombre de nœuds, et sans réglage de paramètres. Nous proposons la première implémentation à grande échelle d'apprentissage de graphe fondée sur une approximation des plus proches voisins. Le coût est de l'ordre de $\mathcal{O}\left(n\log(n)\right)$ pour n nœuds et les paramètres pour un niveau de parcimonie donné sont sélectionnés automatiquement.

Finalement, nous explorons l'apprentissage des graphes évoluant au cours du temps. En supposant un changement lent, nous proposons deux nouveaux modèles fondés sur les résultats précédents. Notre méthode d'apprentissage peut être utilisée avec n'importe quelle fonction de pénalisation différentiable et est facilement parallélisable. Le graphe ainsi obtenu permet un bon compromis entre la précision au niveau temporel et le temps de calcul.

Mots clefs: graph, graph learning, matrix completion, recommendation systems, smooth signals, signal processing on graphs, cumulative energy, total cumulative energy residual, TCER, large scale graph learning, time varying graphs, network inference, graph regression, Laplacian estimation, convex optimization, ADMM, primal-dual optimization, proximal methods.

Contents

Introduction

When we solve a machine learning problem, adding prior information about the structure of our data can lead to stronger models. An effective and flexible way to express structure in data is by the use of graphs, where pairs of objects (nodes) are connected through edges. Depending on the problem and the data in hand, a graph structure might be trivially available. For example in social networks the graph can be defined through friendships, while in a train transportation network the graph connects cities that have a direct train connection. When such prior information is known about our data, a natural question that arises is

1. *How can we use graph structures to solve a machine learning problem more effectively?*

In the context of machine learning, the graph structure is often encoded through its Laplacian matrix, denoted by L (Chapter 1). The importance of the graph Laplacian has long been known as a tool for low-dimensional embedding, manifold learning, clustering and semi-supervised learning, see for example Belkin and Niyogi [2001]; Zhu et al. [2003]; Coifman et al. [2005]; Belkin et al. [2006]; Von Luxburg [2007].

More recently we find an abundance of methods that exploit the notion of smoothness on a graph (Chapter 1) to regularize various machine learning tasks, solving problems of the form

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

Zhang et al. [2006] solve problems of this form in order to enhance web page categorization with graphs, Elmoataz et al. [2008] for image and manifold processing, and Zheng et al. [2011] for graph regularized sparse coding. Cai et al. [2011] use the same smoothness term to regularize NMF, Jiang et al. [2013] for PCA, and Shahid et al. [2015] for Robust PCA. In Chapter 2 we solve a problem of this form in order to enhance matrix completion when similarity graphs are known [Kalofolias et al., 2014].

From data to structures

In many cases, the structure underlying our data in hand is not readily available. Learning such graph structures is in itself an important problem. Not only can they be an powerful statistical analysis tool, but also a necessary step behind the success of the aforementioned

methods.

Given the importance of the problem, a big part of today's machine learning research is devoted in identifying structures from data. The second and more interesting question addressed in this thesis is in the same direction:

2 *How can we learn a graph from given data?*

In order to learn a graph we solve the complementary problem, where we are given the data X and we want to find a graph through its Laplacian L

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

where $\mathcal{L}$ denotes the set of valid graph Laplacians. This is the subject of Chapter 3 and the ones after that, significantly extending the work in [Kalofolias, 2016].

Graphs are a great data analysis and processing tool. In order to afford to use them for real data, we have to keep the *computational cost* low as the number of nodes increases. This is the reasoning behind using Chebyshev polynomials [Hammond et al., 2011] or Lanczos approximations [Susnjara et al., 2015] in order to vastly reduce the cost of graph filtering. This is also the rationale for using approximate nearest neighbor graphs [Muja and Lowe, 2014] instead of exact fully connected or nearest neighbor graphs. For the same reasons, graph learning has to be performed efficiently, leading us to methods like our large scale graph learning of Chapter 6.

Thesis structure and contributions

Chapter 1: We provide properties of graphs (many of which are novel) that will be needed in the rest of the thesis. We show the connection between data smoothness, graph sparsity, and connectivity. We unify different models of smooth signals under the notion of filtering. We provide a new quality measure of how well a linear basis "explains" a data matrix or distribution. The latter will be needed throughout the thesis to measure graphs quality. Notably, we show that minimizing $\operatorname{tr}\left(X^\top LX\right)$ implicitly optimizes this general basis quality measure.

Chapter 2: We introduce a novel matrix completion model that uses graphs to encode proximity information between rows and between columns. Fitting the model leads to a convex function, for which we provide an efficient optimization algorithm. Targeting recommender systems applications, we study and evaluate the proposed matrix completion on synthetic and real data, showing that our model outperforms the standard matrix completion model in many situations.

Chapter 3: We propose a framework to learn 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 $\operatorname{tr}\left(X^\top LX\right)$ is small, where

L is the graph Laplacian. We show that the problem is a weighted ℓ_1 minimization that leads to naturally sparse graphs. We classify models as edge-local, node-local or global depending on the separability of the objective function with respect to the edges.

Chapter 4: We focus on graph learning where each edge can be learnt separately (edge-local). We prove that the standard graph construction with Gaussian weights $w_{ij} = \exp\left(-\frac{1}{\sigma^2}\|x_i - x_j\|^2\right)$ is a special case of our framework, with an objective function related to the edge entropy. The ϵ-neighborhood graphs are also part of our framework. We provide a new model that interpolates between ϵ-neighborhood and Gaussian weight graphs, and we show that in many cases it performs better than both.

Chapter 5: Models where edges are not learnt separably are stronger. We propose a new node-local model and present a fast and scalable primal-dual based algorithm to solve it. We also analyze the previous state of the art model using our framework, so as to simplify it, prove fundamental properties, and provide a scalable algorithm that was missing from the literature. The two algorithms we provide (for our model and the previous state of the art) are the first scalable solutions in the literature to learn a graph under smoothness assumptions. We discuss a Bayesian point of view of graph learning to show the connections between these models and another global model of the literature. We evaluate their performance on artificial and real data. Our model performs best in most settings, providing good connectivity for distant nodes, where the previous state of the art model is struggling.

Chapter 6: We attack the two main problems of our model of Chapter 5 in terms of complexity. We provide an approximate solution scheme based on approximate nearest neighbors graphs, that costs $\mathcal{O}\left(m \log m\right)$ instead of $\mathcal{O}\left(m^2\right)$ per iteration for m-nodes graphs. We analyze the sparsity of the obtained graph as a function of the learning parameters in order to give a practical scheme of setting them automatically. With a simple Matlab implementation we are able to learn a graph between 60000 images of MNIST on a laptop in less than a minute. Our large scale experiments further show that our model vastly outperforms the previous state of the art, that we also implement using our large scale scheme.

Chapter 7: We show how one can learn a graph that changes slowly in time. We show how the same algorithm we used in Chapter 5 can solve the new problem, and how it is easily parallelized. Our experiments show that the new "time smoothness" assumption makes time varying graphs more robust to the number of training signals, fitting data better and achieving greater detail in time.

Chapter 8: We discuss some of the main points of this thesis and new research directions that it can lead to.

1 Graphs and data on graphs

In this thesis we are interested in *weighted undirected graphs* without self-loops. Such a graph, written as $G = \{\mathcal{V}, \mathcal{E}, W\}$, is defined by the set of vertices or nodes $\mathcal{V} = \{1 \dots n\}$, the set of edges $\mathcal{E} \subseteq \{(i, j) \,|\, i, j \in \mathcal{V}\}$, and a weighted adjacency matrix $W \in \mathbb{R}_+^{n \times n}$. An element W_{ij} of the latter is the weight of its associated edge $(i, j) \in \mathcal{E}$, or equal to 0 if such an edge does not exist. The adjacency matrix of an undirected graph is symmetric, and as we do not assume self-loops, has a zero diagonal.

Given a graph G we define a *signal* $x \in \mathbb{R}^n$ on a graph by associating its element x_i to vertex i of the graph. A signal x will in general be called *smooth on* G if its *Dirichlet energy*

$$\|x\|_{\mathcal{D},G}^2 = \frac{1}{2} \sum_{i,j} W_{ij} (x_i - x_j)^2 \tag{1.1}$$

is small. In order for the Dirichlet energy of a signal x to be small, it needs to have similar values x_i and x_j for two well connected nodes i, j (that is, with large W_{ij}). For a set of m signals in $\mathbb{R}^n$arranged as columns of a matrix $X \in \mathbb{R}^{n \times m}$, the Dirichlet energy measuring its smoothness on G is defined as

$$\|X\|_{\mathcal{D},G}^2 = \sum_{k=1}^{n} \|x_k\|_{\mathcal{D},G}^2 = \frac{1}{2} \sum_{i,j} W_{ij} \left\| x_i - x_j \right\|^2, \tag{1.2}$$

where now we use x_i to denote the i-th row of X. In this case, each of the nodes of the graph corresponds to a row of X. Note that in order to keep notation simple, we will follow the same convention from now on, using x_i to denote a vector in $\mathbb{R}^n$ or $\mathbb{R}^m$ according to the context, unless otherwise noted.

A very important mathematical object defined on a graph is its *Laplacian*

$$L = D - W, \quad D = \operatorname{diag}(W\mathbf{1}), \tag{1.3}$$

where $\mathbf{1} = [1, \dots, 1]^\top$ and D is the diagonal matrix containing the weighted degrees of the graph.

Given this definition of the Laplacian, the Dirichlet energy can be written as

$$\|x\|^2_{\mathcal{D},G} = x^\top L x,$$

$$\|X\|^2_{\mathcal{D},G} = \mathrm{tr}\left(X^\top L X\right),$$

using the fact that

$$\begin{aligned}
\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}\|x_i - x_j\|_2^2 &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(x_i - x_j)^\top W_{ij}(x_i - x_j) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i^\top W_{ij} x_i - \sum_{i=1}^{n}\sum_{j=1}^{n} x_i^\top W_{ij} x_j \\
&= \sum_{i=1}^{n} x_i^\top x_i \sum_{j=1}^{n} W_{ij} - \mathrm{tr}\left(X^\top W X\right) \\
&= \mathrm{tr}\left(X^\top D X\right) - \mathrm{tr}\left(X^\top W X\right) \\
&= \mathrm{tr}\left(X^\top L X\right). \qquad (1.4)
\end{aligned}$$

1.1 Properties of the graph Laplacian

A big part of this thesis is dedicated to learning a graph from a data matrix X. Previous approaches to learning a graph from data were based on searching for the Laplacian of the graph [Lake and Tenenbaum, 2010; Hu et al., 2013; Dong et al., 2015]. The space of all valid graph Laplacians, is by definition

$$\begin{aligned}
\mathcal{L} = \Big\{ & L \in \mathbb{R}^{n\times n} : \\
& (\forall i \neq j) \quad L_{ij} = L_{ji} \leq 0, \;\; L_{ii} = -\sum_{j\neq i} L_{ij} \Big\},
\end{aligned}$$

containing constraints about the sign of the off-diagonal elements, the relationship between the diagonal and off-diagonal elements, and the symmetricity. We argue that it is more intuitive to search for a valid weighted adjacency matrix W from the space

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

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

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

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. We use the notation $W = \mathrm{matrixform}(w)$ and $w = \mathrm{vectorform}(W)$ to express the transition between a

Table 1.1 – Equivalent terms for representations from sets $\mathcal{L}$, $\mathcal{W}_m$ and $\mathcal{W}_v$. We use $z = \text{vectorform}(Z)$ and linear operator S that performs summation in the vector form.

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

weight vector $w \in \mathcal{W}_v$ and an adjacency matrix $W \in \mathcal{W}_m$. We can see the vectorform operation as a simple stacking of the upper triangular elements of W (the diagonal not included) in a vector w. In the sequel we use $\mathcal{W}_m$ to analyze the problem in hand and $\mathcal{W}_v$ when we solve the problem using optimization. Table 1.1 exhibits some of the equivalent forms in the three spaces.

1.1.1 Smoothness on a graph means graph sparsity

In this section we see the Dirichlet energy as a function of the weights (and not as a function of X). Let us define the *pairwise distances matrix* $Z \in \mathbb{R}_+^{n \times n}$:

$$Z_{ij} = \left\| x_i - x_j \right\|^2 . \tag{1.5}$$

Using matrix Z we can rewrite the Dirichlet energy as a function of W as

$$\text{tr}\left(X^\top L X\right) = \frac{1}{2}\,\text{tr}\,(WZ) = \frac{1}{2}\,\|W \circ Z\|_{1,1}\,, \tag{1.6}$$

where $\|A\|_{1,1}$ is the elementwise norm-1 of A and $\circ$ is the Hadamard product. The last equality holds because both W and Z are non-negative.

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. We can easily see that adding such a term is equivalent to merely adding a constant to the squared distances in Z:

$$\text{tr}\left(X^\top L X\right) + \gamma\|W\|_{1,1} = \frac{1}{2}\left\|W \circ (2\gamma + Z)\right\|_{1,1}. \tag{1.7}$$

Note that all information of X conveyed by the trace term is contained in the pairwise distances matrix Z, so that the original data could be omitted. Moreover, using the last term of eq. (1.6) instead of the trace enables us to define other kinds of distances instead of Euclidean. In fact, any kind of pairwise dissimilarity between nodes can be used in order to learn a graph with our framework of Chapter 3, depending on the application.

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_{ij}.

1.1.2 Graph sparsity versus connectivity

The number of different connected components of a graph is given by the number of zero eigenvalues of its Laplacian, $n - \text{rank}(L)$. While the rank itself is not a convex function of W or L, we can consider the tightest convex relaxation that is the nuclear norm of L if we want to control connectivity. Then the following interesting property holds:

$$\|L\|_* = \text{tr}(L) = \text{tr}(D) = \|W\|_1, \tag{1.8}$$

where the last equality holds because the weights are non-negative. The interpretation of this property is that when we have a sparser graph (fewer edges), it becomes less connected and a larger number of separate connected components will appear. On the other hand, by keeping the nuclear norm high, the graph is denser as we have a less sparse W.

The nuclear norm of the Laplacian is very particular, as it is a linear function of W. A direct implication of this is that when we learn a graph we can either penalize or promote connectivity (i.e. adding a positive or a negative term $\|L\|_*$), keeping the optimization problem convex. We will see both these cases in Chapter 4.

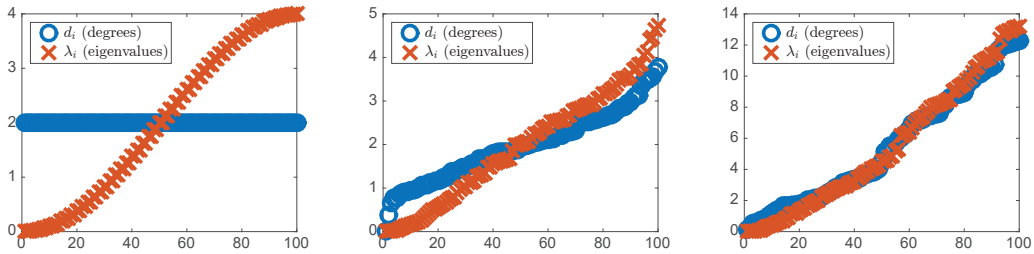
1.1.3 Degrees and eigenvalues

The degrees and the eigenvalues of a graph Laplacian have a special relationship. We already know that they have the same sum:

$$\text{tr}(L) = \sum_{i=1}^{n} \lambda_i = \sum_{i=1}^{n} d_i = \text{tr}(D).$$

More than that, the eigenvalues of a Laplacian majorize the node degrees according to the Schur-Horn theorem [Schur, 1923; Horn, 1954]:

$$\sum_{i=1}^{n} [\lambda^{\downarrow}]_i > \sum_{i=1}^{n} [d^{\downarrow}]_i,$$



(a) Ring graph (b) Random geometric graph (c) Non-uniform graph

Figure 1.1 – Eigenvalues and degrees of different types of graphs. According to the Schur-Horn theorem [Schur, 1923; Horn, 1954], the eigenvalues majorize the degrees of the Laplacians.

where $\lambda^{\downarrow}$ denotes sorting of the eigenvalues in a *decreasing* order, and $[]_i$ is the i-th element of the expression in brackets. One of the implications is that the smallest degrees are always larger in general than the smallest eigenvalues (and the largest degrees smaller than the largest eigenvalues). We can see a few examples in Figure 1.1.

In Chapter 5 we will use the relationship between the degrees and the eigenvectors in order to compare different graph learning models.

1.2 Smooth Signals

The claim that a signal is smooth on a graph if its Dirichlet energy is small is a bit vague. A slightly more rigorous definition can be given using tools from signal processing on graphs, as explained in the work of Shuman et al. [2013]. Similarly to time signals, we can analyze a graph signal $x \in \mathbb{R}^n$ in its *graph frequencies* $\hat{x} \in \mathbb{R}^n$ using the graph Fourier transform:

$$\hat{x} = U^\top x.$$

The *graph Fourier* matrix $U = [u_1, \dots, u_n]$ of a graph is defined through the eigenvectors u_i of its Laplacian, associated to its eigenvalues λ_i, sorted in increasing order: $L = U\Lambda U^\top$. Low graph frequencies correspond to small eigenvalues, and smooth signals are the ones with a big part of their energy concentrated in low graph frequencies. This is revealed if we assume a unit norm graph signal x so that $\sum_i \hat{x}_i^2 = \|x\|^2 = 1$ and write its Dirichlet energy as

$$\|x\|_{\mathcal{D},G} = x^\top L x = \hat{x}^\top \Lambda x = \sum_{i=1}^{n} \lambda_i \hat{x}_i^2.$$

Low frequency coefficients $\hat{x}_i$ are multiplied by small eigenvalues λ_i and obtain a small graph Dirichlet energy.

In the literature, different definitions of what is a smooth signal have been used in different contexts. We unify these different definitions using the notion of filtering on graphs. Filtering

of a graph signal x by a filter $h(\lambda)$ is defined as the operation

$$y = h(L)x = \sum_i u_i h(\lambda_i) u_i^\top x = \sum_i u_i h(\lambda_i) \hat{x}_i. \tag{1.9}$$

Note that by h we denote both the function $h : \mathbb{R} \to \mathbb{R}$ and its matrix counterpart $h : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ acting on the matrix's eigenvalues. As low frequencies correspond to small eigenvalues, a low-pass or smooth filter corresponds to a decaying function 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 figures 1.2 and 1.3.

1.2.1 Smooth signals by Tikhonov regularization.

Given an arbitrary initial signal $x_0 \in \mathbb{R}^n$ we can obtain a smooth signal x not too far from x_0 by directly penalizing its Dirichlet energy $x^\top Lx$:

$$x = \arg\min_x \frac{1}{2}\|x - x_0\|_F^2 + \frac{1}{\alpha} x^\top Lx.$$

The result of this optimization problem, that can be seen as a denoising problem [Elmoataz et al., 2008] of signal x_0 is given by $x = (\alpha L + I)^{-1} x_0$. Equivalently, we can see this as filtering x_0 by

$$h(\lambda) = \frac{1}{1 + \alpha\lambda}, \tag{1.10}$$

where large α values result in smoother signals.

1.2.2 Smooth signals from a linear Gaussian model.

Lake and Tenenbaum [2010] proposed that smooth signals can be generated by a linear Gaussian model

$$x \sim \mathcal{N}\left(0,\ \left(L + I/\sigma^2\right)^{-1}\right). \tag{1.11}$$

Later, Dong et al. [2015] modeled smooth signals on graphs as Gaussian signals from the generative model

$$x \sim \mathcal{N}\left(\bar{x},\ L^\dagger\right), \tag{1.12}$$

where $L^\dagger$ is the pseudo-inverse of L and the mean $\bar{x}$ should also be a smooth signal. The two generative models (1.11) and (1.12) are *equivalent up to an additive constant* in the special

case where $\bar{x}$ is a constant vector, and for $\sigma \to \infty$.[1,2] The Dirichlet energy of a signal is however invariant to additive constants.

To sample from a Gaussian linear model, it suffices to draw an initial white Gaussian signal $x_0 \sim \mathcal{N}(0, I)$ (that is non-smooth), and then compute

$$x = \bar{x} + h(L)x_0, \tag{1.13}$$

with $h(L) = \sqrt{\left(L + I/\sigma^2\right)^{-1}}$ for eq. (1.11) or $h(L) = \sqrt{L^\dagger}$ for eq. (1.12). Equivalently, for the model by Lake and Tenenbaum [2010] we can filter x_0 by

$$h(\lambda) = \sqrt{\left(\lambda + \sigma^{-2}\right)^{-1}} \in \mathbb{R}, \tag{1.14}$$

and for the model by Dong et al. [2015] we can filter x_0 by

$$h(\lambda) = \begin{cases} \sqrt{\lambda^{-1}}, & \lambda > 0 \\ 0, & \lambda = 0 \end{cases} \in \mathbb{R} \tag{1.15}$$

and add the mean $\bar{x} \in \mathbb{R}$.

We point out here that using eq. (1.13) 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. For an analysis of graph stationary signals with arbitrary covariance function $h(L)$, we refer the curious reader to the work of Perraudin and Vandergheynst [2016].

1.2.3 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

$$h(\lambda) = \exp(-t\lambda), \tag{1.16}$$

[1] This happens because the two covariances only differ in how they deal with the 0-th eigenvalue of L, that is associated with a constant eigenvector.

[2] Note also that if we learn a graph between rows of a matrix X using Euclidean distances Z, adding a random constant to any column will not change the result, as Z is translation invariant.

Table 1.2 – Different Types of Smooth Signals.

Concept	Model	Graph filter
Tikhonov	$x = \arg\min_x \left[\frac{1}{2}\|x - x_0\|_F^2 + \frac{1}{\alpha} x^\top L x\right]$	$h(\lambda) = \frac{1}{1+\alpha\lambda}$
Generative model	$x \sim \mathcal{N}\left(0, L^\dagger\right)$	$h(\lambda) = \begin{cases} \frac{1}{\sqrt{\lambda}} & \text{if } \lambda > 0 \\ 0 & \text{if } \lambda = 0 \end{cases}$
Heat diffusion	$x = \exp(-\alpha L)\, x_0$	$h(\lambda) = \exp(-\alpha\lambda)$

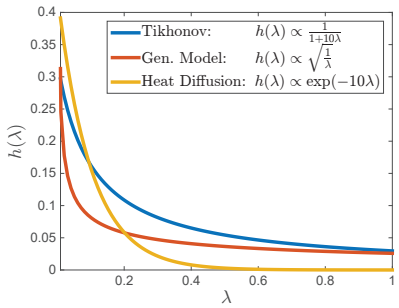


Figure 1.2 – The filters of Table 1.2 for $\alpha = 10$.

where larger values of t result in smoother signals.

1.2.4 Artificial smooth data

Table 1.2 summarizes the different models of smooth signals, that are plotted in Figure 1.2. Samples of these signals on the same non-uniform graph are plotted in Figure 1.3. We see that the three signals are quite similar to each other, as they are smooth versions of the same initial x_0.

1.3 Sample graphs versus feature graphs

Suppose that we have a matrix $X \in \mathbb{R}^{n\times m}$, where rows correspond to samples, for example different images, and columns correspond to features, in this case pixels. In the literature there have been two types of graphs proposed for this kind of data.

The most prevalent type is the one between samples. In this case, the columns can be seen as signals on the columns graph. The rationale behind these is very often that samples reside on low dimensional manifolds, and graphs are their discrete approximations. For example learning the connections between images, forming the graph Laplacian and using the eigenvectors of the smallest eigenvalues we can embed them in a low dimensional space Belkin and Niyogi [2001]. This is the point of view followed by Zhu et al. [2003] as well, and in general by most of the graph models in the literature.

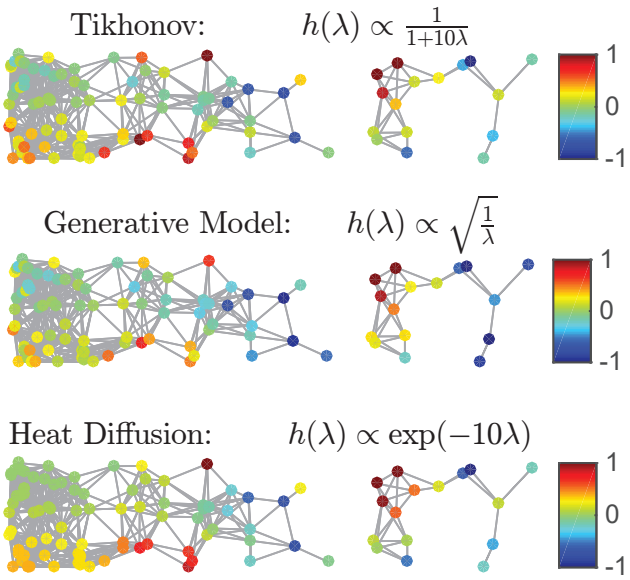


Figure 1.3 – Different smooth signals on the Non Uniform graph used for our artificial data experiments. All signals are obtained by smoothing the same initial $x_0 \sim \mathcal{N}(0, I)$ with three different filters. This instance of the graph is disconnected with 2 components.

More recently, a different functionality of graphs was introduced in the literature. In the work by Shahid et al. [2016a] a graph was introduced between features, in order to obtain a close to low rank matrix X without a nuclear norm term. Later, Perraudin and Vandergheynst [2016] introduced the notion of signals that are stationary on feature graphs, that is, signals whose graph Laplacian and covariance have the same eigenvectors. The signals in this case are the images, or the samples, that reside on the features' graph. There is not necessarily a manifold reasoning behind this choice of graphs. On the other side, we can draw connections between this type of graph and an inverse covariance matrix, or with principal component analysis (PCA).

In this thesis, we experiment with both above cases. To show the contrast between them, we will learn both types of graphs from the same data matrix X, the train part of the MNIST dataset. The latter contains 60000 images (samples) of 784 pixels (features) as the images are of size 28×28. In Figure 1.5b we see the quality of learned *feature graphs* compared to quality of a PCA basis when explaining the whole MNIST dataset using only k samples (signals). As explained in this figure, such graphs are especially interesting when few samples (signals) are available for this task. On the other hand, we also learn a graph between all 60000 images in Chapter 6. There, we have no choice: we *only* have 784 signals (pixels) available in order to learn a graph with 60000 nodes.

Note that it is not always clear which assumption makes more sense for our data. For example in matrix completion of a ratings matrix where columns and rows correspond to users and movies respectively, there is no clear distinction between features and samples. If samples are the users, then features are the movies, and if samples are the movies, then features are

the users. Nevertheless, the use of graphs to encode proximities between users and between movies helps in improving the recovery quality (Chapter 2).

1.4 A new graph quality measure

In our research it is crucial to have a good measure of how well a graph G fits a data matrix X or a distribution $p(X)$. This is important in both fundamental problems that interest us in this thesis: when we recover X knowing G, and when we learn a graph G from the data X. In this section, we propose a new quality measure that associates a graph to a data matrix or distribution, that to the best of our knowledge was missing from the literature.

In order to define the new measure, we will see the graph as a method that provides us with an orthogonal basis. This basis is the graph Fourier transform matrix [Chung, 1997], that is, the eigenvector matrix of the graph Laplacian.

One of the known quality criteria of orthogonal bases is their ability to "explain" most of the data energy or variance with as few basis vectors as possible. The energy is optimized by the singular vectors of a matrix obtained by SVD, and the variance by the eigenvectors of its empirical covariance obtained by PCA. Naturally, when seeking a good basis Q for smooth signals X, we want to quantify this ability of "energy explanation by few vectors", asking the question: *How efficiently does Q accumulate energy of data X or distribution $p(X)$ in its first few vectors?*

We begin by defining how much energy of a matrix X is carried by the first k vectors of an orthogonal basis Q:

Definition 1. Data cumulative energy on a basis Q: Given a data matrix $X \in \mathbb{R}^{n\times m}$ and a sorted orthogonal basis $Q = \left[q_1, \dots q_n\right]$, the cumulative energy of the data on the first k vectors of the basis is defined as

$$\mathcal{S}\{X, Q\}(k) = \sum_{i=1}^{k} \|q_i^\top X\|^2 = \|\hat{X}_k\|_F^2.$$

By $\hat{X}_k$ we denote the projection of X on the first k vectors of the basis: $\hat{X}_k = Q_{:,1:k} Q_{:,1:k}^\top X$.

To understand why the cummulative energy can reveal the basis quality, let us see the error between the projected matrix $\hat{X}_k$ and the original matrix X:

$$\begin{aligned} \|X - \hat{X}_k\|_F^2 &= \|X\|_F^2 + \|\hat{X}_k\|_F^2 - 2\,\mathrm{tr}\left(\hat{X}_k^\top X\right) \\ &= \|X\|_F^2 - \|\hat{X}_k\|_F^2 \\ &= \|X\|_F^2 - \mathcal{S}\{X, Q\}(k). \end{aligned} \tag{1.17}$$

We can see that by increasing the cumulative energy on the first k vectors, we decrease the error between the projection $\hat{X}_k$ and the actual data X. The cumulative energy is a non decreasing

function of k and obviously depends on the order of the basis vectors.

We can summarize the compressibility of a dataset on a given basis Q as expressed by the cumulative energy by its sum over all possible ranks of approximation from 1 to n:

Definition 2. Total cumulative energy of data on a basis Q: Given a data matrix $X \in \mathbb{R}^{n\times m}$ and a sorted orthogonal basis $Q = [q_1, \dots q_n]$, the total cumulative energy of the data X is defined as

$$\mathcal{T}\{X,Q\} = \sum_{k=1}^{n} \mathcal{S}\{X,Q\}(k) = \sum_{k=1}^{n}\sum_{i=1}^{k} \|q_i^\top X\|^2 = \sum_{i=1}^{n}(n+1-i)\|q_i^\top X\|^2.$$

The highest cumulative energy $\mathcal{S}$ of any matrix X for a fixed k is achieved by its left singular vector basis. Consequently, the same basis achieves the highest total cumulative energy of X. In the special case that X has zero sum of columns, the best basis is the one given by principal component analysis (PCA): Suppose that the singular value decomposition of the data matrix is $X = U\Sigma V^\top$ where Σ_{ii} are sorted in decreasing order. Then we have

$$\max_{Q\in\mathbb{O}} \mathcal{T}\{X,Q\} = \mathcal{T}\{X,U\} = \sum_{k=1}^{n}\sum_{i=1}^{k}\Sigma_{ii}^2 = \sum_{i=1}^{n}(n+1-i)\Sigma_{ii}^2, \tag{1.18}$$

where $\mathbb{O}$ is the set of all orthogonal matrices. Note that if X has zero column sum, U is also its PCA basis, as its empirical covariance matrix is $\frac{1}{n}XX^\top = U\Sigma^2U^\top = UEU^\top$, where $E = \Sigma^2$ is the eigenvalue diagonal matrix.

If the columns of X follow a *distribution* $p(X)$ with mean $m_X = \mathbb{E}_{P(X)}\{X\}$ and covariance $C_X = \mathbb{E}_{P(X)}\left\{(X-m_X)(X-m_X)^\top\right\}$, what is more interesting is to measure the *expected cumulative energy*:

Property 1. Expected cumulative energy

$$\begin{aligned}
\mathbb{E}_{P(X)}\left\{\mathcal{S}\{X,Q\}(k)\right\} &= \mathbb{E}_{P(X)}\left\{\sum_{i=1}^{k}\left\|q_i^\top X\right\|^2.\right\} \\
&= \sum_{i=1}^{k} q_i^\top \mathbb{E}_{P(X)}\left\{XX^\top\right\} q_i \\
&= \sum_{i=1}^{k} q_i^\top \left(C_X + m_X m_X^\top\right) q_i \\
&= \sum_{i=1}^{k}\left\|q_i^\top U_X\Sigma_X\right\|^2 \\
&= \mathcal{S}\left\{U_X\Sigma_X, Q\right\}(k),
\end{aligned}$$

where we have used the eigenvalue decomposition

$$C_X + m_X m_X^\top = U_X\Sigma_X^2U_X^\top.$$

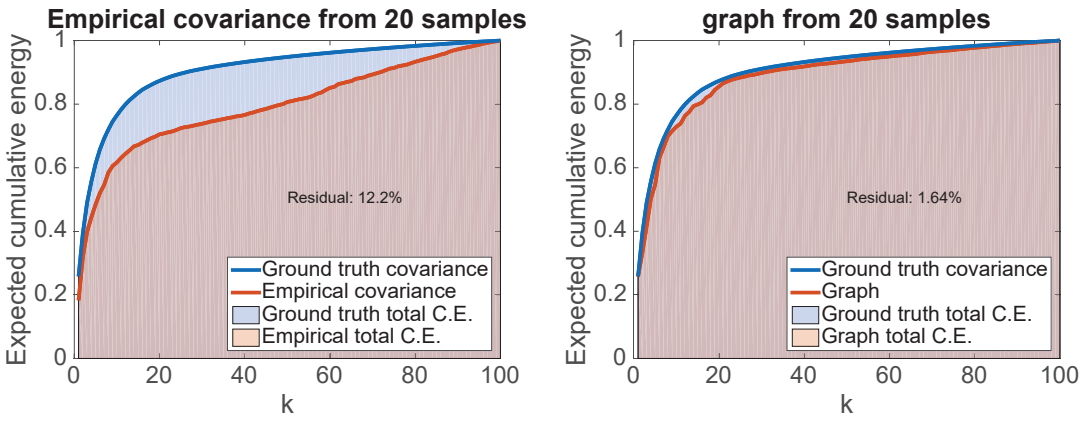


Figure 1.4 – Expected cumulative energy of empirical covariance basis (**left**) and graph basis (**right**) computed from **20 samples** versus maximum expected cumulative energy (blue line). Data is constructed according to the generative model (1.13) from a 100-node sensor graph and normalized. The residual of the expected total cumulative energy, eq. (1.20) is equal to the difference between the red and blue areas over the blue area. The graph used for the right figure is our model explained in detail in Chapter 5.

The maximum *expected* total cumulative energy is now achieved by the basis implied by $\mathbb{E}_{P(X)}\{XX^\top\}$:

$$\begin{aligned}\max_{Q\in\mathbb{O}} \mathbb{E}_{P(X)}\{\mathcal{T}\{X,Q\}\} &= \max_{Q\in\mathbb{O}} \mathcal{T}\{U_X\Sigma_X, Q\} \\ &= \mathcal{T}\{U_X\Sigma_X, U_X\} \\ &= \sum_{i=1}^{n}(n+1-i)\Sigma_{X_{ii}}^2. \end{aligned} \tag{1.19}$$

For zero-mean data, this translates to the eigenvectors of the covariance C_X (also known as the Karhunen–Loève basis) instead of the sample covariance used by PCA.

While the PCA basis of a zero-mean matrix X "explains" best its samples (columns of X), it might fail to explain well other samples of the same distribution, if the number of "training" samples is not sufficient. Furthermore, if the number of samples is smaller than the number of variables, only the first eigenvectors of the covariance matrix are valid, and the rest are rotated arbitrarily in the space orthogonal to the first ones. We see this effect in the left part of Figure 1.4. When only 20 samples are used to approximate the covariance matrix, the quality of the basis of its eigenvectors is far from the perfect basis given by the eigenvalue decomposition of the actual covariance matrix (blue line). To quantify this compression inefficiency, we propose the following *orthogonal basis quality measure*:

Definition 3. Total cumulative energy residual (TCER): Given a data distribution $p(X)$ with mean m_X and covariance C_X, and a sorted orthogonal basis $Q = [q_1, \dots q_n]$, we define the

residual of the total cumulative energy as

$$\begin{aligned}\mathcal{R}\{p(X),Q\} &= 1 - \frac{\mathbb{E}_{P(X)}\{\mathcal{T}\{X,Q\}\}}{\max_{Q\in\mathbb{O}}\ \mathbb{E}_{P(X)}\{\mathcal{T}\{X,Q\}\}} \\ &= 1 - \frac{\mathcal{T}\{U_X\Sigma_X,Q\}}{\mathcal{T}\{U_X\Sigma_X,U_X\}} \end{aligned} \tag{1.20}$$

where we use the eigenvalue decomposition $C_X + m_X m_X^\top = U_X \Sigma_X^2 U_X^\top$ and the denominator of eq. (1.20) is simply $\sum_{k=1}^{n}\sum_{i=1}^{k}{\Sigma_X}_{ii}^2$.

An interesting implication of TCER is that for stationary signals on graphs as defined by Perraudin and Vandergheynst [2016], where eigenvectors of the covariance C_X and the Laplacian coincide, if the graph Fourier coefficients are strictly decreasing in magnitude (the signal is smooth), TCER is minimized. In other words, *TCER is a measure of how smooth and stationary is a signal on a given graph.* This is because for such signals the Karhunen–Loève transform and the graph Fourier transform coincide as well.

1.4.1 Graphs versus PCA and the graphical LASSO

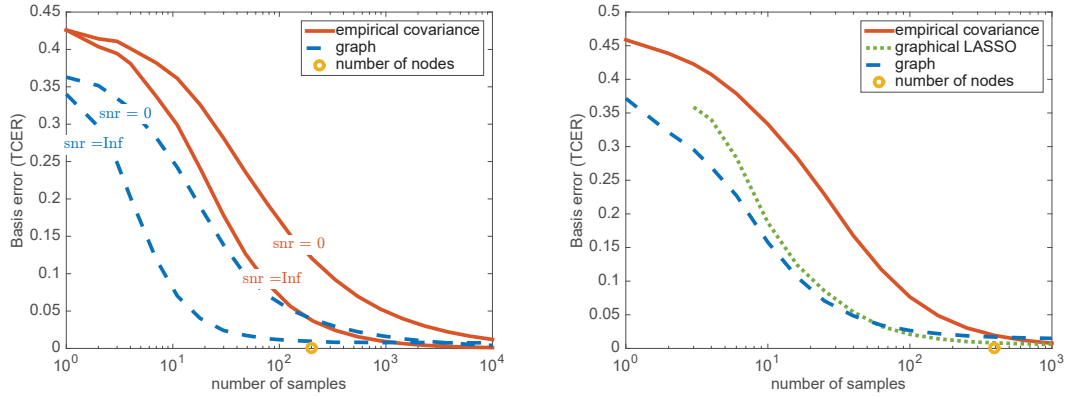
One of the advantages of using graphs is, as we see in Figure 1.5, that they provide a good basis for the whole distribution, even when very few samples are available. To produce this figure we have used zero mean data in order to have a fair comparison against PCA (for MNIST we subtract the per-feature mean from the whole dataset before any computation). We see that the trend on artificial data (left figure) is closely followed by real data as well (right figure). Because for the latter we can not know the exact theoretical covariance matrix (to compute the denominator of eq. (1.20)), we approximate it by the sample covariance of 50000 samples that were not used for computing the bases by PCA or graphs.

For MNIST, we also compare against a sparse inverse covariance estimator (also known as the graphical LASSO) [Banerjee et al., 2008; Friedman et al., 2008], that was proposed as a remedy to the problem of having very few observations for covariance estimation. We see that learning a graph gives a better quality basis unless we have around 100 samples or more for training. Note also that the computational cost of graphical LASSO is in general $\mathcal{O}\left(n^3\right)$ per iteration, while the graph learning model used here (Section 5.2) has a cost of only $\mathcal{O}\left(n^2\right)$ per iteration.

1.4.2 Connection with $\operatorname{tr}\left(X^\top L X\right)$

Let $L = Q\Lambda Q^\top$ be the eigenvalue decomposition of the graph Laplacian. Then we can rewrite the Dirichlet energy of a data matrix X on the graph as:

$$\operatorname{tr}\left(X^\top L X\right) = \sum_{i=1}^{n} \lambda_i \| q_i^\top X\|^2.$$



(a) Artificial data (sensor graph)

(b) Real data (MNIST features)

Figure 1.5 – Total cumulative energy residual of bases computed by sample PCA and graphs for different number of samples. **Left:** Artificial data from a sensor graph of 200 nodes following the generative model (1.13). **Right:** Covariance and graphs bases between features (pixels with at least 5% of the maximum variance) of MNIST train dataset. The theoretical upper bound of the expected cumulative energy was approximated using the 50000 samples of the dataset that were not used for PCA or the graph construction. The yellow circle indicates the number of features (nodes), above which the empirical covariance is invertible. Graph learning is especially attractive for the region to the left of this point. The graph used for the right figure is our model explained in detail in Chapter 5.

For the same basis, the data matrix X has the total cumulative energy

$$\mathcal{T}\{X,Q\} = \sum_{i=1}^{n} (n+1-i)\|q_i^\top X\|^2.$$

The two quantities are complementary, in the sense that the Dirichlet energy has increasing weights and TCER decreasing ones for the terms $\|q_i^\top X\|^2$. *When we learn a graph by minimizing the Dirichlet energy, we implicitly minimize the TCER, obtaining a good representability of the data using the first few eigenvectors of the Laplacian.* This is an explanation of why graph smoothing works, where we are given a graph and we minimize with respect to X. This is also why graph learning works, where we are given data X and we minimize with respect to the Laplacian L.

2 Matrix completion on graphs

The problem of finding the missing values of a matrix given a few known elements is called matrix completion. In order to solve it we need further assumptions, the most common one being that it is low rank. Although under a low rank assumption the problem is NP-hard, Candès and Recht [2009] showed that we can obtain an exact solution with high probability if the number of observed elements is sufficiently large. In this chapter, we introduce a novel matrix completion model that makes use of side information about proximities between rows and between columns in order to achieve better reconstruction quality.

The assumption of known connectivity between rows or columns makes sense in real-world problems like in recommender systems, where there are communities of people sharing preferences, while products form clusters that receive similar ratings. Our main goal is to find a low-rank solution that is structured by the proximities of rows and columns encoded by graphs. Given the graphs of products (for example movies) and users, we can assume that the corresponding signals are smooth thereon. We formulate our matrix recovery model as a convex non-smooth optimization problem, for which we provide an efficient alternating direction multipliers method in order to optimize it. We study and evaluate our model on synthetic and real data, showing that adding the graphs' side information provides better performance than the standard matrix completion model in many situations.

2.1 Related work

Reconstructing signals exactly from very few measurements was first explored by the field of compressed sensing. Exact recovery from few measurements is actually possible if the signal is *sparse* in some representation domain. A related essential question has been recently considered for matrices: is it possible to reconstruct matrices exactly from very few observations? It appears that exact recovery is also possible in this setting if the matrix is *low-rank*. The problem of low-rank recovery from sparse observations is referred as the *matrix completion* problem. Several important real-world problems can be cast as a matrix completion problem, including remote sensing [Schmidt, 1986], recommendation systems [Srebro et al., 2004], and

system identification [Liu and Vandenberghe, 2009]. Throughout this chapter we will consider the application to recommendation systems as an illustration of the matrix completion problem. Such systems have indeed become very common in many applications such as movie or product recommendation (e.g. Netflix, Amazon, Facebook, and Apple).

The Netflix recommendation system tries to predict ratings of movies never seen by users. Collaborative filtering is widely used today to solve this problem [Breese et al., 1998], inferring recommendations by finding similar rating patterns and using them to complete missing values. This is typically a matrix completion problem where the unknown values of the matrix are computed by finding a low-rank matrix that fits the given entries.

How much information is needed for the exact recovery of low-rank matrices? In the case of random uniformly sampled entries without noise, Candès and Recht [2009] showed that, to guarantee perfect recovery, the number of observed entries must be larger than $cn^{1.2}r\log n$ for $n \times n$ matrices of rank r (this bound has been refined more recently, see [Recht, 2011] and references therein). The case of noisy observations was studied in [Candès and Plan, 2010; Negahban and Wainwright, 2012], while a non-uniform sampling setting was considered by Salakhutdinov and Srebro [2010]. In this chapter, we propose to use additional information about rows and columns of the matrix to further improve the matrix completion solution, especially in the noisy case.

In the standard matrix completion problem, rows and columns are assumed to be completely *unorganized*. However, in many real-world problems like the one of Netflix, there exist relationships between users, (reflecting their behavioral similarities) and movies (such as their genre, release year, actors, origin country, etc). This information can be taken advantage of, since people sharing the same tastes for a class of movies are likely to rate them similarly. We make use of *graphs* to encode relationships between users and movies and we introduce a new reconstruction model called *matrix completion on graphs*. Our main goal is to find a low-rank matrix that is *structured* by the proximities between users and movies.

Introducing structure in sparse recovery problems is not new in the literature of compressed sensing [Huang et al., 2009; Baraniuk et al., 2010; Jenatton et al., 2011], while similar structure inducing regularization has been proposed for factorized models for matrix completion [Ma et al., 2011]. Yet, introducing structures via graphs in the convex low-rank matrix recovery setting is novel. We note that a large class of recommendation systems, called content-based filtering, use graphs and clustering techniques to make predictions [Huang et al., 2002]. Along this line, our proposed methodology can be seen as a hybrid recommendation system that combines collaborative filtering (low-rank property) and content-based filtering (graphs of users and movies).

We borrow ideas from the field of manifold learning [Belkin and Niyogi, 2001, 2003] and force the solution to be smooth on the graphs of users and movies. When data resides on a manifold, it is computationally more convenient to work with the corresponding graph, that can be seen as the (non-uniform) discretization of the latter. In the case of clustered data, we can see

each cluster to be a different manifold that corresponds to a community in the graph. In any case, we use the graph Dirichlet energy (1.2), and show that the proposed model leads to a convex *non-smooth* optimization problem. Convexity is a desired property in order for the solution to be unique. Non-smoothness can be challenging, however, our problem belongs to the class of ℓ_1-type optimization problems, for which several recently proposed efficient solvers exist [Boyd et al., 2011; Combettes and Pesquet, 2011; Nesterov and Nemirovski, 2013]. The corresponding algorithm is derived in Section 2.4. It is tested on synthetic and real data [Miller et al., 2003] in Section 2.5.2.

2.2 Original matrix completion problem

The problem of matrix completion is to find the values of an $m \times n$ matrix M given a sparse set Ω of observations $M_{ij} : (i, j) \in \Omega \subseteq \{1, \dots, m\} \times \{1, \dots, n\}$. Problems of this kind are often encountered in recommender system applications, the most famous of which is the Netflix problem, in which one tries to predict the rating that n users (columns of M) would give to m films (rows of M), given only a few ratings provided by each user. A particularly popular model is to assume that the ratings are affected by a few factors, resulting in a low-rank matrix. This leads to the rank minimization problem

$$\min_{X \in \mathbb{R}^{m \times n}} \operatorname{rank}(X) \quad \text{s.t.} \quad \mathcal{A}_\Omega(X) = \mathcal{A}_\Omega(M), \tag{2.1}$$

where $\mathcal{A}_\Omega(M) = (M_{ij \in \Omega})$ denotes the observed elements of M. Problem (2.1) is NP-hard. However, replacing $\operatorname{rank}(X)$ with its convex surrogate known as the *nuclear* or *trace norm* [Srebro et al., 2004] $\|X\|_* = \operatorname{tr}((XX^\top)^{1/2}) = \sum_k \sigma_k$, where σ_k are singular values of X, one obtains a semidefinite program

$$\min_{X \in \mathbb{R}^{m \times n}} \|X\|_* \quad \text{s.t.} \quad \mathcal{A}_\Omega(X) = \mathcal{A}_\Omega(M). \tag{2.2}$$

Under the assumption that M is sufficiently incoherent, if the indices Ω are uniformly distributed and $|\Omega|$ is sufficiently large, the minimizer of (2.2) is unique and coincides with the minimizer of (2.1) [Candès and Recht, 2009; Recht, 2011]. If in addition the observations are contaminated by noise, one can reformulate problem (2.2) as

$$\min_{X \in \mathbb{R}^{m \times n}} \gamma_n \|X\|_* + \ell\left(\mathcal{A}_\Omega(X), \mathcal{A}_\Omega(M)\right), \tag{2.3}$$

where the data term ℓ in general depends on the type of noise assumed, and $\gamma_n \in \mathbb{R}_+$ is a parameter that depends on the amount of noise assumed. If ℓ is the squared Frobenius norm $\|A_\Omega \circ (X - M)\|_F^2$ (A_Ω here is the observations mask matrix, $\circ$ the Hadamard product), the distance between the solution of (2.3) and M can be bounded by the norm of the noise

[Candès and Plan, 2010].

One notable disadvantage of problems (2.2-2.3) is the assumption of a "good" distribution of the observed elements Ω, which implies, in the movie rating example, that on average each user rates an equal number of movies, and each movie is rated by an equal number of users. In practice, this uniformity assumption is far from being realistic: for instance, in the Netflix dataset, the number of movie ratings of different users varies from 5 to 10^4. When the sampling is non-uniform, the quality of the lower bound on $|\Omega|$ deteriorates dramatically [Salakhutdinov and Srebro, 2010], from approximately constant number of observations per row in the former case, to an order of $n^{1/3} - n^{1/2}$ in the latter. In such settings, Salakhutdinov and Srebro [Salakhutdinov and Srebro, 2010] suggest using the weighted nuclear norm $\|X\|_{*(p,q)} = \|\mathrm{diag}(\sqrt{p})X\mathrm{diag}(\sqrt{q})\|_*$, where p and q are m- and n- dimensional row- and column-marginals of the distribution of observations, showing a significant performance improvement over the unweighted nuclear norm. Pathologically non-uniform sampling patterns, such as an entire row or column of M missing, cannot be handled. Furthermore, in many situations the number of observations might be significantly smaller than the lower bounds.

2.3 Matrix completion on graphs

Assuming that the recovered matrix should be low rank implies that its rows or columns are linearly dependent. However, this dependence is unstructured. In many situations, the rows/columns of matrix M possess additional structure that can be incorporated into the completion problem in the form of a regularization. In this work, we assume that rows/columns of M are given on vertices of graphs. In the Netflix example, the users (columns of M) are the vertices of a "social graph" whose edges represent e.g. friendship or similar tastes relations. Thus, it is reasonable to assume that connected users would give similar movie ratings, that is, interpreting the ratings as an m-dimensional vector-valued function on the n vertices of the social graph, such a function would be *smooth*.

More formally, let us be given the undirected weighted *row graph* $G_r = (\mathcal{V}_r, \mathcal{E}_r, W_r)$ and *column graph* $G_c = (\mathcal{V}_c, \mathcal{E}_c, W_c)$. Let $X \in \mathbb{R}^{m\times n}$ be a matrix, which we will regard as a collection of m-dimensional column vectors denoted with subscripts $X = [x_1, \ldots, x_n]$, or of n-dimensional row vectors denoted with superscripts $X = \left[(x^1)^\top, \ldots, (x^m)^\top\right]^\top$. Regarding the columns $x_1, \ldots, x_n$ as a vector-valued function defined on the vertices V_c, the smoothness assumption implies that $x_j \approx x_{j'}$ if $(j, j') \in E_c$. Stated differently, we want

$$\sum_{j,j'} w^c_{jj'} \|x_j - x_{j'}\|_2^2 \quad = \quad \mathrm{tr}\left(XL_cX^\top\right) \quad = \quad \|X^\top\|^2_{\mathcal{D},c} \tag{2.4}$$

to be small, where $L_c = D_c - W_c$ is the Laplacian of the column graph G_c, and $\|\cdot\|_{\mathcal{D},c}$ is the graph Dirichlet semi-norm for columns. Similarly, for the rows we get a corresponding expression $\mathrm{tr}\left(X^\top L_r X\right) = \|X\|^2_{\mathcal{D},r}$ with the Laplacian L_r of the row graph G_r. These smoothness terms are

added to the matrix completion problem as regularization terms (in the sequel, we treat the case where ℓ is the squared Frobenius norm),

$$\min_X \gamma_n \|X\|_* + \ell\left(\mathcal{A}_\Omega(X), \mathcal{A}_\Omega(M)\right) + \frac{\gamma_r}{2}\|X\|^2_{\mathcal{D},r} + \frac{\gamma_c}{2}\|X^\top\|^2_{\mathcal{D},c}. \tag{2.5}$$

2.3.1 Relation to simultaneous sparsity models.

Low rank promoted by the nuclear norm implies sparsity in the space of outer products of the singular vectors, i.e., in the singular value decomposition $X = \sum_k \sigma_k u_k v_k^\top$ only a few coefficients σ_i are non-zero. Recent works [Oymak et al., 2012] proposed imposing additional structure constraints, considering matrices that are simultaneously low-rank (i.e., sparse in the space of singular vectors outer products) and sparse (in the original representation). Our regularization can also be considered as a kind of simultaneously structured model. The column smoothness prior (2.4) makes the rows of X be close to the eigenvectors of the column graph Laplacian L_c, i.e., each row of X can be expressed as a linear combination of a few eigenvectors of L_c (see Section 1.4.2). This can be interpreted as row-wise sparsity of X in the column graph Laplacian eigenbasis. Similarly, the row smoothness prior results in column-wise sparsity of X in the row graph Laplacian eigenbasis. Overall, the whole model (2.5) promotes simultaneous sparsity of X in the singular vectors outer product space, and row/column-wise sparsity in the respective Laplacian eigenspaces.

2.4 Optimization

Problems like (2.5) containing non-differential terms cannot be optimized with pure gradient based approaches, while proximal based methods can be applied instead. We use the *Alternating Direction Method of Multipliers (ADMM)* that has seen great success recently [Boyd et al., 2011][1] by first introducing the equivalent splitting version of (2.5)

$$\min_{X,Y\in\mathbb{R}^{m\times n}} \underbrace{\gamma_n\|X\|_*}_{F(X)} + \underbrace{\frac{1}{2}\|A_\Omega \circ (Y-M)\|_F^2 + \frac{\gamma_r}{2}\|Y\|^2_{\mathcal{D},r} + \frac{\gamma_c}{2}\|Y^\top\|^2_{\mathcal{D},c}}_{G(Y)} \quad \text{s.t.} \quad X = Y. \tag{2.6}$$

This splitting step followed by an augmented Lagrangian method to handle the linear equality constraint is what constitutes ADMM. The success of ADMM for ℓ_1 problems is mainly due to the fact that it does not require an exact solution for the iterative sub-optimization problems, but rather an approximate solution. The augmented Lagrangian of (2.6) is

$$\mathcal{L}(X,Y,Z) \;=\; F(X) + G(Y) + \mathrm{tr}\left(Z^\top(X-Y)\right)\rangle + \frac{\rho}{2}\|X-Y\|_F^2.$$

[1] Other choices could include for example *fast iterative soft thresholding* [Beck and Teboulle, 2009].

Both F and G are closed, proper and convex, and since we have no inequality constraints, Slater's conditions hold and therefore we have strong duality. Then $(X^\star, Y^\star)$ and $Z^\star$ are primal-dual optimal if $(X^\star, Y^\star, Z^\star)$ is a saddle point of the augmented Lagrangian $\mathcal{L}$, i.e.

$$\sup_Z \inf_{X,Y} \mathcal{L}(X,Y,Z) \;=\; \mathcal{L}(X^\star,Y^\star,Z^\star) \;=\; \inf_{X,Y} \sup_Z \mathcal{L}(X,Y,Z) \;\Rightarrow$$

$$\mathcal{L}(X^\star,Y^\star,Z) \;\leq\; \mathcal{L}(X^\star,Y^\star,Z^\star) \;\leq\; \mathcal{L}(X,Y,Z^\star) \quad \forall X,Y,Z.$$

ADMM finds a saddle point with the following iterative scheme

$$X^{k+1} = \operatorname*{argmin}_X \mathcal{L}(X, Y^k, Z^k), \tag{2.7}$$

$$Y^{k+1} = \operatorname*{argmin}_Y \mathcal{L}(X^{k+1}, Y, Z^k), \tag{2.8}$$

$$Z^{k+1} = Z^k + \rho(X^{k+1} - Y^{k+1}). \tag{2.9}$$

Eventually the convergence of the proposed ADMM algorithm (2.7)-(2.9) can be studied (and likely proved) with different mathematical approaches, for example as by Cai et al. [2010].

2.4.1 Solving sub-optimization problems

ADMM algorithms can be very fast as long as we can compute fast approximate solutions to the sub-optimization problems, here (2.7) and (2.8). The first one requires finding X^{k+1} that minimizes $\mathcal{L}(X, Y^k, Z^k)$, that is[2]

$$\begin{aligned} X^{k+1} &= \operatorname*{argmin}_X \; \gamma_n \|X\|_* + \rho/2 \|X - H\|_F^2 \\ &= \text{prox}_{F/\rho}(H), \end{aligned}$$

where $H = Y^k - \rho^{-1} Z^k$. In the case of the nuclear norm, there exists a closed-form solution:

$$X^{k+1} = U \text{soft}_{\gamma_n/\rho}(\Lambda) V^\top,$$

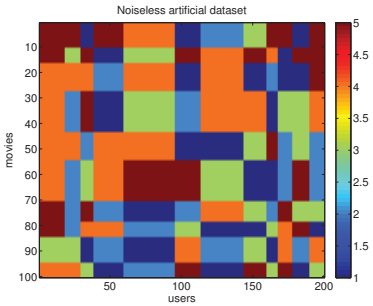
where $U\Lambda V^\top = H$ is the singular value decomposition (SVD) of H, and $\text{soft}_\eta(\lambda) = \max(0, \lambda - \eta)\frac{\lambda}{|\lambda|}$ is the soft-thresholding operator defined for $\lambda \in \mathbb{R}$.

To solve the second subproblem (2.8), we need to find Y^{k+1} that minimizes $\mathcal{L}(X^{k+1}, Y, Z^k)$, that is

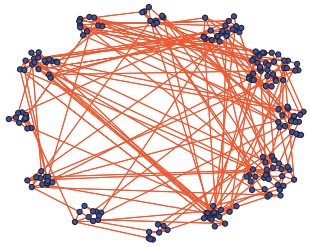
$$\begin{aligned} Y^{k+1} &= \operatorname*{argmin}_Y \; 1/2 \|A_\Omega \circ (Y - M)\|_F^2 + \gamma_r/2 \|Y\|_{\mathcal{D},r}^2 + \gamma_c/2 \|Y^\top\|_{\mathcal{D},c}^2 + \rho/2 \|Y - H\|_F^2 \\ &= \text{prox}_{G/\rho}(H), \end{aligned}$$

where $H = X^{k+1} + \rho^{-1} Z^k$. The solution of this problem can be obtained by solving a linear

[2] The *proximal operator* prox_E of function f is defined as $\text{prox}_f(X) = \text{argmin}_Y \, f(Y) + 1/2\|X - Y\|_F^2$.



(a) Rank-10 matrix M of ideal ratings



(b) Graph of users with 25% of wrong edges

Figure 2.1 – Synthetic 'Netflix' dataset

system of equations. More precisely, the optimality condition of (2.8) is

$$A_\Omega \circ (Y - M) + \gamma_r Y L_r + \gamma_c L_c Y + \rho(Y - H) = 0,$$

which can be re-written in the standard form $Ay = b$ as

$$(\tilde{A}_\Omega + \gamma_r L_r \otimes I_n + \gamma_c I_m \otimes L_c + \rho I_{mn})\mathrm{vec}(Y) = \mathrm{vec}(M + \rho H).$$

Here we have used the column stack vectorization operator $\mathrm{vec}(\cdot)$, the Kronecker product $\otimes$, and $\tilde{A}_\Omega = \mathrm{diag}(\mathrm{vec}(A_\Omega))$. We also use the known formula $\mathrm{vec}(ABC) = (C^\top \otimes A)\mathrm{vec}(B)$. Also note that A is symmetric positive semidefinite (s.p.s.d.) as the Kronecker product of two s.p.s.d. matrices, thus the conjugate gradient (CG) algorithm can be applied to compute a fast approximate solution of (2.8).

Computational complexity. The overall complexity of the algorithm is dominated by the computation of the nuclear proximal solutions by SVD, whose complexity is $O(mn^2)$ per iteration for $m > n$ [Golub and Van Loan, 2012]. The computational complexity of the CG algorithm is $O(kmn)$ for k-NN graphs.

2.5 Numerical experiments

In this section we evaluate our method under different experimental scenarios, both in artificial and on real data.

2.5.1 Synthetic 'Netflix' dataset

We start the evaluation of our matrix recovery model with a synthetic Netflix-like dataset, to study the behavior of model under controlled conditions. The artificial dataset M is generated such that it fulfills two assumptions: (1) M is *low-rank* and (2) its columns and rows are respectively *smooth* w.r.t. the column graph G_c and the row graph G_r. Figure 2.1a shows our

synthetic dataset. It is inspired by the problem of movie recommendations as elements of M are chosen to be integers from $\{1 \dots 5\}$ like in the Netflix prize problem. The matrix in Figure 2.1a is *noiseless*, showing the ideal ratings for each pair of user and movie groups.

The row graph G_r of the matrix M is constructed as follows. The rows of M are grouped into 10 *communities* of different sizes. We connect nodes within a community using a 3-nearest neighbors graph and then add different amounts of erroneous edges, that is, edges between vertices belonging to different communities. The erroneous edges form a standard Erdős-Rényi graph with variable probability. We follow the same construction process for the column graph G_c that contains 12 communities. For both graphs, binary edge weights are used. The intuition behind this choice of graphs is that *users form communities of people with similar taste*. Likewise, movies can be grouped according to their type, so that *movies of the same group obtain similar ratings*. The users graph is depicted in Fig 2.1b, where nodes of the same community are clustered together. Note that matrix M in Figure 2.1a has rank equal to the minimum of user communities and the movie communities, in this case 10.

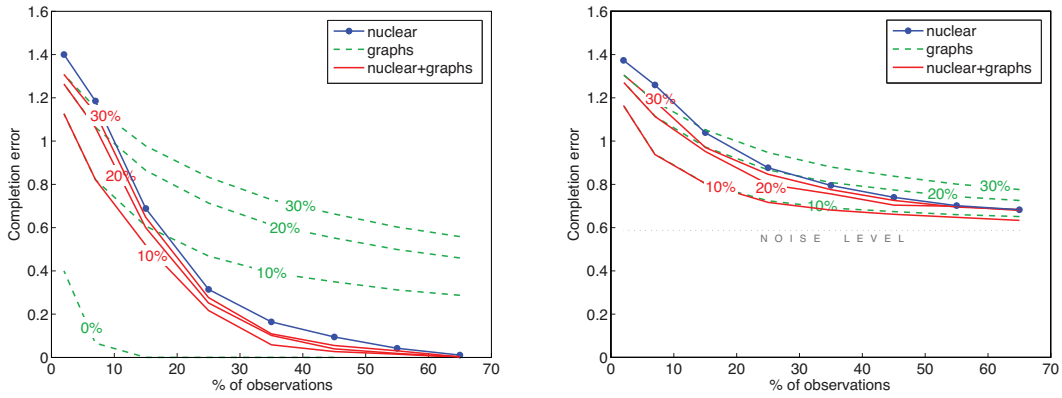
Recovery quality versus number of observations

Two standard assumptions often used in the literature on matrix completion are that the observed elements of the matrix are sampled *uniformly at random*, and that the reconstructed matrix is *perfectly* low-rank (the case that we call *noiseless*).

Noiseless case. We test the performance of our method in this setting, comparing it to the standard nuclear norm-based matrix completion (a particular case of our problem with $\gamma_c = \gamma_r = 0$) and to a method that uses only the graphs ($\gamma_n = 0$). We reconstruct the matrix M using different levels of observed values and report the reconstruction root mean squared error (RMSE) on a fixed set of 35% of the elements that was not observed. The result is depicted in Fig 2.2a. We use graphs with 10%, 20%, and 30% of erroneous edges. Noisy graphs alone (green lines) perform poorly compared to the nuclear norm reconstruction (blue line). However, when we use both graphs and nuclear norm (red lines), we obtain results that are better than any of the two alone.

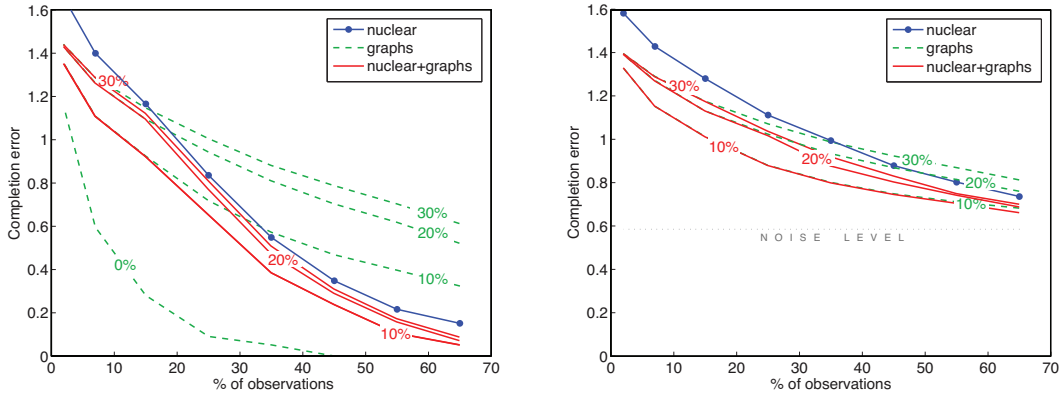
Noisy case. We add noise to M using a discretized Laplacian distribution. This type of noise models the human tendency to impulsively over- or under-rate a movie. In this case, the matrix that we try to reconstruct is *close* to low-rank, and the nuclear norm is still expected to perform well. As we see in Figure 2.2b though, if we have high-quality graphs (green line with 10% erroneous edges), we can expect the same reconstruction quality of the nuclear norm regularization by using just *half of the number of observations* and only with the graph smoothness terms (green dashed line), that computationally is much cheaper to run. In this figure, the dashed black line designates the level of added noise in the data.

Note also that even if we use connectivity information of relatively bad quality (green dashed line with 30% wrong edges), we can still benefit by combining the smoothness and the low-rank



(a) Noiseless observations (b) Noisy observations

Figure 2.2 – Matrix recovery error on synthetic 'Netflix' dataset (uniform sampling). Percentage of erroneous edges in graphs is shown on top of green and red lines.



(a) Noiseless observations (b) Noisy observations

Figure 2.3 – Matrix recovery error on synthetic 'Netflix' dataset (non-uniform sampling). Percentage of erroneous edges in graphs is shown on top of green and red lines.

regularization terms (solid red line). Therefore the combination of nuclear and graph is robust to graph construction errors for low levels of observation. However, when the observation level is high enough (> 50% for this *specific* size of matrices - note that this number may vary significantly depending on the matrix size), this benefit is lost (solid red line) and the nuclear norm regularizer (blue line) works as well without the graph smoothness terms.

Non-uniform sampling. As noted in [Salakhutdinov and Srebro, 2010], the pattern of the observed values in real datasets does not usually follow a uniform distribution. In fact, the observations are such that the rating frequencies of users and movies closely follow *power law distributions*. In our experiment, we assume a simple generative process where users and movies are independently sampled from a power law, that is $pr\left(\text{sample}\{i,j\}\right) = 1/ij$. This is a very sparse distribution with fixed expected number of observations, so we repeat this process identically s times in order to control the overall density. Our final sampling is the logical OR operator of all these s 'epochs', that follows the distribution $p\left(\{i,j\} \in \Omega\right) = 1 - \left(1 - 1/ij\right)^s$. We find that this simple sampling scheme gives results close to the actual ratings of real datasets such as the MovieLens 10M that we use in the following.

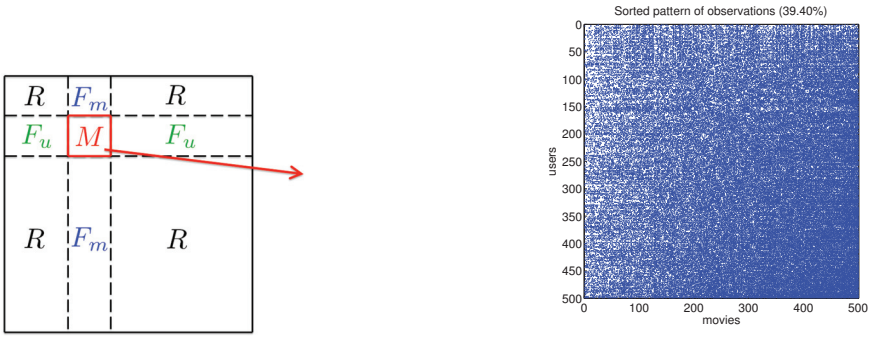
The results of our experiments for this setting are summarized in Figure 2.3a. Not surprisingly, all methods suffer from the non-uniformity of the sampling distribution. Still, the nuclear norm (blue line) crosses the line of a high-quality graph (10% green line) only after 35% observations, while in the uniform case, Figure 2.2a, this happened for less than 20% observed values. A similar behavior is exhibited for the noisy case, Figure 2.3b. There, the nuclear norm regularization quality is better than the medium-quality graph (20% green line) only for more than 45% observations, while in the uniform case, Figure 2.2b, the corresponding percentage was 25%.

2.5.2 Movielens dataset

In this section, we report experiments on real data, which appear consistent with the results on the aforementioned artificial data. We work with the widely used *MovieLens 10M* dataset [Miller et al., 2003], containing ratings ('stars') from 0.5 to 5.0 (increments of 0.5) given by 71,567 users for 10,677 movies. The density of the observations is 1.31%. In our experiments, we use a 500×500 subset of the original matrix for the reconstruction evaluation. This serves two purposes: firstly, we can choose an arbitrary density of the submatrix, and secondly, we can use ratings outside of it as features for the construction of the column and row graphs, as detailed below (see Figure 2.4a). Furthermore, the effect of non-uniformity is weaker.

The density of the observations is selected as follows. We sort the rows (users) and columns (movies) by order of increasing sampling frequency (Figure 2.4b). Then, users and movies are chosen to be close to the 99-th and 95-th percentile of their corresponding distributions.[3] The resulting 500×500 matrix has 39.4% observed values that correspond to the ratings that

[3] Since the number of movies in the full matrix is much smaller than the number of users, we keep more frequently rating users in order to have a dense features matrix when we create the users graph.



(a) Full Movielens matrix A.

(b) Submatrix $M \subset A$.

Figure 2.4 – Movielens 10M dataset. The submatrix M of A is used for training and testing. The blocks F_m and F_u are used to construct the movie and user graphs.

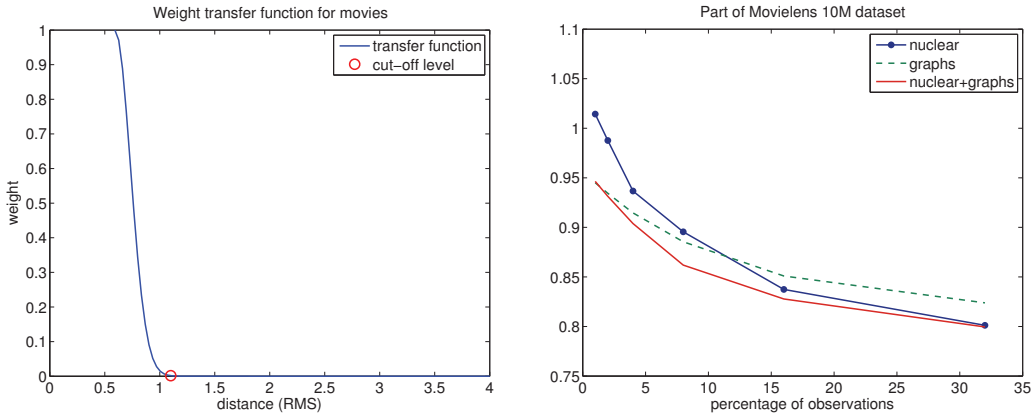
a user has given to a movie. After a row and column permutation, the original MovieLens 10M matrix A is partitioned in blocks $A = [M, F_u; F_m, R]$, where M is the 500×500 matrix that we use for our experiments (Figure 2.4a). We treat F_u as the users feature matrix, F_m as the movies feature matrix and discard the remaining matrix R.

Graph construction

Quality of graphs obviously plays an important role to our matrix recovery algorithm. A detailed analysis of how to construct good graphs is the subject of the next chapters of this thesis, but for this application we will resort to a simple, yet natural way of constructing the graphs for our setting, using the feature matrices F_u and F_m. We adapt the basic algorithm of [Belkin and Niyogi, 2003] to our setting that contains missing values.

The distance we use between two users is the RMS distance between their commonly rated movies $d_{u_{ij}} = \| \left[F_{u_i} - F_{u_j}\right]_{\Omega_{u_{ij}}} \|_{\ell_2} / \sqrt{|\Omega_{u_{ij}}|}$, $\Omega_{u_{ij}} = \Omega_{u_i} \cap \Omega_{u_j}$, where Ω_{u_i} is the set of observed movie ratings for user (row) i in F_u and $|\Omega_{u_{ij}}|$ is the number of movies in F_u that both users i and j have rated. We do the same to construct the movie distances from F_m, that is, for each pair of movies we only take into account the ratings from users that have rated both. Note that distances between movies or between users, that take values from $[0, 4.5]$ stars, share the same scale with the ratings and with the reconstruction error. Since the distances are all Euclidean, choosing the parameters of the graphs becomes more natural. The first choice we make is to use an ϵ-neighborhood graph instead of a k-NN graph. To give weights to the edges, we use a Gaussian kernel, that is, $w_{u_{ij}} = \exp\left[-\left(d_{u_{ij}} - d_{\text{min-}u}\right)^2 / \alpha\right]$ if $d_{u_{ij}} < \epsilon, 0$ otherwise. In the latter, $d_{\text{min-}u}$ denotes the minimum distance among all pairs of users and α controls how fast the weights decay as distances increase. The transfer function used for the movies graph is plotted in Figure 2.5a, while the one for users is nearly identical.

We give weight values equal to 1 for distances close to the minimum one (around 0.6 stars), while the weights decay fast as the distance increases. We choose $\epsilon = 1.1$ star, while α is



(a) Transfer function used for movies graph.

(b) Reconstruction error

Figure 2.5 – Experiments on a part of the Movielens-10M dataset

chosen so that the transfer function is already very close to 0 for $d_{u_{ij}} \to \epsilon$. This means that our model is equivalent to an inf-NN graph with the same exponential kernel. Note that the final reconstruction error is better than 1.1 star in RMS, which justifies that distances that are smaller than that are trusted. We found that the results are indeed much better when a k-NN graph is not used. A possible explanation for this is that in the case that a user that deviates a lot from the habits of other users (s)he would still have k connections. These connections would not contribute positively in the recommendations quality regarding this user.

All this being said, it is here essential to emphasize that the graphs constructed for these experiments are not optimal. We foresee that the results presented in this chapter can be further improved if one has access to detailed profile information about users and movies/products. This information is available to typical companies that sell products to users.

Results

We apply a standard cross-validation technique to evaluate the quality of our completion algorithm. For this purpose, the 39.4% observations of the 500×500 matrix are split into a fixed test set (7.4%) and a varying size training set (from 1% to 32%). We perform 5-fold cross validation to select the parameters γ_n, γ_r and γ_c of our model (2.5) and only use the test set to evaluate the performance of the final models. The recommendation error results are plotted in Figure 2.5b. The behavior of the algorithms is similar to the one exhibited by the noisy artificial data above (medium quality of graphs). For most observation levels our method combining nuclear norm and graph regularization (red line) clearly outperforms the rest. There are however two boundary phases that are noteworthy. When very few observations are available (1%) there seems to be no benefit in adding the expensive nuclear norm term in the optimization problem, as the graph regularization alone (green line) performs best. On the other hand, for very dense observation levels (32%) the nuclear norm (blue line) reaches

the performance of the combined model. In general our combined model is very robust to observation sparsity, while the standard nuclear norm model performs worse even than the much cheaper graphs-only model for up to 8% observations.

2.6 Conclusion

The main message of this chapter is that the standard low-rank matrix recovery problem can be further improved using similarity information about rows and columns. We solve an optimization problem seeking a low-rank solution that is structured by the proximity between rows and columns that form communities. As an application, our matrix completion model offers a new recommendation algorithm that combines the traditional collaborative filtering and content-based filtering tasks into one unified model. The associated convex non-smooth optimization problem is solved with a well-posed iterative ADMM scheme, which alternates between nuclear proximal operators and approximate solutions of linear systems. Artificial and real data experiments are conducted to study and validate the proposed matrix recovery model, suggesting that in real-life applications where the number of available matrix entries (ratings) is usually low and information about products and people taste is available, our model would outperform the standard matrix completion approaches.

Specifically, our model is robust to graph construction and to non-uniformly sampling of observations. Furthermore, it significantly outperforms the standard matrix completion when the number of observations is small.

3 How to learn a graph from smooth signals

In this and the next chapters we are interested in how to learn a meaningful graph from the data. Consider a matrix $X \in \mathbb{R}^{n \times m} = [x_1, \ldots, x_n]^\top$, where each row (sample) $x_i \in \mathbb{R}^m$ resides on one of n nodes of an undirected graph G. In this way, each of the m columns (feature) 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 $x_1, \ldots, x_n \in \mathbb{R}^m$ on a given weighted undirected graph is through the function

$$\frac{1}{2} \sum_{i,j} W_{ij} \|x_i - x_j\|^2 = \operatorname{tr}\left(X^\top L X\right), \tag{3.1}$$

where $W_{ij} \in \mathbb{R}_+$ denotes the weight of the edge between nodes i and j and $L = D - W$ is the graph Laplacian, $D_{ii} = \sum_j W_{ij}$ being the diagonal weighted degree matrix. In words, if two vectors x_i and x_i from a smooth set reside on two well connected nodes (i.e. W_{ij} is large), they are expected to have a small distance $\|x_i - x_j\|$ so that $\operatorname{tr}\left(X^\top L X\right)$ is small.

As discussed in the Introduction and in the previous chapter, many machine learning problems come in the form

$$\underset{X}{\text{minimize}} \;\; g(X) + \operatorname{tr}\left(X^\top L X\right), \tag{3.2}$$

where the Laplacian of a graph is used as a regularizer. *The goal of this and the rest of this thesis is to solve the complementary problem of learning a good graph:*

$$\underset{L \in \mathcal{L}}{\text{minimize}} \;\; \operatorname{tr}\left(X^\top L X\right) + f(L), \tag{3.3}$$

where $\mathcal{L}$ denotes the set of valid graph Laplacians (Section 1.1).

Why is this problem important? Firstly because it enables us to directly learn the hidden graph structure behind our data. Secondly because in problems in the form of eq. (3.2) we are often given a noisy graph, or no graph at all. Therefore, starting from the initial graph and alternating

between solving problems (3.2) and (3.3) we can at the same time get a better quality graph and solve the task of the initial problem.

In this chapter we give a general framework for graph learning from smooth signals. As we see in the next chapter, the most standard weight construction formula, eq. (4.2) can be obtained by our framework for a specific choice of prior. In Chapter 5 we similarly show that the previous state of the art graph learning model is a special case of it. In the same chapter, we use our framework to provide a general purpose graph learning model that becomes the new state of the art.

3.1 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 et al. [2008] proposed solving an ℓ_1 penalized log-likelihood problem to estimate a sparse inverse covariance with an 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 et al. [2009] considered the similar objective $\|LX\|_F^2$ and they approximately minimized it with a greedy algorithm and a relaxation. Jebara et al. [2009] learn a binary edge pattern (b-matching) from a pairwise distance matrix.

Zhang et al. [2010] alternate between problem (3.2) and a variant of (3.3). However, while initializing with a graph Laplacian L, they finally learn a s.p.s.d. matrix that is not necessarily a valid Laplacian.

More recently, Segarra et al. [2016] proposed to learn the topology of a graph given that the eigenvectors of the graph Laplacian (or adjacency matrix) are known, and specifically given by its covariance matrix. However, approximating the actual data covariance with the empirical covariance matrix requires many samples, as we show in Section 1.4 and Figure 1.5. Furthermore, when the number of nodes is big, it is not possible to obtain a full eigenvalue decomposition, while there is no scalable algorithm proposed in the paper even if the latter is given.

The works most relevant to ours are the ones by Lake and Tenenbaum [2010] and by Hu et al. [2013]; Dong et al. [2015]. 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 make its solution impractical. To the

best of our knowledge, there is no scalable algorithm in the literature that solves their model. Hu et al. [2013] propose a model that outperforms the one by Lake and Tenenbaum according to experiments by Dong et al. [2015], while they optimize blindly with respect to the graph Laplacian without exploiting its structure. In Chapter 5 we provide an analysis of their model following our general graph learning framework, that allows us to provide an efficient, scalable algorithm to solve it.

3.2 General framework

In order to learn a graph from smooth signals, we propose to rewrite problem (3.3) using the weighted adjacency matrix W and the pairwise distance matrix Z, as explained in Section 1.1:

$$\underset{W \in \mathcal{W}_m}{\text{minimize}} \quad \|W \circ Z\|_{1,1} + f(W). \tag{3.4}$$

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. Sparse graphs are desirable for large scale applications, while they are often easier to interpret when used to reveal structure.

We want to point out here that using this form instead of $\operatorname{tr}\left(X^\top LX\right)$ enables us to use any dissimilarity measure is meaningful for our data, and not only pairwise ℓ_2 distances. For example oftentimes it is more meaningful to use an ℓ_1 distance between samples, which is perfectly valid using the framework of eq. (3.4). Another example is text document data, where X is a term-document matrix with many zero values, and cosine similarities or TF-IDF represenations are more meaningful than Euclidean distances.

As we show in Section 1.1.1, maximizing smoothness is maximizing sparsity of the adjacency matrix. And since an empty adjacency matrix is not very useful, the role of $f(W)$ is very important. Its most significant tasks are the following:

1. Prevent W from obtaining the trivial solution $W = 0$.
2. Allow W to obtain zero values.
3. Impose further structure using prior information.

Preferably we would like to have a parameter that controls the sparsity level, that is easy to achieve in many models as we see in the next two chapters. In the next section we give an indicative list of possible choices of f in order to give an idea of how it affects the result.

3.2.1 Regularization

In this section we give some possible choices for f that are convex and easy to optimize within our framework, and give a brief explanation of their effect. Note that some of them can be

combined as we will see in the next chapters. While we do not use all of them (and the list could be much longer), we present them here in order to show the flexibility of the framework. The actual choice of the final function $f(W)$ should be based on the problem we want to solve, and the prior knowledge we have on the structure of the graph.

Note that by $\mathbb{1}\{\text{condition}\}$ we denote the indicator function that takes a zero value when the condition holds, and infinity otherwise. We also refer the reader to Table 1.1 for the connections between functions of W and L.

1. Constrain the graph to have at least a given amount c of "connectivity":

$$f^1(W) = \mathbb{1}\{\|W\|_1 \geq c\} = \mathbb{1}\{\|L\|_* \geq c\}. \tag{3.5}$$

   This should not be used alone with the term $\|W \circ Z\|$, as then we have a linear program that assigns weight c to the edge corresponding to the smallest pairwise distance in Z and zero everywhere else. To prevent this, it has to be combined with other regularizers like the ones below.

2. Keep each node connected to at least a neighbor:

$$f^2(W) = -\sum_i \log \sum_j W_{ij} = -\mathbf{1}^\top \log(W\mathbf{1}), \tag{3.6}$$

   where $\mathbf{1} = [1, \dots 1]^\top$, and the last logarithm is applied elementwise to the degrees vector. The log-barrier prevents the degrees $W\mathbf{1}$ from becoming zero.

3. Prevent formation of super-connected nodes:

$$f^3(W) = \sum_i \left(\sum_j W_{ij} \right)^2 = \|W\mathbf{1}\|^2. \tag{3.7}$$

   By preventing the well connected nodes from getting a too big degree, and in combination with term f_1, prevents *indirectly* the distant nodes from being assigned zero edge weights. It has the advantage of being a smooth differentiable function of W that is easy to optimize.

4. Prevent formation of too strong edges:

$$f^4(W) = \frac{1}{p} \sum_{ij} W_{ij}^p, \quad p > 1. \tag{3.8}$$

   By preventing close-by nodes from getting too big weights, and in combination with term f_1, this term prevents *indirectly* the distant nodes from being assigned zero edge weights. It is a differentiable of W (the p-norm of the vector of W to the power p).

5. Prevent many disconnected components:

$$f^5(W) = -\log\det(L + \lambda_0 I) = -\sum_{i=1}^{n} \log(\lambda_i + \lambda_0), \tag{3.9}$$

where $\lambda_i, i = 1 \dots n$ are the eigenvalues of the Laplacian and $\lambda_0 > 0$ is a small number (bigger for big noise) to make $L + \lambda_0 I$ positive definite. This term was used by Lake and Tenenbaum [2010] in a graph learning model that emerged as a MAP estimator of a data generative model.

6. Distance from initial (noisy) graph:

$$f^6(W) = \|W - W_0\|_F^2, \tag{3.10}$$
$$f^7(W) = \|W - W_0\|_{1,1}. \tag{3.11}$$

If there is prior information about a noisy version of the adjacency matrix these priors could be used for graph denoising or imposing structure preference.

3.3 Three levels of locality for graph learning

The term $\|W \circ Z\|_1$ of the objective function of our framework is simple, in the sense that it can be written as a simple sum over all edges. Any structure that relates different edges with each other in the objective function might only come from the term $f(W)$. We can classify the graph models of our framework according to this criterion of edge grouping by $f(W)$, leading to the following three levels of locality:

1. **Edge-local models**: $f(W) = \sum_{i,j} f_{i,j}\left(W_{ij}\right)$
   In these models we can split the problem in n^2 independent subproblems and solve them separately as there is no interaction between different edges. Example: $f^4(W) = \frac{1}{p}\sum_{i,j} W_{ij}^p$ is edge separable.
2. **Node-local models:** $f(W) = \sum_{i} f_i\left(W_{i,:}\right)$
   In these models each row[1] of W is grouped, containing n variables. Example: $f^2(W) = -\sum_{i=1}^{n} \log\left(\|W_{i,:}\|_1\right)$ is node-local.
3. **Global models:** $f(W)$ cannot be written as a sum of element-separable functions of W. Example: $f^5(W) = -\log\det(L + \lambda_0 I)$ cannot be written as a sum of functions operating on disjoint sets of edges.

The above classification is an indicator of the modeling power and the complexity of different models. For example, the edge-local models are weaker in terms of modeling power, as the only information used for setting a weight to an edge is its own pairwise distance. At the same time, they are the cheapest models available, and the first ones used in the literature.

[1] or column, because $W = W^\top$

While the above classification is clear, its correlation with difficulty is not absolute. As we will see in Section 5.2 for example, while the constraint that sets the sum of all edges equal to a constant leads to a global model, the latter can be solved as efficiently as node-local models. The same cannot be said for $f^5(W)$, eq. (3.9), however.

Note also that while we make a distinction between node-local and global models, for both of them changing a distance between only one pair of nodes can lead to the change of all the other weights of the graph. For node-local models, however, this happens indirectly because W is symmetric and not because of $f(W)$ itself.

In Chapter 4 we give an analysis of known and new edge-local models, in Chapter 5 we design a new node-local model, and in Chapter 6 we show how it can be scaled to big datasets.

4 Edge-local graph learning

Imagine the following scenario: *You are given the distance $Z_{ij} = \|x_i - x_j\|^2$ between nodes i and j, and you need to assign a weight to the edge between them without knowing the distances of other neighboring nodes.* How do we solve this problem? The two most common solutions used in the literature that can fit this scenario are the following [Belkin and Niyogi, 2003]:

- **ϵ-neighborhood graph:** we set a threshold ϵ and we assign $W_{ij} = 1$ if $\|x_i - x_j\|^2 < \epsilon$, 0 otherwise.
- **Exponential decay graph:** we connect the two nodes with an exponentially decaying function of the distance $W_{ij} = \exp(-\|x_i - x_j\|^2)/\sigma^2$.

While the idea of graph learning is relatively new, weighted graphs were used long before, and were constructed from the data with simple weighting rules like the ones above. As we show in this chapter, these simple edge-weighting schemes are special cases of the graph learning framework we proposed in the previous chapter. The models that correspond to this type of graph construction are the ones defined by an objective function completely separable by edges:

$$\underset{W \in \mathcal{W}_m}{\text{minimize}} \quad \|W \circ Z\|_{1,1} + f(W), \qquad f(W) = \sum_{i,j} f_{i,j}\left(W_{ij}\right).$$

In these models, because the whole objective function can be written as an independent sum over the graph edges, we can solve the problem by splitting it in the following $n \times (n-1)$ sub-problems[1]:

$$\underset{W_{ij} \in \mathbb{R}_+}{\text{minimize}} \quad W_{ij} Z_{ij} + f_{i,j}(W_{ij}), \qquad j > i. \tag{4.1}$$

Even more, depending on the choice of $f(W)$, these models might enjoy an analytic solution, and "graph learning" is reduced to standard "graph construction" like the cases discussed

[1] We have supposed for convenience that $f_{i,j} = f_{j,i}$. The non-symmetric case $f_{i,j} \neq f_{j,i}$ can easily be solved by replacing $f_{i,j}$ with $f_{i,j}/2 + f_{j,i}/2$ in eq. (4.1).

$f(W)$	Explanation
$+\|L\|_*$	Enhance sparsity (see Section 1.1.1)
$-\|L\|_*$	Prevent disconnectivity (see Section 1.1.1)
$+\|W\|_F^2$	Prevent strong edges
$+\sum_{i,j} W_{ij}^p,\ \ p>1$	Prevent strong edges, effect larger for large p
$-\sum_{i,j}\log\left(W_{ij}\right)$	Prevent zero edges
$+\sum_{i,j} W_{ij}\log\left(W_{ij}-1\right)$	Negative normalized edge entropy (prevents zero edges)

Table 4.1 – List of some convex edge-local functions $f(W)$.

before. Obviously graphs of this family are weaker in terms of modeling power, since they are completely edge-local and do not take into account at all the relationship between different edges. However, depending on the application in hand, they might be the only plausible solution given their extremely low computational complexity.

In Table 4.1 we can see some edge-separable functions and the intuition behind them.

4.1 Exponential decay graphs

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

$$W_{ij} = \exp\left(-\frac{\|x_i - x_j\|_2^2}{\sigma^2}\right). \tag{4.2}$$

We can prove that this choice of weights can be seen as the result of solving problem (3.4) with a specific prior on the weights W:

Proposition 1. The solution of the problem

$$\underset{W\in\mathcal{W}_m}{\text{minimize}}\ \ \|W\circ Z\|_{1,1} \ + \ \sigma^2\sum_{ij} W_{ij}\left(\log(W_{ij})-1\right) \tag{4.3}$$

is given by eq. (4.2).

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

What is most interesting about this result is the intuition behind the objective function. This model tries to find a graph balancing between the *sparsest* solution (first term of eq. (4.3)) and a solution with the *maximum normalized entropy* (second term). Note, however, that despite the sparsity term in the objective function, the logarithm of the second term prevents the weights from being assigned a zero value. As the result is a fully connected graph, any

sparsification has to be imposed explicitly afterwards.

4.2 From exponential decay to ϵ-neighborhood graphs

Another specifically interesting edge-local model is the following:

$$\min_{W} \ \|W \circ Z\|_{1,1} - \lambda \|L\|_* + \frac{\beta}{p} \sum_{ij} W_{ij}^p, \qquad p \in (1, \infty). \tag{4.4}$$

With a first look we can see that parameter λ should control sparsity, p should control how large are the strong edges allowed to be compared to the weak ones, and β should control the edge strength in general. In order to solve this problem, we can solve separately for each edge W_{ij}

$$\min_{W_{ij}>0} \ W_{ij} \|x_i - x_j\|^2 - \lambda W_{ij} + \frac{\beta}{p} W_{ij}^p,$$

that has the analytic solution

$$W_{ij} = \sqrt[p-1]{\frac{\max\left(0, \ \lambda - \|x_i - x_j\|^2\right)}{\beta}}. \tag{4.5}$$

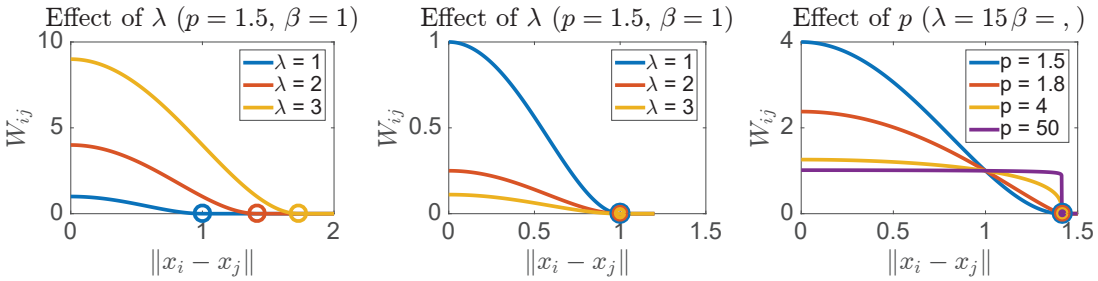
Indeed, as we see in Figure 4.1, the parameters λ, β, p effect the result as expected.

This edge weight construction, however simple it might seem, has very interesting properties. Parameter λ simply sets an upper limit to any pairwise distance, above which a zero weight is assigned. Parameter β is a multiplicative scaling parameter, that can be set to $\beta = \lambda$ in order to assign a maximum weight $W_{ij} = 1$ if $\|x_i - x_j\| = 0$. The most interesting parameter, however, is the order of the norm, p.

The value of p controls how fast is the decay of the weights as a function of the pairwise distance. The smaller the value p, the faster the weight decays given the input distance. What more, we can prove that if we set p to the limits $p \to 1^+$ and $p \to \infty$, we recover exponential decay and ϵ neighborhood graphs respectively. In other words, *model* (4.5) *interpolates between exponential decay graphs* (4.2) *and ϵ-neighborhood graphs.*

Theorem 1. The p-norm model of eq. (4.4) with solution (4.5) converges to the exponential decay model (4.2) in the limit $p \to 1^+$ and for $\beta = \lambda = \frac{\sigma^2}{p-1}$.

Proof. Let $\beta = \lambda$ so that for zero distance we assign weight equal to 1. Let us also fix them to



(a) Sparsity (and scale) (b) Scale (c) Decay speed

Figure 4.1 – The effects of the different parameters λ, β, p of the model (4.4).

$\lambda = \frac{\sigma^2}{p-1}$. Then, in the limit of $p \to 1^+$ and for $\|x_i - x_j\|_2 < \sqrt{\lambda}$ we have

$$\begin{aligned} W_{ij} &= \lim_{p\to 1^+} \sqrt[p-1]{\frac{\lambda - \|x_i - x_j\|_2^2}{\lambda}} \\ &= \lim_{p\to 1^+} \sqrt[p-1]{\frac{\frac{\sigma^2}{p-1} - \|x_i - x_j\|_2^2}{\frac{\sigma^2}{p-1}}} \\ &= \lim_{p\to 1^+} \sqrt[p-1]{1 + (p-1)\frac{-\|x_i - x_j\|_2^2}{\sigma^2}} \\ &= \exp\left(\frac{-\|x_i - x_j\|_2^2}{\sigma^2}\right) \end{aligned}$$

where we have used the known identity[2]

$$e^x = \lim_{\delta\to 0^+} (1 + \delta x)^{\frac{1}{\delta}}.$$

Note also that while we threshold to 0 any weight of distance greater than $\sqrt{\lambda}$, this does not change the result as $\lambda = \frac{\sigma^2}{p-1} \to \infty$ for $p \to 1^+$. □

Theorem 2. The p-norm model of eq. (4.4) with solution (4.5) converges to the ϵ-neighborhood model

$$W_{ij} = \begin{cases} 1 & \text{if } \|x_i - x_j\| < \epsilon, \\ 0 & \text{otherwise} \end{cases}$$

in the limit $p \to \infty$ for $\lambda = \epsilon^2$ and for any finite β.

[2] Another way to explain this limit behavior is to start from the objective function and use the identity $\log(w) = \lim_{\delta\to 0^+} \frac{1}{\delta}\left(w^\delta - 1\right)$ in order to obtain the objective function of the exponential decay weighting scheme,

$$f(W) = \sigma^2 \sum_{i,j} W_{ij}(\log W_{ij} - 1) = \lim_{p\to 1^+} \frac{\sigma^2}{p-1}\left(-\|L\|_* + \frac{1}{p}\sum_{i,j} W_{ij}^p\right).$$

Proof. We set $\lambda = \epsilon^2$. The solution (4.5) of the model can be written as

$$W_{ij} = \begin{cases} \sqrt[p-1]{\frac{\lambda - \|x_i - x_j\|^2}{\beta}} & \text{if } \|x_i - x_j\| < \epsilon, \\ 0 & \text{otherwise.} \end{cases}$$

For the first case, because the variable of the root is positive we can use the known limit

$$\lim_{p\to\infty} \sqrt[p]{\alpha} = 1, \quad \alpha > 0,$$

that concludes the proof. □

In Table 4.2 we gather some of the separable problems that enjoy analytic solutions. Note that in all cases exhibited therein we have included the term $-\|L\|_*$ that always has a translation effect on the squared pairwise distance. As an example, in the standard Laplacian with exponential decay (third row of Table 4.2), it is reasonable to choose $\lambda = \min_{ij} D_{xij}$, so that the maximum output weight is equal to $\exp(0) = 1$.

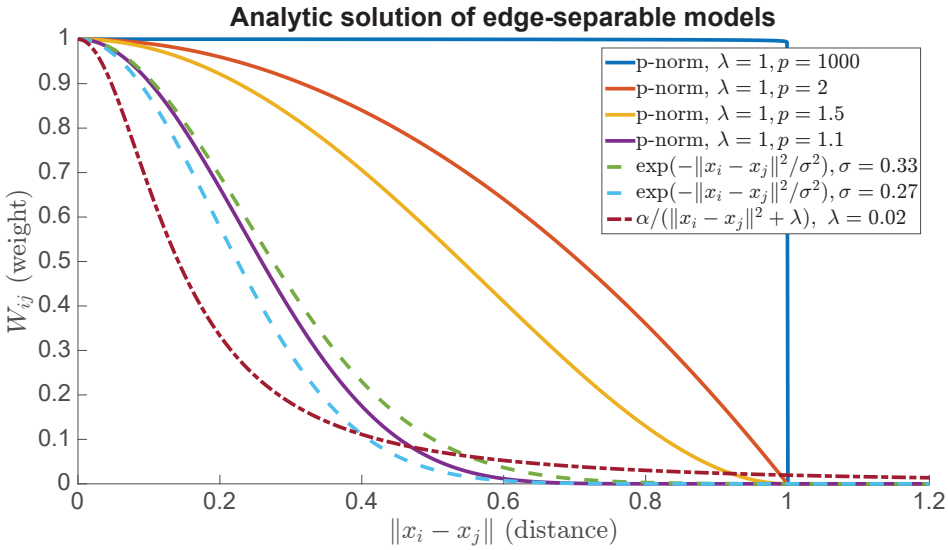


Figure 4.2 – Analytic solution of edge-local models. **Solid lines**: model of second row of Table 4.2. **Dashed lines**: model of third row, that is the standard exponential kernel. **Dashed/dotted line**: last model of Table 4.2. For all models, the parameters not shown in the legend, corresponding to multiplicative scaling, were chosen so that the weight assigned to zero distance is equal to 1. Only the models based on p-norm regularization are sparse, since they become 0 for distances $d > \sqrt{\lambda} = 1$. Note how close is the p-norm model with $p = 1.1$ to the exponential decay models. Note also that for $p = 1000$ we have a model close to the ϵ-neighborhood graph.

Regularization term $f(W)$	Analytic solution	Parameters	Sparse?
$-\lambda\|L\|_* + \frac{\beta}{2}\|W\|_F^2$	$W_{ij} = \max\left(0, \frac{\lambda - Z_{ij}}{\beta}\right)$	λ: cut-off distance β: multiplicative scaling	YES
$-\lambda\|L\|_* + \frac{\beta}{p}\sum_{ij} W_{ij}^p$, $p > 1$	$W_{ij} = \sqrt[p-1]{\frac{\max(0, \lambda - Z_{ij})}{\beta}}$	λ: cut-off distance β: multiplicative scaling p: decay speed	YES
$-\lambda\|L\|_* + \sigma^2 \sum_{ij} W_{ij}(\log W_{ij} - 1)$	$W_{ij} = e^{\frac{\lambda - Z_{ij}}{\sigma^2}}$	λ: multiplicative scaling σ: decay speed	NO
$+\lambda\|L\|_* - \alpha \sum_{ij} \log W_{ij}$	$W_{ij} = \frac{\alpha}{Z_{ij} + \lambda}$	λ: decay speed α: multiplicative scaling	NO

Table 4.2 – List of separable problems with analytic solutions and the explanation of their parameters. The term $\|L\|_*$ in the first three models prevents disconnectivity. In the last model this role is played by the "stronger" logarithmic term, and the nuclear norm has the opposite effect of preventing infinite connectivity between zero distance nodes. We use $Z_{ij} = \|x_i - x_j\|^2$.

4.3 Experiments on real data

We compare different edge-local models using real data, both for sample graphs and for feature graphs. The model with square weight decay $W_{ij} = \frac{\alpha}{Z_{ij} + \lambda}$ performed significantly worse for all cases of our experiments, we thus do not report these results.

4.3.1 Sample graphs

We first perform label propagation between images of the digits "1" and "2" of MNIST. We randomly choose 100 images of the digit "1" and 100 images of the digit "2", stacking them in a matrix $X \in \mathbb{R}^{200\times 768}$. We compute the pairwise distance matrix $Z \in \mathbb{R}^{200\times 200}$ using squared Euclidean distances between the images (features are pixels), and use different models to compute graph adjacency matrices. We keep 10% of their labels as known, and use label propagation as proposed by Zhu et al. [2003] to classify unknown images into classes "1" and "2". The quality measure for this experiment is the classification error.

Note that this problem has a particular difficulty, as explained in Figure 4.3: The distance of pairs of digits from different classes ("1" vs. "2", yellow histogram) has the same distribution as the distance between pairs of digits from the same class "2" (orange histogram). This means, that if we want the edges between images of "2" to pass the threshold, we will automatically allow many "wrong" edges that connect nodes from different classes. For this reason, the graphs have to be fairly dense in order to achieve a good connectivity and therefore a good classification quality.

In the left part of Figure 4.4 we see the quality of artificially thresholded exponential decay graphs that are often used in the literature. The horizontal axis corresponds to different

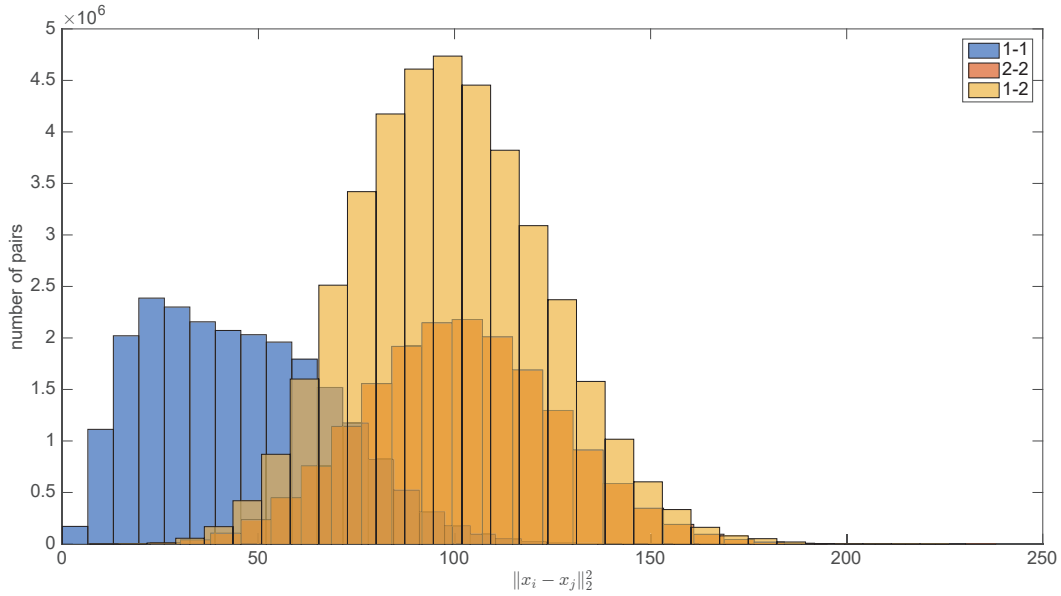
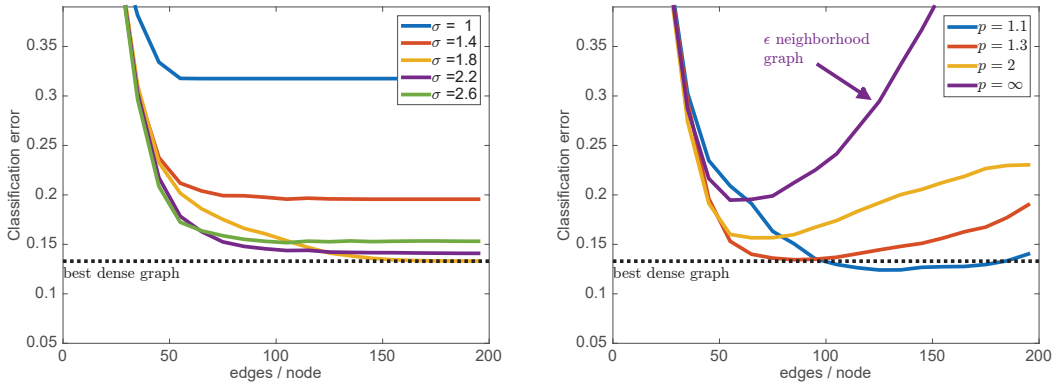


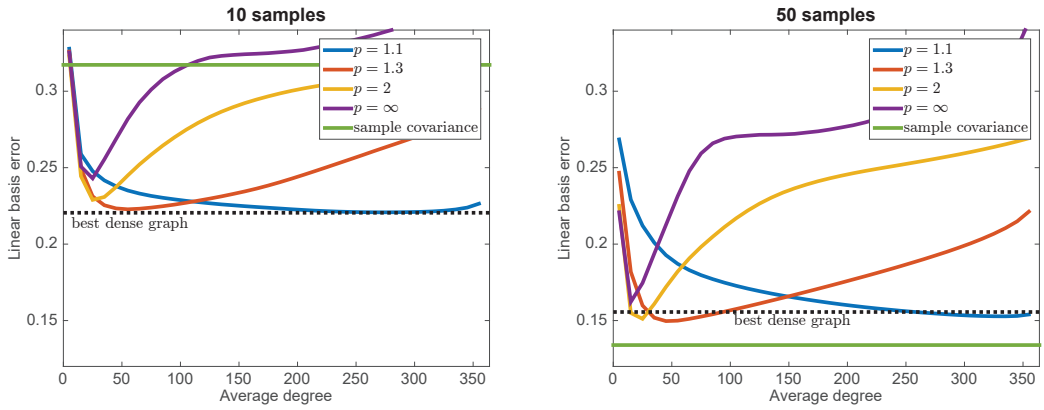
Figure 4.3 – Histogram of squared Euclidean distances between "1"s and "2"s of MNIST, using pixels as features. The distances between pairs of a "1" and a "2" have the same distribution as the distances between pairs of "2", making the problem very difficult. Moreover, the distances between pairs of "1"s are significantly smaller, making edge-local models prefer connecting these over other pairs.



(a) Thresholded dense models from eq. (4.2)

(b) Sparse models using a p-norm

Figure 4.4 – Quality of different graphs between images of MNIST 1 vs 2. The dashed line corresponds to the result of the best dense graph with $w_{ij} = \exp(-z_{ij}^2/\sigma^2)$. In the left figure we see that thresholding a dense graph does not lead to better results, while the result depends heavily on the parameter σ. However, using models that are by construction sparse (right figure, $p = 1.1$), we can achieve better classification quality with a sparser graph.



(a) Graphs learned from 10 images

(b) Graphs learned from 50 images

Figure 4.5 – Quality (total cumulative energy residual, Section 1.4) of graphs between pixels of images of the digit '2' of MNIST. The TCER is measured over the rest of the images (around 6000) of the same digit. The dashed line corresponds to the result of the best dense graph with $w_{ij} = \exp(-z_{ij}^2/\sigma^2)$.

thresholding levels, obtaining on average k neighbors per node. A first observation is that exponential decay graphs are very sensitive to the choice of σ. It seems that it is preferable to choose a σ that is larger than optimal, than one that is too small ($\sigma = 1$). Another observation is that the quality can only decay by decreasing the threshold, and the best result is achieved by a dense graph for any given choice of σ^2.

In the right part of Figure 4.4 we see results for our p-norm model that is naturally sparse, and its thresholding distance is controlled by the choice of λ. The first observation is that *our p-norm model* (4.5) *achieves the best result of all edge-local models for small values of p*, close to the limit of the exponential decay graphs (Theorem 1). Moreover, when sparser graphs are needed, we can choose slightly larger values of p and perform better for any given level of sparsity.

We also see that ϵ-neighborhood graphs (limit $p \to \infty$, Theorem 2) perform worse than any other choice of $p < \infty$. However, they still do better than thresholding an exponential graph with a wrong σ (left part of figure, blue line).

We can conclude that our p-norm model that interpolates between exponential decay and ϵ-neighborhood graphs enjoys the best of two worlds. It is naturally sparse and robust to the choice of thresholding level as long as p is in a relatively small value.

4.3.2 Feature graphs

We continue by evaluating the quality of the graphs when we need to "explain" the distribution between features of the digit "2" using very few samples. This is an experiment similar to the one of Figure 1.5, but with the weaker (and cheaper) edge-local models of this chapter.

As features we use the pixels with at least 5% of the maximum pixel energy across all images of "2"s (364 nodes). We measure how well the distribution is "explained" using the total cumulative energy residual (TCER) that we defined in Section 1.4. The distribution of the features is defined as the distribution of the rest of the images (around 6000) of the same digit.

The results of the basis quality obtained from only 10 or only 50 images of the same digit are plotted in Figure 4.5. The first observation is that for 10 samples the graphs perform better than the empirical covariance, while for 50 their quality is not as good as the one of the empirical covariance. In this case, more expressive graph models like the ones of the next chapter are needed to achieve the good quality exhibited in Figure 1.5.

We see that the quality of our p-norm model is close to or superior (for 50 samples) to the best dense exponential decay graph. However, while the dense model is not robust to the choice of parameter σ (we only plot the best of all choices of σ), the inherently sparse ones are more robust, especially the ones with small parameter p.

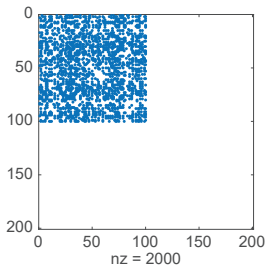
4.4 Weakness of edge-local models

While edge-local models are computationally cheap and have some theoretical interest, when it comes to practice they exhibit a significant weakness. The fact that each edge is computed independently, makes them weaker in terms of modeling, as they fail to capture any interaction between different edges, for example edges of the same node.

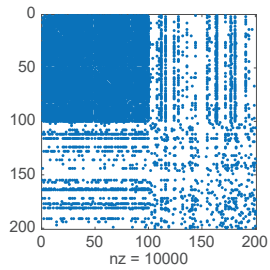
To understand the implications of edge independence, let us see how we control sparsity in such models. Given the distance between a pair of nodes, we connect them or not, depending on whether this distance is below a threshold. As each edge is considered separately, this threshold is the same across edges of the whole graph. In most real data, this can be detrimental, as samples are not uniformly distributed in the original high-dimensional space, and thresholding blindly will inevitably lead to leaving distant nodes completely disconnected.

We illustrate this effect in Figure 4.6, where we plot the sparsity pattern of the adjacency matrix for the problem MNIST "1" vs "2". In this problem, having a very sparse graph leads to connecting only images of digit "1" (upper left block), that have small distances between them (Figure 4.6a). Obviously this is not a good quality graph, which explains why the models of this chapter perform poorly in our label propagation experiments, especially for sparse graphs. When we increase the distance threshold in order to allow connections between digits of "2", we also allow the formation of "wrong" edges, connecting digits from different classes (Figure 4.6b), making also dense graphs perform poorly.

Note that this effect is universal for all edge-local models. If we assume that the weight assigned to an edge is a non-increasing function of the corresponding distance, the sparsity pattern for any given sparsity level (average number of edges per node) is independent of the choice of the function. To avoid this problem, we should use instead node-local or global models, like



(a) 10 edges per node on average.



(b) 50 edges per node on average.

Figure 4.6 – Sparsity pattern of adjacency matrix between images of digits “1” and “2” of MNIST for edge-local models. The first 100 nodes correspond to the “1”s, that have small distances and are well connected. The last 100 nodes correspond to images of “2”s, that have larger distances, with the same distribution as distances between a “1” and a “2”. **Left:** Only connections between “1”s, that have smaller distances, appear. **Right:** Connections between “2”s appear, but so do many erroneous edges (off-diagonal parts of adjacency matrix).

the ones discussed in the next chapter. These models take into account the neighborhood structure, to prioritize edges that are expected to be more meaningful.

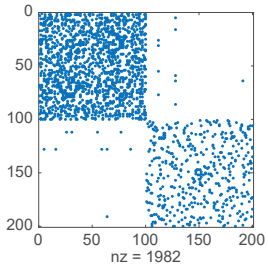
5 Node-local graph learning

As we saw in Section 4.4, learning each edge of a graph independently has limited modeling power. In this chapter, we are interested in models that take into account the *neighborhoods* of the edges, to add structure and obtain better graphs. For these models, the objective function cannot be written as an independent sum over the edges, as explained in Section 3.3. Eventually in this chapter we provide a new general purpose node-local model, that sets the new state of the art in terms of both quality and computational efficiency, and discuss two global models of the literature.

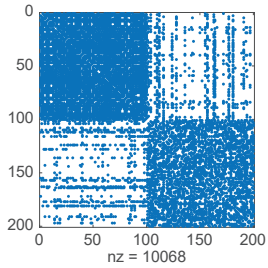
To motivate the need of taking into account the edge neighborhood, we again use the example of graphs between digits "1" and "2" of MNIST. When going from a sparser to a denser graph, we should be able to prioritize edges according to their importance. Looking at the pattern of Figure 4.6a, we can see that adding edges in the lower right block should be prioritized compared to adding edges to the upper right (and lower left) block. This becomes obvious by the fact that the first rows (nodes) already have many edges, while the last ones have no edges at all.

This problem could be avoided following two different approaches: (1) *by penalizing the formation of super-connected nodes* or (2) *by penalizing the formation of completely disconnected nodes*. The first approach is followed by the model of Section 5.2, that was the previous state-of-the-art, while the last one is followed by our *log-degrees* model presented in Section 5.1.

To illustrate the modeling power of node-local models, we plot in Figure 5.1 the pattern of an adjacency matrix using our model of the next section. Compared to Figure 4.6, we see that edges in the off-diagonal blocks are not prefered, as oposed to edges between nodes of the same class, even if the distribution of their distances might be the same (cf. Figure 4.3).



(a) 10 edges per node on average.



(b) 50 edges per node on average.

Figure 5.1 – Sparsity pattern of adjacency matrix between images of digits "1" and "2" of MNIST using our *log-degrees* node-local model of Section 5.1. The first 100 nodes correspond to the "1"s, that have small distances and are well connected. The last 100 nodes correspond to images of "2"s, that have larger distances, with the same distribution as distances between a "1" and a "2". Compared to Figure 4.6 we see that edges of the lower right block (connecting nodes of digit "2") are given higher priority, as oposed to edges between different classes in the off-diagonal blocks, even if the distribution of their distances might be the same (cf. Figure 4.3).

5.1 Our proposed model

Based on our framework (3.4) 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 (1) *each node has at least one edge with another node.* It is also desirable to (2) *have control of how sparse is the resulting graph.*

To meet these expectations, we need to incorporate the neighborhood structure of the graph in our model, that can be encoded by the vector of the node degrees. We eventually propose the following model with parameters $\alpha > 0$ and $\beta \geq 0$ controlling the shape of the edges:

$$\underset{W \in \mathcal{W}_m}{\text{minimize}} \ \|W \circ Z\|_{1,1} \ - \ \alpha \mathbf{1}^\top \log(W\mathbf{1}) \ + \ \frac{\beta}{2}\|W\|_F^2. \tag{5.1}$$

The logarithmic barrier (second term of the objective function) acts on the weighted degree vector $W\mathbf{1}$ of the nodes, unlike the model 4.3 that has a similar barrier on the edge matrix W. This means that we force the degrees to be strictly positive, but do not prevent the weights of individual 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 need an additional term in the objective function to control sparsity. As we showed with eq. (1.7), adding an ℓ_1 norm term for this reason is not very useful: it just adds the same constant to all pairwise squared distances. Instead, adding a Frobenius norm is a wiser choice in this case.

The reason we add the squared Frobenius norm of W in our objective function is to penalize the formation of large edges, but not penalize smaller ones. This leads to more dense edge patterns for larger 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 (5.1) for input distances Z and parameters α, β. Then the following property holds for any $\gamma > 0$:

$$F\left(Z,\alpha,\beta\right) = \gamma F\left(Z,\frac{\alpha}{\gamma},\beta\gamma\right) = \alpha F\left(Z,1,\alpha\beta\right). \tag{5.2}$$

Proof. See Appendix A. □

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 multiply with the scalar that gives $\|W\| = s$.

The main advantage of our model over the state of the art ℓ_2-degree based model (Section 5.2), is that it promotes connectivity by putting a log barrier directly on the node degrees. Even the sparsest possible solution, obtained with $\beta = 0$, will assign at least one edge to each node. In this case, the distant nodes will have smaller degrees (because of the first term), but will still be connected to their closest neighbor similarly to a 1-NN graph.

5.2 Fitting the state of the art in our framework

Hu et al. [2013] proposed the following ℓ_2-degree penalization model (that was also used by Dong et al. [2015] later) for learning a graph:

$$\begin{aligned} &\underset{L\in\mathcal{L}}{\text{minimize}} && \operatorname{tr}\left(X^\top L X\right) + \alpha\|L\|_F^2, \\ &\text{s.t.,} && \operatorname{tr}(L) = s. \end{aligned}$$

Parameter $s > 0$ controls the scale[1] 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.[2] Secondly, solving this optimization problem is complicated, as it has 4 constraints on L: 3 in order to constrain L in space $\mathcal{L}$, and one to keep the trace constant. Hu et al. [2013] optimize it using projected gradient descent on the space of $\mathcal{L}$, ignoring, during the gradient step, the dependence of the diagonal and off-diagonal elements, while Dong et al. [2015] do not provide an algorithm and rely on CVX (a black box general purpose optimization toolbox)

[1] Hu et al. [2013] and Dong et al. [2015] set it to n, but as we prove in Proposition 3, it can be set conveniently to any scale we want without compromising the range of edge shapes that can be obtained by different choices of α.

[2] Actually the authors of [Hu et al., 2013] proposed penalizing $\|W\|_F^2$ but ended up with $\|L\|_F^2$ because the first one "is not derivable with respect to L". However, using $\|W\|_F^2$ would have led to the much weaker edge-local model (4.4) with $p = 2$. Clearly, writing the optimization problem in terms of the weight matrix W increases interpretability and avoids confusion.

that is less than efficient.

We propose to solve this model using our framework: Using transformations of Table 1.1, we obtain the equivalent simplified model[3]

$$\begin{aligned} &\underset{W\in\mathcal{W}_m}{\text{minimize}} \ \|W\circ Z\|_{1,1} + \alpha\|W\mathbf{1}\|_2^2 + \alpha\|W\|_F^2, \\ &\text{s.t.} \qquad \|W\|_{1,1} = s. \end{aligned} \tag{5.3}$$

Using this parametrization, solving the problem becomes much simpler, as we show in Section 5.4. 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 large values, we penalize large 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 (5.3):

Proposition 3. Let $H(Z,\alpha,s)$ denote the solution of model (5.3) for input distances Z and parameters α and s. Then for $\gamma > 0$ the following properties hold:

$$H(Z+\gamma,\alpha,s) = H(Z,\alpha,s) \tag{5.4}$$

$$H(Z,\alpha,s) = \gamma H\left(Z,\alpha\gamma,\frac{s}{\gamma}\right) = sH(Z,\alpha s,1) \tag{5.5}$$

Proof. See Appendix A. □

In other words, model (5.3) 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 α.

5.3 A probabilistic perspective

Lake and Tenenbaum [2010] proposed a linear Gaussian generative model for smooth signals X given a graph with adjacency matrix W. Following the intuition behind standard sparse inverse covariance estimation, they explicitly penalize the ℓ_1 norm of W in order to obtain sparse graphs, which is the result of assuming an exponential prior over its edges. Their hierarchical model reads

$$p(W) \propto \exp\left(-\beta\|W\|_{1,1}\right), \tag{5.6}$$

$$p(x_i|W,\sigma) = \mathcal{N}\left(x_i \,\middle|\, \mathbf{0}, \left(L+\sigma^2 I\right)^{-1}\right), \tag{5.7}$$

with a graphical model like the one of Figure 5.2, left. The existence of $\sigma^2 > 0$ makes the Gaussian proper (that is, non-degenerate), as $\left(L+\sigma^2 I\right)$ is invertible. In order to learn W (and

[3] We ommit a factor $\frac{1}{2}$ of the first term, as it can be absorbed by parameter α.

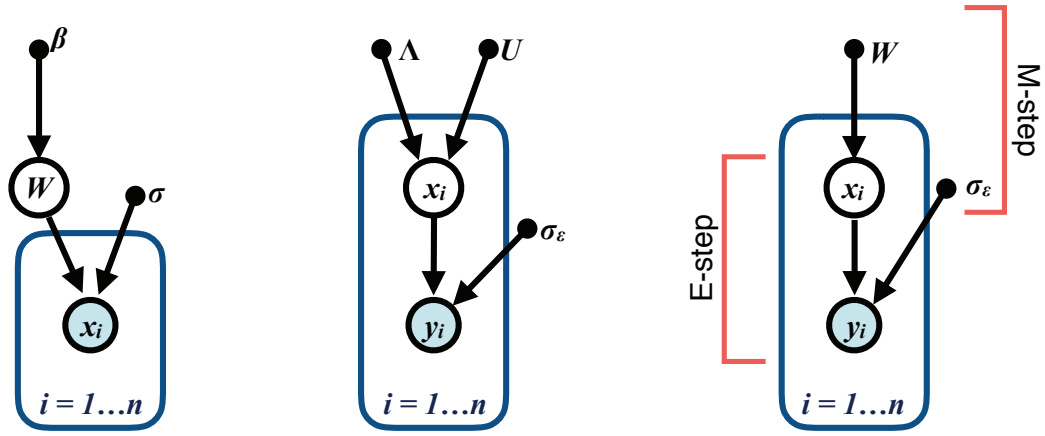


Figure 5.2 – Different graphical models of graph learning. **Left:** The model by [Lake and Tenenbaum, 2010]. **Middle:** The model by [Dong et al., 2015]. **Right:** Equivalent to the model of [Dong et al., 2015] showing E-step and M-step of an expectation maximization algorithm for graph learning from noisy observations. The M-step of this model is equivalent to the model in the left with $\sigma = 0$ (degenerate case) and without a prior on W.

σ^2), Lake and Tenenbaum propose a maximum posterior probability (MAP) estimate, that can be written using our framework[4] as the optimization problem

$$\min_{W \in \mathcal{W}_m, \sigma^2 \in \mathbb{R}_+} \quad \frac{1}{4} \left\| W \circ \left(Z + 4\beta \right) \right\|_{1,1} \; - \; \frac{1}{2} (2\pi)^{-m/2} \log \left| D - W + \sigma^2 I \right|, \tag{5.8}$$

where $|\cdot|$ denotes the determinant of a square invertible matrix. This can be seen as a "global" graph learning model, as the second term cannot be split as a sum over nodes or edges of the graph. Unfortunately, because of the same term this model is computationally expensive (see Table 5.2), needing $\mathcal{O}\left(n^3\right)$ computations per iteration. Nevertheless, being mathematically interesting, it is worth to draw connections between this and the other models of this chapter, showing effectively how the latter can be seen as computationally efficient approximations.

5.3.1 Connections with the model by Dong et al. [2015]

More recently, Dong et al. [2015] proposed a similar generative model, adding a noisy observations assumption, that leads to the hierarchy[5]

$$p(\hat{x}_i) = \mathcal{N}\left(\hat{x}_i \,\middle|\, 0, \Lambda^\dagger \right), \tag{5.9}$$

$$x_i | U, \hat{x}_i = U \hat{x}_i, \tag{5.10}$$

$$p(y_i | x_i, \sigma_\epsilon) = \mathcal{N}\left(y_i \,\middle|\, x_i, \sigma_\epsilon^2 I \right), \tag{5.11}$$

[4] The authors of [Lake and Tenenbaum, 2010] omit the factor $(2\pi)^{-m/2}/2$ of the logarithm of the determinant.

[5] For simplicity, we omit an assumed mean of x_i that should also be a smooth signal – denoted by u_x in [Dong et al., 2015].

where $x_i \in \mathbb{R}^n$ are the latent smooth signals and $y_i \in \mathbb{R}^n$ are noisy observations. As in Section 1.2, we assume the eigenvalue decomposition $L = U \operatorname{diag}([\lambda_1, \ldots, \lambda_n]) U^\top$, and $\lambda_i^\dagger$ is the pseudo-inverse of λ_i.

While this generative model (Figure 5.2, middle) is derived using a factor analysis point of view (where U is the latent loadings matrix), it is more convenient to parameterize them using the graph Laplacian W, obtaining the equivalent simpler graphical model on the right of Figure 5.2. This is because we can only allow a pair of U and Λ that obtains $L \in \mathcal{L}$ (see Section 1.1), and explicitly applying all implied constraints on U and Λ would be much more difficult if possible at all.

Given this model, it is tempting to try to simultaneously learn the latent variables X and the graph adjacency matrix W by maximizing the posterior distribution of x_i

$$p(X|Y, W, \sigma_\epsilon) = \frac{p(Y|X, \sigma_\epsilon) p(X|W)}{p(Y)}, \tag{5.12}$$

where we have used the rule of Bayes. By taking the negative logarithm of the latter, and because $p(X|W) = \mathcal{N}\left(X \middle| \mathbf{0}, L^\dagger\right)$, we can write it as a minimization problem jointly on X and W (keeping σ_ϵ fixed to avoid trivial solutions):

$$\underset{X, W \in \mathcal{W}_m}{\text{minimize}} \quad \frac{1}{2\sigma_\epsilon^2} \|X - Y\|_F^2 + \frac{1}{2} \operatorname{tr}\left(X^\top L X\right) - \frac{1}{2} (2\pi)^{-m/2} \log|L|_+, \tag{5.13}$$

where $|\cdot|_+$ is the pseudo-determinant [Rao, 2009] that takes into account only the positive eigenvalues of L.[6] Such problems are usually solved using expectation maximization, that alternates between learning the mean of X and updating the parameter W:

$$\underset{X \in \mathbb{R}^{n \times m}}{\text{minimize}} \quad \frac{1}{2\sigma_\epsilon^2} \|X - Y\|_F^2 + \frac{1}{2} \operatorname{tr}\left(X^\top L X\right) \qquad \text{(E-step)}, \tag{5.14}$$

$$\underset{W \in \mathcal{W}_m}{\text{minimize}} \quad \frac{1}{4} \|W \circ Z\|_{1,1} - \frac{1}{2} (2\pi)^{-n/2} \log|L|_+ \qquad \text{(M-step)}. \tag{5.15}$$

We see that the M-step of expectation maximization is equivalent to solving problem (5.8) for $\beta = 0$ and $\sigma \to 0$, so that the complete algorithm would cost at least the cost of (5.8) multiplied by the number of iterations until convergence. What the authors of Dong et al. [2015] do instead is keep the M-step intact and replace the E-step by solving Hu et al.'s version of graph learning (5.3) that is arguably cheaper.

5.3.2 Connections with our model

While our model (5.1) does not stem from a probabilistic model and does not assume a specific data generative model, it has an interesting connection with the one by Lake and Tenenbaum.

[6] By assuming a connected graph, we could replace this term by $-\prod_{i=2}^n \log \lambda_i$. This would only yield connected graphs, since all eigenvalues of L except for λ_1 would be bound to be positive by this logarithmic barrier.

In the latter, the log determinant term does not allow for trivial solutions $W = 0$. This is done by putting a log barrier on the eigenvalues of the Laplacian:

$$-\log|L+\sigma^2 I| = -\log\prod_{i=1}^{n}\left(\lambda_i+\sigma^2\right) = -\sum_{i=1}^{n}\log\left(\lambda_i+\sigma^2\right).$$

Instead, in our model, we use a log barrier directly on the degrees of the nodes, so that no eigenvalue decomposition is needed in every iteration. Our barrier term can be written as the log determinant of the degrees matrix D, that is the diagonal part of the graph Laplacian L:

$$-\log|D| = -\log\prod_{i=1}^{n}(d_i) = -\sum_{i=1}^{n}\log(d_i).$$

As explained by Ipsen and Lee [2011], the logarithm of the determinant of a symmetric positive definite matrix can be approximated by the logarithm of the determinant of its diagonal. More precisely, for $\widetilde{L} = L+\sigma^2 I$ and $\widetilde{D}$ its diagonal, we have:

$$\begin{aligned}-\log\left|\widetilde{L}\right| &= -\log\left|\widetilde{D}-W\right| \\ &= -\log\left|\widetilde{D}\right| - \log\left|I-\widetilde{D}^{-1}W\right| \\ &= -\log\left|\widetilde{D}\right| + \sum_{p=2}^{\infty}\frac{(-1)^p}{p}\operatorname{tr}\left(\left(\widetilde{D}^{-1}W\right)^p\right).\end{aligned}$$

For the last equality we have used the matrix Taylor expansion $\log(I+A) = \sum_{p=1}\frac{(-1)^{p-1}}{p}A^p$ that converges because $\rho(A) = \rho\left(\widetilde{D}^{-1}W\right) < 1$,[7] and the fact that $\log|I+A| = \operatorname{tr}\left(\log(I+A)\right)$. Also note that the first term of the sum (for $p=1$) is zero, and that the approximation is tighter for bigger values of σ.

5.3.3 Controlling connectivity and Laplacian eigenvalues

Given the three models we have seen until now, we can summarize their terms that principally control connectivity:

- Log-degree model (5.1): $-\sum_i \log(d_i)$
- L2-degree model (5.3): $\dfrac{\|\lambda\|_2^2}{\|\lambda\|_1}$
- Log-determinant model (5.8): $-\sum_i \log(\lambda_i+\sigma^{-2})$

The first one is a node-local model and decouples the connectivity constraint into disjoint degrees. Had the symmetricity constraint not existed, changing a node degree would only affect the edges of that node.

[7] By ρ we denote the spectral radius of a matrix.

The other two models are global: even without L being symmetric, changing only one eigenvalue changes the whole graph in general (or the whole corresponding component for disconnected graphs).

Our log-degree model has the advantage that, even for very sparse graphs, it always connects each node to at least another one. Our experiments show that for very sparse graphs, our model achieves a good connectivity even for more distant nodes.

The ℓ_2-degree model, on the other hand, implicitly controls connectivity through the sparsity of the eigenvalues. Unfortunately, as we see in our experiments, when sparse graphs are sought, this model tends to produce graphs with many disconnected nodes. This can be expected, as there is a direct sparsity term on the eigenvalue vector λ.

5.4 Optimization

An advantage of using the formulation of problem (3.4) 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 Hu et al. [2013] (and used by Dong et al. [2015]). Using these as examples, it is easy to solve many interesting models from the general framework (3.4).

In order to make optimization easier, we use the vector form representation from space $\mathcal{W}_v$ (see Table 1.1), so that the symmetricity does not have to be imposed as a constraint. We remind the reader that if $W \in \mathcal{W}_m$ is an $n \times n$ adjacency matrix, then $w \in \mathcal{W}_v$ is an $n(n-1)/2 \times 1$ vector. 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 function is

$$\underset{w \in \mathcal{W}_v}{\text{minimize}} \; f_1(w) + f_2(Kw) + f_3(w), \tag{5.16}$$

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. The main advantage of using primal dual splitting methods is that we can avoid explicitly optimizing with respect to w when a part of the function (here f_2) is defined on a linear transformation Kw of w.

In the sequel we explain how this general optimization framework can be applied to the two models of interest, leaving some 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]; 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 words, $S = K$ is the operator that takes the weights w and returns the degrees of the graph $d = W\mathbf{1} = \text{matrixform}(w)\mathbf{1}$.

In the first term of our model 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

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

where

$$
\mathbb{1}\{c\} = \begin{cases} 0, & \text{if } \ c = \text{true} \\ \infty, & \text{if } \ c = \text{false} \end{cases}
$$

is the indicator function of condition c. Given the above splitting, we need to provide the algorithm with the proximal operators of the first two functions, the gradient of the third one, as well as the operator K and the Lipschitz constant of the gradient of f_3. Note again that we only need to provide the proximal of the second function f_2 as defined directly on the dual variable d (that here is very conveniently the node degrees vector), and not w. The final algorithm for our model is given as Algorithm 1.

While many different primal dual algorithms could be used for solving such problems, we choose specifically a *forward-backward-forward* scheme, as it allows for very efficient parallelization of most computations. Specifically, note that in Algorithm 1, we can run completely independently lines in pairs: 3 with 4, 5 with 6, and lines (7, 9) in parallel with lines (8, 10).

Algorithm 1 Primal dual algorithm for log-degrees model (5.1).

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

For model (5.3) 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.

In order to solve the ℓ_2-degrees model (5.3), we provide Algorithm 2 based on the same primal dual template. 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.

Algorithm 2 Primal dual algorithm for ℓ_2-degrees model (5.3).

Input: $z, \alpha, s, w^1 \in \mathcal{W}_v$, $c^1 \in \mathbb{R}_+$, γ, tolerance ϵ
for $i = 1, \ldots, i_{max}$ **do**
  $y^i = w^i - \gamma(2\alpha(2w^i + S^\top Sw^i) + 2c^i)$
  $\bar{y}^i = c^i + \gamma(2\sum_j w^i_j)$
  $p^i = \max(0, y^i - 2\gamma z)$ ▷ Proximal of primal variable
  $\bar{p}^i = \bar{y}^i - \gamma s$ ▷ Proximal of dual variable
  $q^i = p^i - \gamma(2\alpha(2p^i + S^\top Sp^i) + 2p^i)$
  $\bar{q}^i = \bar{p}^i + \gamma(2\sum_j p^i_j)$
  $w^i = w^i - y^i + q^i$; ▷ Update weights
  $c^i = c^i - \bar{y}^i + \bar{q}^i$; ▷ Update sum of weights
  if $\|w^i - w^{i-1}\|/\|w^{i-1}\| < \epsilon$ **and**
    $|c^i - c^{i-1}|/|c^{i-1}| < \epsilon$ **then**
    break
  end if
end for

5.4.1 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]).

5.5 Experiments

We compare our model against the previous state of the art model (5.3) 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.

Table 5.1 – Performance of Different Models on Artificial Data.

	Tikhonov			Generative Model			Heat Diffusion		
	base	**Hu etal**	**Ours**	**base**	**Hu etal**	**Ours**	**base**	**Hu 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**

Other models based on the log-determinant 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. According to the experiments of Dong et al. [2015] on small graphs, the model by Lake and Tenenbaum [2010] performs worse.

5.5.1 Artificial data

The difficulty of solving problem (3.4) 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. (4.2) 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. (4.2) 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. (1.9), using one of the three filter types of Section 1.2. We normalize the Laplacian ($\|L\|_2 = 1$) so that the filters $h(\lambda)$ are defined for $\lambda \in [0, 1]$. See Table 1 (Appendix) for a summary.

1. Tikhonov: $h(\lambda) = \frac{1}{1+10\lambda}$ as in eq. (1.10).
2. Linear Gaussian Model: $h(\lambda) = 1/\sqrt{\lambda}$ if $\lambda > 0$, $h(0) = 0$ from model of eq. (1.13) ($\bar{x} = 0$).
3. Heat Diffusion: $h(\lambda) = \exp(-10\lambda)$ as eq. (1.16).

For all cases we use $n = 100$ nodes, with $m = 1000$ smooth signals as features, 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 between the learned and the ground truth graphs, 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 the use of priors on the degrees vector. 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 (4.2) 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 (4.2) with different thresholds.

Table 5.1 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 (1.13) do not yield better results in terms of graph reconstruction. Using smoother "Tikhonov" signals from eq. (1.10) or "Heat Diffusion" signals from (1.16) by setting $\lambda = 20$ 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 samples of length $m = 100$ smooth signals instead of 1000 (c.f. Table 2 of Appendix). This would be interesting to investigate theoretically.

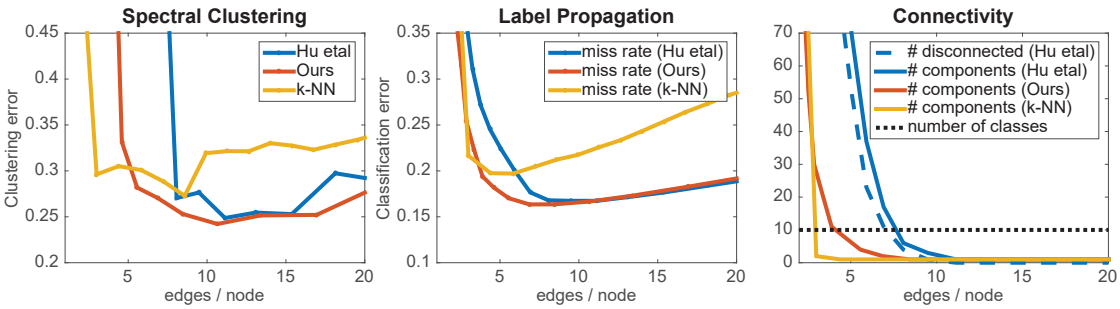


Figure 5.3 – Graph learned from 1001 USPS images. **Left**: Clustering quality. **Middle**: Label propagation quality. **Right**: Number of completely disconnected components (continuous lines) and number of disconnected nodes for ℓ_2-degrees model (blue dashed line). Our model and k-NN have no disconnected nodes.

5.5.2 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 quality. 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 degrees 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 et al. [2003] and sample the class sizes non-uniformly. For each class $i \in \{1 \dots 10\}$ 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 proposed by Ng et al. [2002] 100 times and measure the average error. We also perform label propagation 100 times using different subsets of 10% known labels and report averaged results.

In Figure 5.3 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

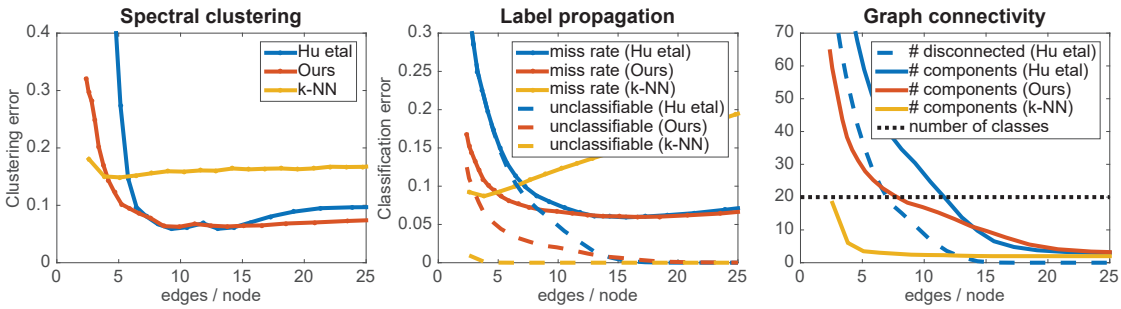


Figure 5.4 – Graph learned from non-uniformly sampled images from COIL 20. Average over 20 different samples from the same non-uniform distribution of images. **Left**: Clustering quality. **Middle**: Label propagation quality. Dashed lines are the number of nodes in components without labeled nodes. **Right**: Number of disconnected components and number of disconnected nodes (Our model and k-NN have no disconnected nodes).

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 the model by Hu et al.. 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 Hu et al. returns a graph with 35 components out of which 22 are disconnected nodes.

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[8] 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 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. 5.5. 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 et al., 2002] but without normalizing the Laplacian) with k-means 100 times. For label propagation 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.

[8] Code for both models is given as part of the open-source toolbox GSPBox by Perraudin et al. [2014] using code from UNLocBoX by Perraudin and Kalofolias.

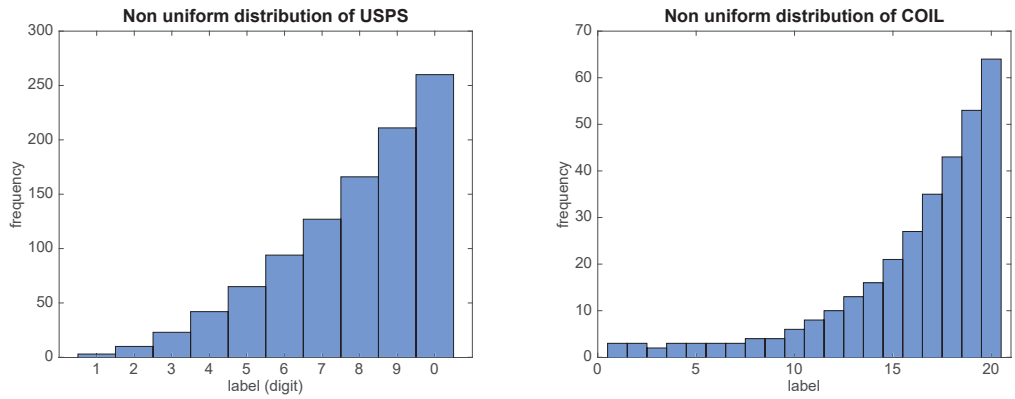


Figure 5.5 – The non uniform distributions of the USPS (**left**) and COIL 20 (**right**) datasets that we used in the experiments of this chapter.

In Figure 5.4 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 cannot be classified.

Learning the graph of MNIST digits 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 5.6 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 cannot be classified. In this case, the ℓ_2-degrees model of [Hu et al., 2013] 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.

5.5.3 Timing comparison on real data

We compare the time needed for different algorightms of graph learning. We use 25 to 200 different images of the USPS dataset to learn graphs from different algorithms and report the time in seconds. The results are given in Table 5.2. Log-det (CVX) denotes the CVX solution of the model proposed by Lake and Tenenbaum [2010]. Dong etal. (CVX) denotes the CVX solution provided by Dong et al. [2015]. We solve the same problem in the form of eq. (14) with Algorithm 2, while our model of eq. (12) is solved by Algorithm 1. CVX is a generic convex

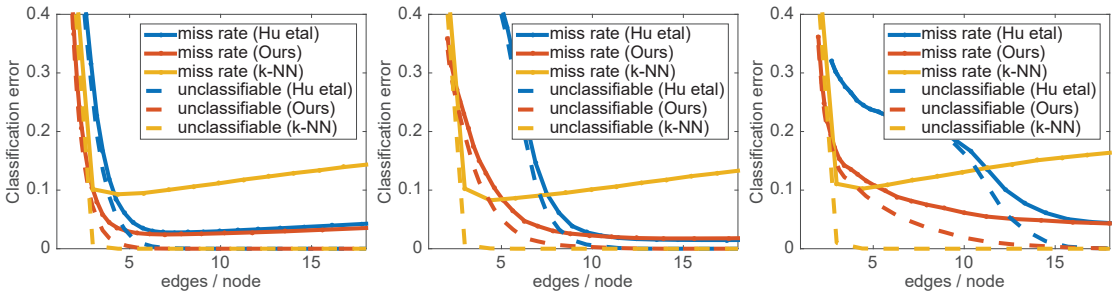


Figure 5.6 – Label propagation for the problem "1" vs. "2" of MNIST with different class size proportions: 1 to 4 **(left)**, 1 to 1 **(middle)** or 4 to 1 **(right)**. Missclassification rate for different number of edges per node.

Problem size	Log-det (CVX)	Dong etal. (CVX)	Algorithm 1	Algorithm 2
25 nodes	34.29	1.49	0.09	0.10
50 nodes	473.13	4.21	0.19	0.20
100 nodes	-	51.41	0.31	0.47
200 nodes	-	2109.87	0.93	1.75
400 nodes	-	-	1.91	11.41

Table 5.2 – Time for solving different graph learning models in seconds. We use a dash for cases that took more than an hour. Log-det is the model by Lake and Tenenbaum [2010], solved with CVX. Our algorithms are run for a tolerance of $\epsilon = 1e-5$.

optimization tool meant to be used as a black box, and therefore it struggles to solve even moderate sized problems. This effect is stronger when the log of the determinant is used in the objective function. On the other hand, our algorithms are fast proximal based methods tailored specifically for our problems, and are therefore much faster. Note also that the time needed by both our algorithms is linear to the number of iterations that varies according to the parameters and the step size. In these experiments they converged after around 300 and 2000 iterations respectively for algorithms 1 and 2.

5.6 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, implying a general framework for learning a graph. We prove that the standard Gaussian weight construction 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 ℓ_2-degree penalization model, 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 with ℓ_2-degree penalization, that was missing from the literature. Our experiments suggest that when sparse graphs are to be learned, but connectivity is crucial, our model outperforms the previous solutions.

We hope not only that our solution will be used for many applications where good quality of graphs is crucial, but also that our framework will be the trigger for new models targeting specific applications.

6 Large scale graph learning

We saw that node-local graphs (Chapter 5) give an arguably better quality than edge-local graphs (Chapter 4) and are more attractive for most applications. While the complexity of the latter ($\mathcal{O}\left(n^2\right)$ per iteration) makes them more scalable than log-determinant based global models ($\mathcal{O}\left(n^3\right)$ per iteration[1]), in order to scale it to realistically big data we have to further diminish the computational cost. A scalable large scale model should have 3 main characteristics:

1. Lower per iteration complexity.
2. No need for parameter tuning.
3. Parallelizable algorithms.

In this chapter we first show that the per iteration complexity of our model can be reduced from $\mathcal{O}\left(n^2\right)$ to $\mathcal{O}\left(nk\right)$ if we know beforehand a subset of on average k allowed edges per node. We also show how the user can have a good estimate of the model parameters so that tweeking them can be avoided. While providing a well parallelized algorithm would require a fair amount of engineering beyond the scope of this thesis, the choice of primal dual algorithms like the ones we propose here are favorable to this direction.

Ultimately, in this chapter we solve the first two of the above points. With a simple matlab implementation of our algorithm and without any parallelization, we already can learn a $10-NN$ graph of 60000 nodes within seconds on a laptop. And this, without any parameter tweeking or even data normalization needed.

We evaluate our large scale graph learning for both our log-degree model (5.1) and the ℓ_2-degree based model (5.3). The experiments show that for such applications where data is big but connections are few, our model vastly outperforms an ℓ_2-degrees one, being fair about assigning edges to more distant nodes.

[1] The state of the art method for sparse inverse covariance (not necessarily Laplacian) estimation runs in $\mathcal{O}\left(n^2k\right)$ per iteration, where nk is the number of non-zeros of the inverse covariance [Hsieh et al., 2011]. Similar algorithms should be applicable to the valid Laplacian case, even though they don't exist in the current literature.

6.1 Constrained edge pattern

It is often useful to constrain the set of allowed edges in a graph. There are two reasons we might want to do this:

- We want to reduce the complexity.
- We want to induce further constraints because of the problem we are solving. For example we have a graph with nodes corresponding to points on the map and we want to avoid any connection between geographically distant nodes.

We will denote by $\mathcal{E}^{\text{allowed}} \subseteq \{(i,j) : i < j\}$ the set of edges that are allowed to have a non-zero value. What we need to solve is the modified problem

$$\underset{W \in \widetilde{\mathcal{W}}_m}{\text{minimize}} \ \|W \circ Z\|_{1,1} \ - \ \alpha \mathbf{1}^\top \log(W\mathbf{1}) \ + \ \frac{\beta}{2}\|W\|_F^2, \tag{6.1}$$

where we optimize in the constrained set of adjacency matrices $W \in \widetilde{\mathcal{W}}_m$.

Changing Algorithm 1 so that it learns only an allowed subset of edges is not difficult. Similarly to Section 5.4, we can gather all free parameters of an adjacency matrix $\widetilde{W} \in \widetilde{\mathcal{W}}_m$ in a vector $\tilde{w} \in \widetilde{\mathcal{W}}_v$. The latter is only of size $|\mathcal{E}^{\text{allowed}}|$, that is, the number of allowed edges each counted only once.

We can continue by using the same primal dual algorithm we used for our initial model, using new primal variable $\tilde{w} = w(\mathcal{E}^{\text{allowed}})$, pairwise distances that only correspond to the allowed edges $\tilde{z} = z(\mathcal{E}^{\text{allowed}})$, and a modified linear operator $\widetilde{S} = S(:, \mathcal{E}^{\text{allowed}})$. We can solve the problem giving the objective function as the sum of the following three functions (see Section 5.4):

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

Note that the dual variable d corresponding to the node degrees has not changed. However, the linear operator K that exchanges between primal and dual spaces was replaced by $\widetilde{S}$. The cost of applying the modified operator in order to exchange between the primal and dual spaces is $\mathcal{O}\left(\mathcal{E}^{\text{allowed}}\right)$.

6.1.1 Approximate nearest neighbors graphs

If we want to learn a large scale graph and no subset of allowed edges is known by the problem, we can reduce the number of parameters by using approximate nearest neighbor graphs. The latter are based on techniques that provide an approximate set of k neighbors for each node. The user can choose the tradeoff between the approximation quality and the computation

speed.

Overall the complexity of these methods is $\mathcal{O}\left(n\log(n)d\right)$ for n nodes and d-dimensional data, and efficient implementations are available. For example see the work by Muja and Lowe [2014] and its references. The current state of the art is apparently the one reported by Malkov and Yashunin [2016]. The details of these methods are beyond the interest of this thesis, but a curious reader can find a fairly good review of the literature in the afforementioned papers.

Approximate nearest neighbors (ANN) are very convenient for our problem: they scale gracefully with the number of nodes n and provide fairly good results. Once we have a subset $\mathcal{E}^{\text{allowed}}$ of edges of cardinality $|\mathcal{E}^{\text{allowed}}| = \mathcal{O}(nk)$ we can learn a graph on this reduced subset of variables. The cost automatically drops from $\mathcal{O}\left(n^2\right)$ to $\mathcal{O}(nk)$ per iteration.

Note however that the advantage of graph learning over simpler techniques like weighted k-NNs is not only due to the weights, but also due to the ability to assign more edges for some nodes than to others. In order to not lose this advantage, if we want on average k edges per node, the constraint set of edges $\mathcal{E}^{\text{allowed}}$ should assign slightly more edges per node. Like this, our graph learning algorithm is free to choose which edges to allow and which to set to 0. If in the end of the graph learning it occurs that a large number of nodes still has all its allowed edges set to non zero values, it is an indication that we should have given a more generous set of allowed edges. In our experiments we use $3k$-ANN to compute $\mathcal{E}^{\text{allowed}}$ when we want to finally obtain a graph with on average k edges per node and obtain very good results.

Overall complexity

Asymptotically, the cost of learning a k-ANN graph is $\mathcal{O}\left(n\log(n)d\right)$ for a graph of n nodes and data with dimensionality d, while additionally learning the edge weights costs only $\mathcal{O}(kn)$. As usually $k < \log(n)d$, the asymptotical complexity of our large scale graph learning is thus dominated by the cost of computing the k-ANN, and not by the cost of learning the weights.

6.2 Automatic parameter selection

A major problem of complex models is the choice of meaningful parameters. When intuition is not enough, we might need to perform grid search and cross validation on our data, making the computational complexity significantly greater. In this section we show how this burden can be avoided for our model 5.1, that initially seems to have two important parameters α and β to be tuned.

In [Kalofolias, 2016] it is argued that model (5.1) effectively has only one parameter changing the shape of the edges, the second changing the magnitude. We can formulate this claim more clearly as follows:

Proposition 4. Let $F(Z, \alpha, \beta)$ denote the solution of model (5.1) for input distances Z and

parameters $\alpha, \beta > 0$. Then the same solution can be obtained with fixed parameters $\alpha = 1$ and $\beta = 1$, by multiplying the input distances by $\theta = \frac{1}{\sqrt{\alpha\beta}}$ and the resulting edges by $\delta = \sqrt{\frac{\alpha}{\beta}}$:

$$F\left(Z, \alpha, \beta\right) = \sqrt{\frac{\alpha}{\beta}} F\left(\frac{1}{\sqrt{\alpha\beta}} Z, 1, 1\right) = \delta F\left(\theta Z, 1, 1\right). \tag{6.2}$$

Proof. Apply Proposition 2 with $\gamma = \sqrt{\frac{\alpha}{\beta}}$ and divide all operants by the same constant $\sqrt{\alpha\beta}$. □

In words, all graphs that can be learned by our model (5.1) can be equivalently computed by multiplying the initial distances Z by a constant θ, using them to learn a graph with fixed parameters $\alpha = \beta = 1$, and multiplying all resulting edges by the same constant δ. Obviously, parameter θ controls sparsity: the larger θ is, the greater the pairwise distances between nodes and therefore the sparser the edges, as connections between distant nodes are penalized.

This proposition shows that the two parameter spaces (α, β) and (θ, δ) are equivalent. The first one was convenient to define our model, while the second one makes the sparsity analysis and the application of the model simpler. This result is important for many graph-based applications that are multiplicative scale invariant. In these applications, multiplying all edges by the same constant does not change the functionality of the graph. In other applications, we want to explicitly normalize the graph to a specific size, for example setting $\|W\|_{1,1} = n$ as in [Hu et al., 2013], or making sure that $\lambda_{\max} = 1$. In these cases, the user needs only search for the value of θ that will obtain the desired sparsity.

In this chapter we take one step further, showing how θ can be set automatically to a value that will approximately give a desired sparsity level set by the user.

6.2.1 Sparsity analysis for one node

In order to analyze the sparsity of the graphs obtained by our model, we take one step back and drop the symmetricity constraint so that we can focus on only one node of the graph. By keeping only one column w of matrix W, we arrive to the simpler optimization problem

$$\min_{w \in \mathbb{R}_+^n} \quad \theta w^\top z - \log(w^\top \mathbf{1}) + \frac{1}{2}\|w\|_2^2. \tag{6.3}$$

Note that the vector w here corresponds to only one node, having only n edges, and should not be confused with the vector $w \in \mathcal{W}_v$ in Section 5.4 that contains all $n(n-1)/2$ potential edges of the graph.

The above problem also has only one parameter θ that controls sparsity, so that larger values of θ yield sparser solutions w^*. Furthermore, *it enjoys an analytic solution if we sort the elements of z*, as stated later by Theorem 3.

Active and inactive variables

In order to solve Problem (6.3) we first introduce a slack variable l for the inequality constraint, so that the KKT optimality conditions are

$$\theta z - \frac{1}{w^\top \mathbf{1}} + w - l = 0, \tag{6.4}$$

$$w \geq 0, \tag{6.5}$$

$$l \geq 0, \tag{6.6}$$

$$l_i w_i = 0, \forall i. \tag{6.7}$$

We can reveal the form of w at the optimum by introducing the term $\lambda^* = \frac{1}{w^{*\top}\mathbf{1}}$ and rewrite (6.4) as

$$w^* = \lambda^* - \theta z + l. \tag{6.8}$$

We can split the elements of w in two sets, $\mathcal{A}$ and $\mathcal{I}$ according to the activity of the inequality constraint (6.5), so that $w_\mathcal{I} > 0$ (inactive) and $w_\mathcal{A} = \mathbf{0}$ (active). Note that at the minimum, the elements of w will also be sorted in a descending order so that $w^* = [w^*_\mathcal{I}; \mathbf{0}]$, according to Theorem 3. We first need a condition for an element of w^* to be positive as expressed in the following lemma:

Lemma 1. An element w^*_i of the solution w^* of problem (6.3) is in the active set $\mathcal{A}$ if and only if it corresponds to an element of z_i for which $\theta z_i \geq \lambda^*$.

Proof. ($\Rightarrow$): If w_i is in the active set we have $w_i = 0$ and $l_i \geq 0$, therefore from eq. (6.8) we have $\theta z_i - \lambda^* \geq 0$. ($\Leftarrow$): Suppose that there exists $i \in \mathcal{I}$ for which $\theta z_i \geq \lambda^*$. The constraint being inactive means that $w^*_i > 0$. From (6.7) we have that $l_i = 0$ and (6.8) gives $w^*_i = \lambda^* - \theta z_i \leq 0$, a contradiction. □

Analytic solution

Using Lemma 1, we can find the exact form of the solution as a function of the input "distances" of vector z. To simplify our analysis we assume that the elements of z are **sorted** in increasing order.

Theorem 3. Suppose that the input vector z is sorted in ascending order. Then the solution of problem (6.3) has the form

$$w^* = \max\left(0, \lambda^* - \theta z\right) = [\lambda^* - \theta z_\mathcal{I}; \mathbf{0}], \tag{6.9}$$

with

$$\lambda^* = \frac{\theta b_k + \sqrt{\theta^2 b_k^2 + 4k}}{2k}.$$

The set $\mathcal{I} = \{1, \dots, k\}$ corresponds to the indices of the k smallest "distances" z_i and b_k is the cumulative sum of the smallest k distances in z, $b_k = \sum_{i=1}^{k} z_i$.

Proof. As elements of θz are sorted in an ascending order, the elements of $\lambda^* - \theta z$ will be in a descending order. Furthermore, we know from Lemma 1 that all positive w_i^* will correspond to $\theta z_i < \lambda^*$. Then, supposing that $|\mathcal{I}| = k$ we have the following ordering:

$$\begin{aligned} -\theta z_1 \geq \cdots \geq -\theta z_k > \ -\lambda^* \ \geq -\theta z_{k+1} \geq \cdots \geq -\theta z_n \Rightarrow \\ \lambda^* - \theta z_1 \geq \cdots \geq \lambda^* - \theta z_k > \quad 0 \quad \geq \lambda^* - \theta z_{k+1} \geq \cdots \geq \lambda^* - \theta z_n. \end{aligned}$$

In words, the vector $\lambda^* - \theta z$ will have sorted elements so that the first k are positive and the rest are non-positive. Furthermore, we know that the elements of l in the optimal have to be 0 for all inactive variables $w_{\mathcal{I}}^*$, therefore $w_{\mathcal{I}}^* = \lambda^* - \theta z_{\mathcal{I}}$. The remaining elements of w will be 0 by definition of the active set:

$$w^* = [\underbrace{\lambda^* - \theta z_1, \cdots, \lambda^* - \theta z_k}_{w_{\mathcal{I}}^*}, \underbrace{0, \cdots, 0}_{w_{\mathcal{A}}^*}].$$

What remains is to find an expression to compute λ^* for any given z. Keeping z ordered in ascending order, let the cumulative sum of z_i be

$$b_k = \sum_{i=1}^{k} z_i. \tag{6.10}$$

Then from the definition of $\lambda^* = \frac{1}{w^{*\top}\mathbf{1}}$ and using the structure of w^* we have

$$\begin{aligned} & w^{*\top}\mathbf{1}\lambda^* = 1 \Rightarrow \\ & \left(k\lambda^* - \theta z_{\mathcal{I}}^{\top}\mathbf{1}\right)\lambda^* = 1 \Rightarrow \\ & \left(k\lambda^* - \theta b_k\right)\lambda^* = 1 \Rightarrow \\ & k(\lambda^*)^2 - \theta b_k \lambda^* - 1 = 0, \end{aligned} \tag{6.11}$$

which has only one positive solution,

$$\lambda^* = \frac{\theta b_k + \sqrt{\theta^2 b_k^2 + 4k}}{2k}. \tag{6.12}$$

□

Algorithm for one node

While Theorem 3 gives the form of the solution for a known k, the latter cannot be known apriori, and is a function of z as well. Here we propose an exact algorithm that solves the problem in $\mathcal{O}(k)$ operations. This algorithm will be needed for automatically setting the parameters for the symmetric case of graph learning.

As k is the number of non-zero edges per node, it can be assumed to be a small number, like the ones used for k nearest neighbor graphs. Therefore, it is cheap to incrementally try all values of k starting from $k = 1$ until we find the correct one, as Algorithm 3 does. Once we try a value of k for which λ is greater than θz_k but smaller or equal to θz_{k+1}, all KKT conditions hold and we have found the solution to our problem. A similar algorithm has been proposed in [Duchi et al., 2008] for projecting a vector on the probability simplex, that could be used for a similar analysis for the ℓ_2-degree constraints model (5.3).

Algorithm 3 Solver of the one-node problem, eq. (6.3).

Input: $z \in \mathbb{R}^n_{*+}$ in ascending order, $\theta \in \mathbb{R}_{*+}$
$b_0 \leftarrow 0$ ▷ Initialize cumulative sum
for $i = 1, \ldots, n$ **do** ▷ will only run $k+1$ times
  $b_i \leftarrow b_{i-1} + z_i$ ▷ Cumulative sum of z
  $\lambda_i \leftarrow \frac{\sqrt{\theta^2 b_i^2 + 4i} + \theta b_i}{2i}$
  if $\lambda_i > \theta z_i$ **then**
    $k \leftarrow i - 1$
    $\lambda^* \leftarrow \lambda_k$
    $w^* \leftarrow \max\{0, \lambda^* - \theta z\}$ ▷ k-sparse output of form $[\lambda^* - \theta z_{\mathcal{I}}; \mathbf{0}]$
    break
  end if
end for

Selecting parameters for a sparsity level

Most interestingly, using the form of the solution given by Theorem 3 we can solve the reverse problem: *If we know the distances vector z and we want a solution w^* with exactly k non-zero elements, what should the parameter θ be?* The following theorem answers this question, giving intervals for θ as a function of k, z and its cumulative sum b.

Theorem 4. By setting θ in the range $\left(\frac{1}{\sqrt{k z_{k+1}^2 - b_k z_{k+1}}}, \frac{1}{\sqrt{k z_k^2 - b_k z_k}} \right]$, the result of problem (6.3) has exactly k non-zero elements.

Proof. From the proof of Theorem 3 we know that $\|w^*\|_0 = k$ if and only if $\lambda^* \in [\theta z_k, \theta z_{k+1})$. We can rewrite this as

$$
\begin{aligned}
\theta z_k \leq \frac{\theta b_k + \sqrt{\theta^2 b_k^2 + 4k}}{2k} < \theta z_{k+1} \Leftrightarrow \\
2k\theta z_k \leq \theta b_k + \sqrt{\theta^2 b_k^2 + 4k} < 2k\theta z_{k+1} \Leftrightarrow \\
2k\theta z_k - \theta b_k \leq \sqrt{\theta^2 b_k^2 + 4k} < 2k\theta z_{k+1} - \theta b_k \Leftrightarrow \\
4k^2\theta^2 z_k^2 + \theta^2 b_k^2 - 4k\theta^2 b_k z_k \leq \theta^2 b_k^2 + 4k < 4k^2\theta^2 z_{k+1}^2 + \theta^2 b_k^2 - 4k\theta^2 b_k z_{k+1} \Leftrightarrow \\
4k^2\theta^2 z_k^2 - 4k\theta^2 b_k z_k \leq 4k < 4k^2\theta^2 z_{k+1}^2 - 4k\theta^2 b_k z_{k+1} \Leftrightarrow \\
k\theta^2 z_k^2 - \theta^2 b_k z_k \leq 1 < k\theta^2 z_{k+1}^2 - \theta^2 b_k z_{k+1} \Leftrightarrow \\
\theta^2 (k z_k^2 - b_k z_k) \leq 1 < \theta^2 (k z_{k+1}^2 - b_k z_{k+1}).
\end{aligned}
$$

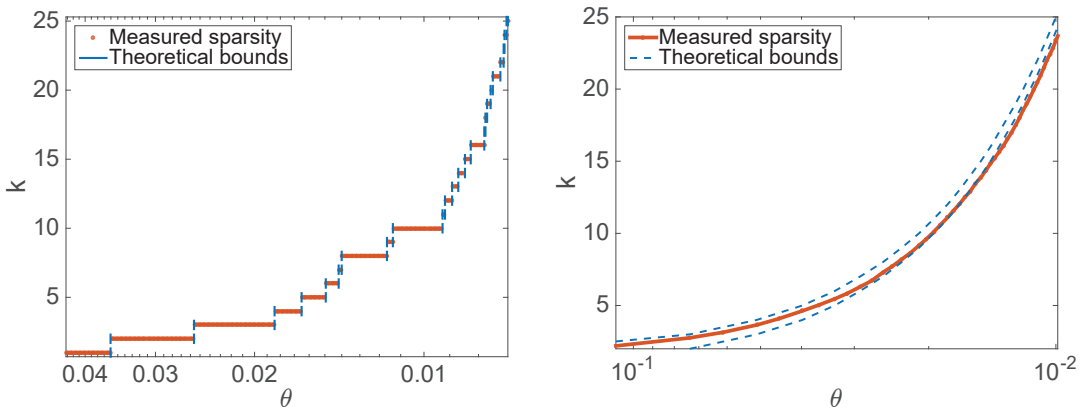


Figure 6.1 – Theoretical bounds of θ for a given sparsity level on 1000 images from MNIST. **Left**: Solving (6.3) for only one column of Z. Theorem 4 applies and for each sparsity level k gives the bounds of θ plotted in blue. **Right**: Solving (5.1) for the whole pairwise distance matrix Z. The bounds of eq. (6.13) plotted by the blue dashed line are used to approximate the sparsity of the solution. The red line is the measured sparsity (average number of edges per node) of the graphs learned using our model (5.1).

As θ is constrained to be positive, the only values that satisfy the above inequalities are the ones proposed in the theorem. □

The idea of Theorem 4 is illustrated in the left part of Figure 6.1. For this figure we have used the distances between one image of MNIST and 999 other ones. For any given sparsity level k we can know what are the intervals of the valid values of θ just by looking at the pairwise distances.

6.2.2 Parameter selection for the symmetric case

In order to approximate the parameter θ that gives the desired sparsity of W, we can use the above analysis for each row or column separately, omitting the symmetricity constraint. By using the arithmetic mean of the bounds of theta we can have a good approximation of the behaviour of the full symmetric problem. In other words, we propose to use the following intervals to obtain a graph with approximately k edges per node:

$$\begin{aligned}\theta_k \in & \left[\frac{1}{n}\sum_{j=1}^{n}\theta_{k,j}^{lower},\ \frac{1}{n}\sum_{j=1}^{n}\theta_{k,j}^{upper}\right] \\ = & \left[\sum_{j=1}^{n}\frac{1}{n\sqrt{k\hat{Z}_{k+1,j}^2 - B_{k,j}\hat{Z}_{k+1,i}}},\ \sum_{j=1}^{n}\frac{1}{n\sqrt{k\hat{Z}_{k,j}^2 - B_{k,j}\hat{Z}_{k,j}}}\right],\end{aligned} \tag{6.13}$$

where $\hat{Z}$ is obtained by sorting each column of Z in increasing order, and $B_{k,j} = \sum_{i=1}^{k}\hat{Z}_{i,j}$. The above expression is the arithmetic mean over all minimum and maximum values of $\theta_{k,j}$

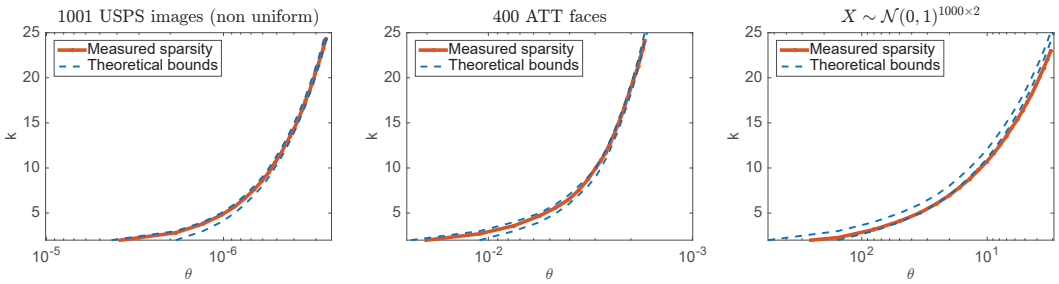


Figure 6.2 – Predicted and measured sparsity levels for different values of the parameter θ. Note that θ is plotted in logarithmic scale and decreasing. **Left**: USPS digits dataset, 1001 non uniform samples according to the distribution used in Section 5.5.2. **Middle**: ATT faces dataset, all 400 images. **Right**: Random 2-D data from a Gaussian distribution. When data comes from a uniform distribution $\mathcal{U}(0,1)^{n\times 2}$ the approximate bounds are better predictors (plot omitted).

that would give a k-sparse result $W_{:,j}$ if we were to solve problem (6.3) for each of columns separately, according to Theorem 4. Even though the above approach does not take into account the symmetricity constraints, it gives surprisingly good results in most cases, and a good starting point in the few cases where it fails.

The right part of Figure 6.1 shows the approximate bounds obtained by this approach for 1000 images of the MNIST dataset. It is clear that for this dataset we can get a very good approximation

6.3 Experiments

In this section we learn the graph between all 60000 images of the train set of MNIST and report our findings. MNIST has relatively uniform sampling between numbers of the different labels as shown in Figure 6.3 (left). However, it has one particularity, namely that the digits "1" are more densely distributed than all other digits (6.3 (right)). This will affect the results of graph learning as we see in the sequel.

Number of allowed edges and time

Running our algorithm for a subset of allowed edges $\mathcal{E}^{\text{allowed}}$ has a cost linear to the size $|\mathcal{E}^{\text{allowed}}|$ of this set as illustrated in the left part of Figure 6.4. For learning a graph with approximately 10 edges per node, we needed 20 seconds to compute $\mathcal{E}^{\text{allowed}}$, and 20 seconds to learn the final graph of 60000 nodes (around 250 iterations).

For computing $\mathcal{E}^{\text{allowed}}$ we compute an approximate nearest neighbors (ANN) graph using the publically available FLANN library[2] that implements the work of Muja and Lowe [2014]. When

[2]Compiled C code run through Matlab, available from http://www.cs.ubc.ca/research/flann/.

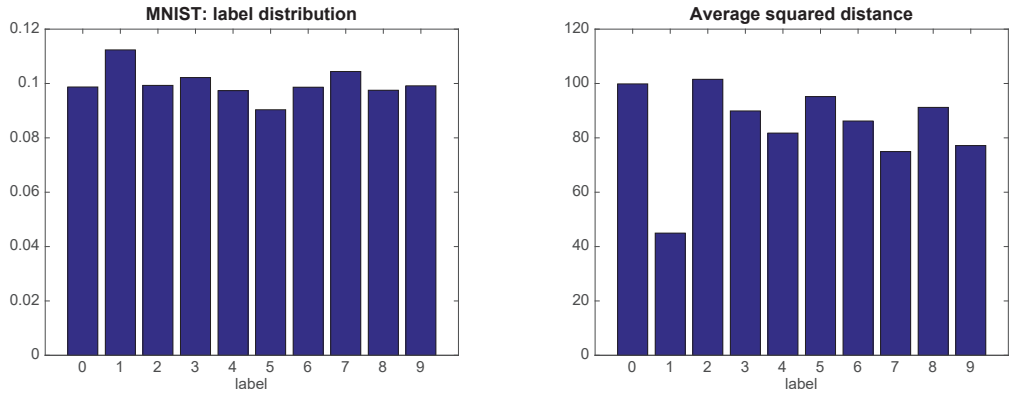


Figure 6.3 – Label frequency (**left**) and average squared distribution (**right**) of MNIST train data (60000 nodes). The distances between digits "2" are significantly smaller than distances between other digits.

we want to learn a graph with on average k neighbors per node (k-NN), we first compute a $3k$-ANN graph (with approximately $3k$ edges per node), we use its edges as $\mathcal{E}^{\text{allowed}}$, and learn a weighted graph on this subset. The choice of number of allowed edges does not only affect the time needed to learn the graph, but also its quality. If we choose a very small restricted edge set, we might prevent the final graph form learning useful edges. The percentage of nodes of the final graph that have the maximum number of edges allowed by $\mathcal{E}^{\text{allowed}}$ is a measure of this quality deterioration due to the edge restriction imposed by the ANN. We plot this percentage for our large scale log-degree model in the right part of Figure 6.4. In the horizontal axis we have k, and different lines correspond to different densities of the ANN graph, from $2k$ to $4k$. We find that an ANN of $3k$ is a good compromise between time and quality, and use it for the rest of the experiments of this section.

Connectivity quality

Next we analyze the quality of the connections retrieved by our large scale log model, by the large scale version of the ℓ_2 model (using only a subset of allowed edges like for our model), and an k-ANN graph as a baseline. We first normalize each graph so that they have equal total "connectivity" $\|W\|_{1,1} = 1$. What we plot in these histograms is the percentage of this connectivity that is spent for edges between images of each label. The last bar corresponds to the total percentage of connectivity wasted on useless edges (edges between any pair of different labels).

Given the distribution of the labels and distances in the MNIST dataset (Figure 6.3), we would like to see hopefully similar connectivity between images of each label, with a small variation due to different average distances (with label "1" more connected). Ideally, the wrong edges should be minimal.

A first obvious result is a general failure of the ℓ_2-degree model (5.3) to give consistent con-

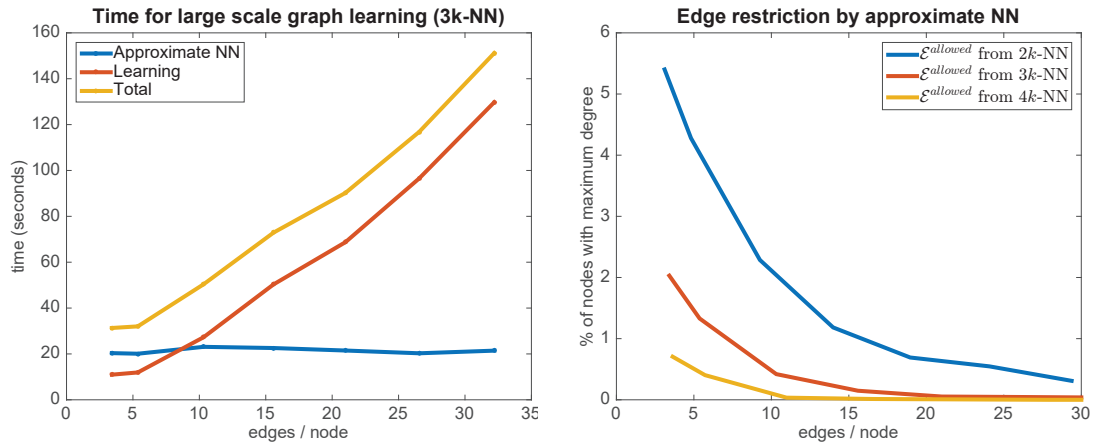


Figure 6.4 – **Left:** Time needed for learning a graph of 60000 nodes (MNIST images) using the large scale version of our model . Experiments of this Chapter were performed on a laptop (MacBook pro 2015, double core, 16GB RAM). Our algorithm converged after 250 to 450 iterations with a tolerance of $1e-4$. The time needed is linear to the number of variables, that is linear to the average degree of the graph. **Right:** Percentage of nodes that after learning have the maximum allowed number of edges.

nectivities accross different labels, being too sensitive to the average distances. Like in the experiments of the previous chapter, this model prefers to assign the vast majority of its connectivity only to the label "1" that has the smallest intra-label image distance. This effect becomes smaller when more dense graphs (30 edges per node, yellow bars) are sought.

On the other hand, our log-degree model does not suffer from this problem: It gives consistent connectivities without depending much on the graph density. Similarly for a k-NN graph, the connectivity is constant across all labels. Note, however, that we use it here only as a baseline, and that a weighted k-NN graph would also follow the intra-label distances depending on its weighting scheme.

Label propagation

We finally repeat the label propagation experiment of Chapter 5, this time on the whole 60000-node graph of MNIST. We have used only 1% of the labels as observations, and perform standard label propagation [Zhu et al., 2003] to predict the rest of the labels. The results are plotted in Figure 6.6. Given our previous observations, it is not surprising that the log model performs best. Here to have a fair comparison we used the best weighting scheme for k-NN, while the quality of the label propagation did not vary significantly by changing weighting schemes.

To explain why the ℓ_2-degree model (blue line) fails even for 30 edges per node, we plot the number of nodes that have no edge (blue dashed), only one edge, only two, or only three edges in graphs of different degrees. The other two types of graphs perform much better to this

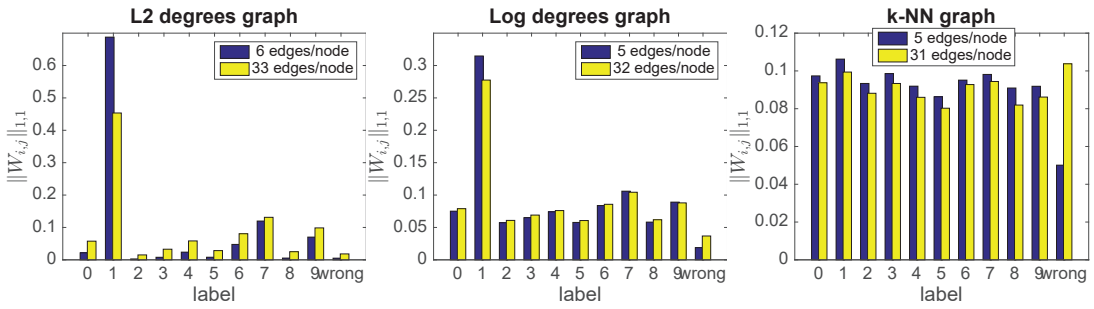


Figure 6.5 – Connectivity across different labels of the full MNIST train dataset (60000 nodes). We have normalized each graph so that $\|W\|_{1,1} = 1$ and measure how large is the percentage of the total weight for pairs of each label. The last columns correspond to the total of the wrong edges, between images of different labels. **Left:** The ℓ_2-degree penalization model (5.3) is too sensitive with respect to sample distances and does not connect well digits with relatively large distance ("2"s and "8"s) even for 30 edges per node graphs. **Middle:** Our model (6.1) while sensitive to the distance between different pairs, does not neglect to connect any cluster, even for very sparse graphs of 5 edges per node. **Right:** The k-NN graph does not take into account enough the distance and connects uniformly pairs of all labels. For 30 edges per node, it introduces many edges that should not exist.

direction.

Given the performance of the k-NN graph, one might wonder why pay the additional cost of learning a graph only for a small improvement in label propagation. Note, however, that the additional cost is not significant. Asymptotically, the cost of learning a k-ANN graph is $\mathcal{O}\left(n\log(n)d\right)$ for a graph of n nodes and data with dimensionality d, while additionally learning the edge weights costs only $\mathcal{O}(kn)$. The asymptotical complexity is thus dominated by the cost of computing a k-ANN, and not by the cost of learning the weights. For the relatively small size of the problem we solve, this corresponds to 20 seconds for the k-ANN graph (using compiled C code), and an additional 45 seconds for our graph learning (using a Matlab-only implementation). Had we used C implementation for our algorithm, the second number would be orders of magnitude less. Furthermore, for really large scale problems the asymptotical complexity would show that the bottleneck is definitely not the graph learning.

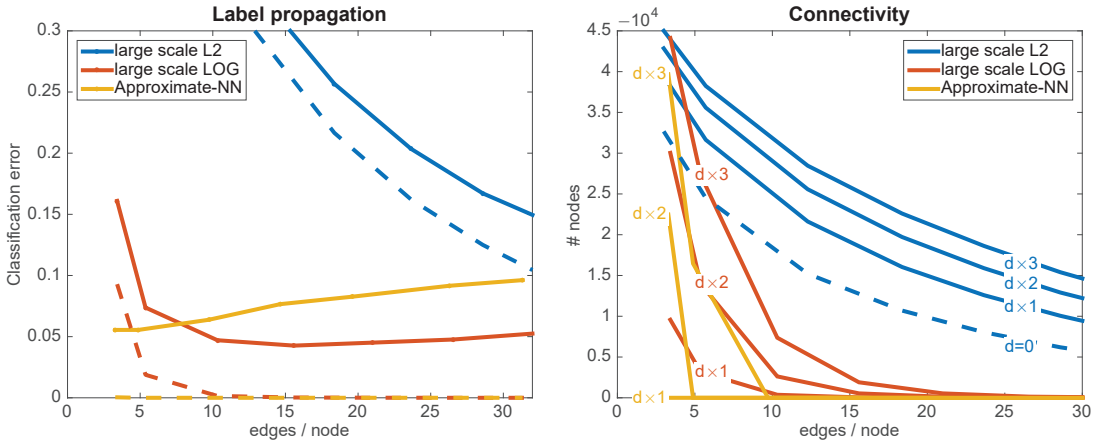


Figure 6.6 – **Left:** Label propagation with only 1% labels known on the whole MNIST train dataset (60000 nodes). **Right:** Connectivity of different graphs: We measure how many nodes have at most 1, 2 or 3 connections.

7 Learning time varying graphs

Until now we have supposed that data reside on a static graph, and we can use them to learn the relationship between different nodes. In this chapter we are interested in data that reside on a graph that changes in time. Such data can occur for example as measurements from a sensor network where the nodes are moving in space, or a social network where friendship relationships change with time.

In this case, given our data, we wish to learn not one graph, but a set of graphs that change slowly in time. This way we can capture the structure between nodes in different time windows instead of learning an "average" graph using all samples and ignoring their time dependence.

One of our objectives is, as before, to have a scalable algorithm that learns meaningful connections between our data. The algorithm we propose is based on our node-local models and scales *linearly* in the size of the variables to be learned. That is, linearly with respect to the number of different graphs we learn times the number of allowed edges in each of them. We show that such an approach explains data better than graphs or other baselines that do not take into account the variability in time.

7.1 Adding time structure

We denote our data matrix $X \in \mathbb{R}^{n \times m}$, containing columns x_j [1] as time samples of a graph with n nodes. We suppose that the graph changes slowly in time, and to make the problem tractable we further discretize time in T windows. We denote by $W^{(t)}, i = 1 \dots T$ the adjacency matrix of a window t. Obviously $T \leq m$, and while this is not binding, we suppose for simplicity that the size of the time windows is constant.

Following our assumption that the graph changes slowly in time, we are based on our previous models and add a term that penalizes fast changes of the adjacency matrices. To keep the model scalable we use a differentiable term that associates each adjacency matrix with its

[1] In this chapter we use x_j to denote column $X_{:,j}$, in order to keep notation simple.

previous one. The model we propose is the following:

$$\underset{\{W^{(t)}\in\mathcal{W}_m\}}{\text{minimize}} \sum_{t=1}^{T}\left[\theta\left\|W^{(t)}\circ Z^{(t)}\right\|_{1,1} - \mathbf{1}^\top\log\left(W^{(t)}\mathbf{1}\right) + \frac{1}{2}\left\|W^{(t)}\right\|_F^2\right] + \gamma\sum_{t=2}^{T} f_{\text{time}}^t\left(W^{(t)}|W^{(t-1)}\right), \tag{7.1}$$

where the last term is a dissimilarity measure between the adjacency matrix at time window t and the one of time window $t-1$. By $Z^{(t)}$ we denote the squared pairwise distances by eq. (1.5), computed only from the columns of X that correspond to the time window t. Setting the parameter γ to big values obtains graphs with slower change over time, while θ controls sparsity as before. Similarly, if we want to allow disconnected nodes for some of the time windows, we can solve a time-varying model based on an ℓ-2 degree penalization

$$\begin{aligned}\underset{\{W^{(t)}\in\mathcal{W}_m\}}{\text{minimize}} & \sum_{t=1}^{T}\left[\left\|W^{(t)}\circ Z^{(t)}\right\|_{1,1} + \frac{\alpha}{2}\left\|W^{(t)}\mathbf{1}\right\|^2 + \frac{\alpha}{2}\left\|W^{(t)}\right\|_F^2\right] + \gamma\sum_{t=2}^{T} f_{\text{time}}^t\left(W^{(t)}|W^{(t-1)}\right),\\ \text{s.t.,} & \quad \left\|W^{(t)}\right\|_{1,1} = s.\end{aligned} \tag{7.2}$$

7.2 Optimization

One feature of these models is that they can be optimized using the same primal-dual based algorithms we used in the original models of Chapter 5. Furthermore, they are trivially parallelizable over the time windows. To use the primal dual optimization framework, we first split the objective function and solve (5.16), using

$$\begin{aligned} f_1(w^{\text{all}}) &= \mathbb{1}\left\{w^{\text{all}}\geq 0\right\} + 2\theta\, w^{\text{all}\top} z^{\text{all}},\\ f_2(d^{\text{all}}) &= -\mathbf{1}^\top\log(d^{\text{all}}),\\ f_3(w^{\text{all}}) &= \|w^{\text{all}}\|^2 + \gamma\sum_{t=2}^{T} f_{\text{time}}^t\left(w^{(t)}|w^{(t-1)}\right),\end{aligned}$$

where we have stacked the vector forms $w^{(t)}$ of the adjacency matrix of each window in a vector w^{all}, and similarly for the distances z^{all} and the degrees d^{all}.

For the first two functions, we see that performing the proximal operator can be done in parallel for each window independently (even more, independently for each element of w^{all} or d^{all} respectively.)

In order to switch from the primal (weights) domain to the dual (degrees) domain, we need to use the linear operator S (see Section 5.4) for each window separately, i.e. $d^{(t)} = Sw^{(t)}$. Equivalently, for the space containing weights of all windows, we use the linear operator $K = \text{diag}(S,\ldots,S)$, that is a block diagonal operator with T repetitions of S in the diagonal.

The gradient of f_3 is easy to compute as well:

$$\nabla_{w^{(t)}} f_3(w^{\text{all}}) = 2w^{(t)} + \gamma \nabla_{w^{(t)}} f^t_{\text{time}}\left(w^{(t)} | w^{(t-1)}\right) + \gamma \nabla_{w^{(t)}} f^{t+1}_{\text{time}}\left(w^{(t+1)} | w^{(t)}\right),$$

for example for $f^t_{\text{time}}\left(W^{(t)} | W^{(t-1)}\right) = \frac{1}{2}\left\| W^{(t)} - W^{(t-1)} \right\|_F^2 = \left\| w^{(t)} - w^{(t-1)} \right\|_2^2$ we have

$$\nabla_{w^{(t)}} f_3(w^{\text{all}}) = (2+4\gamma) w^{(t)} - \gamma\left(w^{(t-1)} + w^{(t+1)}\right).$$

We see that in order to compute the gradient for a window t, we need to use *message passing*, from neighboring windows $t-1$ and $t+1$. The computation of this gradient is the only operation where information from neighboring windows is exchanged during the primal-dual algorithm.

7.3 Experiments

We first evaluate our time varying models on linear Gaussian data that follow our assumptions. We first create a set of m graphs W^i with n nodes. To do so we explore two types of time varying graph:

Data types:

1. Erdős Rényi: Starting from a random binary Erdős Rényi graph [Gilbert, 1959] G_1, we sample each subsequent graph G_{t+1} by keeping 95% of the edges of G_t intact, and 5% of them resampled from the same distribution that gave G_1. In total we sample 700 such graphs of 50 nodes, and we use probability of connection p=5/($m-1$).
2. Random waypoint geometric graph: We suppose that we have 50 sensors moving around the 2 dimensional square space $[0,20]^2$ (meters) according to the mobility model described by Johnson and Maltz [1996], commonly called the *random waypoint* model. By assuming each sensor moves with an average speed of $0.05-0.5 m/s$, we sample their positions every $0.1s$ for a total time of $70s$. Given their positions we construct a total of 700 ϵ-neighborhood graphs with exponential decaying weights.

In this scenario, we are given a matrix $X \in \mathbb{R}^{n\times m}$, where m = 700 denotes time. To make the data more realistic, we suppose that not only the graphs, but also the samples are time dependent, given from the Gaussian Markov chain

$$p\left(x_i \,\middle|\, x_{i-1}, W^{(i)}\right) = \mathcal{N}\left(x_i \,\middle|\, 0, \left(L^{(i)} + \sigma^2 I\right)^{-1}\right)\mathcal{N}\left(x_i \,\middle|\, x_{i-1}, \frac{1}{\mu} I\right), \tag{7.3}$$

starting from a zero vector x_0.

Metric: Given this matrix, we can split it in time windows of a given size and learn a graph for each of them. The number of the windows is therefore inverse proportional to the window size. In order to compute the graph quality, we take the average TCER (Section 1.4) achieved by

the eigenvalue decomposition of the graph Laplacian, averaged over all samples of the given window. For example, the basis error of a given algorithm for the first window $t = 1$ of size k is

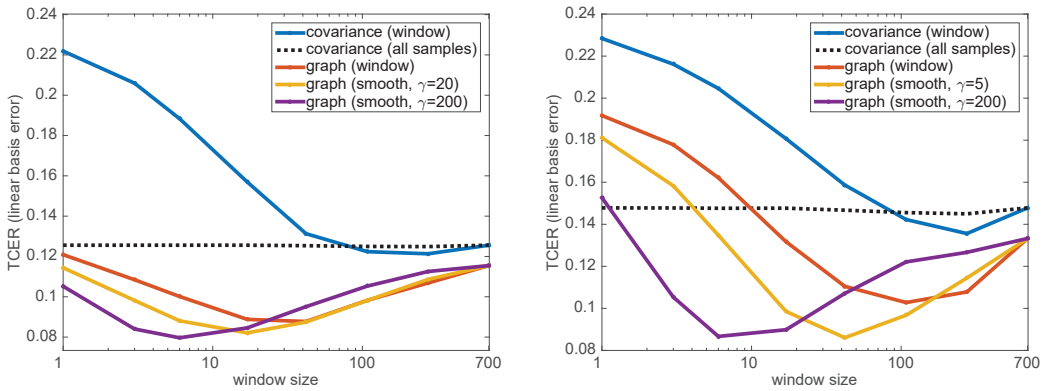
$$\frac{1}{k}\sum_{i=1}^{k}\mathcal{R}\left(p(x_i)\middle|Q^{(1)}\right),$$

where each marginal distribution $p(x_i)$ is "explained" by the same basis $Q^{(1)}$ obtained as the eigenvalue decomposition of the learned graph Laplacian (or empirical covariance) of that window.

In Figure 7.1 we see the graph quality for different window sizes and for the two different types of data. For the Erdős Rényi data we used the log-degree model (7.1) (that performed best), while for the random waypoint data we used the ℓ-2 degree model (7.2), as there are often disconnected nodes in the initial data. Since there is a Gaussian dependence between x_i and x_{i+1}, we use an ℓ-2 distance $f^t_{\text{time}}\left(w^{(t)}|w^{(t-1)}\right) = \left\|w^{(t)} - w^{(t-1)}\right\|_2^2$.

Several observations are in order. Focusing only on the covariance estimation that sets the baseline, we see that because the distribution changes in time, the sample covariance estimation of all samples (**dotted black line**) is a worse estimator than the sample covariance with only half, or even less of the samples (minimum of **blue line**). This justifies that there is a need to split the data in more than one coherent blocks. However, splitting data in more blocks, we keep fewer vectors in each of them, making covariance estimation worse. As suggested by Figure 1.5, this is less of a problem when we learn a graph instead.

The **red line** corresponds to learning a graph from each window independently ($\gamma = 0$), which already does much better than the covariance matrix as it is robust to fewer samples, and therefore can achieve greater precision in time. This effect is stronger when we add the smoothness prior, with the best result in this plot achieved for $\gamma = 200$ (**purple line**). In this case, graph learning is very robust to having very few samples per window (only 6). As the complexity is linear to the number of windows, one might choose to have less windows with more samples each, in which case a smaller γ is more appropriate (**orange line**).



(a) Time varying Erdős Rényi graph (log-degrees prior)

(b) Random waypoint geometric graph (ℓ-2 degrees prior)

Figure 7.1 – Quality of time varying graphs on a matrix $X \in \mathbb{R}^{50\times 700}$ from eq. (7.3) for different sizes of windows. More smoothing in graph learning achieves more detail in time, needing only 6 samples per window to obtain the best result.

8 Discussion

During the preparation of this thesis, some questions of the field were answered explicitly or implicitly. While we leave a detailed list of our contributions in the introduction of this thesis, we stress here explicitly some points that the author finds important.

General graph learning framework. The biggest contribution of this thesis is the general graph learning framework (3.4) that can easily be adapted to many graph learning problems and has an increased interpretability compared to any previous attempt. We argue that before solving a problem, the first step should be to simplify it as much as possible and understand the easiest cases: *Can we see standard graph construction as graph learning? What is the interpretation of each parameter or constraint of our model? Can we omit some of them?*

Traditionally, graph based machine learning problems were solved in a two step fashion: (1) construct a graph given features, then (2) use the graph to solve the problem. With a graph learning point of view available, and given its relatively low cost, we are free to repeat this loop in an expectation maximization fashion, alternating between solving problem (3.2) for X and problem (3.3) for L. We hope that this idea will *lead to adapting our generic framework to specific applications, and provide simultaneously better quality graphs and solutions of existing machine learning problems.*

Graphs, the next PCA? We saw in Figure 1.5 that graphs can obtain a good basis of our data with only very few of them available. This is most important in today's applications, where the number of nodes can grow to large values, and standard structure estimators like the empirical covariance struggle with very few signals available. Moreover, the non-negativity of the edges and the sparsity it automatically leads to, make graphs easier to interpret than a standard covariance matrix.[1] Given the above, *we hope that in the near future, learning a graph from a matrix will be a tool as important in data science as is PCA today.*

[1] Sparse inverse covariance estimation is an interesting alternative, but it struggles with an elevated computational cost due to the log-determinant term.

Graphs for big data. Many advances of graphs make them easily scalable to big data. The greatest examples are graph filtering with Chebyshev polynomials[2] and approximate nearest neighbor graphs.[3] The same direction we follow with our large scale graph learning algorithm, that has a final leading cost same as the one of approximate nearest neighbors algorithms. We see that *graphs are already targeting large scale applications, and this should be one of the usability criteria of new models.*

Reproducible research. We feel that keeping research reproducible is the only way to allow others to correctly review, use, and build upon one's work. Giving code capable to reproduce important results of one's research plays an important role in this. Most algorithms of this thesis are available as part of the open source Matlab toolbox GSPbox [Perraudin et al., 2014]. The latter builds on our general proximal based convex optimization Matlab toolbox UNLocBoX [Perraudin and Kalofolias, 2016]. Code for matrix completion on graphs is available in https://lts2.epfl.ch/research/reproducible-research/matrix-completion-on-graphs/.

[2] Reducing cost from $\mathcal{O}\left(n^3\right)$ needed by eigenvalue decomposition to $\mathcal{O}\left(|\mathcal{E}| \cdot I\right)$ for I iterations.

[3] Reducing cost from $\mathcal{O}\left(n^2 m + n^2 \log(n)\right)$ needed by sorting all pairwise distances to $\mathcal{O}\left(n \log(n) m\right)$ for m signals (features) and n nodes.

A Appendix

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 $\log(\gamma\tilde{W}\mathbf{1}) = \log(\tilde{W}\mathbf{1}) + \text{const.}(W)$. The second equality is obtained from the first one for $\gamma = \alpha$. □

Proof of proposition 3

Proof. For equation (5.4)

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

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

For equation (5.5) we change variable in the optimization and use $\tilde{W} = W/\gamma$ to obtain

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

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

Table A.1 – 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 higher 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	**Hu etal**	**Ours**	**base**	**Hu etal**	**Ours**	**base**	**Hu 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**

List of publications

During my thesis I authored or co-authored the following papers and technical reports (sorted in reverse chronological order of publication):

1. V. Kalofolias. *How to learn a graph from smooth signals.* In The 19th International Conference on Artificial Intelligence and Statistics (AISTATS 2016). Journal of Machine Learning Research (JMLR), 2016. [Kalofolias, 2016].
2. K. Benzi, V. Kalofolias, X. Bresson, and P. Vandergheynst. *Song recommendation with non- negative matrix factorization and graph total variation.* In 2016 IEEE International Confer- ence on Acoustics, Speech and Signal Processing (ICASSP), pages 2439–2443. IEEE, 2016. [Benzi et al., 2016].
3. N. Shahid, N. Perraudin, V. Kalofolias, B. Ricaud, and P. Vandergheynst. *PCA using graph total variation.* In 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4668–4672. IEEE, 2016b. [Shahid et al., 2016b].
4. N. Shahid, N. Perraudin, V. Kalofolias, G. Puy, and P. Vandergheynst. *Fast robust pca on graphs.* IEEE Journal of Selected Topics in Signal Processing, 10(4):740–756, 2016a. [Shahid et al., 2016a].
5. N. Shahid, V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst. *Robust principal component analysis on graphs.* In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 2812–2820, 2015. [Shahid et al., 2015].
6. V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst. *Matrix completion on graphs.* Workshop "Out of the box: robustness and high dimensional data", Neural Information Processing Systems (NIPS), 2014. [Kalofolias et al., 2014].
7. V. Kalofolias. *Analysis and algorithms of variational inference.* Technical report, 2012. [Kalofolias, 2012].

Bibliography

O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *The Journal of Machine Learning Research*, 9:485–516, 2008.

A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439): 509–512, 1999.

R. Baraniuk, V. Cevher, M. Duarte, and C. Hegde. Model-based Compressive Sensing. *IEEE Trans. Information Theory*, 56(4):1982–2001, 2010.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, volume 14, pages 585–591, 2001.

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

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

K. Benzi, V. Kalofolias, X. Bresson, and P. Vandergheynst. Song recommendation with non-negative matrix factorization and graph total variation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2439–2443. IEEE, 2016.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. Conf. Uncertainty in Artificial Intelligence*, 1998.

D. Cai, X. He, J. Han, and T. S. Huang. Graph regularized nonnegative matrix factorization for data representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(8): 1548–1560, 2011.

J.-F. Cai, E. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM J. Optimization*, 20(4):1956–1982, 2010.

E. Candès and Y. Plan. Matrix completion with noise. *Proc. IEEE*, 98(6):925–936, 2010.

E. Candès and B. Recht. Exact matrix completion via convex optimization. *FCM*, 9(6):717–772, 2009.

F. R. Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

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

P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.

S. I. Daitch, J. A. Kelner, and D. A. Spielman. Fitting a graph to vector data. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 201–208. ACM, 2009.

A. P. Dempster. Covariance selection. *Biometrics*, pages 157–175, 1972.

X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst. Learning laplacian matrix in smooth graph signal representations. *arXiv preprint arXiv:1406.7842v2*, 2015.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279. ACM, 2008.

A. Elmoataz, O. Lezoray, and S. Bougleux. Nonlocal discrete regularization on weighted graphs: a framework for image and manifold processing. *IEEE transactions on Image Processing*, 17 (7):1047–1060, 2008.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

E. N. Gilbert. Random graphs. *The Annals of Mathematical Statistics*, pages 1141–1144, 1959.

G. Golub and C. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

A. Horn. Doubly stochastic matrices and the diagonal of a rotation matrix. *American Journal of Mathematics*, 76(3):620–630, 1954.

C.-J. Hsieh, I. S. Dhillon, P. K. Ravikumar, and M. A. Sustik. Sparse inverse covariance matrix estimation using quadratic approximation. In *Advances in Neural Information Processing Systems*, pages 2330–2338, 2011.

C. Hu, L. Cheng, J. Sepulcre, G. El Fakhri, Y. M. Lu, and Q. Li. A graph theoretical regression model for brain connectivity learning of alzheimer's disease. In *2013 IEEE 10th International Symposium on Biomedical Imaging*, pages 616–619. IEEE, 2013.

J. Huang, Z. Zhang, and D. Metaxas. Learning with structured sparsity. In *Proc. ICML*, 2009.

Z. Huang, W. Chung, T.-H. Ong, and H. Chen. A graph-based recommender system for digital library. In *Proc. ACM/IEEE-CS joint conference on Digital libraries*, 2002.

I. C. Ipsen and D. J. Lee. Determinant approximations. *arXiv preprint arXiv:1105.0437*, 2011.

T. Jebara, J. Wang, and S.-F. Chang. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 441–448. ACM, 2009.

R. Jenatton, J. Audibert, and F. Bach. Structured Variable Selection with Sparsity-Inducing Norms. *JMLR*, 12:2777–2824, 2011.

B. Jiang, C. Ding, B. Luo, and J. Tang. Graph-laplacian pca: Closed-form solution and robustness. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3492–3498. IEEE, 2013.

D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile computing*, pages 153–181. Springer, 1996.

V. Kalofolias. Analysis and algorithms of variational inference. Technical report, 2012.

V. Kalofolias. How to learn a graph from smooth signals. In *The 19th International Conference on Artificial Intelligence and Statistics (AISTATS 2016)*. Journal of Machine Learning Research (JMLR), 2016.

V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst. Matrix completion on graphs. *arXiv preprint arXiv:1408.1717*, 2014.

N. Komodakis and J.-C. Pesquet. Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems. *arXiv preprint arXiv:1406.5429*, 2014.

B. Lake and J. Tenenbaum. Discovering structure by learning sparse graph. In *Proceedings of the 33rd Annual Cognitive Science Conference*. Citeseer, 2010.

Z. Liu and L. Vandenberghe. Interior-Point Method for Nuclear Norm Approximation with Application to System Identification. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1235–1256, 2009.

H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 287–296. ACM, 2011.

Y. A. Malkov and D. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *arXiv preprint arXiv:1603.09320*, 2016.

B. Miller, I. Albert, S. Lam, J. Konstan, and J. Riedl. MovieLens unplugged: Experiences with an occasionally connected recommender system. In *Proc. Conf. Intelligent User Interfaces*, 2003.

M. Muja and D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2227–2240, 2014.

S. Negahban and M. J. Wainwright. Restricted strong convexity and weighted matrix completion: Optimal bounds with noise. *JMLR*, 13(1):1665–1697, 2012.

Y. Nesterov and A. Nemirovski. On first-order algorithms for l 1/nuclear norm minimization. *Acta Numerica*, 22:509–575, 2013.

A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

S. Oymak, A. Jalali, M. Fazel, Y. C. Eldar, and B. Hassibi. Simultaneously structured models with application to sparse and low-rank matrices. *arXiv:1212.3753*, 2012.

N. Perraudin and V. Kalofolias. UNLocBoX, A matlab convex optimization toolbox for proximal splitting methods. *ArXiv e-prints*, Oct. 2016.

N. Perraudin and P. Vandergheynst. Stationary signal processing on graphs. *arXiv preprint arXiv:1601.02522*, 2016.

N. Perraudin, J. Paratte, D. Shuman, V. Kalofolias, P. Vandergheynst, and D. K. Hammond. GSPBOX: A toolbox for signal processing on graphs. *ArXiv e-prints*, Aug. 2014.

C. R. Rao. *Linear statistical inference and its applications*, volume 22. John Wiley & Sons, 2009.

B. Recht. A simpler approach to matrix completion. *JMLR*, 12:3413–3430, 2011.

R. Salakhutdinov and N. Srebro. Collaborative filtering in a non-uniform world: Learning with the weighted trace norm. In *Proc. NIPS*, 2010.

R. Schmidt. Multiple emitter location and signal parameter estimation. *IEEE Trans. on Antennas and Propagation*, 34(3):276–280, 1986.

I. Schur. Uber eine klasse von mittelbildungen mit anwendungen auf die determinantentheorie. *Sitzungsberichte der Berliner Mathematischen Gesellschaft*, 22:9–20, 1923.

S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro. Network topology identification from spectral templates. *arXiv preprint arXiv:1604.02610*, 2016.

N. Shahid, V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst. Robust principal component analysis on graphs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2812–2820, 2015.

N. Shahid, N. Perraudin, V. Kalofolias, G. Puy, and P. Vandergheynst. Fast robust pca on graphs. *IEEE Journal of Selected Topics in Signal Processing*, 10(4):740–756, 2016a.

N. Shahid, N. Perraudin, V. Kalofolias, B. Ricaud, and P. Vandergheynst. Pca using graph total variation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4668–4672. IEEE, 2016b.

D. Shuman, S. K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, et al. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *Signal Processing Magazine, IEEE*, 30(3):83–98, 2013.

N. Srebro, J. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. In *Proc. NIPS*, volume 17, pages 1329–1336, 2004.

A. Susnjara, N. Perraudin, D. Kressner, and P. Vandergheynst. Accelerated filtering on graphs using lanczos method. *arXiv preprint arXiv:1509.04537*, 2015.

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

F. Wang and C. Zhang. Label propagation through linear neighborhoods. *Knowledge and Data Engineering, IEEE Transactions on*, 20(1):55–67, 2008.

F. Zhang and E. R. Hancock. Graph spectral image smoothing using the heat kernel. *Pattern Recognition*, 41(11):3328–3342, 2008.

T. Zhang, A. Popescul, and B. Dom. 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*, pages 821–826. ACM, 2006.

Y.-M. Zhang, Y. Zhang, D.-Y. Yeung, C.-L. Liu, and X. Hou. Transductive learning on adaptive graphs. In *AAAI*, 2010.

M. Zheng, J. Bu, C. Chen, C. Wang, L. Zhang, G. Qiu, and D. Cai. Graph regularized sparse coding for image representation. *Image Processing, IEEE Transactions on*, 20(5):1327–1336, 2011.

X. Zhu, Z. Ghahramani, J. Lafferty, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 3, pages 912–919, 2003.

Vassilis Kalofolias

Rue du Maupas 44, CH-1014
Lausanne, Vaud, Switzerland
+41 78 890 73 69
v.kalofolias@gmail.com
3 January 1987, Greek

Strengths:

- Machine Learning
- Optimization, Numerical Analysis
- Strong analytical skills, fast learner

Education

2011 - today PhD candidate in Signal Processing lab (LTS2) and Probabilistic Machine Learning lab (LAPMAL), EPFL, Switzerland.

2004 - 2010 Dipl. Eng. in Computer Engineering and Informatics (CEID), Patras University, Greece. GPA: 9.01 out of 10, ranked 1st.

2008 Exchange student at Informatics Department of Karlsruhe University, Germany (Erasmus Program).

Working Experience

2014, 2015 **EPFL**, Main Lecturer, "Advanced Signal Processing Lab". One of the 2 organizers of the whole lab. Structure, material, grading and supervision on advanced signal processing and optimization techniques.

2013 **EPFL**, Teaching Assistant, "Pattern Classification and Machine Learning". Helped in design and preparation of project on image classification (neural networks, linear regression). Supervised students, assisted exams preparation.

2010 **EPFL**, Teaching Assistant, "Scientific Computing". Gave lectures on high performance computing (vectorization, BLAS, complexity). Assisted examination and project.

Technical Skills

6 years of experience in advanced data analysis techniques

Programming MATLAB (expert), C (fluent), C++ (intermediate)

Parallelization MPI, OpenMP (some experience)

Tools Git, SVN, LaTeX

Projects

UNLocBoX Open source **convex optimization toolbox** for Matlab, implementing various state-of-the-art methods and an abundance of proximal operators/tools. Key contributions in both design and implementation. Link: `https://lts2.epfl.ch/unlocbox/`.

GSPbox Open source **graph signal processing toolbox** for Matlab, implementing from basic operations like filtering to advanced features like graph learning. Key contributions in both design and implementation. Integration of novel methods based on my publications. Link: `https://lts2.epfl.ch/gsp/`.

Graph-based data analysis I proposed a novel model for enhancing **collaborative filtering** based recommendations using graph based side information. The performance of the new system is much better, and especially robust in the regime of many missing data.
I have co-authored a series of papers about enhancing **PCA with graphs**. The performance of the models we propose is superior for tasks like clustering and low rank representation, both in robustness and in computational complexity.
I wrote a paper about **learning the graph from smooth data**, a problem of great importance for graph analysis techniques. I provide the first **scalable algorithm** for learning a graph, using **primal dual** optimization. The quality of the graph is better than the previous state of the art in many tasks.

Nonnegative matrix factorization During my diploma thesis on Nonnegative Matrix Factorization (NMF), I proposed a polynomial time algorithm for exact factorization under additional constraints (the general case is NP-hard). More recently, we used approximate NMF enhanced with graph total variation priors to achieve better results on **song recommendation**. Our algorithm, based on primal dual optimization, is scalable and performs better than previous techniques.

Interests

- Machine Learning
- Optimization
- Numerical Analysis
- Signal Processing
- Graph techniques
- Recommendation Systems

Speaking Languages

Greek	Mother Tongue
English	C2
French	C1
German	B2
Spanish	A2

Selected publications

Detailed list	In my Google Scholar profile, `https://scholar.google.ch/citations?user=Bz1RQ8MAAAAJ&hl=en`.
Graph Learning	V. Kalofolias. **How to learn a graph from smooth signals.** The 19th International Conference on Artificial Intelligence and Statistics (**AISTATS** 2016).
Graph PCA	N. Shahid, V. Kalofolias, X. Bresson, M. Bronstein and P. Vandergheynst. **Robust Principal Component Analysis on Graphs**. International Conference on Computer Vision (**ICCV** 2015). N. Shahid, N. Perraudin, V. Kalofolias and P. Vandergheynst. **Fast Robust PCA on Graphs**, to appear to IEEE Journal of Selected Topics in Signal Processing, 2015.
Matrix Completion	V. Kalofolias, X. Bresson, M. Bronstein and P. Vandergheynst. **Matrix Completion on Graphs**. Neural Information Processing Systems, Workshop "Out of the Box: Robustness in High Dimension", (**NIPS** 2014).
NMF	V. Kalofolias and E. Gallopoulos. **Computing symmetric nonnegative rank factorizations**, in Linear Algebra and its Applications, 2012.