TECHNICAL UNIVERSITY OF CRETE

Regularized Optimization Applied to Clustering and Joint Estimation of Multiple Undirected Graphical Models

SUBMITTED TO THE SCHOOL OF ELECTRONIC AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE MASTER OF SCIENCE DEGREE

8 April 2014

Author:
Alexandros GEORGOGIANNIS

Supervisors:
Vassilis DIGALAKIS
Athanasios LIAVAS
Co-advisors:
Michail LAGOUDAKIS

Contents

List of Figures

List of Tables

This page is intentionally left blank.

Chapter 1

Introduction

Since its earliest days as a discipline, machine learning has made use of optimization formulations and algorithms. Likewise, machine learning has contributed to optimization, driving the development of new optimization approaches that address the significant challenges presented by machine learning applications. This influence continues to deepen, producing a growing literature at the intersection of the two fields while attracting leading researchers to the effort [62]. While techniques proposed twenty years ago continue to be refined, the increased complexity, size, and variety of today's machine learning models demand a principled reassessment of existing assumptions and techniques. This thesis makes a small step toward such a reassessment. It describes novel contexts of established frameworks such as convex relaxation, splitting methods, and regularized estimation and how we can use them to solve significant problems in data mining and statistical learning.

The thesis is organised in two parts. In the first part, we present a new clustering algorithm. The task of clustering aims at discovering structures in data. This algorithm is an extension of recently proposed convex relaxations of k-means and hierarchical clustering [47, 58, 37]. In the second part, we present a new algorithm for discovering dependencies among common variables in multiple undirected graphical models. Graphical models are useful for the description and modelling of multivariate systems [45]. In the appendix, we comment on a core problem underlying the whole study and we give an alternative solution based on recent advances in convex optimization.

Chapter 2

Preliminaries

We will make extensive use of convex optimization to formulate and solve statistical learning problems. In order to illustrate our aim more concretely, in the next pages we make a brief review of convex analysis [11, 7].

2.1 Convex optimization in machine learning

Most machine learning problems reduce to optimization problems. Consider the machine learning analyst in action solving a problem for some set of data. She formulates the problem by selecting an appropriate model among many others and transforms the data into an appropriate format [38]. Then, the model is typically trained by solving a core optimization problem. The research area of mathematical programming intersects with machine learning through these core optimization problems.

In general, we shall be concerned with constrained optimization problems, with real parameter vectors $x \in \mathbb{R}^n$ of the form

$$\begin{aligned} \min_{x \in S} \; & f_0(x) \\ \text{s.t } & f_i(x) \leq 0, \; i = 1, \dots, m \\ & h_i(x) = 0, \; i = 1, \dots, p, \end{aligned} \tag{2.1}$$

where

- the domain $S \subseteq \mathbb{R}^n, \; S = \bigcap_{i=0}^{m} \text{dom} f_i \; \cap \; \bigcap_{i=1}^{p} \text{dom} h_i$ of the optimization problem is a non-empty set,
- $x = ([x]_1 \; \cdots \; [x]_n)^T$ is the vector of optimization variables,
- $f_0 : \mathbb{R}^n \to \mathbb{R}$ is the objective function to be minimized,
- $f_i : \mathbb{R}^n \to \mathbb{R}, \; i = \{1, \dots, m\}$ are the inequality constraints, and
- $h_i : \mathbb{R}^n \to \mathbb{R}, \; i = \{1, \dots, p\}$ are the equality constraints.

A point $x \in S$ is *feasible* if it satisfies the constraints $f_i(x) \leq 0, \; i = 1, \dots, m$ and $h_i(x) = 0, \; i = 1, \dots, p$. The problem (2.1) is said to be feasible if there exists at least one feasible point, and infeasible otherwise. The set of all feasible points is called the feasible set or the constraint set. In Figure 2.1, we plot two convex functions, the l_1 and l_2-norms, and a non-convex one, the l_0-pseudonorm defined as $||x||_0 = I(x) = 1$, if $x \neq 0$ and 0 otherwise.

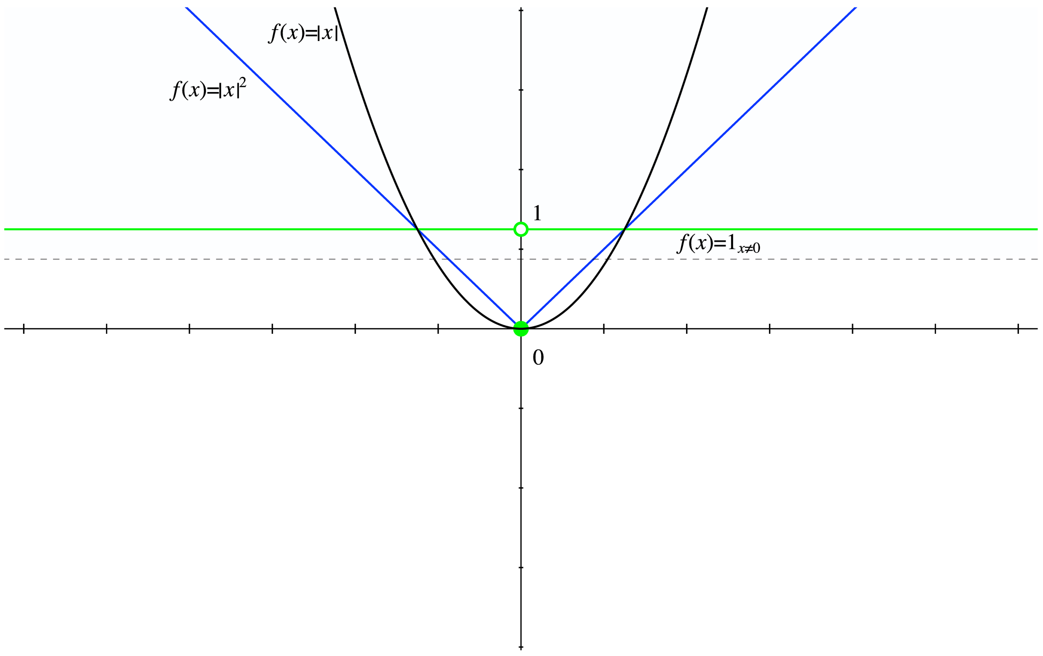


Figure 2.1: Two convex functions and one non-convex with global minima equal to 0. blue: the l_2-norm (convex), black: the l_1-norm (convex), green: the l_0-pseudo-norm (non-convex).

Optimality conditions for convex functions

A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is *convex* if for all $x, y \in \mathbb{R}^n$, and for all $\theta \in \mathbb{R}$ such that $0 \leq \theta \leq 1$, we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \tag{2.2}$$

and *strictly convex* if, for $0 < \theta < 1$,

$$f(\theta x + (1-\theta)y) < \theta f(x) + (1-\theta)f(y). \tag{2.3}$$

The definition of convexity means that the chord (i.e., line segment) between any two points $(x, f(x))$ and $(y, f(y))$ lies above the graph of f. Convexity plays a key role in mathematical programming; every local minimum of a convex problem is also a global one. If a function f is differentiable, the *gradient* $\nabla f(x) \in \mathbb{R}^n$ at the point $x \in \mathbb{R}^n$ is the vector of partial derivatives

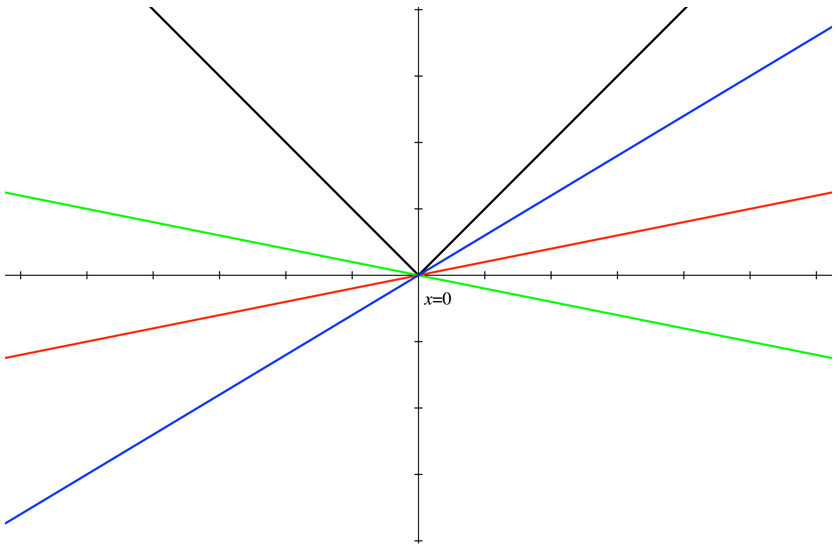
$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1} \cdots \frac{\partial f(x)}{\partial x_n} \right)^T. \tag{2.4}$$

The gradient allows us to characterize the stationary points of f. In particular, we know that any point x such that $\nabla f(x) = 0$ is either a local *minimum* , a local *maximum* or a *saddle* point of f. For every differentiable convex function f that achieves its minimum value $p^\star$ at $x^\star$, we have

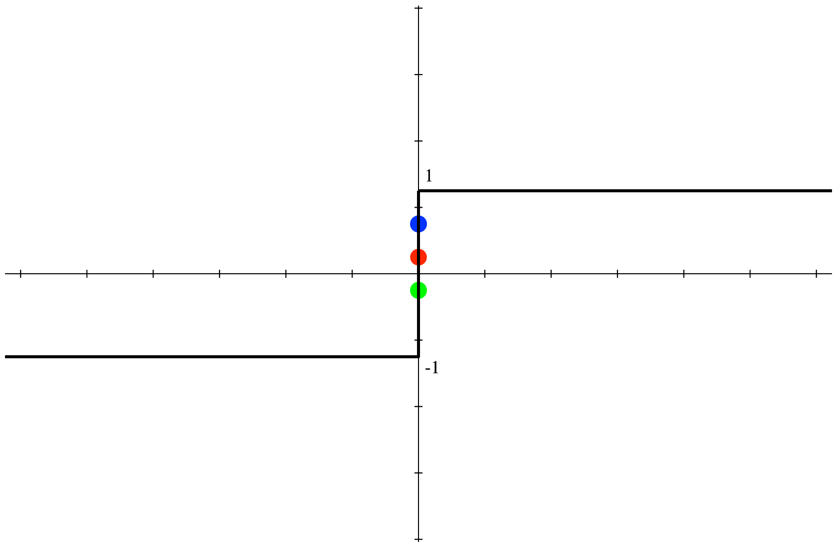
$$f(x^\star) = p^\star \Leftrightarrow \nabla f(x^\star) = 0. \tag{2.5}$$

When a function is nondifferentiable, we use the subgradients. We define the *subgradient* for any convex function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ at a point x, as

$$\partial f(x) = \{d \in \mathbb{R}^n | f(x) - f(y) \geq d^T(x-y) \text{ for any } y \in \mathbb{R}^n\}. \tag{2.6}$$



(a) l_1-norm $|x|$.



(b) Subgradient $\partial|x|$.

Figure 2.2: The function $f(x) = |x|$ and its subdifferential $\partial f(x)$ as a function of x. At $x = 0$, the subdifferential is $\partial|x| \subseteq [0, 1]$.

If f is convex and differentiable at x, then $\partial f(x) = \{\nabla f(x)\}$, i.e., its gradient is its only subgradient. We plot the non-differentiable function, $|x|$, with its subgradient in Figure 2.2. This function has an infinite set of tangent planes at $x = 0$, and so its subgradient at this point is the set of scalars $\{d \in \mathbb{R} | d \in [0, 1]\}$ If the subdifferential of a convex function f at x contains the zero vector $0 \in \partial f(x)$, then x is a minimum of f:

$$f(x) = p^\star \Leftrightarrow 0 \in \partial f(x). \tag{2.7}$$

When f is smooth this reduces to the usual gradient optimality condition $\nabla f(x) = 0$.

The Lagrangian function and duality

The basic idea in *duality* is to take the constraints in (2.1) into account by augmenting the objective function f with a weighted sum of the constraint functions. Then the *Lagrangian* associated with the primal problem (2.1) is the function $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ defined as

$$\mathcal{L}(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x). \tag{2.8}$$

The vectors λ and ν are called the *dual variables* or *Lagrange multiplier vectors* associated with the primal problem. Having the Lagrangian $\mathcal{L}$ we define another important function called the *Lagrange dual function* $\mathcal{D} : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ as the minimum value of the Lagrangian over x: for $\lambda \in \mathbb{R}^m, \nu \in \mathbb{R}^p$,

$$\mathcal{D}(\lambda, \nu) = \inf_{x \in S} \mathcal{L}(x, \lambda, \nu) = \inf_{x \in S} \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \right). \tag{2.9}$$

When the Lagrangian is unbounded below in x, the dual function takes on the value $-\infty$. Since the dual function is the point-wise infimum of a family of affine functions of (λ, ν), it is concave, even when the primal problem is not convex. For any $\lambda \succeq 0$, $\mathcal{D}(\lambda, \nu)$ provides a lower bound for (2.1), that is

$$\mathcal{D}(\lambda, \nu) \le p^\star. \tag{2.10}$$

This property is called *weak-duality* and holds for both convex and non-convex problems. Weak duality can be used to find lower bounds for difficult problems. When the problem is convex and its feasible set has non-empty interior, *strong duality* holds, that is

$$\max_{\lambda \succeq 0, \nu} \mathcal{D}(\lambda, \nu) = p^\star. \tag{2.11}$$

A problem of the form (2.1) is convex when f_0, f_i, and h_i are affine functions.

Regularized optimization

Though the contexts in machine learning vary widely, the objective remains the same; we have a function $f : \mathbb{R}^n \to \mathbb{R}$ that is the loss or empirical risk associated with our chosen model for the data, and we aim to obtain a solution x that minimizes f so as to tune the model. The common thread is to search for solutions x that are *simple* in some well defined sense. In this case, *sparsity* is a synonym for simplicity, that is, we search for solutions $x = ([x]_1 \cdots [x]_n)$ with many components $[x]_i$ equal to 0. There are some reasons behind the search for sparse solutions [70, 3].

Constrained formulation I	Penalized formulation
min $f(x)$ s.t $g(x) \leq 0$	min $f(x) + \gamma g(x)$

Table 2.1: Three main formulations of machine learning optimization problems.

First, simple explanations of the observations are preferable to complicated explanations. Second, sparse solutions sometimes are more robust to data inexactness, and third, sparse solutions conform better to prior knowledge. Prior knowledge on the structure of x is expressed through the constraint g.

We will encounter problems in two main formulations as summarized in Table 2.1. *Penalized formulation* is the Lagrangian associated with the *constrained formulation I*. We focus on the penalized formulation of learning problems, that is problems of the form

$$\underset{x \in S}{\text{minimize}} \quad f(x) + \gamma g(x), \tag{2.12}$$

where f is convex red and g may be nonconvex and nonsmooth. Parameter γ is a non-negative scalar that weights the relative importance of optimality and simplicity of the solution, and $S \subseteq \mathbb{R}^n$ is a convex set.

Algorithms for convex optimization

There are several practical methods or "solvers" that one can use to solve (2.12); from black-box optimizers [63, 42] that we "feed" with the function to be minimized and return the solution, to specific designed algorithms that exploit the special structure of the problem [18]. A major difficulty that arises in solving problems of the form (2.12) is the possible non-differentiability of g, which rules out traditional techniques, e.g., gradient descent, Newtons' method etc.

In this study, we make extensive use of *alternating direction minimization* algorithms [10, 30], especially the *alternating direction method of multipliers* (ADMM) [8] and the *alternating minimization algorithm* (AMA) [68]. These methods have become particularly important in computer vision and machine learning communities. They are designed to solve problems of the form

$$\begin{aligned} \min \quad & f(x) + g(z) \\ \text{s.t.} \quad & Ax + Bz = c, \end{aligned} \tag{2.13}$$

with variables $x \in \mathbb{R}^n$ and $z \in \mathbb{R}^m$, where $A \in \mathbb{R}^{p\times n}, B \in \mathbb{R}^{p\times m}$, and $c \in \mathbb{R}^p$. We will assume that f and g are convex, and g possibly non-smooth. The regularized problem (2.12) is in the form (2.13) with $z = x$, $A = I_{p\times n}$, $B = -I_{p\times m}$, where I is the identity matrix, and $c = 0$.

The optimal value for the problem (2.13) is

$$p^\star = \inf \{f(x) + g(z) | Ax + Bz = c\}.$$

The Lagrangian $\mathcal{L}$ for (2.13) is

$$\mathcal{L}(x, z, \lambda) = f(x) + g(z) + \lambda^T(c - Ax - Bz), \tag{2.14}$$

while the augmented Lagrangian $\mathcal{L}_\rho$ is

$$\begin{aligned} \mathcal{L}_\rho(x, z, \lambda) &= \mathcal{L}(x, z, \lambda) + (\rho/2)||c - Ax - Bz||_2^2 \\ &= f(x) + g(z) + \lambda^T(c - Ax - Bz) + (\rho/2)||c - Ax - Bz|||_2^2. \end{aligned} \tag{2.15}$$

Note that $\mathcal{L}_0 = \mathcal{L}$. Before moving on to discuss ADMM and AMA, it is helpful to recall [22] how ones calculates the dual function $\mathcal{D}$ of the problem (2.13)

$$\begin{aligned}
\mathcal{D}(\lambda) &= \inf_{x,z} \mathcal{L}_0(x, z, \lambda) && (2.16)\\
&= \inf_{x,z} \{f(x) + g(z) + \lambda^T(c - Ax - Bz)\} && (2.17)\\
&= \inf_{x} \{f(x) - \lambda^T Ax\} + \inf_{z} \{g(z) - \lambda^T Bz\} + \lambda^T c && (2.18)\\
&= -f^\star(\lambda^T A) - g^\star(\lambda^T B) + \lambda^T c, && (2.19)
\end{aligned}$$

involving the Fenchel conjugates [49]

$$f^\star(y) = \sup_{x \in \text{dom} f} (y^T x - f(x)) \quad \text{and} \quad g^\star(y) = \sup_{z \in \text{dom} g} (y^T z - g(z)).$$

The ADMM is an iterative algorithm and can be described as follows [10]. If we assume that at iteration k we have computed z^k and λ^k, then at the iteration $(k+1)$ we compute

$$\begin{aligned}
x^{k+1} &:= \arg\min_x \mathcal{L}_\rho(x, z^k, \lambda^k)\\
z^{k+1} &:= \arg\min_z \mathcal{L}_\rho(x^{k+1}, z, \lambda^k)\\
\lambda^{k+1} &:= \lambda^k + \rho(c - Ax^{k+1} + Bz^{k+1}).
\end{aligned} \tag{2.20}$$

A simpler ADM algorithm is AMA that takes a slightly different path and updates x without augmentation, assuming $f(x)$ is strongly convex. This change is accomplished by setting the positive tuning constant ρ to be 0. The overall algorithm iterates according to [68]

$$\begin{aligned}
x^{k+1} &:= \arg\min_x \mathcal{L}_0(x, z^k, \lambda^k)\\
z^{k+1} &:= \arg\min_z \mathcal{L}_\rho(x^{k+1}, z, \lambda^k)\\
\lambda^{k+1} &:= \lambda^k + \rho(c - Ax^{k+1} + Bz^{k+1}).
\end{aligned} \tag{2.21}$$

Although this alternating minimization appears to complicate matters, it often simplifies optimization, as in the cases we study here.

Finally, one of the fastest ways to write the code that solves (2.12) is using disciplined convex programming. Using this technique, a library such as `cvxmod` [32] is used to write code that describes the optimization problem. Then, the `cvxmod` library analyzes the program, translates it into standard form (2.1), and applies a standard solver. `cvxmod` supports a number of standard problem types, including linear and quadratic programs (LPs/QPs), second-order cone programs (SOCPs), and semidefinite programs (SDPs).

Chapter 3

Clustering with Non-Convex Penalties

Abstract of this chapter: In this chapter, we solve the clustering problem by transforming it to a penalized non-convex regression problem. We present an iterative algorithm that calculates the regularization path of solutions, by approximating the initial non-convex problem with a series of l_1-norm constrained problems. We discuss the convergence properties of this iterative procedure and experimentally compare its performance against other state-of-the-art clustering algorithms. Results show that our method is a strong candidate for clustering non-convex datasets and outperforms all other previously proposed convex relaxations of k-means and hierarchical clustering.

3.1 Introduction

Clustering is an unsupervised method where the goal is to partition a set of samples into groups called clusters. Intuitively, the samples within a cluster are more similar to each other than to those from other clusters. Standard clustering algorithms such as k-means, hierarchical clustering, and Gaussian mixture models are effective and robust but they can get trapped in local minima, either because they are cast as hard combinatorial optimization problems [2], or because of the hard thresholding of distances, e.g., cutting the dendrogram of an agglomerative clustering algorithm at a specific height [26].

Regularized methods for regression have been used extensively since the initial works of [12, 65]. These methods minimize a function that generally is the sum of two terms; a loss term and a regularization term. The latter is frequently non-smooth in order to prevent over fitting and promote sparsity, and in recent works non-convex [14, 24], in order to promote sparsity even more, making it hard to minimize.

In this chapter, we present a majoration/minimization based clustering algorithm that solves a *non-convex penalized regression* problem. To find a solution, the algorithm iteratively approximates the non-convex problem with a series of l_1*-norm constrained problems*. Similar clustering formulations to ours were previously presented in [58, 47, 37]. However, all the aforementioned studies approximate the (non-convex) cardinality l_0-norm with a convex norm, where, in our approach, we use a family of non-convex functions, like the l_q quasi-norm for $q \in (0, 1)$, that better approximate the l_0-norm and favor sparse solutions even more. Another work, that of [56], uses a non-convex penalty, the truncated l_1-norm, which is shown to outperform convex norms. The authors of [56] approximate the original non-convex problem using quadratic problems. Instead, we solve a series of l_1-norm problems to better approximate our initial non-convex problem, incorporating the benefits that arise from the sparsity-promoting nature of the l_1-norm. We note that, since the initial problem we solve is non-convex, the algorithms developed in this chapter produce local optimal solutions.

The rest of this chapter is structured as follows. In Section 2, we present the clustering problem as a penalized regression problem. In Section 3, we provide a detailed description of the proposed algorithm. Finally, in Section 4, we compare our method with other state-of-the art clustering methods and conclude.

3.2 Clustering as penalized regression

Let $X \in \mathbb{R}^{n\times p}$ be a data matrix where n is the number of samples in the dataset and p the dimension of each sample. Every point $x_i \in \mathbb{R}^p$, $i = 1,\ldots,n$, has an associated *centroid* $\mu_i \in \mathbb{R}^p$. All centroids are aggregated in a matrix $M \in \mathbb{R}^{n\times p}$. Now, consider a function $g : \mathbb{R}^m_+ \to \mathbb{R}_+$ that is separable in its input:

$$g(|z|) = \sum_{l=1}^{m} \zeta(|[z]_l|), \tag{3.1}$$

where $z \in \mathbb{R}^m$, $[z]_l$ is the l-th element of z, and $\zeta : \mathbb{R}_+ \to \mathbb{R}_+$. We use the notation $\mathbb{R}_+ = \{t \in \mathbb{R} \ : \ t \geq 0\}$ and $\mathbb{R}_{++} = \{t \in \mathbb{R} \ : \ t > 0\}$. The absolute value vector $|z|$ is to be understood coordinate-wise, that is $|z| = (|[z]_1|,\ldots,|[z]_m|)^T$. We assume that $\zeta(t)$ has the following properties:

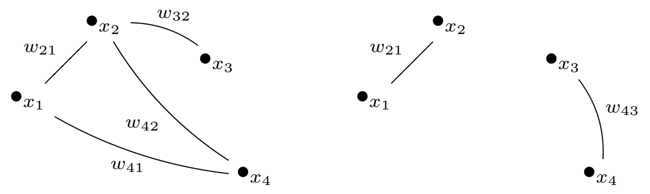
1. it has a global minimum at $t = 0$,
2. it is twice continuously differentiable on $\mathbb{R}_{++}$,
3. $\zeta'(t) > 0, \ \forall t \in \mathbb{R}_{++}$ (increasing on $\mathbb{R}_{++}$),
4. $\zeta''(t) \leq 0, \ \forall t \in \mathbb{R}_{++}$ (concave on $\mathbb{R}_{++}$), and
5. $\zeta''(0_+) \leq \zeta''(t), \ \forall t \in \mathbb{R}_{++}$ (maximally concave at 0).

Later, we will make use of functions ζ parametrized by a positive scalar parameter $\epsilon > 0$ and we shall use the notation ζ_ϵ to denote the parametrized form.

A clustering of the rows of matrix X can be obtained by solving the optimization problem [58, 16]

$$\min_M f(M;\gamma) = \frac{1}{2}||X - M||_F^2 + \gamma \sum_{j<i} w_{ij} g(|\mu_i - \mu_j|), \tag{3.2}$$

where $||\cdot||_F$ is the Frobenius norm, $\gamma \geq 0$, and $w_{ij} \geq 0$. The function g is used as a penalty for the number of different centroids. Parameter γ controls the number of different centroids and the weights w_{ij} are chosen to improve clustering quality (and computational efficiency). The set of solutions $\{M(\gamma_0),\ldots,M(\gamma_\infty)\}$ of the problem (3.2) for $\gamma_0 < \cdots < \gamma_\infty$ is called *clusterpath* [37]. To provide a graphical interpretation for the clustering problem, consider that every sample x_i corresponds to a node in a weighted graph $\mathcal{G} = (V, E, W)$. The number of nodes $|V|$ is equal to the number of points n and an edge $e_{ij} \in E$ between x_i and x_j exists whenever $w_{ij} > 0$ (Figure 3.1). The number of edges $|E|$ is equal to the number of non-zero weights w_{ij} in the set W.



(a) Connected graph with 1 connected component.

(b) Unconnected graph with 2 connected components.

Figure 3.1: Examples of clustering graphs.

Constraining the weights to be non-zero in a neighborhood around a point x_i corresponds to defining a nearest-neighbor graph $\mathcal{G}$. Our goal is to have a nearest-neighbor graph such that subgraphs induced by points from the same cluster are connected, while subgraphs corresponding to different clusters are not connected to each other. Let kNN(x_i) denote the set of the k_{nn} nearest neighbors of x_i. We choose weights of the form [16, 37]

$$w_{ij} = I_{\{i,j\}} \exp(-\phi||x_i - x_j||_2^2), \tag{3.3}$$

where $I_{\{i,j\}} = 1$ if $x_i \in$ kNN(x_j) and/or $x_j \in$ kNN(x_i). As such, we measure the density of different assigned data points in a local neighborhood employing a similar mechanism as in the histogram density estimator. The kind of k_{nn}-nearest neighbor graph, e.g., symmetric or mutual nearest-neighbor graph, has little importance compared to the number k_{nn} of nearest-neighbors. As a rule of thumb, k_{nn} should be small enough in order to reduce computational load, but not as small so as to loose in clustering accuracy [50].

As $\gamma \to \infty$, if $\mathcal{G}$ is connected (Figure 3.1(a))[1] then all points have equal centroids and coalesce into a single cluster with centroid $\bar{x}$, where $\bar{x}$ is the mean of x_i [58, 16]. On the other hand, assume that $\mathcal{G}$ is partitioned into κ subgraphs (connected components) $\mathcal{G}^k$, for $k = 1, \ldots, \kappa$, each one with n_k nodes (Figure 3.1(b)). In that case, for $\gamma \to \infty$, the points coalesce into κ clusters with centroids

$$\bar{x}^k = \frac{1}{n_k} \sum_{i \in \mathcal{G}^k} x_i, \quad k = 1, \ldots, \kappa. \tag{3.4}$$

Note that, in case of κ connected components, $f(M; \gamma)$ can be rewritten as

$$\sum_{k=1}^{\kappa} \left\{ \sum_{i \in \mathcal{G}^k} ||x_i - \mu_i||_2^2 + \gamma \sum_{i,j \in \mathcal{G}^k} w_{ij} g(|\mu_i - \mu_j|) \right\}. \tag{3.5}$$

By inspection of (3.5) and the fact that g has global minimum at 0 (property 1 of ζ), after simple algebraic computations, we arrive at the solution (3.4). Thus, with an appropriate choice of weights w_{ij}, we are able to decompose the original problem into κ small problems. Summarising, when using penalized regression for clustering, the problem of clustering n points into K clusters is transformed to the problem of identifying a set of K centroids $\{\mu_1, \ldots, \mu_K\}$.

3.2.1 Decomposability of the objective function

We proceed to examine the issues that arise when we solve problem (3.2) using functions g of different forms. Since g is acting coordinate-wise, the cost function f in (3.2) can be decomposed

[1]The assumption of connectivity implies that all nodes interact with each other.

as

$$\begin{aligned}
f(M;\gamma) &= \frac{1}{2}||X-M||_F^2 + \gamma\sum_{j<i} w_{ij} g(|\mu_i-\mu_j|) \\
&= \frac{1}{2}\sum_{i=1}^{n}||x_i-\mu_i||_2^2 + \gamma\sum_{j<i} w_{ij} g(|\mu_i-\mu_j|) \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{l=1}^{p}|[x_i]_l-[\mu_i]_l|^2 \\
&\qquad + \gamma\sum_{j<i} w_{ij}\sum_{l=1}^{p}\zeta(|[\mu_i]_l-[\mu_j]_l|) \\
&= \sum_{l=1}^{p}\left(\tfrac{1}{2}||x^l-\mu^l||_2^2 + \gamma\sum_{j<i} w_{ij}\zeta(|[\mu^l]_i-[\mu^l]_j|)\right) \\
&= \sum_{l=1}^{p} f_1(\mu^l;\gamma),
\end{aligned}$$

where $\mu^l \in \mathbb{R}^n$ and $x^l \in \mathbb{R}^n$ are the l-th columns of matrices M and X, respectively. Thus, the minimization with respect to every column μ^l corresponds to solving p problems of the form

$$\min_{\mu} f_1(\mu;\gamma) = \frac{1}{2}||x-\mu||_2^2 + \gamma\sum_{j<i} w_{ij}\zeta(|[\mu]_i-[\mu]_j|), \tag{3.6}$$

dropping the superscript l from x and μ for ease of notation.

The set of $m = \binom{n}{2}$ distinct differences $[\mu]_i - [\mu]_j$ form a column vector $z \in \mathbb{R}^m$ that can be written as

$$z = D\mu = \begin{bmatrix} [\mu]_2 - [\mu]_1 \\ \vdots \\ [\mu]_n - [\mu]_{n-1} \end{bmatrix},$$

where $D \in \mathbb{R}^{m\times n}$ is a sparse matrix, with every row containing only two nonzero elements, -1 and 1, placed at the appropriate positions. With D at hand, we can write (3.6) as

$$\min_{\mu} f_1(\mu;\gamma) = \frac{1}{2}||x-\mu||_2^2 + \gamma\sum_{l=1}^{m} w_l\zeta(|[D\mu]_l|), \tag{3.7}$$

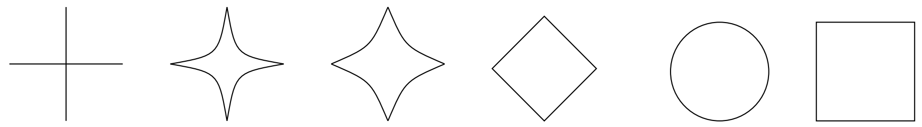
where l corresponds to a pair (i,j) with $j < i$.

As far as function ζ is concerned, there are many choices that promote sparsity. A popular choice is $\zeta(|t|) = |t|$, that leads to an l_1-norm regularization problem. In this work, we focus on its non-convex variants, the l_q-quasinorm

$$\zeta_\epsilon(|t|) = (|t|+\epsilon)^q, \tag{3.8}$$

for $0 < q < 1$, and the log penalty

$$\zeta_\epsilon(|t|) = \log(\epsilon|t|+1). \tag{3.9}$$



(a) $q = 0$, $\epsilon = 0$ (b) $q = 0.5$, $\epsilon = 0.01$ (c) log, $\epsilon = 2$ (d) $q = 1$, $\epsilon = 0.01$ (e) $q = 2$, $\epsilon = 0.01$ (f) $q = \infty$, $\epsilon = 0.01$

Figure 3.2: Unit balls in two dimensions for $g(|x|) = \sum_1^2 \zeta_\epsilon$.

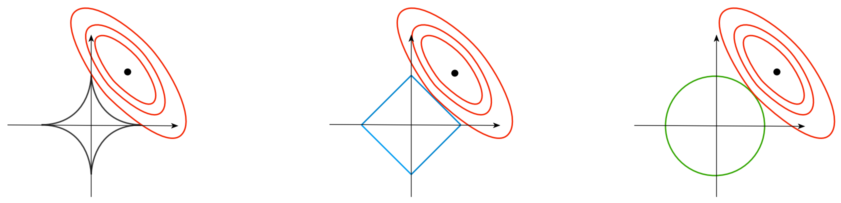


Figure 3.3: Contours (red ellipses) of a quadratic objective function and constraint regions for the $l_{0.5}$ (left), l_1 (middle), and l_2-norms (right).

The unit balls of g for the previous parametrized functions ζ_ϵ are plotted in Figure 3.2. The motivation behind the use of concave penalties is that they induce more sparsity than the l_1-norm, while they can be optimized with continuous optimization [13, 55] as opposed to greedy methods. Figure 3.3 helps understand this motivation; it depicts the $l_{0.5}$, l_1, and l_2-norm constrained minimization of a quadratic function $f : \mathbb{R}^2 \to \mathbb{R}$. The objective function has elliptical contours, centered at the unconstrained global minimum. The constraint regions are $(|[x]_1|^{0.5} + |[x]_2|^{0.5}) \leq a$, $(|[x]_1| + |[x]_2|) \leq a$, and $(([x]_1)^2 + ([x]_2)^2) \leq a$, where $a \in \mathbb{R}$ is a constant. The solution for each problem occurs when the elliptical contours first hit the constraint region. When the solution occurs at a corner, then one of the estimated parameters is equal to zero. Figure 3.3 suggests that, it seems more likely that the solution will be at a corner of the constraint region for the case of the non-convex norm $l_{0.5}$.

3.3 Non-Convex Clusterpath

We now present an algorithm for the computation of non-convex clusterpath for the general case of a concave penalty ζ. The matrix X is assumed to be standardized with every column having zero mean and unit l_2-norm. The basic idea is as follows. Starting from a sufficiently large γ, we iteratively compute $M(\gamma)$ for smaller values, using the estimates from previous iterations as warm start. Details are presented in Algorithm 1. For a specific value of γ, at every loop of the repeat statement in line 4, a column $\mu \in \mathbb{R}^n$ of $M(\gamma)$ is estimated via the solution of a non-convex problem. The general framework we shall use to solve this non-convex problem is the majorization/minimization (MM) algorithm [40]. According to the MM, at the $(k+1)$-th iteration a convex over-estimator of the objective $f_1(\mu;\gamma)$ is minimized. The over-estimator is constructed so as to be tangent to the graph of f_1 at the previous estimate. Surrogate convex functions with these properties are obtained easily by exploiting the concavity of the penalty function which always lies below its tangent [13].

The minimization problem (3.7), and its equivalent in line 4 of Algorithm 1, is an instance of

Algorithm 1: Algorithm to compute the clusterpath

```
Data: Data $X^{n\times p}$, matrix $D^{m\times n}$ with $m = \binom{p}{2}$, weights $w_{ij}$, grid of decreasing $\gamma$ values
      $\{\gamma_\infty, \dots, \gamma_0\}$
Result: Clusterpath $M(\gamma_\infty), \dots, M(\gamma_0)$
1 foreach value of $\gamma$ do
2     foreach column $\mu$ of $M(\gamma)$ do
3         repeat
4             $\mu = \arg\min_{\mu} f_1(\mu;\gamma)$,  where $f_1(\mu;\gamma) = \frac{1}{2}||x^l - \mu||_2^2 + \gamma \sum_{l=1}^m w_l \zeta(|[D\mu]_l|)$;
5         until until convergence;
6     end
7     store $M(\gamma)$;
8 end
9 return $\{M(\gamma_\infty), \dots, M(\gamma_0)\}$
```

the general form of problems encountered in machine learning and computer vision [55]

$$\arg\min_{\mu} F(\mu) = F_1(\mu) + F_2(|D\mu|), \tag{3.10}$$

where $F_1 : \mathbb{R}^n \to \mathbb{R}_+$ is a smooth convex function, bounded from below, and $F_2 : \mathbb{R}_+^m \to \mathbb{R}_+$ is a continuous, concave and increasing. In our case

$$F_1(\mu) = \tfrac{1}{2}||x - \mu||_2^2 \tag{3.11}$$

and

$$F_2(|D\mu|) = \gamma \sum_{l=1}^m w_l \zeta_\epsilon(|[D\mu]_l|). \tag{3.12}$$

For solving the more general optimization problem (3.10) we use the following iterative procedure [55] where at the $(k+1)$-th iteration we solve the following approximation of (3.10)

$$\begin{aligned} \mu^{(k+1)} &= \arg\min_{\mu} F^{(k)}(\mu) \\ &:= \arg\min_{\mu} F_1(\mu) + \nabla F_2(|D\mu^{(k)}|)^T |D\mu|, \end{aligned} \tag{3.13}$$

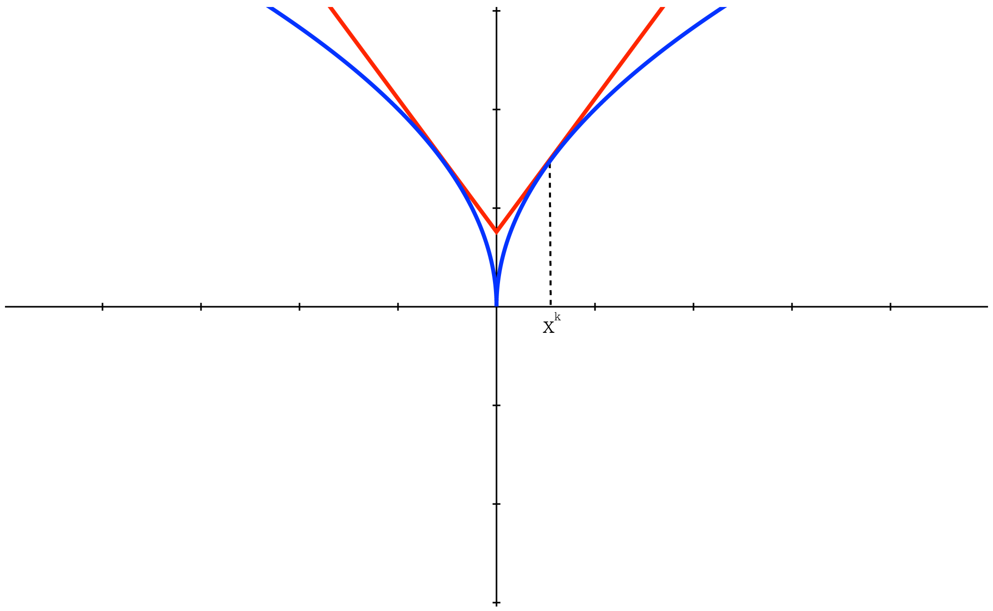
where $D\mu \in \mathbb{R}^m$. Since F_2 is by assumption increasing, the components of $\nabla F_2(|D\mu^{(k)}|)$ have non-negative values. To explain this, consider the first order Taylor approximation of F_2 around $|D\mu^{(k)}|$

$$F_2(|D\mu^{(k)}|) + \nabla F_2(|D\mu^{(k)}|)^T(|D\mu| - |D\mu^{(k)}|). \tag{3.14}$$

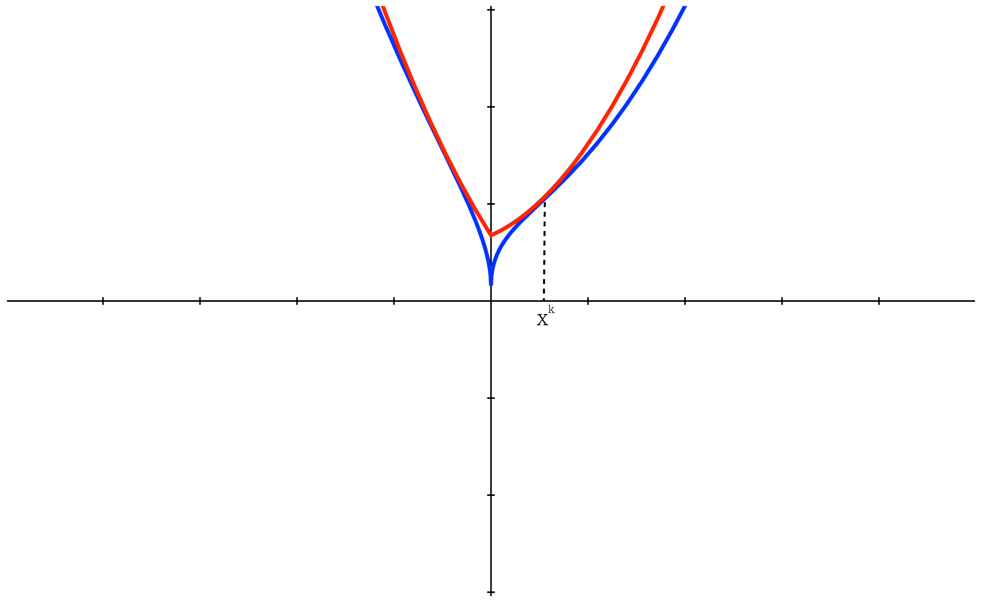
Minimizing the sum of F_1 plus (3.14) leads to a convex problem of the form

$$\arg\min_{\mu} F_1(\mu) + \nabla F_2(|D\mu^{(k)}|)^T |D\mu|,$$

which is equivalent to (3.13). This is our building block and the analysis of convergence for Algorithm 1 boils down to the analysis of the iterative scheme (3.13).



(a) Blue: $\zeta_\epsilon(|x|) = (|x| + \epsilon)^q$. Red: convex surrogate $\zeta'_\epsilon(|x^{(k)}|)|x| + C$ at $x^{(k)}$.



(b) Blue: $f(x) + \zeta_\epsilon(|x|)$. Red: $f(x) + \zeta'_\epsilon(|x^{(k)}|)|x| + C$

Figure 3.4: Functions and their surrogates involved in the MM Algorithm 1 for the one dimensional case. f is quadratic.

For $\zeta_\epsilon(|t|) = (|t| + \epsilon)^q$ and $\zeta_\epsilon(|t|) = \log(1 + \epsilon|t|)$, the derivatives with respect to $|t|$ are

$$\zeta'_\epsilon(|t|) = \frac{q}{(|t| + \epsilon)^{1-q}} \tag{3.15}$$

and

$$\zeta'_\epsilon(|t|) = \frac{\epsilon}{1 + \epsilon|t|}, \tag{3.16}$$

respectively. When F_2 is of the form (3.12), the gradient of F_2 at $|D\mu^{(k)}|$ is

$$\nabla F_2(|D\mu^{(k)}|) = (\zeta'_\epsilon(|[D\mu^{(k)}]_1), \ldots, \zeta'_\epsilon(|[D\mu^{(k)}]_m|))^T. \tag{3.17}$$

In order to solve the l_1-convex problem (3.13), we use the algorithm in [16] which is an instance of the alternating minimization algorithm (AMA)[68, 30]. In our case, the computational complexity per iteration of AMA is $\mathcal{O}(|E|)$, where $|E|$ is the number of non-zero weights w_{ij}.

3.3.1 Analysis of convergence

In order to analyze the iterative scheme (3.13):

1. we show in Proposition 1 that the sequence $\{F(\mu^{(k)}\}$ is monotonically decreasing and converges, and
2. we show in Proposition 2 the existence of an accumulation point for the sequence $\{\mu^{(k)}\}$.
3. we show in Proposition 3 that when F_2 is of the form (3.12), any accumulation point of the sequence $\mu^{(k)}$ is a stationary point for (3.10).

Our analysis proceeds the same way as [15, 55].

Proposition 1. *The sequence $\{F(\mu^{(k)})\}$ generated by Algorithm (3.13) monotonically decreases and converges.*

Proof. Le $\mu^{(k+1)}$ be a local minimizer of $F^{(k)}(\mu)$. According to the Karush-Kuhn-Tucker conditions

$$0 \in \partial F^{(k)}(\mu^{(k+1)}) = \partial F_1(\mu^{(k+1)}) + \xi^{(k)} \odot \partial||D\mu^{(k+1)}||_1 D^T, \tag{3.18}$$

where $\odot$ denotes the component-wise product of vectors. By assumption, we have

$$\xi^{(k)} \in \partial F_2(|D\mu^{(k)}|) = \{\nabla F_2(|D\mu^{(k)}|)\}, \tag{3.19}$$

that is, the gradient of F_2 is its only subgradient. Equivalently, there exist vectors

$$d^{(k+1)} \in \partial F_1(\mu^{(k+1)}) = \{\nabla F_1(\mu^{(k+1)})\} \text{ and } c^{(k+1)} \in \partial||D\mu^{(k+1)}||_1, \tag{3.20}$$

such that

$$0 = d^{(k+1)} + \xi^{(k)} \odot c^{(k+1)} D^T \Leftrightarrow d^{(k+1)} = -\xi^{(k)} \odot c^{(k+1)} D^T. \tag{3.21}$$

In the sequel, we prove that the difference $F(\mu^{(k)}) - F(\mu^{(k+1)})$ is non-negative.

$$\begin{aligned}
&F(\mu^{(k)}) - F(\mu^{(k+1)}) \\
&= F_1(\mu^{(k)}) - F_1(\mu^{(k+1)}) + F_2(|D\mu^{(k)}|) - F_2(|D\mu^{(k+1)}|) \\
&\geq (d^{(k+1)})^T(\mu^{(k)} - \mu^{(k+1)}) \\
&\qquad + (\xi^{(k)})^T(|D\mu^{(k)}| - |D\mu^{(k+1)}|) \\
&\overset{(a)}{=} (-\xi^{(k)} \odot c^{(k+1)} D^T)^T(\mu^{(k)} - \mu^{(k+1)}) \\
&\qquad + (\xi^{(k)})^T(|D\mu^{(k)}| - |D\mu^{(k+1)}|) \\
&= (\xi^{(k)})^T(|D\mu^{(k)}| - |D\mu^{(k+1)}| \\
&\qquad - c^{(k+1)} \odot D\mu^{(k)} + c^{(k+1)} \odot D\mu^{(k+1)}) \\
&\overset{(b)}{=} (\xi^{(k)})^T(|D\mu^{(k)}| \\
&\qquad - |D\mu^{(k+1)}| - c^{(k+1)} \odot D\mu^{(k)} + |D\mu^{(k+1)}|) \\
&= (\xi^{(k)})^T(|D\mu^{(k)}| - c^{(k+1)} \odot D\mu^{(k)}) \\
&\overset{(c)}{\geq} 0,
\end{aligned} \tag{3.22}$$

which means that the sequence decreases. In the first inequality, we use the definition of subgradient

$$||D\mu^{(k)}||_1 - ||D\mu^{(k+1)}||_1 \geq (c^{(k+1)})^T(|D\mu^{(k)}| - |D\mu^{(k+1)}|),$$

and the fact

$$\begin{aligned}
&F_2(|D\mu^{(k)}|) + (\xi^{(k)})^T(|D\mu^{(k+1)}| - |D\mu^{(k)}|) \\
&\quad \geq F_2(|D\mu^{(k+1)}|) \\
&\qquad\qquad \Longleftrightarrow \\
&F_2(|D\mu^{(k)}|) - F_2(|D\mu^{(k+1)}|) \\
&\quad \geq (\xi^{(k)})^T(|D\mu^{(k)}| - |D\mu^{(k+1)}|),
\end{aligned} \tag{3.23}$$

since F_2 is concave. In equality (a) we use (3.21) and in (b) the fact that $c^{(k+1)} \odot D\mu^{(k+1)} = |D\mu^{(k+1)}|$. The last inequality (c) holds because $\xi^{(k)} \in \mathbb{R}^m_+$ and $|[D\mu^{(k)}]_l| \geq [c^{(k+1)}]_l[D\mu^{(k)}]_l$, as $|[c^{(k+1)}]_l| \leq 1$. The sequence $\{F(\mu^{(k)}\}$ decreases and, by property (assumption) of F_1 is bounded from below. Hence, it converges. □

In order to show that the sequence $\{\mu^{(k)}\}$ has an accumulation point, it suffices to show F is coercive, that is

$$F(\mu) \to \infty \quad \text{as} \quad ||\mu||_2 \to \infty.$$

This is true, since F is the sum of a bounded from below convex function plus an increasing function.

Proposition 2. *Let $\{\mu^{(k)}\}$ be a sequence generated by Algorithm (3.13), then the sequence $\{\mu^{(k)}\}$ is bounded and has at least one accumulation point.*

Proof. By Proposition 1, $\{F(\mu^{(k)})\}$ is a monotonically decreasing sequence, and therefore the sequence $\{\mu^{(k)}\}$ is contained in the closed level set

$$S(\mu^{(0)}) := \{\mu : F(\mu) \leq F(\mu^{(0)})\}.$$

Since F is coercive, we conclude boundedness of the set $S(\mu^{(0)})$. This allows us to apply the Bolzano-Weierstrass theorem, which guarantees he existence of an accumulation point. $\square$

For further analysis of Algorithm (3.13), we need to make additional assumptions about the function F_2. If we assume that F_2 satisfies the conditions (C1) and (C2) stated in [55] then, we can show that any accumulation point $\mu^\star$ of the sequence $\{\mu^{(k)}\}$ generated by Algorithm (3.13) is a local minimizer for F in (3.10). In fact, Lemma 3 from [55] proves that F_2 as defined in (3.12) fulfills the following conditions:

(C1) F_2 is twice continuously differentiable in $\mathbb{R}^m_+$ and there exists a subspace $\mathbb{R}^m_c \subset \mathbb{R}^m$ such that for all $D\mu \in \mathbb{R}^m$ holds: $h^T \partial^2 F_2(|D\mu|)h < 0$ if $h \in \mathbb{R}^m_c$ and $h^T \partial^2 F_2(|D\mu|)h = 0$ if $h \in (\mathbb{R}^m_c)^\perp$.

(C2) F_2 can be expressed as the sum of a convex function and a smooth continuously differentiable function.

Assuming (C1) and (C2) and using Lemma 1 from [55] that states

$$\lim_{k\to\infty} (\partial F_2(|D\mu^{(k)}|) - \partial F_2(|D\mu^{(k+1)}|)) = 0, \tag{3.24}$$

where 0 denotes the zero vector, we have the following Proposition.

Proposition 3 ([55]). *Let $\{\mu^{(k)}\}$ be a sequence generated by Algorithm (3.13) and consider F_2 defined in (3.12)*

$$F_2(|D\mu|) = \gamma \sum_{l=1}^{m} w_l \zeta_\epsilon(|[D\mu]_l).$$

Suppose that $\mu^\star$ is an accumulation point of $\{\mu^{(k)}\}$. Then, $\mu^\star$ is a stationary of $F(\mu) = F_1(\mu) + F_2(|D\mu|)$, that is, $0 \in \partial F(\mu^\star)$.

Proof. Proposition 2 guarantees the existence of an accumulation point, that is the existence of the limit of a subsequence $\{\mu^{(k_j)}\}$ of $\{\mu^{(k)}\}$ converging to $\mu^\star$. Passing to a subsequence $\{\mu^{(k_j)}\}$ and using (3.21) we have:

$$0 = d^{(k_j+1)} + \xi^{(k_j)} \odot c^{(k_j+1)} D^T.$$

Combining the previous equation with (3.24), we conclude that at the limit $j \to \infty$ we have

$$\lim_{j\to\infty} \omega^{(k_j+1)} = 0, \quad \omega^{(k_j+1)} := d^{(k_j+1)} + \xi^{(k_j+1)} \odot c^{(k_j+1)}, \tag{3.25}$$

and

$$\omega^{(k_j+1)} \in \partial F(\mu^{(k_j+1)}). \tag{3.26}$$

Using the following well known property (Proposition 16.28 in [6] and Remark 2.2 in [23]) about the subdifferential of a C^1-perturbation of a convex function, that is, a function of the form $F = F_1 + F_2$, where F_1 is lower semi-continuous, and F_2 is of class C^1:

$$\omega^{(k_j+1)} \in \partial F(\mu^{(k_j+1)}),\ \mu^{(k_j+1)} \to \mu^\star,\ \omega^{(k_j)} \to \omega \implies F(\mu^{(k_j)}) \to F(\mu^\star),\ \omega \in \partial F(\mu^\star),$$

and combining with (3.25)-(3.26), we have that $0 \in \partial F(\mu^\star)$, which concludes the proof. So, every accumulation point $\mu^\star$ of Algorithm (3.13) is also a local minimizer for F. $\square$

3.3.2 Algorithm implementation

As a convex minimization problem, the inner problem of (3.13) can be solved efficiently with the AMA algorithm. We focus on the (outer) non-convex problem (3.7). Let $\mu^{(k)}$ be the sequence generated by the MM algorithm in Algorithm 1, where the index k refers to the outer iterations. Execution is terminated when

$$\frac{f(\mu^{(k)}) - f(\mu^{(k+1)})}{f(\mu^{(0)})} < \tau \quad \text{or} \quad k > m_{\text{outer}}, \tag{3.27}$$

where τ is a threshold defining the desired accuracy and m_{outer} the maximal number of iterations.

The output of the AMA algorithm in (3.13), consists of the vectors μ^l and $z^l = D\mu^l$ for every column l of the data matrix X. The columns z^l form a matrix $Z \in \mathbb{R}^{m \times p}$. The euclidean norm $||z_r||_2^2$ of row z_r of Z, where r represents a pair $(i, j),\ j < i$, defines the distance between the centroids μ_i and μ_j. In order to make cluster assignments, we form the matrix $C = (c_{ij})$ with $c_{ii} = 0$, and $c_{ij} = 1$ if the distance $||z_r||_2^2$ is below a user defined threshold τ, otherwise $c_{ij} = 0$. Then final clusters are the connected components of the undirected graph with adjacency matrix C [64].

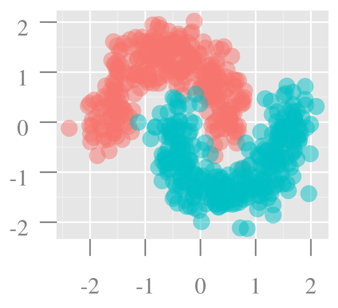
The proposed iterative algorithm can be directly applied to problems with hundreds of points. Moreover, it scales well for thousands of points by taking advantage of the decomposability of the problem across its p dimensions. In this case, we can solve p distributed problems on different processors. As an alternative, we can first cluster the data into a large number of clusters using k-means or minimax linkage, and then cluster the resulting centroids into bigger clusters using our proposed method.

3.4 Experiments

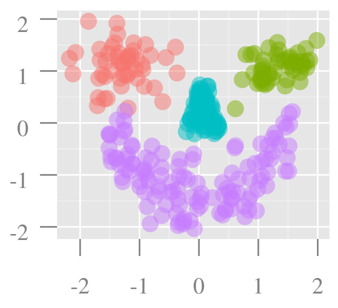
In this section, we test the accuracy of our method. Our proposed method, the non-convex clusterpath with $\zeta_\epsilon(|t|) = (|t| + \epsilon)^q$, $q = 0.5$, and $\epsilon = 0.001$, is compared to the following algorithms: i) convex clusterpath with $\zeta(|t|) = |t|$, ii) convex clusterpath with $\zeta(|t|) = |t|^2$, iii) k-means, iv) spectral k-means, v) Gaussian mixtures, and vi) minimax linkage clustering.

The convex clusterpath with either the l_1 or the l_2-norm is the method proposed in [37, 16]. In both cases, convex and noncovex, a threshold value of $\tau = 0.0001$ was set for the convergence of the algorithms. Spectral k-means is the usual k-means algorithm applied to the matrix formed by the k-largest eigenvectors of the similarity matrix L defined in [54]. Minimax linkage [9] is a new agglomerative clustering algorithm based on the definition of a new linkage function called minimax linkage. Last, Gaussian mixtures [26] were trained using the default setting from the `R` package [25].

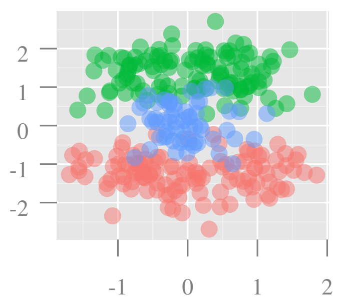
The performance of the algorithms is measured using the normalized rand index [39] which varies from 1, that denotes a perfect match, to 0, denoting a completely random assignment. For the experimental datasets, we consider a scenario with noise added to the original samples. The noise follows a two-variate normal distribution $\mathcal{N}\big(\mu = (0,0)^T, \Sigma^{-1} = \left[\begin{smallmatrix} \sigma & 0 \\ 0 & \sigma \end{smallmatrix}\right]\big)$, where σ is a positive scalar. As far as the choice of weights (3.3) is concerned, in the experiments with artificial data, we set $k_{nn} = 15$ and $\phi = 0$ and 0.5. In Algorithm 1, we set the number of inner iterations $m_{\text{inner}} = 1200$ and the number of outer iterations $m_{\text{outer}} = 6$. In all experiments, the number of clusters are known to the algorithms.



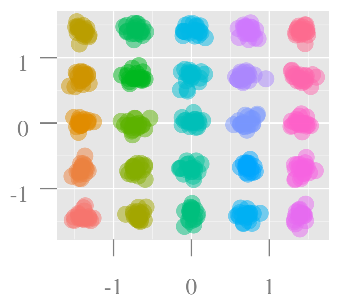
(a) Half-moon, $\sigma = 0.25$



(b) Smiley, $\sigma = 0.25$



(c) Cassini, $\sigma = 0.1$



(d) Gaussian grid, $\sigma = 0.5$

Figure 3.5: Example plots of the artificial datasets when the Gaussian noise added to each dimension has zero mean and variance σ. Different colors denote different clusters.

3.4.1 Results on non-convex clusters

We consider three datasets [46]: the“half-moon”, the “cassini”, and the “smiley” which have non-convex shapes that we proceed to describe (see also Figures 3.5(a)-(c)).

Half-moon dataset

The $n = 600$ samples in this dataset come from two half-moon entagled clusters, which we used as input for the algorithms we test. Table 3.1 shows the adjusted rand index for each algorithm for 20 simulations of the half-moon dataset. Hierarchical clustering and k-means fail because the datasets have non-convex shapes. The performance of convex and non-convex clusterpath is comparable for low level noise ($\sigma = \{0, 0.25\}$) and both of them, along with spectral k-means, are able to identify the true clusters. In the noisy case of $\sigma = 0.5$, clusterpath is more robust compared to all other methods. We note that the choice of ϕ for the weights does not influence the performance of clusterpath methods.

Smiley dataset

The smiley dataset has $n = 300$ points and consists of: 2 Gaussian eyes, a trapezoid nose and a parabola mouth (with vertical Gaussian noise). The standard deviation for eyes and for mouth is σ. Noise does not affect the performance of our method, as shown in Table 3.2. A possible explanation for this phenomenon is the increased convexity of the dataset as the noise level gets higher. Also, the best results for the clusterpath algorithms are obtained for $\phi = 0.5$.

Cassini dataset

The $n = 300$ samples in the cassini dataset are uniformly distributed on a two dimensional space within three structures. The two external classes are banana-shaped structures and in between them there is a circle. The robustness of non-convex penalty in the presence of noise is evident (Table 3.3); non-convex clusterpath is able to identify the correct clusters in most of the cases. Also, the value of ϕ has little influence on its performance, compared to the convex clusterpath which is more sensitive to the choice of ϕ.

3.4.2 Verification on convex-clusters

We tested our algorithm in with a dataset of 25 Gaussian clusters arranged in a 5×5 two dimensional grid. For each cluster, we generate 20 data points (Figure 3.5(d)). Clusterpath goes remarkably well and scores the highest rand index along with Gaussian mixtures (Table 3.4). Note that Gaussian mixtures can be considered as the optimal method to cluster such samples, since it assumes that all the data points are generated from a mixture of a finite number of Gaussian distributions, and this is the actual case.

3.4.3 Fisher’s Iris dataset

This Fisher’s iris data set gives the measurements of four variables, for 50 flowers from each of 3 species of iris. Two out of three classes overlap, so Gaussian mixture model is the only algorithm capable of accurately detecting these clusters. Surprisingly, as shown in Tables 3.5-3.6, the performance of non-convex clusterpath is 1 when $k_{nn} \in [15, 20)$, regardless the value of ϕ. To further investigate this case, we plot the estimated regularization paths in Figure 3.8 when $\phi = 0.5$

Table 3.1: Adjusted rand index for 20 simulation on the half moon dataset for several values of σ.

		σ		
	algorithm	0.0	0.25	0.5
	k-means	0.56	0.51	0.51
	spectral k-means	**1.00**	0.73	0.49
	Gaussian mixtures	0.25	0.54	0.52
	minimax linkage	0.5	0.47	0.44
$\phi = 0.0$	clusterpath $q = 0.5$	**1.00**	**1.00**	0.99
	clusterpath $q = 1$	**1.00**	0.99	0.99
	clusterpath $q = 2$	**1.00**	0.99	**1.00**
$\phi = 0.5$	clusterpath $q = 0.5$	**1.00**	**1.00**	**1.00**
	clusterpath $q = 1$	**1.00**	0.95	0.72
	clusterpath $q = 2$	**1.00**	0.95	0.55

Table 3.2: Adjusted rand index for 20 simulation on the smiley dataset for several values of noise levels σ.

		σ		
	algorithm	0.0	0.25	0.5
	k-means	0.52	0.52	0.43
	spectral k-means	0.74	0.7	0.5
	Gaussian mixtures	0.99	0.87	0.65
	minimax linkage	0.38	0.62	0.41
$\phi = 0.0$	clusterpath $q = 0.5$	0.7	0.85	**1.00**
	clusterpath $q = 1$	0.59	0.54	0.54
	clusterpath $q = 2$	0.83	0.81	0.8
$\phi = 0.5$	clusterpath $q = 0.5$	**1.00**	**1.00**	**1.00**
	clusterpath $q = 1$	**1.00**	0.98	0.55
	clusterpath $q = 2$	1.00	0.97	0.52

Table 3.3: Adjusted rand index for 20 simulation on the cassini dataset for several values of σ.

		σ		
	algorithm	0.0	0.1	0.2
	k-means	0.21	0.19	0.18
	spectral k-means	0.55	0.3	0.22
	Gaussian mixtures	0.99	0.55	0.35
	minimax linkage	0.28	0.2	0.18
$\phi = 0.0$	clusterpath $q = 0.5$	**1.00**	**1.00**	**0.99**
	clusterpath $q = 1$	0.6	0.5	0.4
	clusterpath $q = 2$	0.6	0.52	0.41
$\phi = 0.5$	clusterpath $q = 0.5$	0.98	0.97	0.98
	clusterpath $q = 1$	0.93	0.4	0.39
	clusterpath $q = 2$	0.44	0.35	0.32

Table 3.4: Adjusted rand index for 20 simulation on the Gaussian grid dataset for several values of σ.

	algorithm	σ 0.0	0.5	1
	k-means	0.81	0.85	0.85
	spectral k-means	0.84	0.8	0.83
	Gaussian mixtures	**1.00**	**1.00**	**0.95**
	minimax linkage	**1.00**	0.99	**0.95**
$\phi = 0.0$	clusterpath $q = 0.5$	0.99	0.99	0.88
	clusterpath $q = 1$	0.09	0.13	0.1
	clusterpath $q = 2$	0.11	0.13	0.1
$\phi = 0.5$	clusterpath $q = 0.5$	**1.00**	**1.00**	0.85
	clusterpath $q = 1$	**1.00**	**1.00**	0.42
	clusterpath $q = 2$	0.34	0.3	0.26

Table 3.5: Iris dataset. Performance of benchmark algorithms.

algorithm	rand index
k-means	0.58
spectral k-means	0.56
Gaussian mixtures	0.90
minimax linkage	0.6

Table 3.6: Iris dataset. Performance of clusterpath for $q \in \{0.5, 1, 2\}$ when $k_{nn} \in [15, 20)$ and $k_{nn} \in [20, 50)$.

$k_{nn} \in [15, 20]$		
	algorithm	rand index
$\phi = 0.0$	clusterpath $q = 0.5$	**1.00**
	clusterpath $q = 1$	0.56
	clusterpath $q = 2$	0.56
$\phi = 0.5$	clusterpath $q = 0.5$	**1.00**
	clusterpath $q = 1$	0.57
	clusterpath $q = 2$	0.57

$k_{nn} \in (20, 50]$		
	algorithm	rand index
$\phi = 0.0$	clusterpath $q = 0.5$	**0.6**
	clusterpath $q = 1$	0.56
	clusterpath $q = 2$	0.56
$\phi = 0.5$	clusterpath $q = 0.5$	**0.6**
	clusterpath $q = 1$	0.57
	clusterpath $q = 2$	0.57

Table 3.7: Glass dataset. Performance of tested algorithms.

	algorithm	rand index
	k-means	0.22
	spectral k-means	0.28
	Gaussian mixtures	0.20
	minimax linkage	0.18
$\phi = 0.0$	clusterpath $q = 0.5$	**0.49**
	clusterpath $q = 1$	0.1
	clusterpath $q = 2$	0.1
$\phi = 0.5$	clusterpath $q = 0.5$	**0.48**
	clusterpath $q = 1$	0.1
	clusterpath $q = 2$	0.1

and $k_{nn} = 15$. The upper row in Figure 3.8 corresponds to the l_1-regularization path while the bottom row to the $l_{0.5}$-regularization path. The structure of $l_{0.5}$-regularization path is much more balanced, especially for the features sepal length and petal width, making it possible to infer that the number of true classes in the dataset is three. Also, we note that in the case of the l_1-clusterpath, we are not able to guarantee that $||\mu_i - \mu_j||_1 = 0$ as $\gamma \to \infty$ even approximately; in the l_1-solution path the centroids tend to remain in two clusters as γ increases. The issue of an l_q-norm penalty with $q \geq 1$ in yielding possibly severely biased estimates is well known in penalized regression, which partially motivated the development of non-convex penalties [56]. In Figure 3.6, we plot the clusterpaths (black lines) for the l_1-norm and the $l_{0.5}$-norm when $k_{nn} \in [15, 20]$. Parameter ϕ has less influence on the clusterpath solutions compared to k_{nn} which dramatically affects their performance.

3.4.4 Dentition of mammals

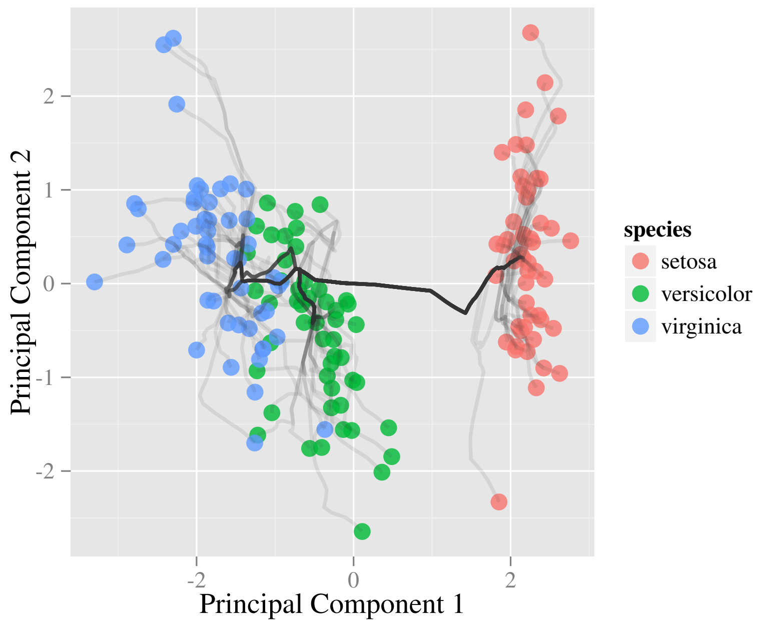
We consider the problem of clustering 27 mammals based on their dentition [20]. This experiment is motivated by a recently proposed method to solve the one-way ANOVA problem [34]. The tooth patterns concerned are the number of top incisors, bottom incisors, top canines, bottom canines, top premolars, bottom premolars, top molars, and bottom molars. Figure 3.7 shows the resulting clustering paths when $k_{nn} = 5$ for $\phi = 0$ and 0.5. The nonconvex clusterpath, despite the highly "zig-zag" lines, gives a different and perhaps more sensible solution than the convex clusterpath. For example, in Figure 3.7(a) the opposum, htailmole, and common mole are considered more similar even though the distance between them is too large in the first two PCA coordinates. Also, fur seal and sea otter are considered more similar in non-convex clusterpath rather than the convex one. Finally, the wolf and raccoon are considered less similar in the non-convex rather than the convex clusterpath, although their corresponding rows in the data matrix differ only in one element.

3.4.5 Glass Identification dataset

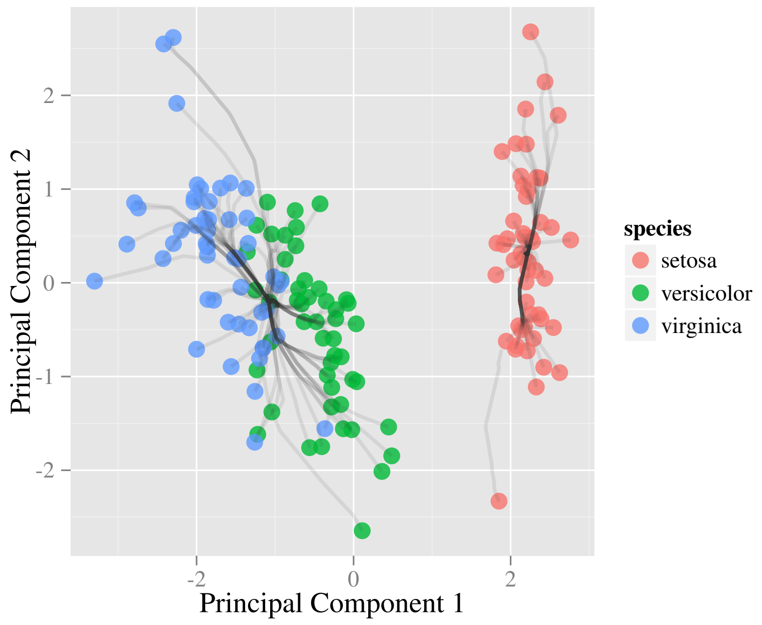
In this setting, we test our algorithm on a dataset initially used for classification, the glass identification dataset [4]. This dataset contains 214 observations on 10 variables of the chemical analysis of 6 different types of glass. The study was motivated by criminological investigation; at the scene of the crime, the glass left can be used as evidence. The classification problem lies on forecasting the type of glass based on its chemical analysis. Here, instead of the classification problem we solve the clustering problem. Once again, we test the performance of our method against the benchmarks of the previous subsections. The results are shown in Table 3.7. Non-convex clusterpath achieves the highest score followed by spectral k-means and the usual k-means. Convex clusterpath fails due to the increased overlap of the true classes.

3.5 Conclusions

In this work, we extend the previous convex relaxations of k-means and hierarchical clustering to also cover the use of non-convex penalties in clustering. For the penalty minimization problem we used a majorization/minimization algorithm, that iteratively minimizes l_1-norm approximations of the objective function. Experimental evaluations showed that the use of non-convex penalties in clustering, is a strong candidate for analyzing non-convex datasets. As future work, we consider to use more general loss function rather than the squared error loss of k-means and we plan to build a distributed implementation and test it on very large datasets. The decomposable nature of the objective function, implies that our algorithm will scale well.

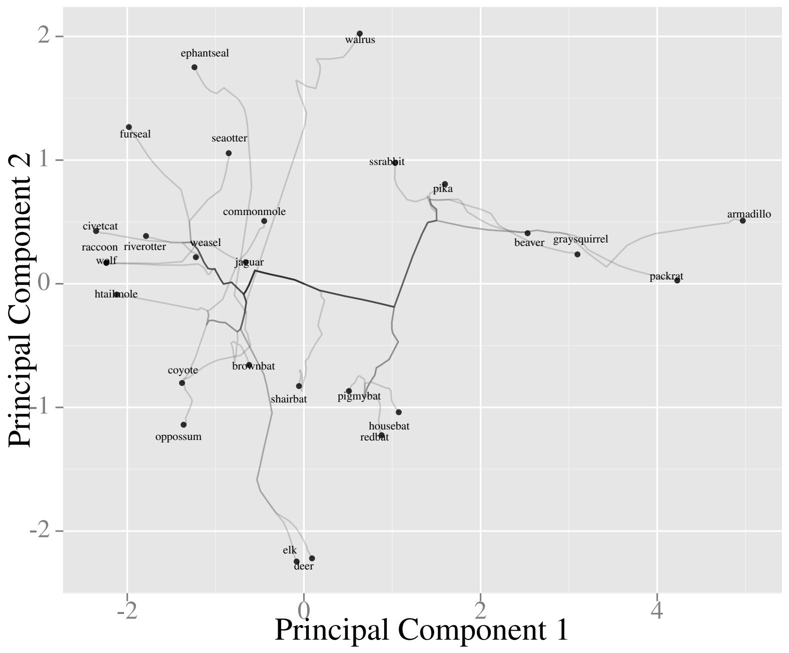


(a) $l_{0.5}$ clusterpath

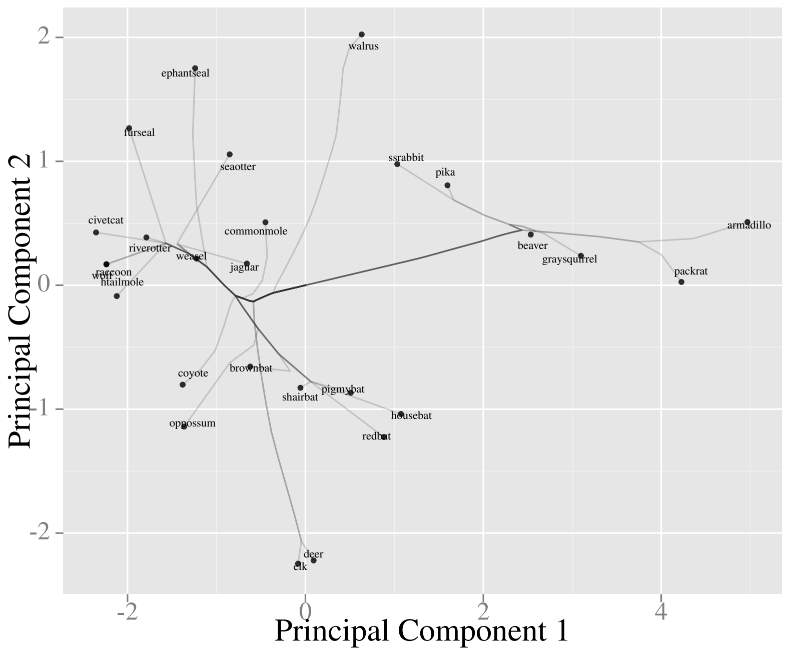


(b) l_1 clusterpath

Figure 3.6: The calculated $l_{0.5}$ non-convex and l_1 convex clusterpaths (black lines) projected onto the first and second principal components. The weights w_{ij} are decreasing and constrained to be non-zero in a neighborhood $k_{nn} = 15$.



(a) $l_{0.5}$ clusterpath



(b) l_1 clusterpath

Figure 3.7: Non-convex and convex clustering paths for the mammals dataset projected onto the first two principal components. The weights w_{ij} are decreasing and constrained to be non-zero in a neighborhood $k_{nn} = 5$.

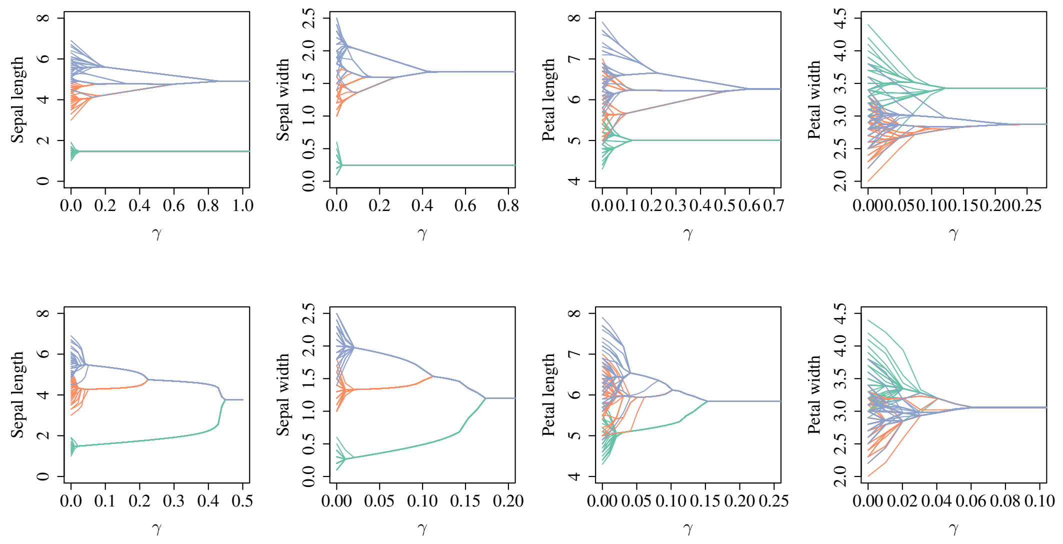


Figure 3.8: Solution paths μ_i for the l_1-clusterpath (upper row) and the $l_{0.5}$-clusterpath (bottom row). In both cases $\phi = 0.5$ and $k_{nn} = 15$.

Chapter 4

Joint Estimation of Multiple Undirected Graphical Models

Abstract of this chapter: Gaussian graphical models are of great interest in statistical learning. Since the conditional independence between different nodes correspond to zero entries in the inverse covariance matrix of the Gaussian distribution, one can learn the structure of the graph by estimating a sparse inverse covariance matrix from sample data. This is usually done by solving a convex maximum likelihood problem with an l_1-regularization term. In this chapter, we develop an estimator for such models appropriate for data from several datasets that share the same set of variables and a common network substructure. We assume that the differences among the network under study are generated due to edge perturbations; there are a few different edges among the networks while the others (edges) are common. To this end, we form an optimization problem that exploits the problem's special structure and we propose a first-order method for its solution based on the alternating direction method of multipliers. We confirm the performance improvement of our method over existing methods in finding the dependence structure on simulated and real datasets.

4.1 Introduction

The aim of this chapter is to analyze the underlying interactions or dependencies among variables that are common across multiple datasets. A major challenge in complex systems, e.g., stock market, gene regulatory networks, or sensors systems, is not to simply construct an accurate predictive model, but rather to discover meaningful interactions among the variables of the system. The properties of samples from such systems dynamically emerge over time or due to changes in surrounding environment. Such effects cause data to have different behavior under different conditions. Probabilistic graphical models, such as Markov random networks, are a popular tool to analyze complex dependencies among many variables. We adopt Markov random networks over Gaussian random variables, also known as Gaussian graphical models (GGM) [45], as the basis of our method.

In the GGM framework, we aim to infer the graph of dependencies among p variables of a Gaussian random vector $x \in \mathbb{R}^p \sim \mathcal{N}_p(\mu, \Sigma)$ from a given set of samples. This task is equivalent to learning the zero entries of precision matrix $\Theta = \Sigma^{-1}$. A precision matrix is translated to an adjacency matrix for a graph $\mathcal{G}$ through the following process [45]. If the entry θ_{ij} of Θ is zero, then there is no edge between x_i and x_j, indicating that x_i and x_j are conditional independent given all other variables. Otherwise, there exists an edge between x_i and x_j, indicating that, given all other variables, x_i and x_j are correlated. This approach produces a graph from an inferred precision matrix Θ. The latter, cannot be produced by an unconstrained maximum likelihood estimation (ML), since ML produces a full matrix with probability 1 and thus a useless fully connected graph.

Identification of zero entries in Θ was first studied by [21], as a covariance selection problem where the task is formulated as a combinatorial problem of optimizing the location of zeros in a matrix. Since then, a number of algorithms have been proposed ranging from stepwise selection methods, for detecting significant edges, to convex optimization methods. Stepwise selection procedures start from a fully connected graph and remove edges until all remaining edges are significant according to a partial correlation test. A significant drawback of the latter methods is large computational complexity. Usually, only a small number of the relevant search space is searched leading to inaccurate network estimates [35].

More recently, the focus has shifted to a relaxed setting [51, 5, 72, 27] where covariance selection is formulated as a convex optimization problem using a l_1-regularization that induces zeros in the resulting precision matrix. However, these formulations assume that observations are identically drawn from a single Gaussian distribution. In our context, the objective is not to estimate the structure of a single graph from a single dataset, but the structure of several graphs coming from distinct but related datasets.

There is some recent work in this direction, that is the estimation of multiple graphical models, where many sparse regression methods have been utilized for this purpose. For example, group lasso [41] was generalized to joint estimation of undirected gaussian networks with the assumption that all networks share a common pattern of sparsity [69]. To overcome the latter assumption, authors in [33] considered a similar method although the problem they solve is not convex. Fused lasso [66] is another regreesion method used to find invariant patterns among many networks [19, 71]. Sign coherence on the precision estimated precision matrices was proposed in [17], while other studies have used time series data to define time varying networks [1, 73].

Our contribution. First, we formulate a convex optimization problem for joint estimation that unifies existing methods in the field and, second, we develop a simple alternating direction method of multipliers (ADMM) algorithm for its solution. The closeness assumption between graphical models is rendered into i) an empirical prior for the sample covariance matrix and ii) l_1-norm differences among precision matrices we expect to be similar. We confirm the validity of our method in finding the dependence structure on simulated and real datasets.

4.2 Methodology

Assume we are given $K \geq 2$ datasets $X^{(k)} \in \mathbb{R}^{n_k \times p}$, for $k = 1, \dots, K$, where p is the number of features common to all datasets. The total number of samples is $n = \sum_{k=1}^{K} n_k$, where n_k is the number of samples in dataset k. We further assume that samples within each $X^{(k)}$ are identically distributed according to $\mathcal{N}_p(\mu^{(k)}, \Sigma^{(k)})$. We seek to estimate $(\Sigma^{(k)})^{-1}$, for all k, by formulating a convex optimization problem with arguments $\{\Theta\} = \{\Theta^{(1)}, \dots, \Theta^{(K)}\}$, where $\Theta^{(k)} = (\Sigma^{(k)})^{-1}$. The solution $\hat{\Theta}^{(k)}$ is an estimate of $(\Sigma^{(k)})^{-1}$. The most direct way to find a solution is to minimize the negative log likelihood [10] for the data in k-th category

$$l(\Theta^{(k)}; S^{(k)}) = -n_k(\log \det \Theta^{(k)} - \operatorname{tr}(S^{(k)}\Theta^{(k)})), \tag{4.1}$$

where $S^{(k)}$ is the sample covariance matrix. However, the estimated matrix $\hat{\Theta}^{(k)}$ will be dense and the corresponding network will be a useless fully connected graph.

4.2.1 Separable estimation method

The most direct way to deal with K different is to separately estimate K individual sparse graphical models. We can compute l_1-regularized estimators [27], by solving K problems of the form $\min_{\Theta^{(k)}} l(\Theta^{(k)}) + \lambda_1 ||\Theta^{(k)}||_1$ or equivalently the problem

$$\min_{\{\Theta\}=\{\Theta^{(1)},\ldots,\Theta^{(K)}\}} \quad \sum_{k=1}^{K} l(\Theta^{(k)}; S^{(k)}) + \lambda_1 \sum_{k=1}^{K} ||\Theta^{(k)}||_1, \tag{4.2}$$

where the minimum is taken over all symmetric positive definite matrices $\Theta^{(k)}$. Term λ_1 is a non-negative tuning parameter that controls the sparsity in the estimated precision matrices. We will refer to this approach as separate estimation method [27] and use it as a baseline to compare with the joint estimation method we propose next and other already proposed joint estimation methods.

4.2.2 Joint estimation method

To benefit from the fact that networks may share an unknown and common substructure, we propose the following. We modify the data fitting term $S^{(k)}$ in (4.1) as the convex combination

$$\tilde{S}^{(k)} = \alpha S^{(k)} + (1-\alpha)\bar{S}, \tag{4.3}$$

where $0 \leq \alpha \leq 1$, and $\bar{S} = n^{-1} \sum_{k=1}^{K} n_k S^{(k)} \in \mathbb{R}^{p \times p}$ is the weighted average of the K empirical covariance matrices. We call $\bar{S}$ the pooled covariance matrix [28]. The presence of α leads to a reduction of variance in the estimated precision matrices at the expense of potentially increased bias between them [17]. The value $\alpha = 0$ implies no interaction among the estimated networks, while $\alpha = 1$ pools all the estimated networks towards a common one. The greater α is the more biased the estimated graphs are. The idea behind this approach is taken from regularized discriminant analysis [28] but here we aim to bias empirical distributions towards a common model. Intuitively, if $\log p/n_k \to 0$ for all k, then $\alpha \to 0$, since in this case we have a large sample dataset to accurately estimate each precision matrix $\Theta^{(k)}$ [59].

In the case of known interactions among the different datasets and their corresponding graphical networks, we introduce another set of parameters, called $\lambda_{(k,k')}$, and consider the modified penalty

$$P(\{\Theta\}) = \lambda_1 \sum_{k=1}^{K} ||\Theta^{(k)}||_1 + \sum_{\substack{(k,k') \in E^\star \\ k \neq k'}} \lambda_{(k,k')} ||\Theta^{(k)} - \Theta^{(k')}||_1, \tag{4.4}$$

where $E^\star$ is the edge set of a known graph $\mathcal{G}^\star = (V^\star, E^\star)$ with $|V^\star| = K$ nodes. In Figure 4.1(a), the solid black lines correspond to known interactions among four graphs $\mathcal{G}^{(k)},\ k = 1, \ldots, 4$. In order to promote similarity between two graphs, we apply a l_1-norm penalty on the difference of the corresponding precision matrices. The graph $\mathcal{G}^\star$ that defines the fusion penalties among the K graphical networks is essentially a *network of networks*, where each node corresponds to a precision matrix $\Theta^{(k)}$. If the edge (k, k') appears in $\mathcal{G}^\star$, then there is a fusion between the corresponding elements of the pair $(\Theta^{(k)}, \Theta^{(k')})$. In this case, a term of the form $\lambda_{(k,k')} ||\Theta^{(k)} - \Theta^{(k')}||_1$ appears in (4.4) and when $\lambda_{(k,k')}$ is large this pair of matrices will also have close elements.

Table 4.1: Special cases of (4.5) for different values of α and $\lambda_{(k,k')}$.

method	α	$\lambda_{(k,k')}$	$\mathcal{G}^\star$
graphical lasso [27]	$\alpha = 1$	$\lambda_{(k,k')} = 0$	empty
intertwined lasso [17]	$\alpha \neq 1$	$\lambda_{(k,k')} = 0$	empty
fused graphical lasso [19, 71]	$\alpha = 1$	$\lambda_{(k,k')} \neq 0$	fully connected, 1-d chain
proposed method	$\alpha \neq 1$	$\lambda_{(k,k')} \neq 0$	arbitrary

To better understand our rationale behind the "network of networks" $\mathcal{G}^\star$, consider a complex system, like a network of sensors [36] or the interaction of proteins in a biological cell [57], that is evolving over time. This system varies into more specialized systems with known interactions among them. These red interactions can be represented as a graph $\mathcal{G}^\star$ that directly defines the relations among the estimated networks. Our approach makes use of $\mathcal{G}^\star$ to define specific fusion penalties for joint estimation of multiple undirected graphical models. Putting it all together we come up with the following problem

$$\begin{aligned} \min_{\{\Theta\}} \quad & \sum_{k=1}^{K} l(\Theta^{(k)}; \tilde{S}^{(k)}) + P(\{\Theta\}) \\ \text{s.t} \quad & \Theta^{(k)} \succ 0, \ \text{for } k = 1, \dots, K, \end{aligned} \tag{4.5}$$

with $\tilde{S}^{(k)}$ in place of $S^{(k)}$. The solution $\{\hat{\Theta}\} = \{\hat{\Theta}^{(1)}, \dots, \hat{\Theta}^{(K)}\}$ consists of the jointly estimated precision matrices.

We note that the reliance of our algorithm on tuning parameters λ_1 and $\lambda_{(k,k')}$ is an advantage rather than a drawback: unlike the proposals which involve a single tuning parameter that controls both sparsity and similarity [33], by adopting our method one can configure separately the enforced amount of similarity and sparsity in the network estimates. Also, our method is closely related to the fused graphical lasso (FGL) [19] and TREEGL [57]. However, FGL makes the rather unrealistic assumption that each graph is connected to all the remaining graphs. In TREEGL, the penalty has many similarities with (4.4) but there is no modification in the log-likelihood (in our case we replace the sample covariance matrix S with $\tilde{S}$ in (4.3)). Our algorithm incorporates the graph $\mathcal{G}^\star$ and is much more flexible. In Table 4.1, we summarise some known forms of (4.5) as the tuning parameters α, $\lambda_{(k,k')}$, and $\mathcal{G}^\star$ vary.

4.3 Optimization via ADMM

ADMM [10] is an algorithm that is intended to blend the decomposability of dual ascent with the convergence properties of the method of multipliers. ADMM solves the problem

$$\min_{\{x,z\}} \quad f(x) + g(z) \quad \text{s.t} \quad Ax + Bz = c, \tag{4.6}$$

where $\{x, z\} \in \mathbb{R}^n$, $c \in \mathbb{R}^p$, and A, B are matrices with appropriate dimensions. Many regularized sparse estimation problems in statistical learning, including the one studied here, have the form of (4.6), where $f(\cdot)$ is a proper loss function and $g(\cdot)$ the regularizer. Next, we present an ADMM reformulation of (4.5) which exhibits separable structure in both the objective and the constraints and we further describe the procedure for solving the subproblems emerging in the implementation.

4.3.1 ADMM reformulation

We introduce an additional set of variables $\{Z\} = \{Z^{(1)}, \ldots, Z^{(K)}\}$ and rewrite problem (4.5) as

$$\begin{aligned} \min_{\{\Theta\},\{Z\}} \quad & \sum_{k=1}^{K} l(\Theta^{(k)}; \tilde{S}^{(k)}) + P(\{Z\}) \\ \text{s.t} \quad & \Theta^{(k)} - Z^{(k)} = \mathbf{0},\ \Theta^{(k)} \succ \mathbf{0} \text{ and } Z^{(k)} \succ \mathbf{0},\ \text{ for } k = 1, \ldots, K, \end{aligned} \tag{4.7}$$

where $P(\{Z\})$ has the form of (4.4). Also, we assign Lagrange multipliers $Y^{(k)} \in R^{(p\times p)}$ to the linear constraints $Z^{(k)} - \Theta^{(k)} = 0$. Then, the scaled augmented Lagrangian [10] for (4.7) is

$$\begin{aligned} L_\rho(\{\Theta\}, \{Z\}, \{U\}) = & \sum_{k=1}^{K} l(\Theta^{(k)}; \tilde{S}^{(k)}) + P(\{Z\}) \\ & + \frac{\rho}{2} \sum_{k=1}^{K} ||\Theta^{(k)} - Z^{(k)} + U^{(k)}||_F^2 \end{aligned}$$

where $U^{(k)} = (1/\rho)Y^{(k)}$ are the scaled dual variables and $\rho > 0$ is the penalty parameter for the violation of the linear constraints. The i-th iteration of the ADMM algorithm consists of three steps

$$\begin{aligned} 1. \quad & \{\Theta_{(i)}\} := \operatorname*{argmin}_{\{\Theta\}} L_\rho(\{\Theta\}, \{Z_{(i-1)}\}, \{U_{(i-1)}\}) \\ 2. \quad & \{Z_{(i)}\} := \operatorname*{argmin}_{\{Z\}} L_\rho(\{\Theta_{(i)}\}, \{Z\}, \{U_{(i-1)}\}) \\ 3. \quad & \{U_{(i)}\} := \{U_{(i-1)}\}\} + \{\Theta_{(i)}\} - \{Z_{(i)}\}. \end{aligned} \tag{4.8}$$

Steps 1 and 2 are further separable as described next.

4.3.2 Solving subproblems of ADMM

Step 1 is equivalent to solving K problems of the form

$$\operatorname*{argmin}_{\Theta^{(k)}\succ 0} \left\{ -n_k(\log\det\Theta^{(k)} - \operatorname{tr}(\tilde{S}^{(k)}\Theta^{(k)})) + \frac{\rho}{2} ||\Theta^{(k)} - Z_{(i-1)}^{(k)} + U_{(i-1)}^{(k)}||_F^2 \right\}. \tag{4.9}$$

Let $V^{(k)}D^{(k)}(V^{(k)})^T$ denote the eigendecomposition of $\tilde{S}^{(k)} - \rho/n_k \left(Z_{(i-1)}^{(k)} + U_{(i-1)}^{(k)}\right)$. Then the solution to (4.9) is given by $\hat{\Theta}^{(k)} = V^{(k)}\tilde{D}^{(k)}(V^{(k)})^T$ [72], where $\tilde{D}^{(k)}$ is the diagonal matrix with jth diagonal element

$$\tilde{D}_{jj}^{(k)} = \frac{n_k}{2\rho} \left(-D_{jj}^{(k)} + \sqrt{(D_{jj}^{(k)})^2 + 4\rho/n_k} \right).$$

The complexity of Step 1 is $O(Kp^3)$ since the cost of computing the eigenvalue decomposition of a $p \times p$ matrix is $O(p^3)$.

Step 2 can be written as

$$\{Z_{(i)}\} := \operatorname*{argmin}_{Z^{(k)}\succ 0} \left\{ P(\{Z\}) + \frac{\rho}{2} \left\| \{Z\} - \{A_{(i)}\} \right\|_F^2 \right\} \tag{4.10}$$

where

$$\begin{aligned} \{A_{(i)}\} &= \{A_{(i)}^{(1)}, \dots, A_{(1)}^{(K)}\} = \{\Theta_{(i)}\} + \{U_{(i-1)}\} \\ \Leftrightarrow\ A_{(i)}^{(k)} &= \Theta_{(i)}^{(k)} + U_{(i-1)}^{(k)},\ k = 1, \dots, K. \end{aligned} \tag{4.11}$$

Equation (4.10) is separable with respect to each pair of matrix elements (i, j) and thus one can solve p^2 problems of the form

$$\begin{aligned} \underset{\{Z_{ij}\}}{\operatorname{argmin}} \Bigg\{ \lambda_1 \sum_{k=1}^{K} \left| Z_{ij}^{(k)} \right| + \sum_{\substack{(k,k')\in E^\star \\ k\neq k'}} \lambda_{(k,k')} \left| Z_{ij}^{(k)} - Z_{ij}^{(k')} \right| \\ + \frac{\rho}{2} \sum_{k=1}^{K} \left(Z_{ij}^{(k)} - A_{ij}^{(k)} \right)^2 \Bigg\}, \end{aligned} \tag{4.12}$$

for each element $Z_{ij}^{(k)}$ of $Z^{(k)}$, with $\{Z_{ij}\} = \{Z_{ij}^{(1)}, \dots, Z_{ij}^{(K)}\}$.

This is a special case of the generalized lasso problem [67] that we solve using the split Bregman method [31], and is called sparse fused lasso over an arbitrary graph $\mathcal{G}^\star = (V^\star, E^\star)$. Equation (4.12) can be rewritten in the form

$$\begin{aligned} &\min_{x,d} ||d||_1 + \frac{\rho}{2}||a - z||_2^2 \\ &\text{s.t } d = Dz. \end{aligned} \tag{4.13}$$

Now the coordinates of $a \in \mathbb{R}^K$ correspond to nodes in the graph $\mathcal{G}^\star$ (recall that $|V^\star| = K$), and we penalize the difference between each pair of nodes joined by an edge. The matrix $D = [\Lambda; \lambda_1 I] \in \mathbb{R}^{(m+K)\times K}$ consists of two matrices, $\Lambda^{m\times K}$, where $m = |E^\star|$, and the identity matrix $I^{K\times K}$. The rows of Λ have $-\lambda_{(k,k')}$ and $+\lambda_{(k,k')}$ in the appropriate positions, corresponding to an edge in the graph $\mathcal{G}^\star$. This yields the constrained problem

$$\begin{aligned} &\min_{z,d} ||d||_1 + \frac{\rho}{2}||a - z||_2^2 \\ &\text{s.t } d = Dz. \end{aligned} \tag{4.14}$$

To enforce the constraints in this formulation, we add the penalty function term

$$\min_{z,d} ||d||_1 + \frac{\rho}{2}||a - z||_2^2 + \frac{\lambda}{2}||d - Dz||_2^2. \tag{4.15}$$

Finally, we apply the Bregman iteration [31] to get

$$\min_{z,d} ||d||_1 + \frac{\rho}{2}||a - z||_2^2 + \frac{\lambda}{2}||d - Dz - b(t)||_2^2, \tag{4.16}$$

where $b(t+1)$ is given by

$$b(t+1) = b(t) + (Dz(t+1) - d(t+1)). \tag{4.17}$$

We alternatively minimize (4.16) over z and d, that is at the t-th iteration, we first solve

$$z(t+1) = \min_{z} \frac{\rho}{2}||a - z||_2^2 + \frac{\lambda}{2}||d(t) - Dz - b(t)||_2^2 \tag{4.18}$$

to get z, then solve

$$d(t+1) = \min_{d} ||d||_1 + \frac{\lambda}{2}||d - z(t+1) - b(t)||_2^2, \tag{4.19}$$

to get d, and last we update b as in (4.17). After some simple algebraic computations, (4.18) becomes

$$z(t+1) = \min_{z} \frac{1}{2} z^T(\rho I + \lambda D^T D)z - (\rho a^T + \lambda g(t)^T D)z + \frac{\rho a^T a}{2} + \frac{\lambda g(t)^T g(t)}{2}, \tag{4.20}$$

where the superscript T denotes the transpose of a vector and $g(t) = d(t) - b(t)$. This is an unconstrained quadratic problem, where the matrix $P = (\rho I + \lambda D^T D)$ is positive definite and this has a unique solution,

$$z(t+1) = -P^{-1}q, \tag{4.21}$$

with $q = -(\rho a + \lambda D^T g(t))$.

We can explicitly compute the optimal value for d in (4.19) using the element-wise shrinkage operator [12] as

$$d_j(t+1) = \text{shrink}(Dz_j(t+1) + b_j(t), 1/\lambda) \tag{4.22}$$

where

$$\text{shrink}(z, \lambda) = \frac{z}{|z|} \cdot \max(|z| - \lambda, 0)$$

and the subscript of vectors denotes an element from them. We note that, instead of the split Bregman method, we could use again the ADMM algorithm to solve (4.10). In fact, there is an equivalence [22] between the alternating split Bregman and ADMM algorithm and the updates in (4.18), (4.19), and (4.17) are the same for ADMM.

In our implementation, we used the screening algorithm proposed in [19]. The screening algorithm serves the purpose of identifying block diagonal structure in the estimated precision matrices $\Theta^{(k)}$ and thus reducing the high dimension p to moderate size. For instance, suppose that, for a given choice of λ_1 and λ_2, we determine that matrices $\Theta^{(k)}$ are block diagonal after some permutation of features, each with the same $R = 2$ blocks, the rth of which contains p_r features, $\sum_{r=1}^{R=2} p_r = p$. Then, in each iteration of the ADMM, we only compute eigendecompositions of matrices of dimension $p_1 \times p_1$ and $p_2 \times p_2$, leading to reduction in computational complexity from $O(p^3)$ to $O(\max(p_1^3, p_2^3))$. This screening rule is derived from the perspective of convex optimization (Karush-Kuhn-Tucker conditions), and does not affect the statistical accuracy.

4.4 Joint estimation of multiple Gaussian copula graphical models

The proposed method can be applied directly to jointly estimate nonparanormal graphical models. Nonparanormal (NPN) [48] models are a special case of semiparametric Gaussian copulas [53] and are useful in cases where the normality assumption for data is violated. They extend Gaussian graphical models by marginally transforming the variables using smooth monotone functions. The primary goal is to estimate the underlying sample covariance matrix to better recover the underlying undirected graph. Still, the underlying distribution is assumed to be a p-variate Gaussian distribution $\mathcal{N}_p(0, \Sigma)$ by introducing a collection of monotone functions $\hat{f}_j$'s such that $(\hat{f}_1(x_1), \ldots, \hat{f}_p(x_p))^T \sim \mathcal{N}_p(0, \Sigma)$. The Gaussian copula family is much richer than the normal family. However, the conditional independence graph is still encoded by the sparsity pattern of $\Theta = \Sigma^{-1}$ [48]; that is, $\theta_{ij} = 0 \Leftrightarrow x_i \perp x_j | x \backslash \{x_i, x_j\}$.

Let $\hat{S}^{(k)} = (\hat{S}_{ij}^{(k)})_{p\times p}$ be the correlation matrix of transformed data in the k-th dataset. A two-step procedure to jointly estimate multiple nonparanormal graphical models is:

1. transform the observations in $X^{(k)}$, by the corresponding transformations $\hat{f}_j^k$, and
2. replace the sample covariance matrix $S^{(k)}$ with $\hat{S}^{(k)}$ and solve (4.5).

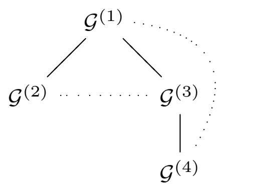
The transformation functions for a sample $x = (x_1, \ldots, x_p) \in X^{(k)}$ are given by

$$\hat{f}_j^k(x_j) = \Phi^{-1}(T_{\delta_{n_k}}[\hat{F}_j^k(x_j)]), \quad j = 1, \ldots, p,$$

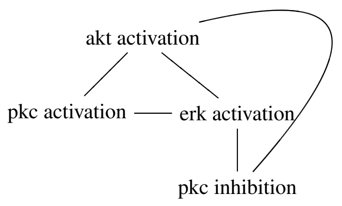
where $\Phi^{-1}(\cdot)$ is the inverse cumulative distribution function (CDF) of a standard Gaussian distribution, T_{δ_n} is a Winsorization (or truncation) operator, and $\hat{F}_j^k(\cdot)$ is the empirical CDF. The operator T_{δ_n} is defined as

$$\begin{aligned} T_{\delta_n}(x) = \delta_n \cdot \mathbb{I}(x < \delta_n) + x \cdot \mathbb{I}(\delta_n \leq x \leq 1 - \delta_n) \\ + (1 - \delta_n) \cdot \mathbb{I}(x > 1 - \delta_n), \end{aligned}$$

where $\mathbb{I}(\cdot)$ is the indicator function, and $\hat{F}_j^k(x) = 1/n_k \sum_{i=1}^{n_k} \mathbb{I}(x_{ij} < x),\ x_{ij} \in X^{(k)}$, is the empirical CDF. In fact, we can use any nonparametric rank-based statistics including Spearman's rho and Kendall's tau [44] to directly estimate the correlation matrices $S^{(k)}$. Then, the estimated correlation matrices, $\hat{S}_\tau^{(k)}$ and $\hat{S}_\rho^{(k)}$ for Kendall's tau and Sperman's rho, respectively, can be directly plugged into (4.5) to obtain the final precision matrix and graph estimates. In a variety of applications with nonparanormal models, the truncation level δ_n is set to $1/(n+1)$ or to $1/(4\pi^{1/4}\sqrt{\pi \log n})$. The latter value controls the trade-off of bias and variance in high dimensions [48].



(a) Simulation set-up. Solid lines correspond to the genealogy of graphs. Dotted lines are extra penalties imposed by our algorithm.



(b) Real world application. We don't impose a similarity penalty between the networks from pkc inhibition and pkc activation.

Figure 4.1: Dependence graphs among the networks in simulated and real datasets. Two adjacent graphs are constrained to be similar.

4.5 Experiments

4.5.1 Artificial datasets

We compare the following estimation methods on simulated and real datasets: i) graphical lasso (GL), ii) intertwined graphical lasso (iGL), iii) fuse graphical lasso (FGL) over a fully connected

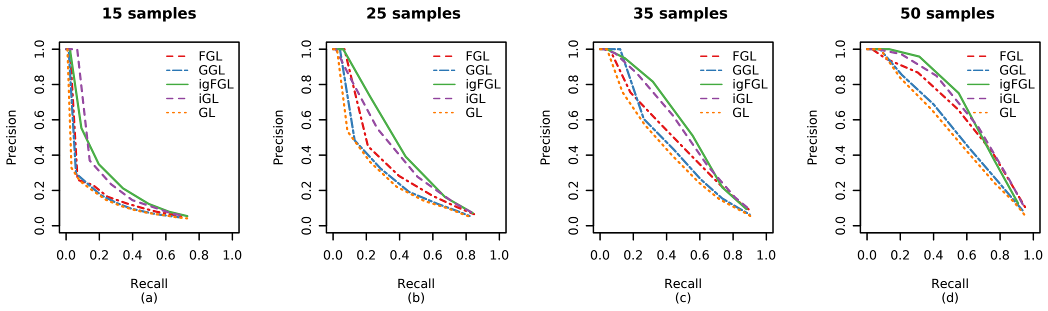


Figure 4.2: ROC curves. Number of perturbed edges between adjacent graphs is $\delta = 17$ (45% of their edges are different).

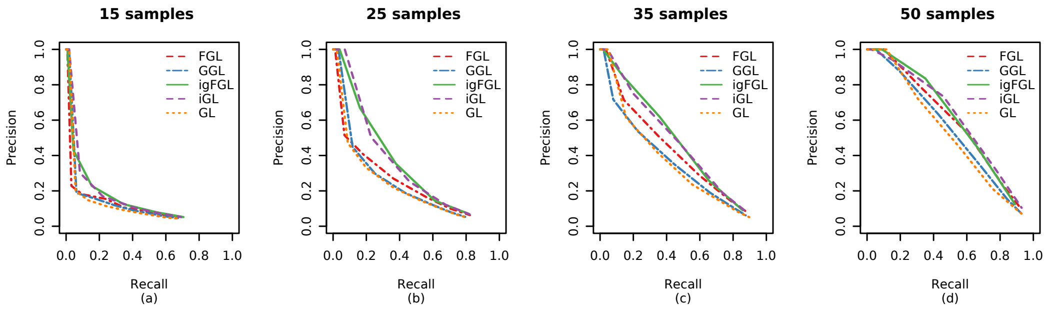


Figure 4.3: ROC curves. Number of perturbed edges between adjacent graphs is $\delta = 25$ (65% of their edges are different).

graph, vi) group graphical lasso (GGL), and v) our method (igFGL); see also Table 4.1. We decided to left out of comparison the method proposed in Guo et al.[33] since authors in [19] compared their algorithm FGL to that of [33] and showed that FGL performs better.

We created $K = 4$ graphs $\mathcal{G}^{(k)} = (V, E^{(k)})$, for $k = 1, \ldots, K$. We set the number of nodes $|V| = 50$, the number of edges $|E^{(k)}| = 37$, and the maximum node degree to be 4. The graphs were generated following the genealogy shown in Figure 4.1(a) (solid black edges). The dotted edges define extra penalties imposed by our method igFGL, that is we fuse every pair of graphs except from $\mathcal{G}^{(2)}$ and $\mathcal{G}^{(4)}$. In the FGL algorithm this pair of graphs is fused. We conducted different experimental scenarios, each time varying the sample size n_k and the number of different edges δ between the adjacent networks. As such, we expect to evaluate the accuracy of the algorithms for small sample size n_k, that is for $n_k \leq p$, a common case in biological studies.

The dataset for the experiment is generated as follows:

1. Generate an Erdós Rényi random graph $\mathcal{G}^{(1)} = (V, E^{(1)})$ for $k = 1$ (root node in the tree of Figure 4.1(a)). Then for $k = 2$ to K randomly add and then remove δ edges from the parent graph $\mathcal{G}^{\pi(k)}$, where $\pi(k)$ is the parent of $\mathcal{G}^{(k)}$. In Figure 4.1(a), we plot the genealogy graph $\mathcal{G}^{\star}$ for the simulated networks with solid black lines.
2. To get the samples from each graph $\mathcal{G}^{(k)}$, we generate the inverse covariance matrix $\Theta^{(k)}$ as

$$\Theta^{(k)} = \begin{cases} 1 & \text{if } i = j \\ 0.3 & \text{if } (i,j) \in E^{(k)} \\ 0 & \text{otherwise,} \end{cases}$$

   where the value 0.3 guarantees positive definiteness of $\Theta^{(k)}$ when max degree is 4.
3. For each k, we sample n_k data points $x_i \in \mathbb{R}^p$, for $i = 1, \ldots, n_k$, from a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma^{(k)})$, with mean $\mu = (0.5, \ldots, 0.5)$ and covariance matrix $\Sigma^{(k)} = (\Theta^{(k)})^{-1}$. The sample mean μ is common for all the Gaussian distributions and only the covariance matrix varies.

We consider two cases, a low perturbation case, where the number of different edges between adjacent graphs is $\delta = 17$, and a high perturbation case, where the number of different edges is $\delta = 25$; these cases correspond to $\{45\%, 65\%\}$ of different edges between the parent graph and the child, respectively. We performed experiments with $n_k \in \{15, 25, 35, 50\}$, that is, we start with datasets having a small number of samples and continue with bigger datasets, until the number of samples in each dataset becomes equal to the number of features $p = 50$.

Let $\hat{\mathcal{G}}^{(k)}_{\lambda_1} = (V, \hat{E}^{(k)}_{\lambda_1})$ be an estimated graph using the regularization parameter λ_1; here λ_1 varies from 1 to 0.1 with step size 0.08. For simplicity, we manually set $\lambda_{(k,k')} = \lambda_2 = 0.05$ for a pair of adjacent graphs $(\mathcal{G}^{(k)}, \mathcal{G}^{(k')})$ with $(k, k') \in \mathcal{G}^{\star}$. The same value for λ_2 was also used for the FGL and GGL algorithm. This value for λ_2 gave the most interpretable results for FGL, GGL, and our method. Note that in most cases, network estimation is performed as a part of exploratory data analysis and the tuning parameters λ_1 and $\lambda_{(k,k')}$ should be guided by practical considerations, such as network interpretability, stability, and the desire for an edge set with low false discovery rate [52, 19]. Since λ_2 is considered fixed, the precision and recall rate for each value of λ_1 are

$$\mathbf{Precision}_{\lambda_1} = \frac{1}{K} \sum_{k=1}^{K} \frac{|\hat{E}^{(k)}_{\lambda_1} \cap E^{(k)}|}{|\hat{E}^{(k)}_{\lambda_1}|}$$

and

$$\mathbf{Recall}_{\lambda_1} = \frac{1}{K}\sum_{k=1}^{K}\frac{|\hat{E}^{(k)}_{\lambda_1} \cap E^{(k)}|}{|E^{(k)}|},$$

respectively. To avoid parameter tuning, term α in (4.3) was set to $1/2$. More refined choices for α are postponed for future research. We independently simulate the above procedure 50 times and adopt precision-recall rates to evaluate the performance of each method in retrieving the true graph edges. Each point in the $(\text{precision}, \text{recall})$ plane is the average of 50 points obtained for each simulation.

Figures 2(a)-(d) and 4.3(a)-(d) display the ROC curves for six prototype conditions. In Figures 2(a)-(d) the number of perturbed edges is $\delta = 17$ while in Figure 4.3(a)-(d) is $\delta = 25$. From subfigure (a) to (d) the sample size n_k for each graph increases from $0.3p$ to p. As we can see, the lower the samples size n_k and the higher the recall rate, the more our method outperforms the others; except from Figure 4.3(d). This figure represents the large-sample high-perturbation condition where all joint estimation methods have comparable performance. In this case, networks differ significantly and there are enough data to estimate each network independently; in this case we have $\log p/n_k = 0$ for $k = 1, \ldots, K$, an indication that fusion is unnecessary. Our method (igFGL), followed by intertwined graphical lasso (iGL), is very robust in the sense that always performs favorably compared to the best baseline method over the whole spectrum of situations.

4.5.2 Inferring protein signaling networks

Only a few real data sets come with a reliable and exhaustive ground-truth allowing quantitative assessments. We make use of a biological dataset measuring protein concentration levels in a T-cell signalling pathway [60]. A cell signalling pathway describes a group of proteins in a cell that interact to control one or more cell functions, such as cell division or cell death. In our case, the T-cell signaling pathway involves 11 proteins and 20 known interactions described extensively in the literature [60]. Figure 4.5(a) describes these interactions in terms of an undirected graphical model; the nodes represent the proteins and the edges the interactions among them. Fourteen experimental conditions (datasets), each one with roughly 700 samples, have been conducted, aiming to reveal different parts of the network. These datasets were created under different experimental conditions by activating or inhibiting the production of a given protein on the network. Here, we use only $K = 4$ random chosen conditions (akt inhibition, pkc inhibition, pkc activation, erk activation) and try to infer the true underlying network of interactions in Figure 4.5(a). We use a small number of conditions to show that our algorithm in the case of "wet" lab data is able to produce accurate results with small sample size datasets. When applying our method (igFGL), the graph $\mathcal{G}^\star$ describing the interactions among the networks, each one of them representing a different experimental condition, is shown in Figure 4.1(b); all conditions are dependent except from two, the pair $(\text{pkc inhibition}, \text{pkc activation})$. Graphs inferred separately, using samples from one condition at a time, show that each dataset really focus on different part of the true network (Figure 4.4). The samples from each dataset were modeled as nonparanormals, since they remained non-normal even after log transformation. To show this, we conduct the normality test at significance level of 0.05 as in Table 4.2. The truncation level in NPN transformation functions was set to $\delta_{n_k} = 1/(4\pi^{1/4}\sqrt{\pi \log n_k})$, where n_k is the number of samples from each dataset $X^{(k)}$.

As the estimated protein signaling network for a particular method, we consider the graph sum of the four graphs estimated by this method. The graph sum of two graphs $\mathcal{A}$ and $\mathcal{B}$ is the graph $\mathcal{C}$ with adjacency matrix given by the sum of adjacency matrices of $\mathcal{A}$ and $\mathcal{B}$.

Table 4.2: Values indicate the number out of 11 proteins rejecting the null hypothesis of normality.

	normality test			
log-transformed dataset	Cramer-von Mises	Lilliefors	Shapiro-Francia	Andreson-Darling
1	11	10	9	11
2	10	11	10	11
3	11	11	10	11
4	11	11	10	11

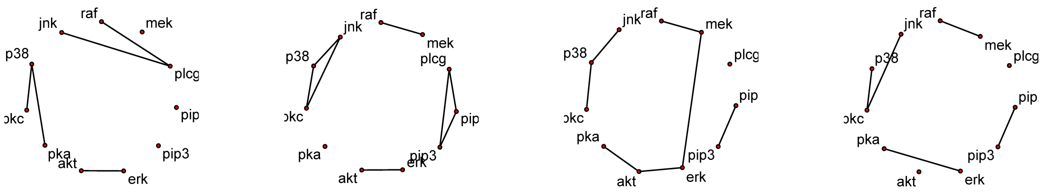


Figure 4.4: Networks inferred separately. From left to right, we have respectively graphs inferred from a dateset : inhibiting akt, inhibiting pkc, activating pkc, activating erk.

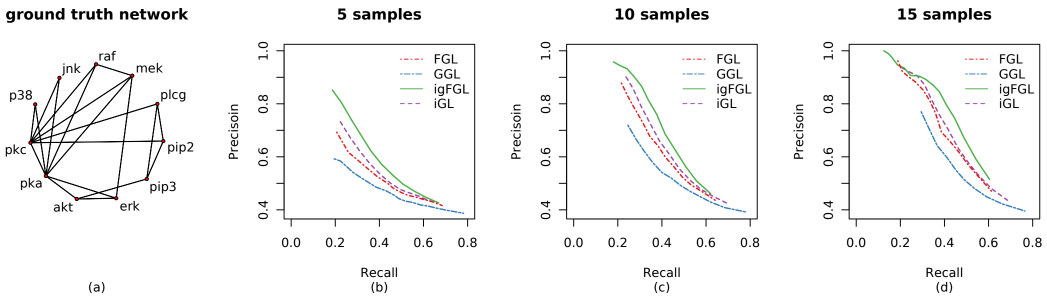


Figure 4.5: Ground truth pathway (a) and precision-recall curves (b)-(d) when $n_k = 5, 10$ and 15.

The various inference algorithms perform almost equally well for large sample size. Figures 4.5(b)-(d) display the results obtained for small sample sizes $n_k = \{5, 10, 15\}$. Here also, the precision recall curves are averaged over 50 independent random draws with $n = 4n_k$ observations over the four datasets. Parameter λ_1 varies from 0.7 to 0.1 with step size 0.035 while λ_2 is fixed to 0.05 for both igFGL, FGL, and GGL. In this situation, our proposal dominates the rest joint estimation methods. It is worth noticing that the large sample size limit is almost obtained for $n_k = 20$, that is, more samples do not benefit our algorithm neither the others.

4.5.3 The 4 Universities data set

This data set contains www-pages collected from computer science departments of various universities in January 1997 by the World Wide Knowledge Base (WebKB) project of the CMU text learning group. The 8.282 pages were manually classified into the following $K = 8$ categories: student (1641 webpages), faculty (1124), staff (137), department (182), course (930), project (504), and other (3764). The original dataset was preprocessed by Cardoso-Cachopo and is availiable at `http://web.ist.utl.pt/~acardoso/datasets/`. The log-entropy weighting function was used to calculate the term document matrix $X^{(k)} = (x_{ij})_{n^{(k)} \times p}$ for each category k, $k = 1, \ldots, 8$, with $n^{(k)}$ and p denoting the number of webpages in each class and distinct words, respectively. We applied the nonparanormal transformation and jointly estimate $\Sigma^{(k)}$ with a fusion penalty to every pair of precision matrices. We choose the largest categories and $p = 100$ terms with the highest log-entropy weights out of total 4800 terms. The networks inferred with our method are shown in Figures 4.6-4.9. The area of the circle representing a node is proportional to its log-entropy weight, while the thickness of an edge is proportional to the magnitude of the associated partial correlation.

The inferred graphs share a common structure but also have major differences. As an example we provide the comparative graph for the categories faculty and student. The comparative is a merged graph where edges in both categories and common edges are distinguished by colors, while nodes are represented as pie charts according to the proportion of edges belonging to either categories. It can be seen that nodes like “number”, “note”, “fall”, and ’colleg’ appear only in the “student” category, while nodes like “assist”, “architect”, “group”, and paper only in the “project” category. Also, common nodes that have high degree in one category happens to have low degree in the other, e.g., the nodes “graduat” and “studi”. Node “graduat” has high degree in “project” with links to nodes like “research”, “work”, and “update”, although, in “student” category it has only one link with the node site. Similarly, node “studi” has high degree in “student” category with links to nodes like “person”, “recent”, and “resourc”, while, it has only one link in category “student” with the node “problem”. Overall, the method gFGL captures the basic common semantic structure of the websites, but also identifies meaningful differences across the various categories. A similar analysis on the same dataset was done by [33] and their inferred graphs exhibit similarities with the graphs the graphs estimated by our algorithm.

4.5.4 Application for discriminant analysis

The proposed joint estimation method has potential applications beyond the framework of sparse GMMs, like discriminant analysis with adaptively pooled covariance matrices [61]. The sparsity of the difference between two precision matrices again shows up if we consider the decision boundaries of discriminant analysis. The general quadratic classifier [29] for the usual two class problem may be expressed as

$$h(x) = \beta_0 + \beta^T + x^T \Gamma x/2, \tag{4.23}$$

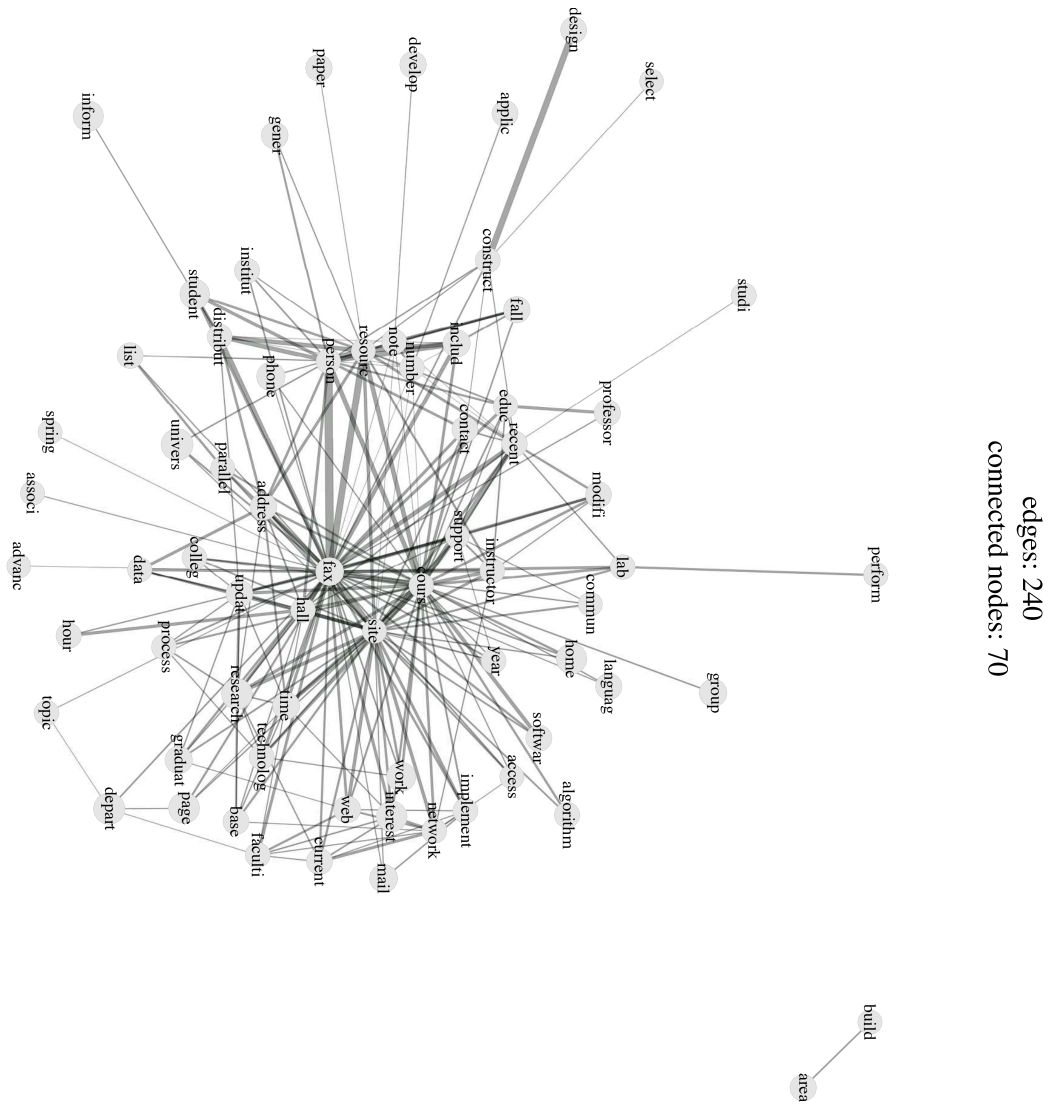


Figure 4.6: “Faculty” graph

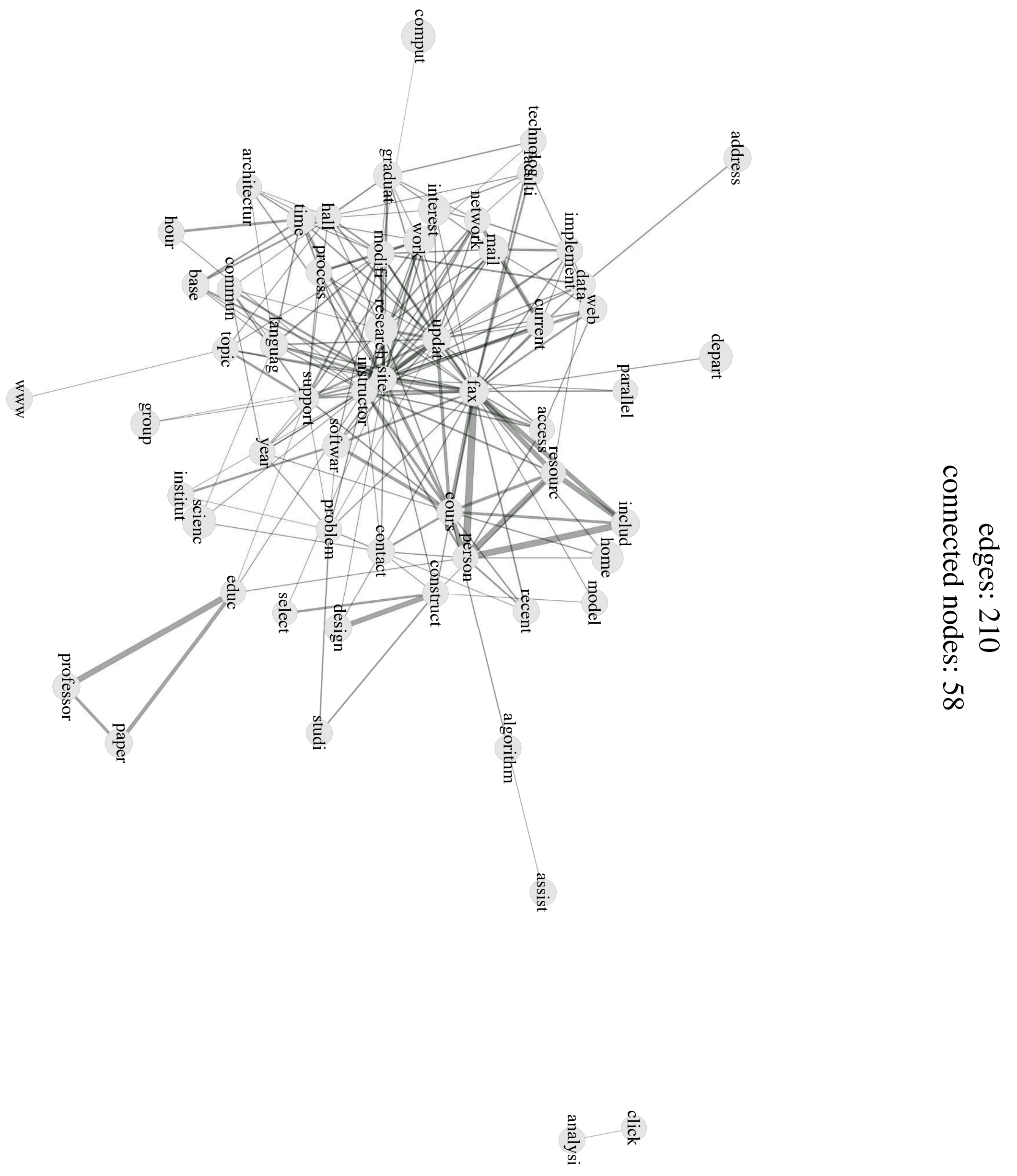


Figure 4.7: “Project” graph

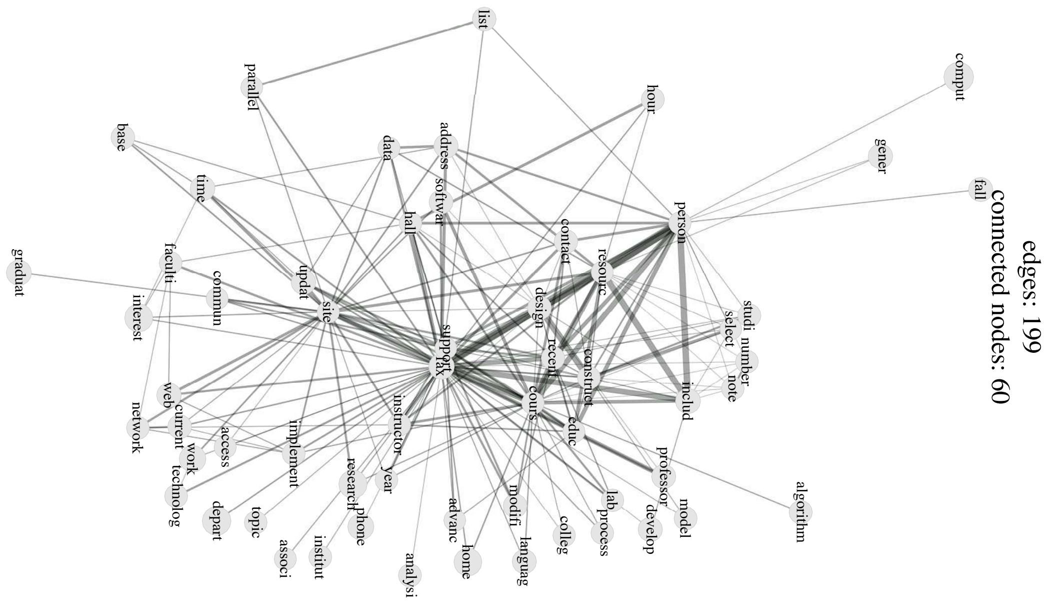


Figure 4.8: "Student" graph

and if $h(x) > 0$ then sample x is classified to class 2, otherwise to class 1. Here, β_0, β^T, and Γ are a scalar, a vector, and a matrix respectively. When we have $n = n_1 + n_2$ total number of samples (with n_1 samples in class 1 and n_2 in class 2) and assume that within each class $k = \{1, 2\}$ the samples are independent normally distributed according to $\mathcal{N}_p(\mu^{(k)}, \Sigma^{(k)})$, then

$$\begin{aligned} \beta_0 &= \log(\pi_1/\pi/2) + (\mu^{(2)})^T(\Sigma^{(2)})^{-1}\mu^{(2)}/2 \\ &\qquad - (\mu^{(1)})^T(\Sigma^{(1)})^{-1}\mu^{(1)}/2 \\ \beta &= (\Sigma^{(1)})^{-1}\mu^{(1)} - (\Sigma^{(2)})^{-1}\mu^{(2)} \\ \Gamma/2 &= (\Sigma^{(2)})^{-1} - (\Sigma^{(1)})^{-1}. \end{aligned} \tag{4.24}$$

In linear discriminant analysis (LDA) we assume that $\Sigma = \Sigma^{(k)}$, $\forall k$, and thus the resulting estimation Γ is identically zero. On the other hand, in quadratic discriminant analysis (QDA) we assume that $\Sigma^{(k)} \neq \Sigma^{(k')}$, $\forall k \neq k'$ and Γ is entirely non-zero. As long as $S_{\text{pool}} = (n_1 S^{(1)} + n_2 S^{(2)})/(n_1 + n_2)$ is full rank, we can use our joint estimation method in (4.5) to get a sparse estimate $\hat{\Gamma}$ and thus to define a classifier intermediate between LDA and QDA; The nonzero terms in $\hat{\Gamma}$ correspond to pairs of dimensions in which the decision boundary is quadratic rather than linear.

4.6 Conclusions

We have proposed a method for estimating sparse precision matrices on the basis of distinct though related datasets. The observations are modeled either as multivariate Gaussian distributions or as semiparametric Gaussian copulas. We use an ADMM algorithm to solve a convex optimization problem whose solution is the jointly estimated precision matrices. Our method has potential applications beyond estimation of sparse precision matrices. For example, in Linear Discriminant

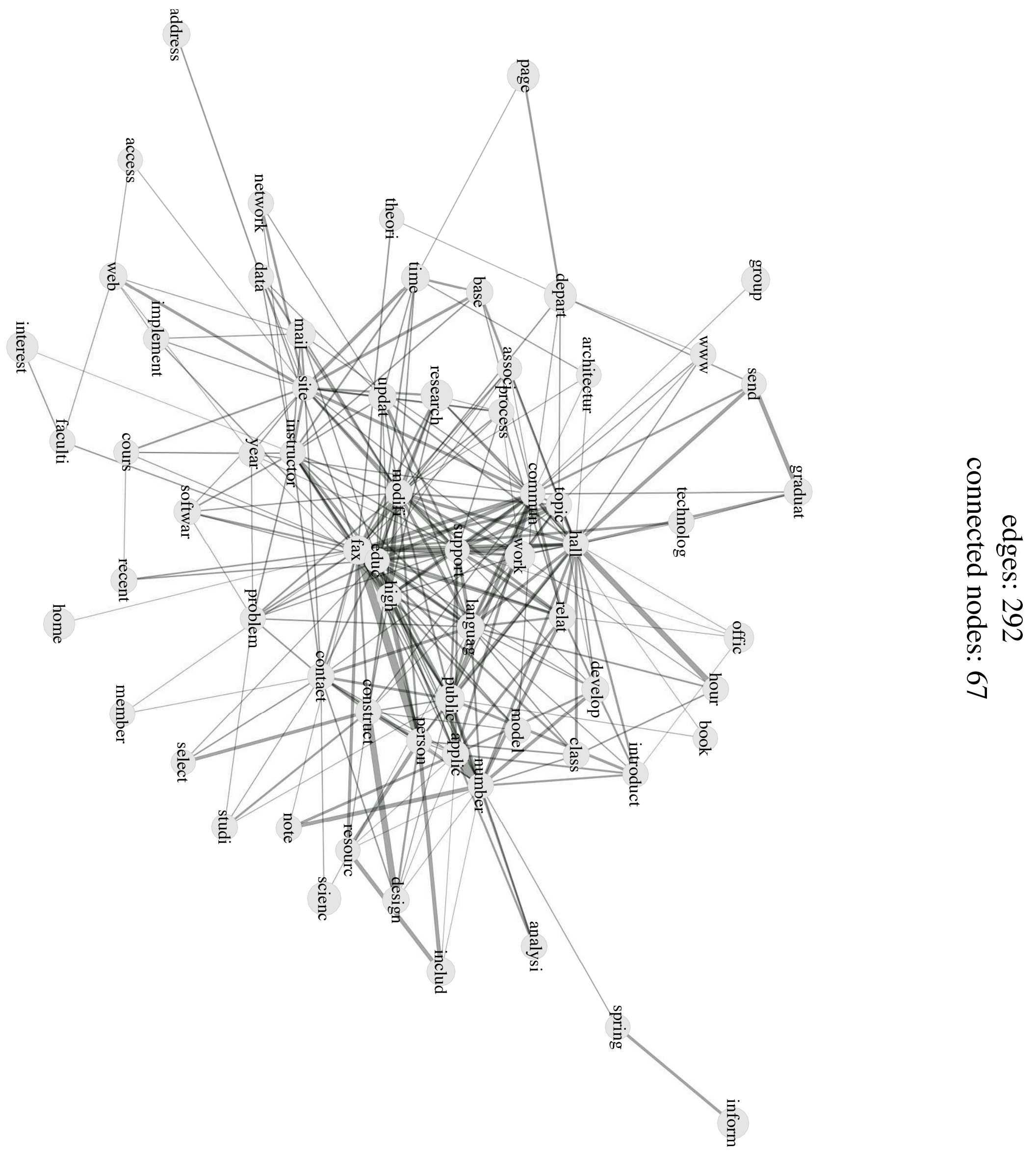


Figure 4.9: “Course” graph

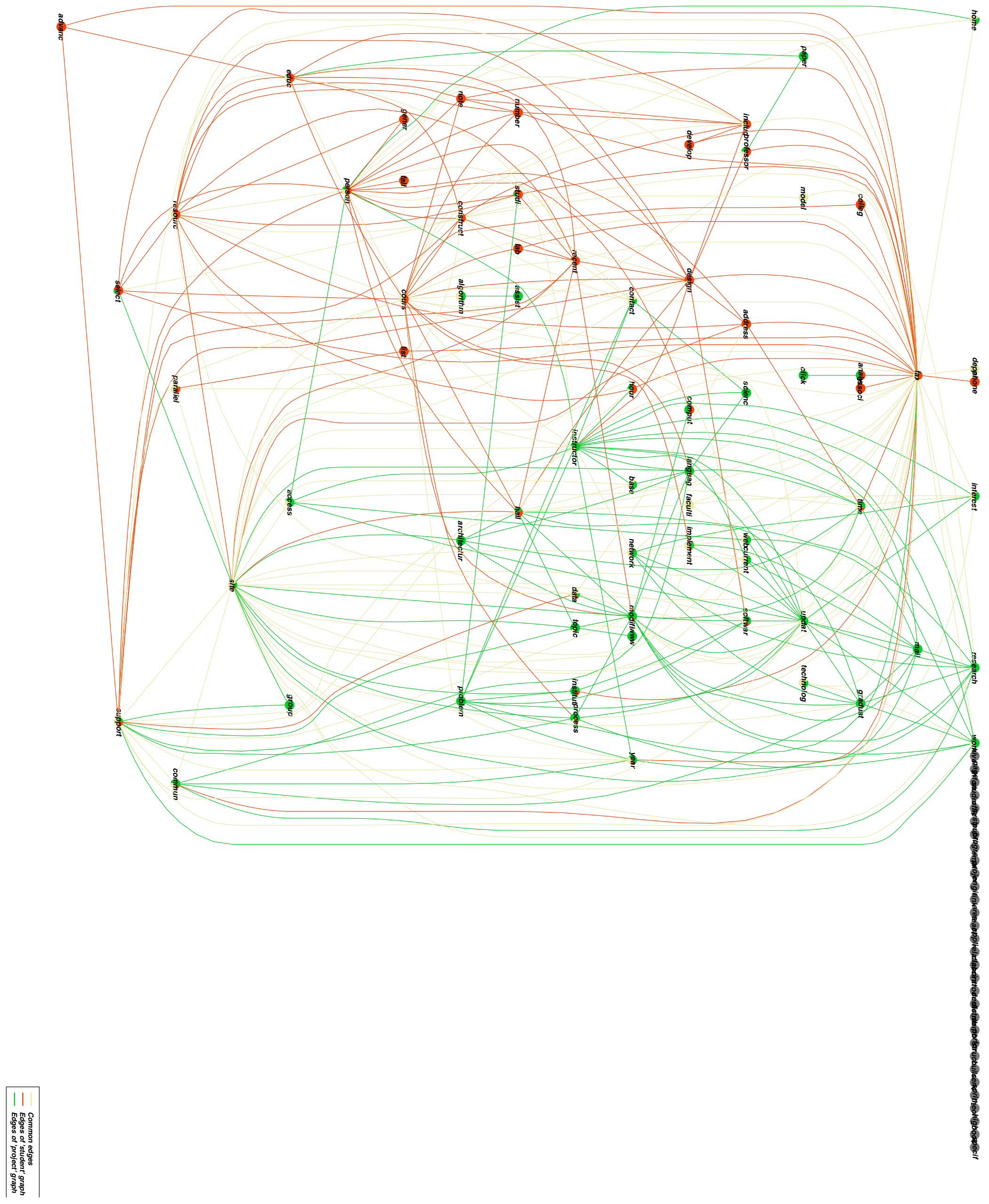


Figure 4.10: Comparative graph for "student" and "project" categories. Common edges are marked light gold, while edges in "students" and "project" are marked red and green, respectively.

analysis (LDA) we estimate a single, pooled covariance matrix, while for Quadratic Discriminant analysis we estimate a separate covariance matrix for each group; one could use our method to define classifiers intermediate between QDA and LDA. Also, it can be used with Gaussian mixture models to jointly estimate the covariance matrices of mixture components. Our experimental results show that our proposal is valuable and robust, consistently performing at least as well as the best of the two baseline solutions. The algorithms developed in this paper are made available in the R programming language from the authors' site. As future work, we plan provide asymptotic sparsistency and consistency of the proposed estimator, here it corresponds to the asymptotic convergence of the set of detected edges towards the set of true edge, and also we plan to experiment with non-convex penalties in place of the convex ones.

Appendix A

A split Bregman algorithm for the generalized signal approximation problem

The core problem underlying Chapters 3 and 4 is the generalized lasso problem

$$\min_{x \in \mathbb{R}^p} \frac{1}{2}||y - Ax||_2^2 + \gamma||Dx||_1, \tag{A.1}$$

where $A \in \mathbb{R}^{n \times p}$, $D \in \mathbb{R}^{m \times p}$ are known matrices, and $y \in \mathbb{R}^n$. This problem penalizes the l_1-norm of a matrix D times the estimated regression vector and has a wide range of applications. When $A = I$, the case of an identity matrix, the problem has a special name: generalized signal approximation. In Chapter 3, where we discussed convex clustering with sparsity inducing norms, the matrix D defined the support for a sample y, that is the number of nearest neighbors taking into account. In Chapter 4, D defines the network of interactions among the datasets under study. In this appendix, we give a brief overview of the applications for the general problem of signal approximation and we present a Bregman iterative algorithm for its solution [31]. Recently, Bregman algorithms have attracted much attention because of their performance for solving this kind of problems. Also, their increased study is a result of their neat connections to classical Lagrangian methods for minimizing the sum of two convex functions under linear constraints.

A.1 Applications of the general signal approximation problem

Taking $A = I$, the generalized lasso (A.1) gives an interesting class of problems

$$\min_{x \in \mathbb{R}^p} \frac{1}{2}||y - x||_2^2 + \gamma||Dx||_1. \tag{A.2}$$

In this setup, we observe data $y \in \mathbb{R}^n$ which is a noisy realization of an underlying signal, and the rows of $D \in \mathbb{R}^{m \times n}$ reflect some believed structure or geometry in the signal. The solution of problem (A.2) fits adaptively to the data while exhibiting some of these structural properties [67]. We begin by looking at piecewise constant signals, and then address more complex applications.

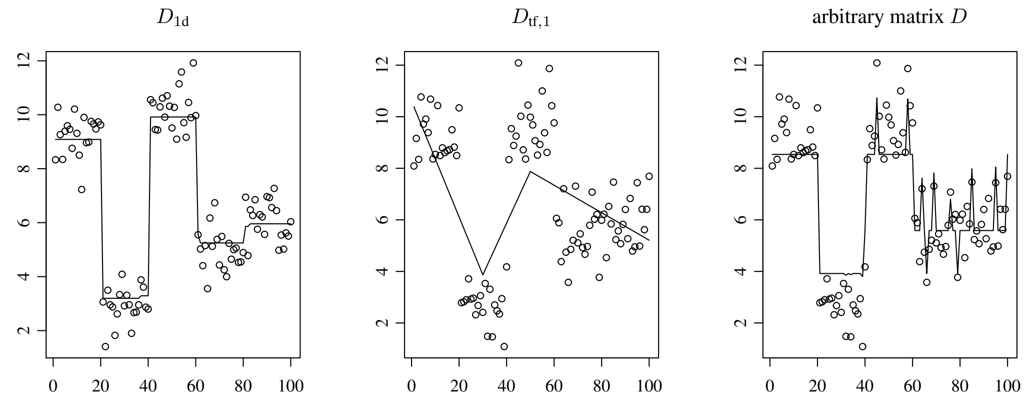


Figure A.1: Solution of (A.1) with $A = I$ and $D = D_{1\text{d}}$ (left panel) and $D_{\text{tf},1}$ (middle panel), and arbitrary D (right panel)

The 1d fused lasso

Suppose that y follows a 1-dimensional structure, that is the coordinates of y correspond to successive positions on a straight line. If D is the $(n-1) \times n$ matrix

$$D_{1\text{d}} = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ & & & \dots & & \\ 0 & 0 & 0 & \dots & -1 & 1 \end{bmatrix}, \tag{A.3}$$

then problem (A.1) penalizes the absolute differences in adjacent coordinates of x, and is known as the 1d fused lasso [66]. This gives a piecewise constant fit, and is used in settings where coordinates in the true model are closely related to their neighbors. Figure A.1 shows for an example of the 1d fused lasso.

Linear and polynomial trend filtering

Suppose again that y follows a 1-dimensional structure, but now D is the $(n-2) \times n$ matrix

$$D_{\text{tf},1} = \begin{bmatrix} -1 & 2 & -1 & \dots & 0 & 0 & 0 \\ 0 & -1 & 2 & \dots & 0 & 0 & 0 \\ & & & \dots & & & \\ 0 & 0 & 0 & \dots & -1 & 2 & -1 \end{bmatrix}. \tag{A.4}$$

Then problem (A.1) is equivalent to linear trend filtering (also called l_1 trend filtering [43]). Just as the 1d fused lasso penalizes the discrete first derivative, this technique penalizes the discrete second derivative, and so it gives a piecewise linear fit. This has many applications, namely, any settings in which the underlying trend is believed to be linear with (unknown) changepoints (Figure A.1).

General fused lasso

We can further extend this idea by defining adjacency according to an arbitrary graph structure, with n nodes and m edges. Now the coordinates of $y \in \mathbb{R}^n$ correspond to correspond to nodes in the graph, and we penalize the difference between each pair of nodes joined by an edge. Hence D is $m \times n$, with each row having a -1 and 1 in the appropriate spots, corresponding to an edge in the graph. In Chapter 3, D defines the neighborhood region for a sample x; for each x there exists a row in D penalizing the distance between the attached centroid to x and the centroids of its neighbors. In the right panel of Figure A.1, we plot the estimated signal when the matrix D penalizes the distance between a sample y and all of the remaining samples.

A.2 Split Bregman method for the l_1-norm minimization

The general form of problems that Split Bregman method solves is

$$\min_{u} \quad H(u) + ||\Phi(u)||_1, \tag{A.5}$$

where both $\Phi(u)$ and $H(u)$ are convex functions. The key to this method is that it ”decouples” the l_1 and l_2 portions of (A.5) and considers the equivalent problem,

$$\begin{aligned} \min_{u,d} \quad & H(u) + ||d||_1 \\ \text{s.t} \quad & d = \Phi(u). \end{aligned} \tag{A.6}$$

To solve the above problem, first we convert it into an unconstrained problem,

$$\min_{u,d} \quad H(u) + ||d||_1 + \frac{\lambda}{2}||d - \Phi(u)||_2^2, \tag{A.7}$$

and then apply the split Bregman iteration,

$$(u(t+1), d(t+1)) = \min_{u,d} \quad ||d||_1 + H(u) + \frac{\lambda}{2}||d - \Phi(u) - b(t)||_2^2 \tag{A.8}$$

$$b(t+1) = b(t) + (\Phi(u(t+1)) - d(t+1)). \tag{A.9}$$

Now, the problem (A.7) is reduced to a sequence of unconstrained optimization problems and updates. In order to minimize (A.8) we iteratively minimize with respect to u and d separately,

$$\begin{aligned} \text{Step 1:} \quad u(t+1) &= \min_{u} \quad H(u) + \frac{\lambda}{2}||d(t) - \Phi(u) - b(t)||_2^2 \\ \text{Step 2:} \quad d(t+1) &= \min_{d} \quad ||d||_1 + \frac{\lambda}{2}||d - \Phi(u(t+1)) - b(t)||_2^2. \end{aligned}$$

To solve Step 1, the optimization problem that we must solve for u is now differentiable and thus we can use a variety of optimization techniques. In Step 2, there is no coupling between elements of d thus we can explicitly compute the optimal value of d using shrinkage operators. The complete generalized Split Bregman method is the following,

Generalized Split Bregman Algorithm

```
while $\frac{||u(t+1)-u(t)||_2}{||u(t)||_2} > tol$
    for $n = 1$ to $N$
        $u(t+1) = \min_u \quad H(u) + \frac{\lambda}{2}||d(t) - \Phi(u) - b(t)||_2^2$
        $d(t+1) = \min_d \quad ||d||_1 + \frac{\lambda}{2}||d - \Phi(u(t+1)) - b(t)||_2^2$
    end
    $b(t+1) = b(t) + (\Phi(u(t+1)) - d(t+1))$
end
```

When $N = 1$, this becomes ADMM (2.20) which can be interpreted as alternately minimizing the augmented Lagrangian with respect to u, then d and then updating the Lagrange multiplier λ [22]. Note that this equivalence between split Bregman and ADMM is not in general true when the constraints are not linear. It is found empirically that for many application where high accuracy is not the goal, optimal efficiency is obtained when only one iteration ($N = 1$) of the inner "for" loop is performed [31]. It may be necessary to perform more iterations ($N > 1$) in applications where high precision is important.

Application of split Bregman method to the general signal approximation case.

To apply Bregman splitting on (A.2), we first replace $\gamma D a^k$ with d. This yields the constrained problem

$$\begin{aligned} \min_x \quad & ||d||_1 + \frac{1}{2}||y - x||_2^2 \\ \text{s.t} \quad & d = \gamma D x. \end{aligned} \tag{A.10}$$

To weakly enforce the constraints in this formulation, we add the penalty function term

$$\min_{x,d} \quad ||d||_1 + \frac{1}{2}||y - x||_2^2 + \frac{\lambda}{2}||d - \gamma D x||_2^2. \tag{A.11}$$

Finally, we strictly enforce the constraints by applying the Bregman iteration (A.8)-(A.9) to get

$$\min_{x,d} \quad ||d||_1 + \frac{1}{2}||y - x||_2^2 + \frac{\lambda}{2}||d - \gamma D x - b(t)||_2^2, \tag{A.12}$$

where the proper values for $b(t)$ are chosen through Bregman iteration.

To solve this minimization problem, we iteratively apply the minimization algorithm, which requires us for the t-th iteration to solve

$$x(t+1) = \min_x \quad \frac{1}{2}||y - x||_2^2 + \frac{\lambda}{2}||d(t) - \gamma D x - b(t)||_2^2 \tag{A.13}$$

and

$$d(t+1) = \min_d \quad ||d||_1 + \frac{\lambda}{2}||d - \gamma D x(t+1) - b(t)||_2^2. \tag{A.14}$$

After some simple algebraic computations (4.18) becomes

$$\begin{aligned} x(t+1) = \min_x \quad & \frac{1}{2}x^T(I + \gamma^2 \lambda D^T D)x - (y^T + \gamma \lambda z(t)^{kT} D)x \\ & + \frac{y^T y}{2} + \frac{\lambda z(t)^{kT} z(t)^k}{2}, \end{aligned} \tag{A.15}$$

where the superscript T denotes the transpose of a vector and $z(t)^{kT} = d(t) - b(t)$. This is an unconstrained quadratic problem, where the matrix $P = (I + \gamma^2 \lambda D^T D)$ is positive definite and thus have a unique solution,

$$x(t+1) = -P^{-1}q, \tag{A.16}$$

with $q = -(y + \lambda\gamma D^T z(t))$.

We can explicitly compute the optimal value for d in (A.14) using the element-wise shrinkage operator as

$$[d]_j(t+1) = \text{shrink}(\gamma[Dx]_j(t+1) + [b]_j(t), 1/\lambda) \tag{A.17}$$

where

$$\text{shrink}(x, \lambda) = \frac{x}{|x|} * \max(|x| - \lambda, 0).$$

Bibliography

[1] Amr Ahmed and Eric P Xing. Recovering time-varying networks of dependencies in social and biological studies. *Proceedings of the National Academy of Sciences*, 106(29):11878–11883, 2009.

[2] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. Np-hardness of euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 2009.

[3] Francis Bach and Guillaume Obozinski. Sparse methods for machine learning theory and algorithms. *ECML/PKDD Tutorial*, 2010.

[4] Kevin Bache and Moshe Lichman. Uci machine learning repository. *Irvine, CA: University of California, School of Information and Computer Science*, 2013.

[5] Onureena Banerjee, Laurent El Ghaoui, and Alexandre 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.

[6] Heinz H. Bauschke and Patrick L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, 2011.

[7] Dimitri P Bertsekas. Nonlinear programming. 1999.

[8] Dimitri P Bertsekas and John N Tsitsiklis. Parallel and distributed computation. 1989.

[9] Jacob Bien and Robert Tibshirani. Hierarchical clustering with prototypes via minimax linkage. *Journal of the American Statistical Association*, 106(495):1075–1084, 2011.

[10] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[11] Stephen Poythress Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[12] Alfred M Bruckstein, David L Donoho, and Michael Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review*, 51(1):34–81, 2009.

[13] Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. Enhancing sparsity by reweighted l_1 minimization. *Journal of Fourier Analysis and Applications*, 14(5-6):877–905, 2008.

[14] Rick Chartrand. Exact reconstruction of sparse signals via nonconvex minimization. *Signal Processing Letters, IEEE*, 14(10):707–710, 2007.

[15] Xiaojun Chen and Weijun Zhou. Convergence of the reweighted l1 minimization algorithm for l2-lp minimization.

[16] Eric C Chi and Kenneth Lange. Splitting methods for convex clustering. *arXiv preprint arXiv:1304.0499*, 2013.

[17] Julien Chiquet, Yves Grandvalet, and Christophe Ambroise. Inferring multiple graphical structures. *Statistics and Computing*, 21(4):537–553, 2011.

[18] Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212. Springer, 2011.

[19] Patrick Danaher, Pei Wang, and Daniela M Witten. The joint graphical lasso for inverse covariance estimation across multiple classes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2013.

[20] Jan de Leeuw and Patrick Mair. Gifi methods for optimal scaling in R: The package homals. *Journal of Statistical Software*, 31(4):1–20, 2009.

[21] Arthur P Dempster. Covariance selection. *Biometrics*, pages 157–175, 1972.

[22] Ernie Esser. Applications of lagrangian-based alternating direction methods and connections to split bregman. *CAM report*, 9:31, 2009.

[23] Massimo Fornasier and Francesco Solombrino. Linearly contrained nonsmooth and nonconvex minimization. Technical report, 2012.

[24] Simon Foucart and Ming-Jun Lai. Sparsest solutions of underdetermined linear systems via lq-minimization for 0¡q¡1. *Applied and Computational Harmonic Analysis*, 26(3):395–407, 2009.

[25] Chris Fraley and Adrian E Raftery. Mclust version 3: an r package for normal mixture modeling and model-based clustering. Technical report, DTIC Document, 2006.

[26] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer Series in Statistics, 2001.

[27] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[28] Jerome H Friedman. Regularized discriminant analysis. *Journal of the American statistical association*, 84(405):165–175, 1989.

[29] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 2 edition, 1990.

[30] Tom Goldstein, Brendan O'Donoghue, and Simon Setzer. Fast alternating direction optimization methods. *CAM report*, pages 12–35, 2012.

[31] Tom Goldstein and Stanley Osher. The split bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.

[32] Michael Grant, Stephen Boyd, and Yinyu Ye. Cvx: Matlab software for disciplined convex programming, 2008.

[33] Jian Guo, Elizaveta Levina, George Michailidis, and Ji Zhu. Joint estimation of multiple graphical models. *Biometrika*, 98(1):1–15, 2011.

[34] Pierre Gutierrez, Guillem Rigaill, and Julien Chiquet. A fast homotopy algorithm for a large class of weighted classification problems.

[35] David J. Hand. Graphical models with r by sren hjsgaard, david edwards, steffen lauritzen. *International Statistical Review*, 81(2):316–316, 2013.

[36] Satoshi Hara and Takashi Washio. Learning a common substructure of multiple graphical gaussian models. *arXiv preprint arXiv:1203.0117*, 2012.

[37] Toby Hocking, Jean-Philippe Vert, Francis Bach, and Armand Joulin. Clusterpath: an algorithm for clustering using convex fusion penalties. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 745–752, New York, NY, USA, June 2011. ACM.

[38] Toby Dylan Hocking. *Learning algorithms and statistical software, with applications to bioinformatics*. PhD thesis, École normale supérieure de Cachan-ENS Cachan, 2012.

[39] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[40] David R Hunter and Kenneth Lange. A tutorial on mm algorithms. *The American Statistician*, 58(1):30–37, 2004.

[41] Laurent Jacob, Guillaume Obozinski, and Jean-Philippe Vert. Group lasso with overlap and graph lasso. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 433–440. ACM, 2009.

[42] Steven G Johnson. The nlopt nonlinear-optimization package, 2010.

[43] Seung-Jean Kim, Kwangmoo Koh, Stephen Boyd, and Dimitry Gorinevsky. \ell_1 trend filtering. *Siam Review*, 51(2):339–360, 2009.

[44] John Lafferty, Han Liu, Larry Wasserman, et al. Sparse nonparametric graphical models. *Statistical Science*, 27(4):519–537, 2012.

[45] Steffen L Lauritzen and Nanny Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, pages 31–57, 1989.

[46] Friedrich Leisch and Evgenia Dimitriadou. mlbench: Machine learning benchmark problems.

[47] Fredrik Lindsten, Henrik Ohlsson, and Lennart Ljung. Just relax and come clustering! a convexification of k-means clustering. Technical report, Tech. Rep. LiTH-ISY, 2011.

[48] Han Liu, John Lafferty, and Larry Wasserman. The nonparanormal: Semiparametric estimation of high dimensional undirected graphs. *The Journal of Machine Learning Research*, 10:2295–2328, 2009.

[49] Yves Lucet. What shape is your conjugate? a survey of computational convex analysis and its applications. *SIAM Journal on Optimization*, 20(1):216–250, 2009.

[50] Markus Maier, Matthias Hein, and Ulrike von Luxburg. Optimal construction of k-nearest-neighbor graphs for identifying noisy clusters. *Theoretical Computer Science*, 410(19):1749–1764, 2009.

[51] Nicolai Meinshausen and Peter Bühlmann. High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, pages 1436–1462, 2006.

[52] Nicolai Meinshausen and Peter Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010.

[53] Roger B Nelsen. An introduction to copulas (lecture notes in statistics). 1998.

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

[55] Peter Ochs, Alexey Dosovitskiy, Thomas Brox, and Thomas Pock. An iterated l1 algorithm for non-smooth non-convex optimization in computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[56] Wei Pan, Xiaotong Shen, and Binghui Liu. Cluster analysis: Unsupervised learning via supervised learning with a non-convex penalty. *Journal of Machine Learning Research*, 14:1865–1889, 2013.

[57] Ankur P Parikh, Wei Wu, Ross E Curtis, and Eric P Xing. Treegl: reverse engineering tree-evolving gene networks underlying developing biological lineages. *Bioinformatics*, 27(13):i196–i204, 2011.

[58] Kristiaan Pelckmans, Joseph De Brabanter, JAK Suykens, and B De Moor. Convex clustering shrinkage. In *PASCAL Workshop on Statistics and Optimization of Clustering Workshop*, 2005.

[59] Pradeep Ravikumar, Martin J Wainwright, Garvesh Raskutti, and Bin Yu. High-dimensional covariance estimation by minimizing 1-penalized log-determinant divergence. *Electronic Journal of Statistics*, 5:935–980, 2011.

[60] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A Lauffenburger, and Garry P Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529, 2005.

[61] Noah Simon and Rob Tibshirani. Discriminant analysis with adaptively pooled covariance. *arXiv:1111.1687*, 2011.

[62] Suvrit Sra, Sebastian Nowozin, and Stephen J Wright. *Optimization for Machine Learning*. Mit Press, 2011.

[63] Jos F Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1-4):625–653, 1999.

[64] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[65] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[66] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

[67] Ryan Joseph Tibshirani. *The solution path of the generalized lasso*. Stanford University, 2011.

[68] Paul Tseng. Applications of a splitting algorithm to decomposition in convex programming and variational inequalities. *SIAM Journal on Control and Optimization*, 29(1):119–138, 1991.

[69] Gaël Varoquaux, Alexandre Gramfort, Jean-Baptiste Poline, Bertrand Thirion, et al. Brain covariance selection: better individual functional connectivity models using population prior.

[70] John Wright, Yi Ma, Julien Mairal, Guillermo Sapiro, Thomas S Huang, and Shuicheng Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031–1044, 2010.

[71] Sen Yang, Zhisong Pan, Xiaotong Shen, Peter Wonka, and Jieping Ye. Fused multiple graphical lasso. *arXiv preprint arXiv:1209.2139*, 2012.

[72] XiaoMing Yuan. Alternating direction methods for sparse covariance selection. Chicago, 2009. 20th International Symposium of Mathematical Programming.

[73] Shuheng Zhou, John Lafferty, and Larry Wasserman. Time varying undirected graphs. *arXiv preprint arXiv:0802.2758*, 2008.