Università degli Studi di Padova
Dipartimento di Scienze Statistiche
Corso di Laurea Triennale in

Statistica Economia e Finanza

RELAZIONE FINALE

MAXIMIN ESTIMATION FOR GAUSSIA GRAPHICAL MODEL

Relatore Prof. Livio Finos
Dipartimento di Psicologia dello Sviluppo e della Socializzazione

Correlatore Prof. Ernst Wit
Faculty of Mathematics and Natural Sciences, RUG University

Laureando: Nome Cognome
Matricola N 1053144

Anno Accademico 2015/2016

I would like to express special thanks to professor Ernst Wit and professor Livio Finos for helping me to complete this project.

Abstract

Large scale data, from a statistically point of view, has often computational problems in the estimation caused by the *high dimension* of the dataset and also statistical problems caused by *inhomogeneities* (outliers, shifting distribution...) of the data.
A lot of "usual" statistical procedures fails when the data are inhomogeneous. In this case there are useful results in the field of robust statistic, but these methods often tend to overparameterize the model, and sometimes can not be used because in the tipical large scale data analisys there are relatively very few observations, compare to the parameters of the model.
The main point is the assumption of *sparsity*, which is really often fulfilled in high dimensional data; in the last two decades have been developed better methods for these cases, called *shrinkage* methods.
The method used in my thesis is based on the lasso (shrinkage method which use a l_1-norm to penalize the parameters), with a different penalty function to increase robustness in the estimation.

Contents

Chapter 1

Maximin effect

In this chapter the definition and the properties of maximin effect for linear model and a possible extension to gaussian graphical model will be exposed.

1.1 Maximin effect for linear model

In this section we refer to Meinshausen and Bühlmann - 2015 [14], here we have a summary from the section 2 with the most useful topics for my thesis.

1.1.1 Definition of maximin effects

To give an intuitive definition, we will focus on a *mixture model*: for $i = 1, ..., n$

$$Y_i = X_i B_i + \epsilon_i \tag{1.1}$$

where Y_i is a real response variable, $X_i \in \mathbb{R}^p$ is a predictor variable, $B_i \in \mathbb{R}^p$ and $B_i \sim F_B$ for an unknown distribition F_B.
The predictors are random, indipendent and identically distributed with population Gram matrix Σ, the noise fulfills $E(\epsilon) = 0$ and $E(\epsilon^T X) = 0$ and the coefficients B_i are indipendent from X_i. The data are inhomogeneous because B_i depend on the observations.

For a fixed regression coefficient $b \in \text{support}(F_B) \subseteq \mathbb{R}^p$ we define the *explained variance of predictions* with $\beta \in \mathbb{R}^p$:

$$V_{\beta,b} = 2\beta^T \Sigma b - \beta^T \Sigma \beta \tag{1.2}$$

or (under the condition $E(\epsilon^T X) = 0$):

$$\begin{aligned} V_{\beta,b} &= E(\|Y\|_2^2/n) - E(\|Y - X\beta\|_2^2/n) \\ &= E(\|Xb + \epsilon\|_2/n) - E(\|X(b - \beta) + \epsilon\|_2/n) \end{aligned}$$

(note that E We want an effects that guarantee good performance in all possible parameters values, especially we want maximize the minimum (respect the all possible values of the parameters) of the explained variance, we define the *maximin effect* in that way:

$$\beta_{maximin} = \arg\min_{\beta} \max_{b \in F} (-V_{\beta,b}) = \arg\max_{\beta} \min_{b \in F} (V_{\beta,b}) \tag{1.3}$$

where $F = \text{support}(F_B)$ or a smallest region of the support with probability $1 - \alpha$.
With this definition we can see that the maximin effects is a parameter that represents the maximization of the objective (explained variance) under the worst possible scenario in the support of F_B.

1.1.2 Charaterization and robustness of maximin effects

Here we can show the most important theoretical proprieties of maximin effects, we begin to show the differences and the advantages respect to the simpliest *pooled effect*, which is defined as:

$$\beta_{pool} = \arg\min_{\beta} E_B\left(-V_{\beta,B}\right)$$

The pooled effect, intuitively, is the value that maximize the average of the explained variance, the main problem of this effect is that, for some values in the support of F_B, the explained variance, respect to these values, can be really low.

Suppose that the coefficient of the model is $B = (1, \eta)$ where $\eta \sim U(a,6), a < 6$ and the Gram matrix is the identity: $\Sigma = I_2$. Applying the definition we have that:

$$\begin{aligned}\beta_{pool} &= \arg\min_{\beta} E_B\left(\beta^T I_2 \beta - 2\beta^T I_2 (1,\eta)^T\right) = \arg\min_{\beta}\left\{\beta_1^2 + \beta_2^2 - 2E_B\left(\beta_1 + \beta_2\eta\right)\right\} \\ &= \arg\min_{\beta}\left\{\beta_1^2 - 2\beta_1 + \beta_2^2 - 2\beta_2 E(\eta)\right\} = (1, a/2+3)^T\end{aligned}$$

instead the maximin:

$$\begin{aligned}\beta_{maximin} &= \arg\min_{\beta}\max_{\eta\in(a,6)}\left(\beta^T I_2\beta - 2\beta^T I_2(1,\eta)^T\right) = \arg\min_{\beta}\left\{\beta_1^2 + \beta_2^2 - 2\min_{\eta}\left(\beta_1 + \beta_2\eta\right)\right\} \\ &= (1,(a)_+) \qquad \text{where} \qquad (a)_+ = \begin{cases} 0 & \text{if } a \le 0 \\ a & \text{if } a > 0 \end{cases}\end{aligned}$$

In Figure 1.1 there is a simulation which compare the maximin effects and the pooled effects. We can see that the values 0 in the maximin effects has an intrinsic meaning, intuitively the effect of β_2 is 0 if β_2 can assume values of different signs. It will show better in the next theorem.

Theorem 1. *Assume that the predictor variables are choosen randomly from a design with full-rank population Gram matrix Σ. Let $F = \text{support}(F_B)$. The maximin-effect is:*

$$\beta_{maximin} = \arg\min_{\gamma\in\text{Conv}(F)} \gamma^T\Sigma\gamma$$

where $\text{Conv}(A)$ *denote the convex hull of the set of points A.*
As a particular case, if $0 \in \text{Conv}(F)$*, then* $\beta_{maximin} = 0$.

The proof is in [14]. The maximin effect parameter is the closest to the origin in the convex hull of the support of F_B.
This is the main point about the robustness of maximin: adding a new point in the dataset the new maximin effect will have lower or equal distance to 0 than the original.
Figure 1.2 show some possible values of maximin effect respect to the support of F_B.

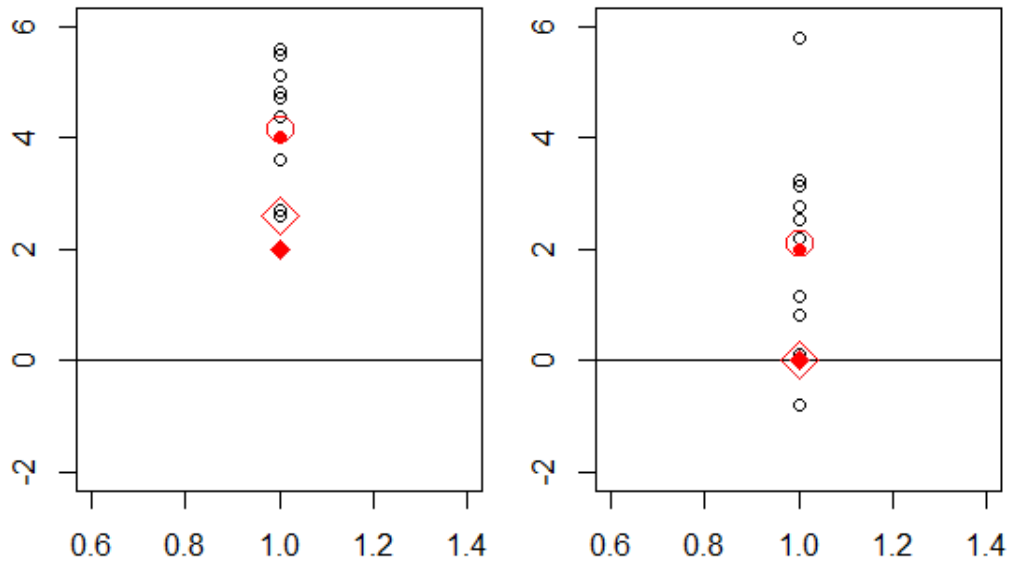


Figure 1.1. Left panel: 10 values from an $U(2,6)$, right panel: 10 values from $U(-2,6)$, the full red square is the maximin effect and the empty is his estimation with the sampled values (the estimator will be discuss in chapter 2, in this case correspond to the sample values closer to 0, if all the values have the same sign, instead is 0), the red full circle is the pooled effect and the empty is his estimation, which is the mean of the values.

1.2 Maximin effect for gaussian graphical model

1.2.1 Basic concept of (gaussian) graphical model

An *undirected graphical model* is an useful way to represent the dipendence between random variables using an undirected graph.
A *graph* is a set of object called *nodes* which each pair of them can be connected with a link, called *edge*, in our case the nodes represents the casual variables in our model, the edges the dependence between them, if the nodes A and B are connected in the graph, it means that there is conditional dipendence between the variables A and B.
A graph is *undirected* if the edges has no direction.

If the only dependence between the p variables in which we are intrested is linear, and the variables are normal, the undirected graphical model can be represented also with a p-dimensional multivariate normal distribution (*gaussian graphical model*).
There are two important results in this model.
The first is that a gaussian graphical model can be seen as a set of regressions in which each variable is regressed on the others simultaneously, formally, we have an i.i.d. sample from the distribution $X \sim N(0, \Sigma)$, we can consider the block:

$$X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12}^T & \Sigma_{22} \end{pmatrix}, \quad \Theta = \Sigma^{-1} = \begin{pmatrix} \Theta_{11} & \Theta_{12} \\ \Theta_{12}^T & \Theta_{22} \end{pmatrix} \tag{1.4}$$

where X_1 is a vector of dimension $p-1$, X_2 is a real value, Σ_{11} and Θ_{11} are a $(p-1) \times (p-1)$ matrix, Σ_{12} and Θ_{12} are $p-1$ vector, Σ_{22} and Θ_{22} are real values.

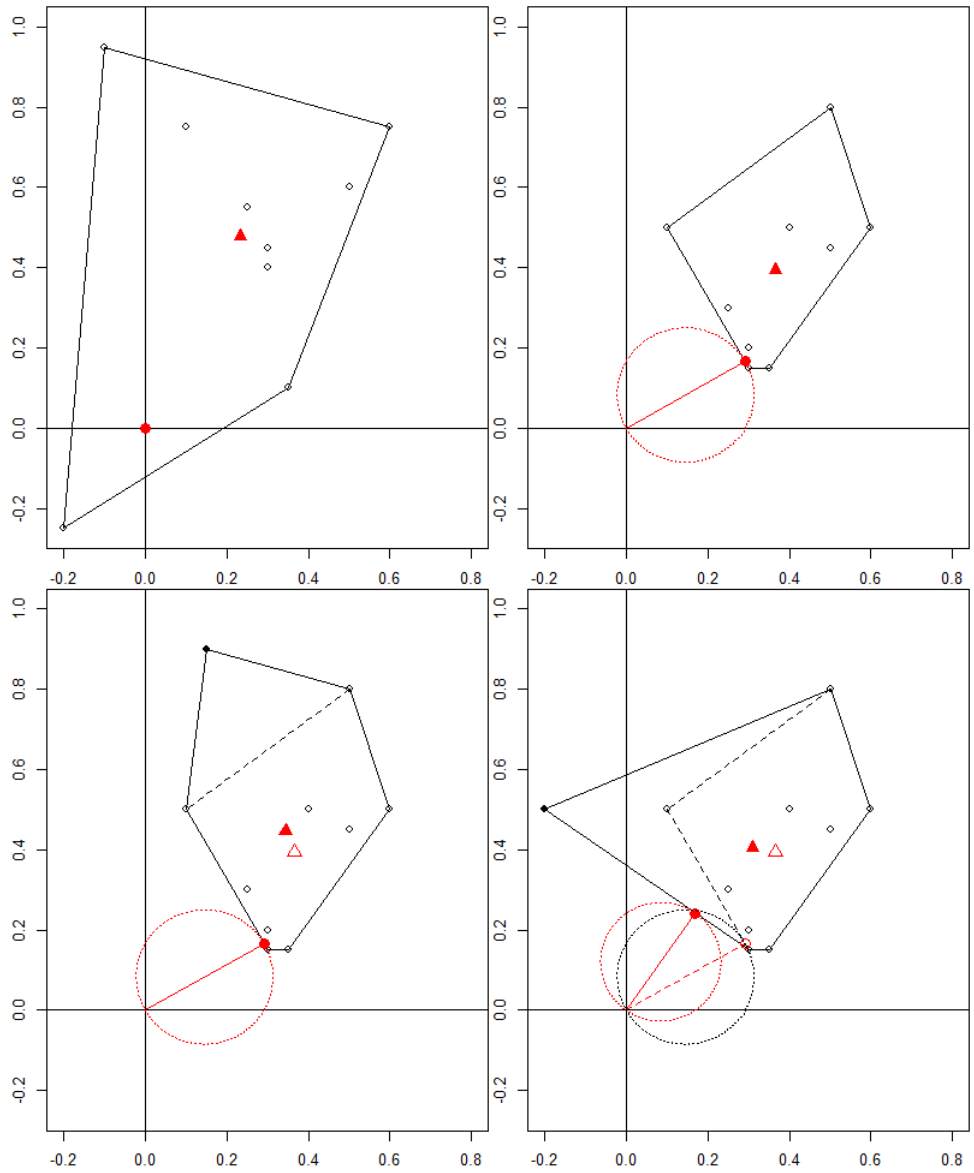


Figure 1.2. here we can see some proprieties of maximin effect using Theorem 1. The first panel shows that if 0 is inside the convex hull of the support of F_B (in these example F_B is discrete, and the black empty points are the values which the function can assume), $\beta_{maximin}$ (red circle) is 0, unlike β_{pool} (red triangle) is different. The seconds shows the result of Theorem 1, the maximin is the point in the convex hull which minimize the distance $d(a,b) = (a-b)^T\Sigma(a-b)$ from 0. The lower panels shows the robustness proprieties of maximin, if we add a new point in the support of F_B, the maximin estimation will change only if there are some new points in the convex hull which are closer to 0 and the maximin can not change to a value with higher distance to 0. Note that whichever point we add to the support, $\beta_{maximin}$ can take a value only in the area inside the red dotted circle.

The conditional density of X_2 given X_1 is univariate normal with parameters:

$$X_2|X_1 \sim N\left(\Sigma_{12}^T\Sigma_{11}^{-1}X_1, \Sigma_{22} - \Sigma_{12}^T\Sigma_{11}\Sigma_{12}\right) \tag{1.5}$$

In this way the linear model associated to each variable of the graphical model is

$$X_2 = X_1\beta + \epsilon, \qquad \begin{array}{l} \text{where } \beta = (\Sigma_{12}^T\Sigma_{11}^{-1})^T = \Sigma_{11}^{-1}\Sigma_{12} \\ \text{and } \epsilon \sim N(0, \Sigma_{22} - \Sigma_{12}^T\Sigma_{11}\Sigma_{12}) \end{array} \tag{1.6}$$

The other important result is the explicit relationship between the coefficients of the associated linear models and the values of the precision matrix.
Let $\beta = (\beta_1, ..., \beta_{p-1})^T$ the coefficient vector in the model (1.6), it holds that

$$\beta_i = -\frac{[\Theta_{12}]_i}{\Theta_{22}}$$

or more generally, if β^j is the coefficient vector of the associated model $X_j = X_{-j}\beta^j + \epsilon$ related to the j-th variable conditional to the others

$$\beta_i^j = -\frac{\Theta_{i,j}}{\Theta_{j,j}} = -\frac{\Theta_{j,i}}{\Theta_{j,j}} \tag{1.7}$$

(Meinshausen and Bühlmann - 2006 [13]).
This result implies that the variable X_i is indipendent from X_j conditionally to the other variables if and only if the element (i, j) in the precision matrix (inverse of covariance matrix) is equal to 0:

$$X_i \perp\!\!\!\perp X_j \mid X_{k_1}, ..., X_{k_{p-2}}, \text{ where } \{k_1, ..., k_{p-2}\} = \{1, ..., p\} \setminus \{i, j\} \iff \Theta_{i,j} = 0 \tag{1.8}$$

1.2.2 Definition of maximin for gaussian graphical model

The maximin effect for a gaussian graphical model can be defined in different ways, we can define it directly in the covariance matrix or consider the "effect" related to the correlations behind the variables.

Proceeding straightforward in the way that has been defined in section 1.1 we can consider the model:

$$X_i \sim N(0, U) \text{ where } U_{i,j} \in \mathbb{R},\ U = U^T \text{ and } U \sim F_U$$

for some distribution F_U, and consider the maximin effect:

$$\widetilde{\Sigma}_{maximin} = \arg\max_{\Sigma} \min_{U} f(U, \Sigma)$$

for some function $f : R \in \mathbb{R}^{p^2} \times \mathbb{R}^{p^2} \to \mathbb{R}$ like, for example, the likelihood.
However a definition like this does not have an intuitive connection with the correlations between the variables, so we prefer define $\Sigma_{maximin}$ in a different way, directly related to the correlations, using the relationship between a gaussian graphical model and a linear model showed in the previous section.

In the model (1.6) we have a maximin effect using the equation (1.3), we define $\Sigma_{maximin}$ as the matrix which respect the p relations:

$$\Sigma_{12;maximin} = \Sigma_{11;maximin}\beta_{maximin}, \tag{1.9}$$

for each block of variables as in (1.4).
Using the usual maximin definition for a linear model:

$$\begin{aligned}\beta_{maximin} &= \arg\min_{\beta} \max_{b\in \text{supp}(F_B)} (-V_{\beta,b}) \\ &= \arg\min_{\beta} \max_{b\in \text{supp}(F_B)} \left(\beta^T \Sigma_{11;maximin}\beta - 2\beta^T \Sigma_{11;maximin} b\right)\end{aligned}$$

now, using the fact that $\beta = \Sigma_{11}^{-1}\Sigma_{12}$, we can reparametrize the space $\text{supp}(F_B)$ whereas $F^* = \Sigma_{11;maximin}^{-1} \cdot \text{supp}(F_B)$, where with the moltiplication we intend that F^* is the set of all values $f^* = \Sigma_{11;maximin}^{-1} b$, for all $b \in \text{supp}(F_B)$. The minimization becomes:

$$\Sigma_{12;maximin} = \Sigma_{11;maximin} \cdot \arg\min_{\beta} \max_{\widetilde{\Sigma}_{12}\in F^*} \left(\beta^T \Sigma_{11;maximin}\beta - 2\beta^T \widetilde{\Sigma}_{12}\right) \tag{1.10}$$

So we haven't an explicit definition of $\Sigma_{maximin}$, but, as we will see in chapter 2, with an iterative method we can estimate this matrix.
Following this method we give an extension to graphical model of the Theorem 1, using $\Sigma_{maximin}$ and $\beta_{maximin}$ as are defined in (1.9) and (1.10), which are the maximin effect of the associated models (1.6), we have the following:

Theorem 2. *Let $X_2 = X_1\beta + \epsilon$ the linear models associated to the variable $X \sim N(0, \Sigma)$. $\Sigma_{maximin}$ is the matrix which fulfills the p relations*

$$\Sigma_{12;maximin} = \Sigma_{11;maximin} \left(\arg\min_{\gamma\in \text{Conv}(F^*)} \gamma^T \Sigma_{11;maximin}\gamma \right)$$

where $F^* = \{f^* : f^* = \Sigma_{11}^{-1} b \ \forall\ b \in \text{supp}(F_B)\}$.

Chapter 2

Maximin estimation

Here we will show the estimation of the maximin effect defined in chapter 1, for linear and graphical model.
The estimation is done with the empirical counterpart of the quantities defined in the previous chapter and adding a l_1-norm penalty (will be explained later) to increase sparsity.
In the first section there are the basic concepts of penalized regression by l_q-norm, especially with $q = 1$ (called lasso), which is the foundation of these kinds of estimation.

2.1 Regression shrinkage by penalized likelihood

Here we show a method to estimate a linear model which has better proprieties respect to the "usual" OLS estimator under some assumptions of sparsity in the vector of coefficients. This section is a summary from the paper Tibshirani - 1996 [17]
The sparsity assumption is useful in high dimensional setting because is often fulfills and offers a better interpretation of the parameters, this method tends to shrink to 0 the values of some coefficients.
There are two main reason to use shrinkage methods instead of the OLS estimator, the first is prediction accuracy: estimation by ordinary least squares tend to have low bias but large variance, by penalized likelihood with some additional bias we can reduce the variance and have a better prediciton accuracy. The second reason is the interpretability of the coefficients: in high dimensional setting is useful find a subset of the coefficients which contains the strongest effects, and overlook to the (many) others.

2.1.1 Adding a l_q-norm penalty

Let the usual linear model $Y = X\beta + \epsilon$, where X is a $n \times p$ matrix of fixed values, β is a p-dimensional vector and $\epsilon \sim N(0, \sigma^2 I_n)$.
The *OLS estimator* minimize the residual variance, i.e. the quantities

$$\sum_{i=1}^{n} \left(y - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 = \|Y - X\beta\|_2^2$$

To increase sparsity we adding a penalty which imposes a bound to the l_q-norm of the coefficients vector, this estimator minimize

$$\|Y - X\beta\|_2^2 \quad \text{subject to} \quad \|\beta\|_q \leq t, \quad \text{for} \quad t > 0 \tag{2.1}$$

or equivalently, by Lagrange duality, the function

$$\|Y - X\beta\|_2^2 + \rho\|\beta\|_q, \quad \text{for} \quad \rho > 0 \tag{2.2}$$

This estimator is called lasso for the value $q = 1$ and ridge for $q = 2$.
The *lasso estimator* for a linear model is:

$$\begin{aligned} \hat{\beta}_{lasso} &= \arg\min_{\beta} \left\{ \|Y - X\beta\|_2^2 + \rho\|\beta\|_1 \right\} \\ &= \arg\min_{\beta} \left\{ \|Y - X\beta\|_2^2 \right\} \quad \text{s.t.} \ \sum_i |\beta_i| \leq t \end{aligned} \tag{2.3}$$

where there is a bijective function from ρ to t.
Then the *ridge estimator*:

$$\begin{aligned} \hat{\beta}_{ridge} &= \arg\min_{\beta} \left\{ \|Y - X\beta\|_2^2 + \tilde{\rho}\|\beta\|_2 \right\} \\ &= \arg\min_{\beta} \left\{ \|Y - X\beta\|_2^2 \right\} \quad \text{s.t} \ \left(\sum_i |\beta_i|^2 \right)^{\frac{1}{2}} \leq \tilde{t} \end{aligned}$$

or more simply, by reparametrization of the penalty $\rho = \tilde{\rho} \cdot \|\beta\|_2 / \|\beta\|_2^2$ and $t = \tilde{t}^2$

$$\begin{aligned} \hat{\beta}_{ridge} &= \arg\min_{\beta} \left\{ \|Y - X\beta\|_2^2 + \rho\|\beta\|_2^2 \right\} \\ &= \arg\min_{\beta} \left\{ \|Y - X\beta\|_2^2 \right\} \quad \text{s.t} \ \sum_i |\beta_i|^2 \leq t \end{aligned} \tag{2.4}$$

In figure 2.1 we can see that the values $q = 1$ increase sparsity respect $q = 2$ because some values of β can be shrinked exactly to 0. Precisely, more q is smaller, more sparsity increase, but for $0 \leq q < 1$ the penalty set is not convex and this lead to a problem in the optimization. So the lasso penalty is the one which most increase sparsity in a convex set (figure 2.2).

Using a Bayesian approach we have an intresting view of the lasso estimation: we see in equation (2.1) with $q = 1$, that in the lasso estimation is added the penalty term $\rho \sum |\beta_i|$ to the residual variance, each $|\beta_j|$ is proportional to the minus log-density of the Laplace distribution

$$f(\beta_j) = \frac{1}{2b} \exp\left(-\frac{|\beta_j - a|}{b} \right) \tag{2.5}$$

when the location parameter $a = 0$ [17].
The lasso estimate is the Bayes posterior mode when the prior for the coefficients β_j are indipendent Laplace distributions, with location parameter equal to 0 and scale parameter $b = 1/\rho$.

2.1.2 Lasso and likelihood

The results in the previous section can be easilly extended to generalized linear model. This is useful to us because for GLM (so also for the linear model) the lasso estimate is equivalent to the

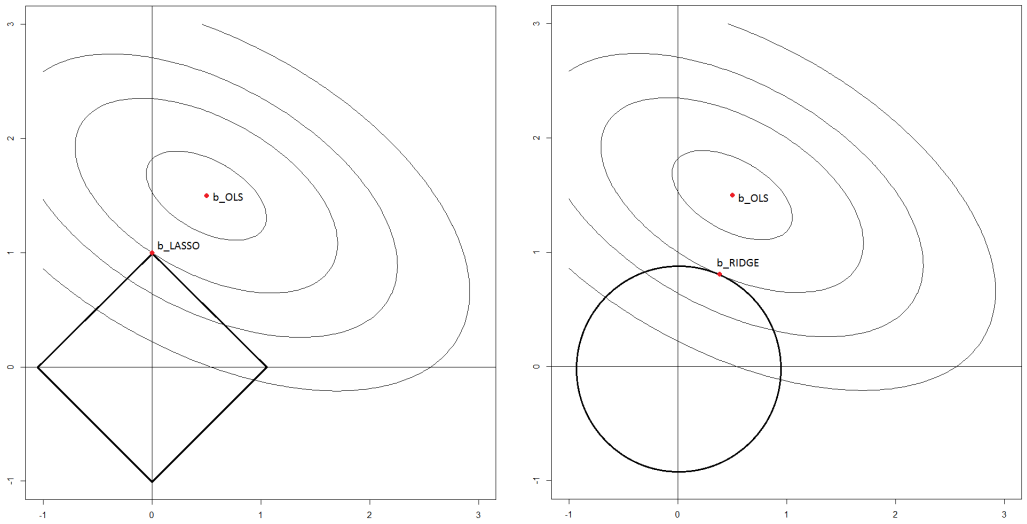


Figure 2.1. In this plot we can see how lasso and ridge work and their difference. Respect to the OLS estimator these methods shrink to 0 the parameters. The penalty forces the coefficients vector to stay inside the square (for lasso) or the circle (for the ridge), the estimation is the point inside the boundary where the value of the function is higher. The difference between lasso and ridge is how the parameters has been shrinked to 0, in this case the estimated value for the x-axis is exactly 0 for the lasso, this is a propriety of l_q-norm with $q \leq 1$ and show how lasso penalty increase sparsity.

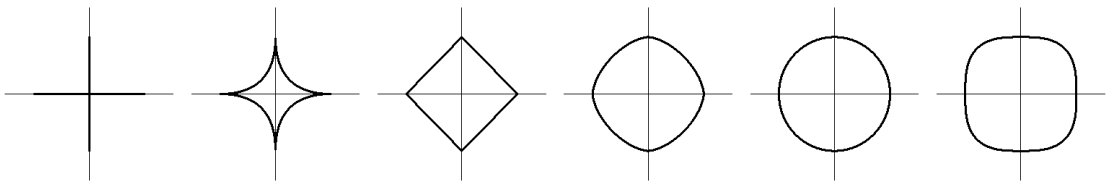


Figure 2.2. contour of equivalues of l_q-norm for, from left to right, $q = 0,\ 0.5,\ 1,\ 1.6,\ 2,\ 3$. The l_q-norm is defined as $\|x\|_q = (\sum_i |x_i|^q)^{1/q}$ when q is greather than 0, the values for $q = 0$ is the limit of $\|x\|_q$ for $q \to 0$. The set in which $\|x\|_q$ is less than some constant k is convex for $q \geq 1$, the contour is not derivable when $q \leq 1$ in the points in which the vector has one element equal to k in absolute values (and of course the other elements are 0).

maximum l_q-penalized likelihood estimation. This method is exposed in Park, Hastie - 2007 [16]. Lasso for generalized linear model is defined as:

$$\hat{\beta}_{lasso,GLM} = \arg\min_{\beta} \{-\log L(Y, \beta) + \rho \|\beta\|_1\} \tag{2.6}$$

where L is the likelihood function.
For a linear model we have that $-\log L(Y, \beta)$ is proportional to $\|Y - X\beta\|_2$, so $\hat{\beta}_{lasso,GLM}$ is equal to $\hat{\beta}_{lasso}$ in equation (2.3).

2.1.3 Lasso and soft-thresolding

Here we show that with a single predictor, lasso estimate is a function of the OLS estimate. This is useful because the lasso optimization can be solved efficiently with coordinate descent algoritm (more detail in chapter 3 and in the paper Friedman Hastie Hoeffling Tibshirani, 2007 [8]).
The Lagrange form of equation (2.3), with a single predictor and standardized X, is:

$$\hat{\beta}_{lasso} = \arg\min_{\beta} \left\{ \beta^2 - 2\beta \cdot \frac{1}{n} \sum_i X_i Y_i + \rho|\beta| \right\} = \arg\min_{\beta} \left\{ \beta^2 - 2\beta\hat{\beta}_{OLS} + \rho|\beta| \right\}$$

where n is the number of observations.
If $\beta > 0$ the derivative of the objective function

$$2\beta - \beta_{OLS} + \rho = 0$$

lead to the solution

$$\hat{\beta}_{lasso} = (\hat{\beta}_{OLS} - \rho)_+ \qquad \text{where } (a)_+ = \begin{cases} a & \text{when } a \geq 0 \\ 0 & \text{when } a < 0 \end{cases}$$

The same steps can be done if $\beta < 0$, the general solution is:

$$\begin{aligned} \hat{\beta}_{lasso} &= \begin{cases} \hat{\beta}_{OLS} - \rho & \text{if } \hat{\beta}_{OLS} > 0 \text{ and } \rho < |\hat{\beta}_{OLS}| \\ \hat{\beta}_{OLS} + \rho & \text{if } \hat{\beta}_{OLS} < 0 \text{ and } \rho < |\hat{\beta}_{OLS}| \\ 0 & \text{if } \rho \geq |\hat{\beta}_{OLS}| \end{cases} \\ &= S(\hat{\beta}_{OLS}, \rho) = \text{sign}(\hat{\beta}_{OLS})(|\hat{\beta}_{OLS}| - \rho)_+ \end{aligned} \tag{2.7}$$

This is useful when the p predictors are uncorrelated because we can split the problem in p one-dimensional minimizations, but also when the predictors are correlated because we can write the objective function in (2.3) as

$$f(\tilde{\beta}) = \sum_{i=1}^{n} \left(y_i - \sum_{k \neq j} x_{ik}\tilde{\beta}_k - x_{ij}\beta_j \right)^2 + \rho \sum_{k \neq j} |\tilde{\beta}_j| + \rho|\beta_j| \tag{2.8}$$

which is a function of the j-th parameter. If the values of β_k for $k \neq j$ are fixed the minimization for the j-th variable has an explicit solution:

$$\tilde{\beta}_j \leftarrow S\left(\sum_{i=1}^{n} x_{ij}(y_i - \tilde{y}_i^{(j)}), \rho \right)$$

where $\tilde{y}_i^j = \sum_{k \neq j} x_{ik}\tilde{\beta}_k$ or equivalently

$$\tilde{\beta}_j \leftarrow S\left(\tilde{\beta}_j + \sum_{i=1}^{n} x_{ij}(y_i - \tilde{y}_i), \rho \right) \tag{2.9}$$

[8]. This is how the lasso minimization can be implemented by coordinate descent, starting with some values and update one by one the elements until convergence.

2.2 Maximin estimator for linear model

In this section we want give an estimation of the quantities (1.3), for doing this, we make an assumption in the distribution of the coefficients F_B. We assume that the observations can be divided in different (unknown) groups with the same regression coefficient, which can varies between groups.
Formally, suppose there are G groups $g = \{1, ..., G\}$ of n_g observations, $\mathcal{I}_g \subset \{1, ..., n\}$ is the set of observations in group g, denote $X_g = X_{\mathcal{I}_g}$ the $n_g \times p$ submatrix of X, which rows are the observations in g, same for $Y_g = Y_{\mathcal{I}_g}$ and $\epsilon_g = \epsilon_{\mathcal{I}_g}$.
The random coefficient β_g is fixed in a group, the model is:

$$Y_g = X_g\beta_g + \epsilon_g, \qquad g = 1, ..., G \tag{2.10}$$

and the empirical counterpart of (1.2) (empirical explained variance) for the g-th group is:

$$\widehat{V}_\beta^g = \frac{1}{n_g}\left(2\beta^T X_g^T Y_g - \beta^T X_g^T X_g \beta\right) \tag{2.11}$$

So, the natural *maximin estimator* is:

$$\hat{\beta}_{maximin} = \arg\min_{\beta\in\mathbb{R}^p} \max_{g=1,...,G} \left(-\widehat{V}_\beta^g\right)$$

or his l_1-penalized version, to increase sparsity:

$$\hat{\beta}_{maximin} = \arg\min_{\beta\in\mathbb{R}^p} \max_{g=1,...,G} \left(-\widehat{V}_\beta^g\right) + \rho\|\beta\|_1 \tag{2.12}$$

Comparing this estimator with the theoretical quantities defined in (1.2) and (1.3)

$$\beta_{maximin} = \arg\min_{\beta} \max_{b\in \text{ supp}(F_B)} (-V_{\beta,b}) \text{ where } V_{\beta,b} = 2\beta^T\Sigma b - \beta^T\Sigma\beta$$

note that we can write the quantities 2.11 as:

$$\widehat{V}_\beta^g = 2\beta^T\widehat{\Sigma}_g b_g - \beta^T\widehat{\Sigma}_g\beta, \quad \text{where } b_g \text{ solve } Y_g = X_g b_g$$

b and, of course, the support of F_B are unknown, so in (2.11), using the models (2.10) for each group, through Y_g we obtain an estimation of b for each group, i.e. an estimation of the support of F_B.
The estimator is consistent, and objective function is convex (as is show in figure 2.3) but not derivable.
The proprieties of this estimatior is given in [14] for different setting of the model.

2.2.1 L_0-norm approximation

Can be useful to give an approximation to the maximin estimator, this is motivated because we can solve the optimization iteratively with the weighted lasso (which is computationally faster) and, if the approximation is good, we can find the function which is maximized to obtain the maximin estimator. This result works only when $\widehat{V}_\beta^g > 0$ for all g.

The *weighted lasso* is defined as

$$\hat{\beta}_{Wlasso} = \arg\min_{\beta} \left\{\|P^{1/2}(Y - X\beta)\|_2^2 + \rho\|\beta\|_1\right\} \tag{2.13}$$

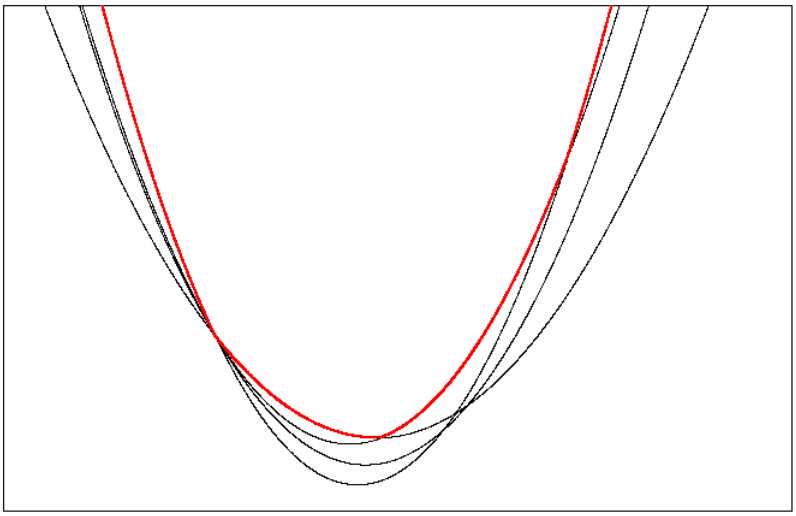


Figure 2.3. convessity of maximin objective function: in black we can see the curve of the quantities $-\widehat{V}_\beta^g$ for 4 different groups, in red we have $\max_g(-\widehat{V}_\beta^g)$

where P is the $n \times n$ matrix of weights, is diagonal with elements greather or equal than 0 and $\sum_{i,j} P_{i,j} = \text{tr}(P) = 1$, we can relax the last assumption using the matrix αP, with $\alpha > 0$ (is relaxed the assumption of the sum of weights, the results is the same if the weights does not sum to one, because of the function arg min), of course the penalty which guarantee the equivalence between the two estimators (with $\text{tr}(P) = 1$ and with $\text{tr}(P) \neq 1$) will change using the different matrix.
Solving inside the norm in the previous equation we have

$$\begin{aligned} \hat{\beta}_{Wlasso} &= \arg\min_\beta \left\{ \|P^{1/2}Y - P^{1/2}X\beta)\|_2^2 + \rho\|\beta\|_1 \right\} \\ &= \arg\min_\beta \left\{ \sum_{i=1}^{n} -P_{i,i}\text{log}L(Y_i, \beta) + \rho\|\beta\|_1 \right\} \end{aligned} \tag{2.14}$$

which is the normal lasso applied to the "weighted" variables (same as before, the penalty will change).

The method for the approximation has an approach similar to the MM-algorithm. This obtain a maximum of the function $f(x)$ using iteratively function $g_k(x)$ such that $g_k(x) < f(x)$ for all x. Let x^* the maximum of f, we start with the values x_0 such that $f(x_0) = g_1(x_0)$, the maximum of g_1 is x_1, so we take g_2 such that $f(x_1) = g_2(x_1)$ and maximize g_2, continuing on the iterations, we will reached x^* in the iteration in which $x^* = \max_x g_k(x)$ for some k (can be also infinity). The algorithm is show in figure 2.4

We will use a different approach, which does not assures the convergence, we need an approximation of $\max_g \widehat{V}_\beta^g$. We will show the approximation of $\hat{\beta}_{maximin}$ in (2.12) in the case of $\rho = 0$. Let write (2.12) as

$$\hat{\beta}_{maximin} = \arg\min_\beta \max_{g=1,\dots G} \left(-\widehat{V}_\beta^g\right) = \arg\min_\beta -\min_g \left(\widehat{V}_\beta^g\right) = \arg\max_\beta \min_g \left(\widehat{V}_\beta^g\right)$$

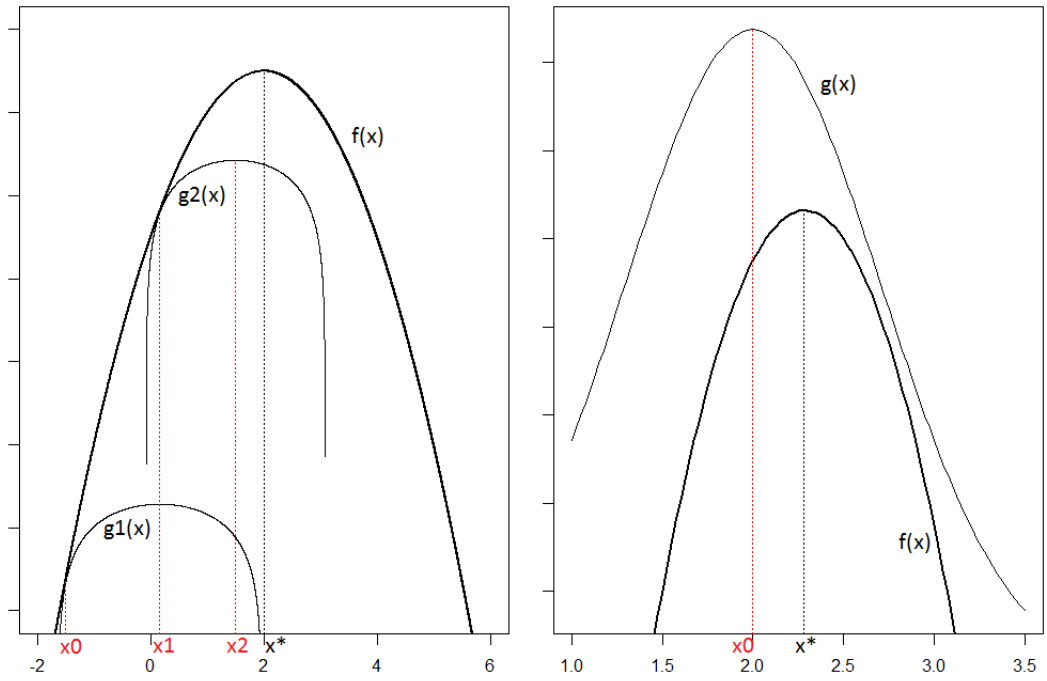


Figure 2.4. in the left we can see the MM algorithm for the maximization of the function f, the function g_1 is equal to f in the point x_0, maximizing g_1 we find x_1 and the function g_2, the last one lead to the values x_2, with more iteration we can find the objective x^*. In the right we show how our approximation works, we want maximize f and find the point x^*, instead we maximize g which dominate f and use the point x_0 as an approximation of x^*, there are no iteration here, with this method we can not find exactly x^* (excluding the trivial case in which $\widehat{V}_\beta^1 = ... = \widehat{V}_\beta^G$)

now, if $x_g > 0$ for all $g \in \{1, ..., G\}$

$$\lim_{\zeta \to 0} \frac{\left(\sum_{g=1}^G x_g^\zeta\right)^{\frac{1}{\zeta}}}{G^{\frac{1}{\zeta}} \cdot \prod_{g=1}^G x_i^{\frac{1}{G}}} = 1 \tag{2.15}$$

the limit can be easily verified by simulation, we will give an intuitive motivation at the end of this section.
Now, $\min_g(x_g) \leq \prod_{g=1}^G (x_g)^{1/G} = \lim_{\zeta \to 0} (\sum_g x_g^\zeta)^{1/\zeta} / G^{1/\zeta}$.
If we fix a small ζ we can write that

$$\hat{\beta}_{maximin} \sim \arg\max_\beta G^{-\frac{1}{\zeta}} \left(\sum_g (\widehat{V}_\beta^g)^\zeta\right)^{\frac{1}{\zeta}} = \arg\max_\beta \sum_g (\widehat{V}_\beta^g)^\zeta = \arg\max_\beta \sum_g (\widehat{V}_\beta^g)^{\zeta-1} \cdot \widehat{V}_\beta^g$$

because $G^{-1/\zeta}$ is a positive constant and the function $x \to x^\zeta$ is a strictly increasing function.
The approximate optimization can be done with the estimator

$$\hat{\beta}_{l_0 maximin} = \arg\max_\beta \sum_g (\widehat{V}_\beta^g)^{\zeta-1} \cdot \widehat{V}_\beta^g - \lambda \|\beta\|_1 \tag{2.16}$$

which is the weighted lasso with weights proportional to the explained variance of the group in which the observation belong to, elevated to the power $\zeta - 1$.

First of all, is possible to derive the connection between ρ in (2.12) and λ in (2.16), but the function between this values depend on ρ, and also on β, G, and ζ, in the optimization we will fix a values of the penalty like here, in the last point.

An intuitive explanation of this method is in figure 2.4.
The main problem is that we do not know how good is the optimization. Excluding the choice of the penalty to replacing the objective with the limit we approximate $\min_G \widehat{V}_\beta^g$ with $\prod_g (\widehat{V}_\beta^g)^{1/G}$, which is the same if $\widehat{V}_\beta^g$ take the same values in each group, but usually do not.
In the next chapter will be exposed the computational problems of this approximation related to the graphical model.

The intuitive relationship between the maximin estimator and the weighted lasso is that the weights, for small ζ, are about $(\widehat{V}_\beta^g)^{-1}$, so instead of taking the minimum explained variance (maximin), the observations inside the group with low explained variance has high weight, so high importance in the estimation. When this approximation works (remind, the quantities $\widehat{V}_\beta^g$ have to be higher than 0 and same, if this assumption is fulfills we still do not know exactly how close is this result with the maximin) we can interpret maximin estimator close to the values of β which minimize the penalized weighted likelihood (see equation (2.14)), with weights proportional to the inverse of the explained variance in the group in which the observation belong to.

At the end of this section we give an intuitive motivation of the limit (2.15).
If the vector x has only one values higher than 0, without lose of generality suppose the first, the limit of the numerator can be simply solved

$$\lim_{\zeta \to 0} \left(\sum_{g=1}^{G} x_g^\zeta \right)^{\frac{1}{\zeta}} = \lim_{\zeta \to 0} \left(x_1^\zeta \right)^{\frac{1}{\zeta}} = x_1$$

If we have more than one values higher than 0, for example in our case, where all the values has to be positive, the results of the limit is infinity. To see that we can write the limit as

$$\lim_{\zeta \to 0} \left(\sum_{g=1}^{G} x_g^\zeta \right)^{\frac{1}{\zeta}} = \lim_{\zeta \to 0} \|x\|_\zeta$$

as is show in figure 2.5 this limit is not finite because the distance from the point to 0 goes to infinity if the point is not in one of the axis (in that case remains the same). Now, each x_g^ζ goes to 1 for $\zeta \to 0$, so the numerator goes to infinity with the same order of $G^{1/\zeta}$, by simulation is easy to see the exact results of the limit.

2.2.2 Explicit solution for maximin

Here we follow the results in section 2.1.3 to give an explicit solution of the estimator (2.12).
We write the equation in a different way

$$\hat{\beta}_{maximin} = \arg\min_\beta \left\{ \frac{1}{2} \max_g \left(\beta^T \frac{X_g^T X_g}{n_g} \beta - \frac{2}{n_g} \beta^T X_g^T Y_g \right) + \rho \|\beta\|_1 \right\}$$

which is the same with a different parametrization of the penalty ρ caused by the factor 1/2, if we have one explicative variable ($p = 1$) standardized, i.e. for all g $\widehat{\Sigma}_g = 1$ the objective function

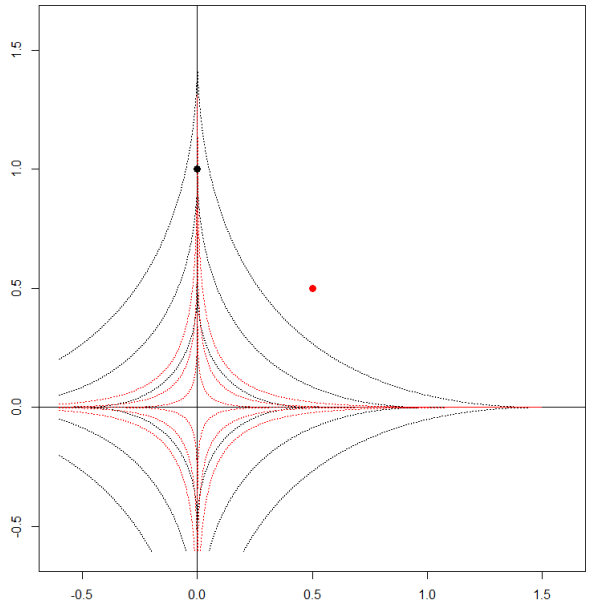


Figure 2.5. we can see the contours of equidensity for $l_{0.5}$-norm in black and for $l_{0.3}$-norm in red for the values 0.5,1 and 1.5. We can see that the black point (0,1) keeps the same distance moving from $q = 0.5$ to $q = 0.3$, instead the red point (0.5,0.5) increases his distance to the origin passing from 2 (with $q = 0.5$) to almost 5 with $l_{0.3}$-norm, if q moving to 0 the distance of this point increases really fast (for $q = 0.1$ is almost 512), becoming infinity when $q = 0$. The rate is faster in high dimension (in dimension $p = 5$ the values of the distance of $(0.5, ...,0.5)^T$ from the origin is around 12.5 for $q = 0.5$, 106.9 for 0.3 and 4.89×10^6 for 0.1)

becomes:

$$\frac{1}{2}\max_g\left(\beta^2 - 2\beta\frac{1}{n_g}\sum_i x_{i;g}y_{i;g}\right) + \rho|\beta| =$$
$$\frac{1}{2}\max_g\left(\beta^2 - 2\beta\hat{\beta}_{OLS;g}\right) + \rho|\beta| =$$
$$\frac{1}{2}\max_g\left(\beta - \hat{\beta}_{OLS;g}\right)^2 + \rho|\beta| =$$
$$\begin{cases}\frac{1}{2}\beta^2 - \beta\min_g\hat{\beta}_{OLS;g} + \rho|\beta| & \text{if } \beta > 0\\ \frac{1}{2}\beta^2 - \beta\max_g\hat{\beta}_{OLS;g} + \rho|\beta| & \text{if } \beta < 0\end{cases}$$

when $\beta > 0$ the derivative of the penalty equalized to 0 is

$$\beta - \min_g\hat{\beta}_{OLS;g} + \rho = 0 \iff \hat{\beta}_{maximin} = \left(\min_g\hat{\beta}_{OLS;g} - \rho\right)_+ \tag{2.17}$$

and in the other case

$$\beta - \max_g\hat{\beta}_{OLS;g} - \rho = 0 \iff \hat{\beta}_{maximin} = \left(\max_g\hat{\beta}_{OLS;g} - \rho\right)_- \tag{2.18}$$

where the function

$$(a)_+ = \begin{cases}a & \text{if } a \geq 0\\ 0 & \text{if } a < 0\end{cases} \quad \text{and} \quad (a)_- = \begin{cases}a & \text{if } a \leq 0\\ 0 & \text{if } a > 0\end{cases}$$

Now we try to extend the result to the general case, where $p > 1$ and the explicative can be correlated. For doing this we need a objective which is a function of the j-th explicative, to try to

find an explicit solution fixing the others.
To isolate β_j, note that the quadratic form $\beta^T \widehat{\Sigma}_g \beta$ can be write as:

$$(X\beta)^T(X\beta) = (X\beta_{-j} + \beta_j X_j)^T(X\beta_{-j} + \beta_j X_j) = \beta_{-j}^T X^T X \beta_{-j} + \beta_j^2 X_j^T X_j + 2\beta_j X_j^T X \beta_{-j}$$

where β_{-j} is the vector β with the j-th coordinate equal to 0.
The objective function becomes

$$\frac{1}{2}\max_g \left(\frac{1}{n_g} \left(\beta_{-j} X_g^T X_g \beta_{-j} + 2\beta_j X_{j;g}^T X_g \beta_{-j} + \beta_j^2 X_{j;g}^T X_{j;g} - 2\beta_{-j}^T X_g^T Y_g - 2\beta_j X_{j;g}^T Y_g \right) \right) + \rho \|\beta\|_1 =$$
$$\frac{1}{2}\max_g \left(\widehat{V}_{\beta_{-j}}^g + \widehat{V}_{\beta_j}^g + \frac{2}{n_g}\beta_j X_{j;g}^T X_g \beta_{-j} \right) + \rho|\beta_j| + \rho \sum_{h \neq j} |\beta_h|$$

we can not have an explicit solution because we can not write β_j only outside the function max, which is not derivable. So in a hypothetical coordinate descent approach in the optimization we have to solve numerically each step, we will talk later that a derivative free optimization function becomes really slow when p increase, so the results can be useful also without an explicit form solution.

2.3 Magging estimation for linear model

Another way to estimate the maximin effect is using Theorem 1:

$$\beta_{maximin} = \underset{\gamma \in \text{ Conv}(F)}{\arg\min}\ \gamma^T \Sigma \gamma, \quad \text{where } F = \text{supp}(F_B)$$

The *magging estimator* is a simply weighted aggregation of an usual estimator (like $\hat{\beta}_{OLS}$ or $\hat{\beta}_{lasso}$) where the weights are a convex combination which minimize the euclidean norm of the fitted values, formally:

$$\begin{aligned} \hat{\beta}_{magging} := \sum_{g=1}^G p_g \hat{\beta}_g \qquad \text{where } p := \underset{p \in C_G}{\arg\min} \left\| \sum_g X \hat{\beta}_g p_g \right\|_2, \\ C_G := \left\{ p : \min_g p_g \geq 0 \text{ and } \sum_g p_g = 1 \right\} \end{aligned} \tag{2.19}$$

and $\hat{\beta}_g$ is a (simply) estimator for the group g, for example $\hat{\beta}_g = \arg\min_\beta(\|Y_g - X_g\beta\|_2 + \rho\|\beta\|_1)$.
If the solution is not unique we choose the one which minimize $\|p\|_2$.
The connection with $\beta_{maximin}$ is that using Theorem 1, if we rewrite γ as:

$$\gamma = \sum_g p_g \beta_g = \sum_{g'} p_{g'} \beta_{g'}$$

the equation of the theorem becomes:

$$\beta_{maximin} = \sum_{g=1}^G p_g^0 \beta_g \qquad \text{where}$$
$$p^0 = \underset{p \in C_G}{\arg\min} \sum_{g,g'=1}^G p_g p_{g'} \beta_g^T \Sigma \beta_{g'} = \underset{p \in C_G}{\arg\min}\ E_X \left(\left\| \sum_{g=1}^G p_g X \beta_g \right\|_2^2 \right)$$

we obtain the magging estimation replacing $E(\|\sum_g p_g X \beta_g\|_2^2)$ with $\|\sum_g p_g X \hat{\beta}_g\|_2^2$.
The magging estimator is treated with more details in Meinshausen, Bühlmann - 2014 [4].

2.4 Extension to gaussian graphical model

Following the section 1.2 we want give a maximin estimation of the covariance matrix, as is defined in equations (1.9) and (1.10), before doing that, we show two approach to estimate the correlations between variables in a gaussian graphical model, the first (Meinshausen, Bühlmann - 2006 [13]) can be view as an approximation of the second (Friedman, Hastie, Tibshirani - 2008 [6]) which lead to the exact penalized maximum likelihood estimation.
As it has been said in the previous chapter a variable X_i in a gaussian graphical model is correlated with the variable X_j conditional to the other variables if the element (i, j) in the precision matrix Θ is not null, and the fact that Θ is symmetric it means that also the element (j, i) has to be not null, we'll talk later about the "problem" of the symmetry in the estimation.

2.4.1 Neighborhood selection

This method is proposed by Meinshausen and Bühlmann, is based to the estimation of the neighborhood (it will be defined later) for all variables to understand which of them is correlated to the target variable. There are some assumptions which, if are fulfill, made the method consistent.

We define the *neighborhood* ne_a of the node $a \in \Gamma$ as the set of all nodes $b \in \Gamma \setminus \{a\}$ which the correspondent variables X_b are correlated with X_a conditionally of all the remaining variables, i.e. the neighborhood of a is all the nodes $b \in \Gamma \setminus \{a\}$ in which the edge $(a, b) \in E$.
From a statistically point of view, the estimation of the neighborhood of a is a standard regression, this approach solve that with the lasso. So the definition of neighborhood is:

$$\begin{aligned} ne_a &= \{b \in \Gamma \setminus \{a\} : \beta_b^a \neq 0\} \qquad \text{where} \\ \beta^a &= \underset{\beta:\beta_a=0}{\arg\min}\, E\left(X_a - \sum_{k\in\Gamma} \beta_k X_k\right)^2 \end{aligned} \tag{2.20}$$

and the correspondent lasso estimation is:

$$\begin{aligned} \widehat{ne}_a^{\rho} &= \left\{b \in \Gamma \setminus \{a\} : \hat{\beta}_b^{a,\rho} \neq 0\right\} \qquad \text{where} \\ \hat{\beta}^{a,\rho} &= \underset{\beta:\beta_a=0}{\arg\min}\left(n^{-1}\|X_a - X\beta\|_2^2 + \rho\|\beta\|_1\right) \end{aligned} \tag{2.21}$$

For the consistence has to be fulfilled some assumptions of high dimensionality, nonsingularity, sparsity, magnitude of partial correlations and neighborhood stability, moreover the tuning parameter ρ has to be appropriate.
Note that with this method we can have different estimations of the correlation from a to b and the correlation from b to a, where with "correlation from, to" we mean that the estimation of the correlation of a and b is a priori diffrent if we use the neighborhood of a or the neighborhood of b. This can be a problem in the estimation of the edges of a graph, infact the edge set $E \subseteq \Gamma \times \Gamma$ is

$$E = \{(a, b) : a \in ne_b \wedge b \in ne_a\} \quad (= \{(a, b) : a \in ne_b \vee b \in ne_a\})$$

the two natural estimations

$$\hat{E}^{\rho,\wedge} = \{(a, b) : a \in \widehat{ne}_b^{\rho} \wedge b \in \widehat{ne}_a^{\rho}\} \tag{2.22}$$

$$\hat{E}^{\rho,\vee} = \{(a, b) : a \in \widehat{ne}_b^{\rho} \vee b \in \widehat{ne}_a^{\rho}\} \tag{2.23}$$

can be different, the last one is more conservative.
All the assumptions, the choice of the tuning parameter and all the proprieties of these estimations are showed in detail in the paper Meinshausen, Bühlmann - 2006 [13].

2.4.2 L_1–penalized maximum likelihood estimation

The method proposed in this section is described in detail in Friedman, Hastie, Tibshirani - 2008 [6], it leads to an exact l_1-*penalized maximum likelihood estimation* of the covariance matrix (and, by reparametrization, of the precision matrix). Is implemented by the function `glasso` from the namesake package [5].
Suppose that we have an i.i.d. sample of a multivariate normal of dimension p, mean μ and covariance matrix Σ, let $\Theta = \Sigma^{-1}$ and S the empirical covariance matrix. The partial log-likelihood of the data (maximized respect to μ) is

$$\log \det(\Theta) - \operatorname{tr}(S\Theta) - \rho\|\Theta\|_1 \tag{2.24}$$

where $\operatorname{tr}(A)$ denote the trace of the matrix A and $\|A\|_1$ denote the sum of the absolute values of A.
The maximization problem is convex and, if the algorithm starts with a positive definite matrix, after the all procedure the matrix remains positive definite, even if $p > n$ [2].
For the maximization consider estimation of Σ rather than $\Theta = \Sigma^{-1}$. Let W the current estimate of Σ, consider the partition

$$W = \begin{pmatrix} W_{11} & w_{12} \\ w_{12}^T & w_{22} \end{pmatrix}, \quad S = \begin{pmatrix} S_{11} & s_{12} \\ s_{12}^T & s_{22} \end{pmatrix}$$

where W_{11} and S_{11} are $(p-1)\times(p-1)$ matrix, w_{12} and s_{12} are vectors of dimension $p-1$, w_{22} and s_{22} are real values.
The algorithm starts with an initial W positive definite, usually $W = S + \rho I_p$, in each step solve the function

$$\begin{aligned} \hat{\beta} &= \arg\min_{\beta} \left\{ \frac{1}{2}\|W_{11}^{1/2}\beta - W_{11}^{-1/2}s_{12}\|_2^2 + \rho\|\beta\|_1 \right\} \\ &= \arg\min_{\beta} \left\{ \frac{1}{2}\left(\beta^T W_{11}\beta - 2\beta^T s_{12}\right) + \rho\|\beta\|_1 \right\} \end{aligned} \tag{2.25}$$

and update W (the current estimate of Σ) as $w_{12} = W_{11}\beta$.
Here we show that the solution of the problem (2.24) is the same of the (2.25).
W is the current estimation of Σ, so

$$\begin{pmatrix} W_{11} & w_{12} \\ w_{12}^T & w_{22} \end{pmatrix} \begin{pmatrix} \Theta_{11} & \theta_{12} \\ \theta_{12}^T & \theta_{22} \end{pmatrix} = \begin{pmatrix} I_{p-1} & 0 \\ 0^T & 1 \end{pmatrix}.$$

The sub-gradient of the partial penalized log-likelihood, evaluates in the maximum is

$$W - S - \rho \cdot \Gamma, \tag{2.26}$$

where $\Gamma_{ij} \in \operatorname{sign}(\Theta_{ij})$, i.e. $\Gamma_{ij} = \operatorname{sign}(\Theta_{ij})$ if $\Theta_{ij} \neq 0$, else $\Gamma_{ij} \in [-1,1]$.
In the upper right block we can write the previous equation

$$w_{12} - s_{12} - \rho \cdot \gamma_{12} = 0.$$

Then we have that the sub-gradient in the minimum of the equation (2.25) is

$$W_{11}\beta - s_{12} + \rho \cdot \nu = 0, \tag{2.27}$$

where $\nu \in \text{sign}(\beta)$.
In [6] is shown that if (W, Γ) solves (2.26), then $(\beta, \nu) = (W_{11}^{-1} w_{12}, -\gamma_{12})$ solves (2.27).
If $W_{11} = S_{11}$ the solution $\hat{\beta}$ is the lasso estimate for the pth variable respect to the others, and this is like the approach showed in the previous section, which do not lead a penalized maximum likelihood estimation.
Note that in this approach we haven't the problem of symmetry of the covariance matrix, because the current estimation W start symmetric (because S is) and in each step is updated w_{12} and consequently w_{12}^T, so in each iteration of the algorithm the current estimation remain symmetric.
The steps of the algorithm can be:

1. Start with $W = S + \rho I$
2. For each $j = 1, ..., p$ consider the j-th column (and the correspondent j-th row) as the last in W and S, solve the equation (1.10) and update $w_{12} = W_{11}\beta$,
3. Continue until convergence.

2.4.3 Maximin estimation

Here we give an extension to maximin using an approach similar to section 2.3.2, but first we briefly show an approach like the section 2.3.1 to understand the main problem of this method.
The extension of equation (2.21) is:

$$\widehat{ne}^{\rho}_{a;maximin} = \left\{ b \in \Gamma : \hat{\beta}^{a,\rho}_{b;maximin} \neq 0 \right\} \qquad \text{where}$$
$$\hat{\beta}^{a,\rho}_{maximin} = \underset{\beta:\beta_a=0}{\arg\min} \left(n^{-1} \max_g \left(\|X_a - X\beta\|_2^2 \right) + \rho \|\beta\|_1 \right)$$

This approach does not lead to an estimation of the covariance matrix. With equation (1.7) if we know the elements $\Theta_{i,i}$we can find the precision matrix, but is not a real estimation because we get a matrix which is not symmetric.
If we are intrested only to find the neighborhood (i.e. only where the correlation exist or not) we can obtain symmetry using an estimator like (2.23), but this method does not give an estimation of the "maximin quantities".
Also note that this method is not the empirical counterpart of the definition of maximin for a graphical model given in section 1.2.2: the objective function is the empirical counterpart of $\beta^T \Sigma_{11} \beta - 2\beta^T \widetilde{\Sigma}_{12}$, not of $\beta^T \Sigma_{11;maximin} \beta - 2\beta^T \widetilde{\Sigma}_{12}$, which is the one we have used in equation (1.9).
We can obtain an estimation of $\Sigma_{maximin}$ using an algorithm similar to the one in the previous section (Friedmann, Hastie, Tibshirani approch) replacing the loss function $L(\beta)$ with the "usual" maximin loss $L_{maximin}(\beta) = \max_{g=1,...,G}(-\widehat{V}_\beta^g)$, so replace:

$$w_{12} = W_{11} \cdot \underset{\beta}{\arg\min} \left\{ \left(\beta^T W_{11} \beta - 2\beta^T s_{12} \right) + \rho \|\beta\|_1 \right\}$$

with the equation:

$$w_{12;maximin} = W_{11} \cdot \underset{\beta}{\arg\min} \left\{ \max_{g=1,...,G} \left(\beta^T W_{11} \beta - 2\beta^T s_{12;g} \right) + \rho \|\beta\|_1 \right\} \tag{2.28}$$

Following this approach we start with a matrix $W = S + \rho I$ as our current estimation of $\Sigma_{maximin}$, instead of $S_g = X_g^T X_g / n_g$, in equation (2.12) we use our current estimator W. The

estimation of $\beta_{maximin}$ now is

$$\hat{\beta}_{maximin} = \arg\min_{\beta} \left\{ \max_{g=1,\dots,G} \left(\beta^T W_{11}\beta - 2\beta^T s_{12;g}\right) + \rho\|\beta\|_1 \right\}, \tag{2.29}$$

where $s_{12;g} = X_g^T Y_g/n_g$, the estimator of $\Sigma_{12;maximin}$ in equation (1.7) is:

$$w_{12,maximin} = W_{11}\hat{\beta}_{maximin} \tag{2.30}$$

As in the step 2. and 3. of the algorithm in section 2.3.2, updating the rows and columns of W and repeating the cycle until convergence, we get the estimation

$$\widehat{\Sigma}_{maximin} = W \quad \text{and} \quad \widehat{\Theta}_{maximin} = W^{-1} \tag{2.31}$$

2.4.4 L_0-norm approximation

Same as linear model, to approximate the estimator (2.29), which leads to the estimation of the quantities defined in section 1.2.2, we need to replace the values

$$\widehat{V}_{\beta}^{g} = 2\beta^T s_{12;g} - \frac{1}{n_g}\beta^T X_g^T X_g \beta$$

with

$$2\beta^T s_{12;g} - \beta^T W_{11}\beta$$

both in weights and penalty. The problems of this approximation is the same as the linear model, moreover (it will be discuss better in the next chapter) when the vector $s_{12,g}$ is close to 0, i.e. the target variable in this step of the algorithm have low correlations with the others, the estimate of the quantities $2\beta^T s_{12;g} - \beta^T W_{11}\beta$ are often (more than linear model scenario) negative.

2.4.5 Magging estimation

As in the previous sections to give a magging estimation of $\Sigma_{maximin}$ we replace the estimator (2.19) for linear model

$$\hat{\beta}_{magging} := \sum_{g=1}^{G} p_g \hat{\beta}_g \quad \text{where } p := \arg\min_{p\in C_G} \left|\left| \sum_g X\hat{\beta}_g p_g \right|\right|_2$$

with the estimator for graphical model:

$$\begin{aligned} w_{12;magging} := W_{11} \cdot \sum_{g=1}^{G} p_g \hat{\beta}_g \quad \text{where } p := \arg\min_{p\in C_G} \left|\left| \sum_g \widetilde{W}_{11}\hat{\beta}_g p_g \right|\right|_2, \\ C_G := \left\{ p : \min_g p_g \geq 0 \text{ and } \sum_g p_g = 1 \right\} \end{aligned} \tag{2.32}$$

where $\widetilde{W}_{11}$ is the Cholesky decomposition of the current estimator of $\Sigma_{11;maximin}$: $W_{11} = \widetilde{W}_{11}^T\widetilde{W}_{11}$. As in section 2.3, we have the connection with $\beta_{maximin}$ and $\Sigma_{maximin}$ for gaussian graphical model defined in equation (1.9), using Theorem 2 with $\gamma = \sum_g p_g\beta_g = \sum_{g'} p_{g'}\beta_{g'}$ we have:

$$\Sigma_{12,maximin} = \Sigma_{11,maximin} \cdot \sum_{g=1}^{G} p_g^0 \beta_g \quad \text{where } p^0 = \arg\min_{p\in C_G} \sum_{g,g'=1}^{G} p_g, p_{g'} \beta_g^T \Sigma_{11;maximin}\beta_{g'}$$

which is the theoretical counterpart of (2.32).

Chapter 3

Optimization problem

Here we show some approach to the optimization, we will talk about the conditions which has to be fulfilled for the convergence in the optimization. We'll talk also about the computational time of these methods, which in graphical model case, more than linear model, is really important, cause the optimization is reached cycling through the parameters until convergence, and each cycle is composed by p optimizations in $p-1$ dimensions.

3.1 Derivative-free method

The objective function (2.28) is convex but not everywere derivable, so in the optimization we can not use methods which involve the gradient of the function. For a derivative-free optimization there are the function `nmk` and `hjk` in R library `dfoptim` [18]. We can use these function in the optimization, but these are really slow in high dimension because the computational time increase more than linearly with respect the number of parameters.

I have used the function `nmk`, which using the simplex method of Nelder and Mead [15], all the results in this section are taken from that paper.
This algorithm, in a minimization in p parameters uses a simplex (generalization of a triangle in more than two dimensions) of $p+1$ dimensions, and move the polytope in the opposite direction respect to the vertex in which the value of the function is higher.
Let $y(P_k)$ the value of the function y in the k-th vertex, define $P_h = \arg\max_{P_i} y(P_i)$ and $P_l = \arg\min_{P_i} y(P_i)$ the vertex in the higher and lower values of the function respectively, let $\bar{P}$ the centroid of $\{P_1, ...P_{p+1}\}$, the minimization is done by three operations.
A *reflection* of P_h is the operation:

$$P^* = (1+\alpha)\bar{P} - \alpha(P_h), \quad \text{where } \alpha > 0$$

P^* is in the line which pass trought P_h and $\bar{P}$, in the opposite direction respect to the last one. If $y(P_l) < y(P^*) < \max_{i \neq h} y(P_i)$ we replace P_h with P^* and start with a new simplex. If $y(P^*) < y(P_l)$ is the new minimum, so expand the simplex in this direction with the second operation.
An *expansion* in the direction of P^* lead to the point

$$P^{**} = \gamma P^* + (1-\gamma)\bar{P}, \quad \text{where } \gamma > 1$$

If $y(P^{**}) < y(P_l)$ replace P_h by P^{**} and restart, else the expansion failed, so replace P_h by P^* and restart.

If the reflection fails ($y(P^*) > \max_{i \neq h} y(P_i)$), define a new $P_h = \min(P_h, P^*)$ and use the *contraction*:

$$P^{**} = \beta P_h + (1 - \beta)\bar{P}, \quad \text{where } 0 < \beta < 1$$

if $y(p^{**}) < \min(y(P_h), y(P^*))$ replace P_h with P^{**} and restart, else we have a failed contraction, replace all P_i by $(P_i + P_l)/2$ and restart.
The three operations are show in figure 3.1.
The algorithm will be over when the standard error of the function $\left(\sum_i (y(P_i) - y(\bar{P}))^2/p\right)^{1/2}$ is

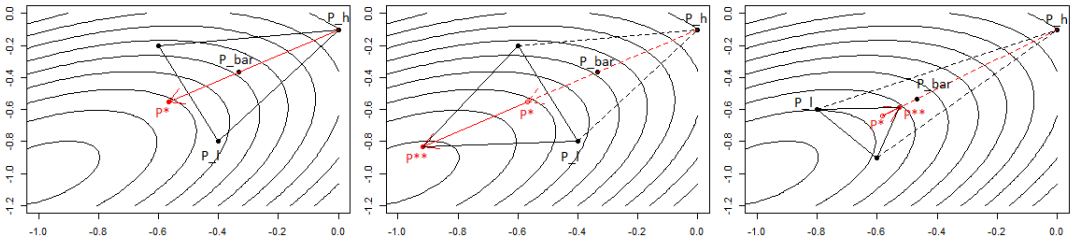


Figure 3.1. In the left we can see a reflection from the point P_h to the point P^*, seeing that the reflection is successfull in the second figure we can see the correspondent expansion to P^{**}. In the right we can see the contraction to P^{**} after a failed reflection

lower than a pre-fixed value.
This method give a more precise optimization if the slope close to the minimum value of the function is high, if not the precision is worse. This is not a big problem because usually in statistics we have a low slope close to the minimum if the parameter has an high variance, so has no sense obtain a too precise value related to the variance.
The method will lead to a good optimization if the simplex become relatively small only close to the minimum.
In figure 3.2 we have the flow diagram of the algorithm.

In our algorithm for using a non-derivative optimization method we simply use the function `nmk` inside a cycle for minimize respect to β the equation (2.29) and find $\hat{\beta}_{maximin}$, and after that trasform it in the estimation of the covariance matrix using the equation (2.30) $w_{12;maximin} = W_{11}\hat{\beta}_{maximin}$.
A possible way to implement a cycle in R using this optimization is in the appendix B.

3.2 Coordinate descent

Usually in lasso and his extension, the optimizations by coordinate descent are used. These methods converge to a global minima if the objective function is convex and derivable, but the assumption of derivability can be relaxed in some cases.
Thess methods minimize a p-dimensional function through smaller (usually univariate) optimizations in each dimension. The simpliest pseudo-code of a coordinate descent method can be:

1. start with some values $\{x_1^0, ..., x_p^0\}$ and fix the tollerance parameter $\delta > 0$
2. in the i-th iteration all the parameters are updated by the univariate optimization:
$$x_j^i = \arg\min_{x_j} \left(x_1^i, ..., x_{j-1}^i, x_j, x_{j+1}^{i-1}, ..., x_p^{i-1}\right)$$
3. stop if $\max_j(|x_j^k - x_j^{k-1}|) < \delta$

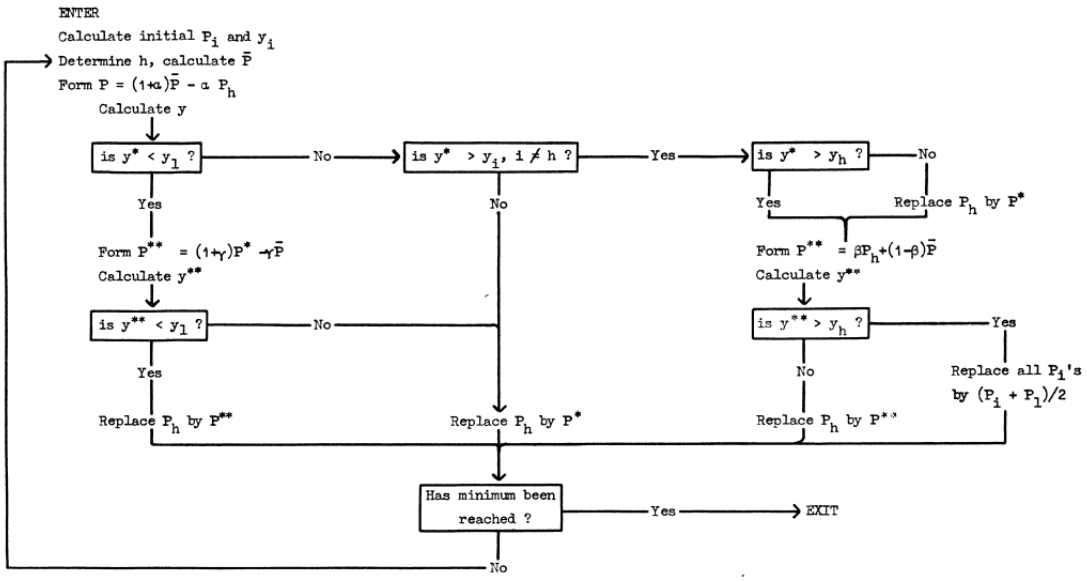


Figure 3.2. flow diagram of the algorithm, this figure is from Nelder and Mead, 1965 [15]

We can see in the code that these methods are useful if the smaller optimizations in point 2 can be performed really fast.
In section 2.1.3 we have seen that the lasso has a close form for the univariate optimization of the j-th parameter, fixed the others, this is why this method works really well in that case.
We have seen in chapter 2 that we can not find an explicit solution for the univariate optimization, like the lasso, so each step have to be solved numerically, but the main problem is that this method converges if the non-derivable objective function f can be written as:

$$f(\beta) = g(\beta) + \sum_i h_i(\beta_i) \tag{3.1}$$

where g is convex and derivable and the functions h_i are convex. So, the method converges if the non-derivable part of the penalty can be splitted in univariate functions of the parameters [8].
In equation (2.29) the non-derivable part is $-2\max_g(\beta^T s_{12;g}) + \rho\|\beta\|_1$, which is not separable because the function $\max_g(\beta s_{12,g})$ can not be splitted in a sum of univariate functions, so the convergence is not assured.
In figure 3.3 we can see how the non-derivable part of the objective function can be a problem in these methods.

3.3 L_0-norm approximation

In this method the optimization is done by weighted lasso, which (see equation (2.13)) is the "usual" lasso estimator with reweighted variables, as we see in the previous section, this optimization can be done by coordinate descent because the non-derivable part of the function, now, is separable.
To obtain the estimation we need to minimize the function

$$\sum_g (\widehat{V}_\beta^g)^{\zeta-1}(-\widehat{V}_\beta^g) + \lambda\|\beta\|_1 \tag{3.2}$$

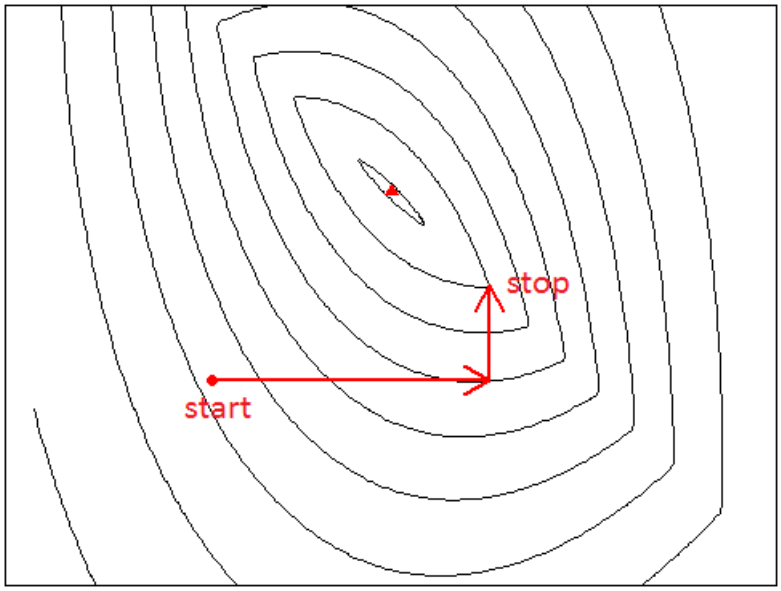


Figure 3.3. here we can see what the violated condition of derivability can cause (as we have said, the condition can be relaxed in some ways so that what happens in this example is not possible). In this case the algorithm does not reach the minimum of the function (red triangle), because in the point stop, in each direction following the axis, the objective function become higher, so the algorithm converges but not in the minima (neither local) of the function.

which, for the linear model, is the same of minimize

$$-\frac{2}{n}\sum_{i=1}^{n} p_i Y_i (X\beta)_i + \frac{1}{n}\sum_{i=1}^{n} p_i (X\beta)_i^2 + \lambda\|\beta\|_1 \tag{3.3}$$

the weights p_i are proportional to $(\widehat{V}_{\beta}^{g_i})^{\zeta-1}$, where g_i is the group in which observation i belong to (the function `glmnet` does not need that the weights sum to 1, only need that weights are greather or equal than 0).
The former equation can be write as

$$-\frac{2}{n}\beta^T X^T P Y + \frac{1}{n}\beta^T X^T P X \beta + \lambda\|\beta\|_1 \tag{3.4}$$

where the matrix P is diagonal and $P_{i,i} = p_i$.
The weights depend on ζ and β, for the first, note that we have derived this using the limit for $\zeta \to 0$, so in the optimization function we can fix a small value of ζ, like 0.01. For the fact that weights are function of β, we can find them iteratively using this algorithm:

1. start with uniform weights (for example equal to 1), i.e. start with $P = I_n$
2. minimizing (3.4) to obtain $\hat{\beta}_{par}$
3. update the weights as $P_{i,i} = (\widehat{V}_{\hat{\beta}_{par}}^{g_i})^{\zeta-1}$
4. repeat 2. and 3. until convergence, at the end the estimator is $\hat{\beta}_{l_0 maximin} = \hat{\beta}_{par}$

For linear model we can use in point 2. the function `glmnet`, from the namesake library [7].

In case of graphical model, as in the others optimization methods, we need to consider the current covariance estimation W.

First of all, consider again the linear model case, the lasso estimation

$$\begin{aligned}\hat{\beta}_{lasso} &= \arg\min_{\beta} \left\{\beta^T \cdot (X^T X) \cdot \beta - 2\beta^T \cdot (X^T Y) + \lambda \|\beta\|_1\right\} \\ &= f(X, Y) \\ &= g\left(\frac{X^T X}{n}, \frac{X^T Y}{n}\right)\end{aligned}$$

can be view both as a function of X and Y or a function of S and $X^T Y/n$. The same for the weighted lasso:

$$\begin{aligned}\hat{\beta}_{Wlasso} &= \arg\min_{\beta} \left\{\beta^T X^T P X \beta - 2\beta^T X^T P Y + \lambda \|\beta\|_1\right\} \\ &= g\left(\frac{X^T P X}{n}, \frac{X^T P Y}{n}\right) \\ &= g\left(S^P, \frac{1}{n}\left(P^{\frac{1}{2}} X\right)^T P^{\frac{1}{2}} Y\right)\end{aligned} \tag{3.5}$$

where S^P is the weighted empirical covariance matrix (i.e. the covariance of the weighted variables). Since we consider the influence of the weights in P, we have an "usual" lasso estimation of the weighted variables, our approximate estimator of $\Sigma_{maximin}$, following Friedman, Hastie, Tibshirani approach, have to be the common estimation W^P.
We need to split the dataset in block, like (1.4), where the target variable is X_2, the algorithm of a cycle is:

1. start with $W^P = S$

2. using W_{11}^P, find the estimation of w_{12}^P:
   (a) start with $P = I_n$
   (b) evaluate $\hat{\beta}_{par} = g\left(W_{11}^P, X_1^T P X_2/n\right)$, with g in (3.5)
   (c) update the weights as $P_{i,i} = \left(n_{g_i}^{-1} \hat{\beta}_{par}^T X_{1;g_i}^T X_{2;g_i} - \hat{\beta}_{par}^T W_{11}^P \hat{\beta}_{par}\right)^{\zeta-1}$ for $i = 1, ..., n$, $g_i \in \{1, ..., G\}$ is the group in which the i-th observation belong to
   (d) repeat (b) and (c) until convergence

3. update W^P as $w_{12}^P = W_{11}^P \hat{\beta}_{par}$

4. change the target variable and repeat from 2. for all the variables

For the optimization in 2.(b) we can use the function `lassoshooting` from the namesake library [1], which can use as input (for the linear model) the quantities $X^T X$ and $X^T Y$ instead of X and Y. In our case (see equation (3.5)) we have to give as input the quantities nW_{11}^P and $X_1^T P X_2$.

In chapter 2 we talked about the statistical problem of this method, which is that do not lead to the maximum of the original objective function. Considering that the maximum is reached only in the trivial case, in which all the empirical explained variances is equal, the approximation will be better if the variances are similar in all groups.

The other problem is that the approximation works only if all the values are higher than 0, but if there is almost no correlation between the target variable and the others, the values inside the brackets in step 2.(c) can be less than zero. Considering that the power is negative, we can not find the weight in this case. So this method can not be use in a tipical high dimensional analysis, because does not work well with the sparsity assumption.
An example of the code is in the appendix B.

3.4 Magging

To find the weights in the magging estimator defined in the equation (2.32) we need to minimize the function

$$\left\| \sum_g \widetilde{W}_{11} \hat{\beta}_g p_g \right\|_2 \quad \text{where } p \in C_G$$

this is a quadratic problem with constraint C_G (is a QP in standard form). To solve this, we use `solve.QP` from library `quadprog` [3], this function use the method of Goldfarb and Idnani (1982) [10], which is a *projection type dual algorithm.*

First of all a *quadratic problem (QP)* is:

$$\min_{x \in \Omega} \; q(x) = \frac{1}{2} x^T H x + d^T x \tag{3.6}$$

where Ω is a set of linear constraints:

$$a_i^T x = b_i \;\; \text{for } i \in \mathcal{E} \quad \text{and} \quad a_i^T x \geq b_i \;\; \text{for } i \in \mathcal{I} \tag{3.7}$$

The $p \times p$ matrix H is symmetric, the QP problem are convex if H is positive definite and Ω is convex, this lead to a global minimum. We will focus in convex QP problem.
The problem is called in *standard form*:

$$\min_{x \in \mathbb{R}^p} q(x) = \frac{1}{2} x^T H x + d^T x \qquad \text{s.t. } a_i^T x = b_i \text{ for } i \in \mathcal{E}, \; x \geq 0 \tag{3.8}$$

if $a_i = 1$ and $b_i = 0$ for all $i \in \mathcal{I}$.

The KKT conditions of (3.6) are

$$\begin{aligned} A_{\mathcal{E}} x = b_{\mathcal{E}}, \;\; A_{\mathcal{I}} x \geq b_{\mathcal{I}} \qquad & (feasibility) \\ \nabla q(x) = A_{\mathcal{E}}^T \pi + A_{\mathcal{I}}^T z \qquad & (stationarity) \\ z \geq 0 \qquad & (nonnegativity) \\ z \cdot (A_{\mathcal{I}} x - b_{\mathcal{I}}) = 0 \qquad & (complementarity) \end{aligned} \tag{3.9}$$

where π and z are vectors of Lagrange multipliers associated to the constraints. The variable x is called *primal variable*, the vectors π and z are called *dual variables* [20].

Let's define the *active set* in the solution:

$$\mathcal{A}(x^*) = \left\{ i \in \mathcal{E} \cup \mathcal{I} : a_i^T x^* = b_i \right\} \tag{3.10}$$

which represents the active constraints in the minimum. If $\mathcal{A}(x^*)$ is known the solution of the full problem (3.6) is the same of the reduced

$$\min_x \frac{1}{2} x^T H x + d^T x \quad \text{s.t.} \quad a_i^T x = b_i \text{ for } i \in \mathcal{A}(x^*) \tag{3.11}$$

An *active set method* is an iterative way to estimate the active set at the solution. Given a feasibility point x_0 (x_0 respect the constraints (3.7)), active set method compute a sequence of feasible iterates $x_{k+1} = x_k + \alpha_k s_k$ such that $q(x_{k+1}) \leq q(x_k)$, where s_k is a nonzero direction vector, $\alpha_k \geq 0$ is the step length.
Starting from $x_0 \in \Omega$, we choose a set $\mathcal{W}$ so that $\mathcal{E} \subset \mathcal{W} \subset \mathcal{A}(x_0)$, solve the reduced QP

$$\min_s q(x_0 + s) \quad \text{s.t } A_{\mathcal{W}}(x_0 + s) = b_{\mathcal{W}} \tag{3.12}$$

where $A_{\mathcal{W}}$ is the matrix of gradients of the constraints.
If the solution is $s = 0$, x_0 could be the full solution (has to respect the first order conditions), if it does not change the set $\mathcal{W}$.
If $s \neq 0$, determine $\alpha \leq 1$ the maximum values where the iteration $x_1 = x_0 + \alpha s \in \Omega$, if $\alpha = 1$ we are in a solution, if not, we have a new constraint not included in $\mathcal{W}$, we have to update this set and going in (3.12), the iterations are over until the solution of (3.12) is $s = 0$ [12].

A *projection method* with respect of the "usual" active set method, has the advantage that in each iteration many constraints can change, it is based on project the gradient of the function evaluate in the current point in a subset of Ω which represents the active constraints in that point. More formally, we are in the point $x_k \in \Omega$, the set which respect the active constraints is

$$\Omega_k = \{x_k + N(A_k)\}$$

where $N(A_k)$ is the null space of A_k, the full-rank matrix of gradients of the set of active constraints. Let q'_k the gradient of q evaluate in x_k, to minimize the function q from x_k we want move following the direction given by $-q'_k$, i.e. minimize $x_k - \alpha q'_k$ respect to α, but normally, for each $\alpha \neq 0$ we will not be anymore in Ω_k, so we project the gradient in $N(A_k)$ and solve the problem

$$\min_\alpha q(x_k - \alpha P_{N(A_k)} q'_k) \qquad \text{s.t.} \quad x_k - \alpha P_{N(A_k)} q'_k \in \Omega \tag{3.13}$$

where $P_{N(A_k)} = I - A_k^T (A_k A_k^T)^{-1} A_k$ is the projection operator. The disadvantage of this method is the computation of $P_{N(A_k)}$ [12].

A *dual method* is a way to solve the QP problem in standard form (3.8) by his dual:

$$\min_{w,z \in R^p, \pi \in R^m} q_D(w, \pi) = \frac{1}{2} w^T H w - b_{\mathcal{E}}^T \pi \qquad \text{s.t.} \quad Hw - A_{\mathcal{E}}^T \pi - z = -c, \quad z \geq 0 \tag{3.14}$$

where m is the number of equalities constraints
If the solution of (3.14) is bounded, and H is positive definite, the solution of (3.14) $w^* = x^*$ is also a solution of (3.8) [20].

Back to the magging estimator, first of all we find the group estimate $\hat{\beta}_g$ for all g, we can use the function `glmnet` (from the namesake package [7]) to find a lasso estimate, then to solve (2.32) we use a QP algorithm with the vector d in (3.6) equal to 0 and $H = B^T W_{11} B$, where B is the $(p-1) \times G$ matrix where the g-th column is the estimate $\hat{\beta}_g$.
For the constraint set C_G we use the $(G+1) \times G$ matrix

$$A = \begin{pmatrix} 1 \\ I_G \end{pmatrix}$$

and the $(G+1)$ vector $b = (1,0,...,0)^T$. We set in the function that the first constraint is an equalities and the others are inequalities.

So the optimization solves the quadratic function:

$$(p_1\ p_2\ ...\ p_g)\begin{pmatrix}\hat{\beta}_1^T\\ \hat{\beta}_2^T\\ \vdots\\ \hat{\beta}_G^T\end{pmatrix} W_{11}(\hat{\beta}_1\ \hat{\beta}_2\ ...\ \hat{\beta}_G)\begin{pmatrix}p_1\\ p_2\\ \vdots\\ p_G\end{pmatrix} \quad \text{s.t} \quad \begin{pmatrix}1 & 1 & \dots & 1\\ 1 & 0 & \dots & 0\\ 0 & 1 & \dots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \dots & 1\end{pmatrix}\begin{pmatrix}p_1\\ p_2\\ \vdots\\ p_G\end{pmatrix} \geq \begin{pmatrix}1\\ 0\\ 0\\ \vdots\\ 0\end{pmatrix} \tag{3.15}$$

where the first constraint ($\sum_i p_i$) is an equalities (set in the function the imput `meq=1`).
For an example of the code in a cycle see the appendix.

3.5 Optimization function comparing

In this section we compare the results using a derivative free optimization (which uses the maximin estimation in section 2.4.3) and the magging estimation (section 2.4.5).

First of all let's talk about time, we compare the methods using the same dataset and groups divisions. We evaluate the time using different number of groups G and especially different dimensions p.
To do this, we set a tollerance ϵ between the current iteration and the previous, the cycle continues until convergence, i.e. the maximum difference between the current estimation in the k-th iteration $W^{(k)}$ and $W^{(k-1)}$ is lower than the tollerance, in each step is useful print this difference to understand how the algorithm converge, we fix also a maximum number of iterations.
The pseudocode is:

1. set $W^{(0)}$ as $W^{(1)} - 0.5 \cdot \mathbf{1}$, where $\mathbf{1}$ is the matrix with all elements equal to 1.
2. while $\max_{i,j} |[W^{(k)}]_{i,j} - [W^{(k-1)}]_{i,j}| > \epsilon$ (if we haven't reached the maximum number of iterations) obtain the $(k+1)$-th estimation $W^{(k+1)}$ and compare it with the previous one.

We see that the maximin estimator has worst computationally proprieties than magging, is slower and has problems of convergence also in low/moderate dimension. Even the magging estimator has some problem of convergence, it happens when the lower eigenvalue of the matrix $(\hat{\beta}_1, ..., \hat{\beta}_G)^T W_{11}(\hat{\beta}_1, ..., \hat{\beta}_G)$ is close to 0, but it happens rarely when the values of G and ρ are too low respect the dimensions p. Before beginning a cross validation (section 4.2) it's useful to check the convergence for the lower values of G and ρ, if the algorithm converges in this case, will do it also for the higher values of G and ρ.
Usually the problem of convergence in high dimension with low values of (G, ρ) do not disturb us, because in high dimension we need a penalty, when ρ is too low the estimator has worst statistical proprieties, because the coefficients are not shrinked enough.
In table 3.1 we have a time comparison between non derivative and quadratic optimization for different values of groups in low dimensions ($p = 8$ and $p = 16$ with 5 groups), we measure also the time of magging optimization in higher dimensions.

	$\epsilon = 0.01$, maximin	$\epsilon = 0.01$,magging	$\epsilon = 0.05$, maximin	$\epsilon = 0.05$,magging
$p = 8$:				
$G = 5$	3.74 (3), 5.25 (4)	0.76 (12), 0.62 (10)	2.68 (2), 2.81 (2)	0.37 (6), 0.27 (4)
$G = 20$	27.85 (10), 29.43 (12)	1.61 (7), 1.69 (7)	5.36 (2), 8.87 (3)	0.93 (4), 0.92 (4)
$G = 40$	nc, nc	2.25 (5), 2.24 (5)	14.62 (3), 19.39 (4)	1.80 (4), 0.97 (3)
$p = 16$:				
$G = 5$	nc, nc	2.64 (20), 1.97 (15)	77.57 (5), 46.64 (3)	1.31 (10), 1.49 (11)
$G = 20$		4.47 (11), 4.91 (11)		4.10 (9), 3.67 (8)
$G = 40$		9.56 (11), 9.50 (11)		6.31 (7), 7.03 (7)
$p = 32$:				
$G = 5$		4.72 (16)		2.72 (10)
$G = 20$		12.37 (13)		8.85 (9)
$G = 40$		12.63 (7)		10.80 (6)
$p = 64$:				
$G = 5$		9.66 (14)		6.31 (9)
$G = 20$		22.31 (10)		18.00 (8)
$G = 40$		36.80 (9)		29.41 (7)
$p = 128$:				
$G = 5$		29.81 (15)		21.97 (11)
$G = 20$		59.32 (10)		46.95 (8)
$G = 40$		81.25 (8)		60.97 (6)
$p = 256$:				
$G = 5$		127.70 (17)		842 (11)
$G = 20$		238.64 (12)		166.39 (8)
$G = 40$		254.73 (8)		188.02 (6)

Table 3.1. time comparing between derivetive free optimization (maximin) and quadratic (magging) for different values of dimensions p, number of groups G and tollerance ϵ, for each case we measure maximin and magging optimization for two simulated dataset. The time are in seconds and inside the round brackets there are the number of cycle to reach the convergence, the values nc indicate the non convergence of the optimization. We stop the comparison at $p = 16$ and $G = 5$ because at that point seems clear the superiority of magging estimation method. The other rows are the time measured for magging estimator in higher dimensions, we see that with higher groups the optimization is reached in less cycle, i.e. the convergence is more stable, even if the time for each cycle is higher.

Chapter 4

Complements

4.1 Dimensionality reduction in the optimization

A first improvement in the optimization can be reached using the sparsity assumption, the following theorems are an adaptation to maximin of the theorem 1 and 2 from the paper of Witten, Friedman, Simon [19] for the graphical lasso problem.

Theorem 3. *If it's know that the precision matrix Θ is block diagonal, the maximin problem can be solved with separate optimization for each block, using the correspondent block of the empirical covariance matrix S.*

Theorem 4. *Let $P_1, ..., P_K$ a partition of the p variables, if*

$$\max_g |S_{i,i';g}| \leq \rho, \forall i \in P_k, i' \in P_{k'} \text{ with } k \neq k'$$

then the solution of maximin problem can be solved in a diagonal block structure.

The proofs is given in the appendix A.
These theorems are useful because we can split the p dimensional problem in smaller optimization respectively of $p_1, ..., p_I$ parameters, with $\sum_i (p_i) \leq p$.
This is useful because if a variable is in absolute value less correlated than ρ respect all the other variables in all G empirical covariance matrix can be excluded from the problem, also, the optimization time is not linear respect p, so if we can is better split the problem to decrese the computational cost.
We need an algorithm to find the partition given the empirical covariance matrix $S_1, ..., S_G$. We introduce some other concepts in the graph theory.
A *path* from X_1 to X_2 is a sequence of edges which connects the node X_1 with X_2, the length of a path is the number of edges crossed for moving from X_1 to X_2 (or vice versa) in a determinate course.
The *adiacency matrix* A of an undirected graph is a matrix that has value 0 in the position (i, j) if there isn't an edge between the nodes X_i and X_j, otherwise has value 1 (of course $A_{i,i} = 0$ for all i).
The matrix A^n is symmetrical for each value of $n \in \mathbb{N} \setminus 0$, and the value $[A^n]_{i,j}$ represents the number of possible paths of length n from the node X_i to the node X_j (note that a diagonal element of A^n can be greather than 0).

The variable X_i and X_j are indipendent if does not exist a path between them, more formally:

$$\text{for } i \neq j,\ X_i \perp\!\!\!\perp X_j \iff [A^n]_{i,j} = 0\ \forall\ n \geq 1 \iff \left[\sum_{n=1}^{p} A^n\right]_{i,j} = 0 \tag{4.1}$$

Note that we do not need to evaluate and sum the matrix A^n with $n > p$ because all the paths from X_i to X_j of length greather than p have to contain a smaller path between these two variables.
An algorithm to find the groups, using this formula, can be:

1. evaluate the matrix $Smax$, where $Smax_{i,j} = \max_g |[S_g]_{i,j}|$
2. the adiacency matrix is A, where $A_{i,j} = 1$ if $Smax_{i,j} > \rho$, else $A_{i,j} = 0$
3. evaluate the matrix $A^* = I + A + A^2 + ... + A^p$
4. scanning all the rows [columns] we can find the groups, which are formed by the columns [rows] of the element greather than 0, note that in A^* we have sum also the identity matrix for find the groups composed by isolate variables

An improvement of this algorithm can be done in the step 3, if p is big that step can be really slow because we have to do at list $p - 1$ matrix moltiplications for find A^*.
When p is high, can be used the function `expm` from the namesake library [11], this function takes as imput a matrix M and evaluates its exponential, defined as

$$\text{expm}(M) = \sum_{i=0}^{\infty} \frac{M^i}{i!}, \quad \text{where } M^0 = I, \tag{4.2}$$

in a faster ways.
Since for all n the elements of A^n are greather or equal than 0

$$\left[I + \sum_{n=1}^{p} A^n\right]_{i,j} = 0 \iff \left[\sum_{i=0}^{\infty} \frac{A^i}{i!}\right]_{i,j} = 0$$

so in the step 3 of the algorithm we can replace A^* with $\text{expm}(A)$.

4.2 Cross validation

In all optimization methods that we have discuss the penalty ρ is unknown, so we need to choose a value from the data. Moreover, if also the groups of observations are unknown, we need to learn also G.
In the general case (the groups in which observations belong to are unknown) we need a bi-dimensional cross validation, where the parameters are (ρ, G), the penalty and the numbers of groups, we will see that there are other problems.

For now let's focus in linear model case, the maximin quantities are defined to isolate the common effect in all the possible scenario, i.e. the effect in the worst possible case (see Figure 1.2), in Meinshausen Bühlmann - 2015 [14] maximin effect is characterized in another way:

$$\beta_{maximin} = \arg\max_{\beta} E\left(\|X\beta\|_2^2\right) \quad \text{s.t.} \min_{b \in \text{supp}(F_B)} E\left((X\beta)^T(Xb - X\beta)\right) \geq 0 \tag{4.3}$$

We can see that the maximin effect is the coefficient that maximizes the prediction imposing that remain positively correlated with the residual, so we can under-explain a signal, never over-explain. This is a problem in the cross validation because the values that minimize the prediction error over-estimate the maximin quantities (i.e. the penalty in the estimator tend to be too small), in [14] is showed a possible way to implement cross validation.
The procedure to evaluate the maximin estimator is based on split randomly the test set in g groups (the values g has to be previously fixed), as the maximin is the value which maximize the explained variance in the worst possible scenario, we use to evaluate the estimate (with a prediction error) the group in which the empirical explained variance (evaluated in $\hat{\beta}_{maximin}$ obtained with the training set) is lower. The algorithm can be:

1. split the dataset in two blocks randomly without replacing. The first block is the training set, the second the test set
2. split the training set in G groups and evaluate $\hat{\beta}_{maximin}$, this estimation depends on G and the penalty ρ
3. split the test set in g groups and evaluate the empirical explained variance (equation (2.11)) in $\hat{\beta}_{maximin}$ for all the groups in the test set, call g_{min} the group in which $\widehat{V}_{\hat{\beta}_{maximin}}$ is lower
4. obtain a prevision for the elements in group g_{min}, evaluate the prediction error respect some norm
5. repeat 2. - 4. for different values of G and ρ
6. repeat 1. - 5. various time, average the prediction errors obtained with different training and test set

The parameter g has to be chosen in advance and has to remain constant over all the algorithm, higher values of g intuetivelly give better estimations of the worst scenario (there are more explained variances to choose the worst one), but give worst estimations of the variance in all the scenarios. In [14] is suggest to choose g as big as possible but the groups have to be at least a few hundred of observations.
In 1 the sampling scheme is uniformly without replacing, in 2. and 3. the split depends of the scenario in which the data are from. If we have cronological observations we have to split the two half-samples in blocks of consecutive observations, if we are in a mixture model scenario we have to split randomly without replacing inside each group, but with replacing behind the groups (this method allows to increase g without shrink too much the number of observations).

In case of graphical model we have an additional problem, we haven't a response variable in which evaluate the prediction error, we need to cycling predict all the variables using the others as explicative (see models (1.6)), or (if the dimension p is too high) choose some variables in which evaluate the prediction error, we can choose these randomly or not, if we have a group of variables in which we are more intrested.
The choosed variables have to remain constant in all the procedure, with a cross-validation as in the previous algorithm we obtain a prediction of all the intrested variables (we fix null column in the prediction for the not-intrested variables), the matrix error is the difference between the observed data and the prediction in the column correspondent to the intrested variables, we have a measure of the error evaluating the norm (fixed a priori) of this matrix.

More preciselly, in each step we obtain a forecast (which depend on G and ρ) on the target variable in group g_{min}, let's call $X_{p_1}, X_{p_2}, ..., X_{p_i}$ the variables in which we are intrested,

$\{p_1, p_2, ..., p_i\} \subseteq \{1, ..., p\}$.
Let's call $X_{p_j}^{g_j}$ the observed variables in the group $g_j \in \{1, ..., g\}$, fixed G and ρ, let's the target variable the j-th, we obtain a prevision $\widehat{X}_j^{g_{min;j}}$ (note that the group in which the explained variance is lower depend on the target variable, g_{min} is not the same group for all the variables), then we evaluate the error in the matrix with j-th column $X_j^{g_{min;j}} - \widehat{X}_j^{g_{min;j}}$ if j is in $\{p_1, p_2, ..., p_i\}$, instead the column is the null vector.
The error is the norm of this matrix, doing this method for some values of (ρ, G), we have an estimation of the error for various number of groups and penalties, the estimation is better if we repeat the whole procedure with different training and test set and average the error.
The main problem of this method is that, especially in high dimension, can be really slow because, considering the whole procedure, we have to compute $f \cdot n_G \cdot n_\rho$ magging optimization, where n_G and n_ρ are the lengths of the paths of the number of groups and the penalty function in which we are intrested to evaluate the error, f is the number of repetition of the procedure. Then we need to evaluate $f \cdot n_G \cdot n_\rho \cdot g \cdot i$ empirical variances (i is the number of variables in which we are intrested), and obtain the prediction of $f \cdot n_G \cdot n_\rho \cdot n_g \cdot i$ values, where n_g is the number of observations in the test groups.
A possible implementation is showed in appendix B.

As we have said the error is evaluated using a matrix norm, now we give an intuitive definition and discuss a bit some norms.
A matrix norm is an extension of a "usual" vector norm, there are two main extensions.
The *induced l_p-norm* of the $m \times n$ matrix A corresponding to the vectorial l_p-norm is the value

$$\|A\|_{p,ind} = \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p} \tag{4.4}$$

which has not an explicit solution in general, the more used induced norms are:

1. $\|A\|_{1,ind} = \max_j \left(\sum_i |a_{ij}|\right)$
2. $\|A\|_{\infty,ind} = \max_i \left(\sum_j |a_{ij}|\right)$
3. $\|A\|_{2,ind} = \max_i(s_i(A))$, where $s_i(A)$ is the i-th singular values of A

Another way to define a matrix norm is consider a matrix A as a $m \cdot n$-dimensional vector with components a_{ij}, and simply using as a norm the "usual" vectorial l_p-norm. Is called *Schatten norm* the values

$$\|A\|_{p,Sch} = \left(\sum_{i=1}^{\min\{m,n\}} (s_i(A))^p \right)^{\frac{1}{p}} \tag{4.5}$$

where $s_i(A)$ is the i-th singular values of A.
The more used Schatten norm are:

1. $\|A\|_F := \|A\|_{2;Sch} = \sqrt{\sum_i \sum_j |a_{ij}|^2} = \sqrt{\mathrm{tr}(A^T A)}$, called *Frobenius norm*
2. $\|A\|_* := \|A\|_{1;Sch} = \mathrm{tr}\left((A^T A)^{1/2}\right)$, called *nuclear norm*

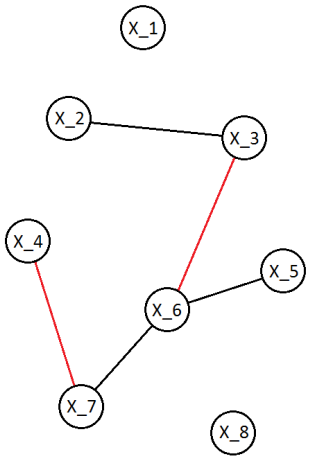
Chapter 5

Simulation

In this chapter we show some simulations with generated datasets to show the main advantages and problems of this method, in each simulation we will compare the method with the estimation obtained with the empirical covariance matrix.

The first simulation is in "low" dimension to show better how magging estimator works, the sparse graph in the right represents the variable $X = (X_1, ..., X_8)^T$ with precision matrix

$$\Theta = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -0.8 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.8 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 1 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$



for the normal values, and for the outliers

$$\Theta_{out} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -0.8 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.8 & 1 & 0 & 0 & -0.3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -0.5 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.5 & 0 & 0 \\ 0 & 0 & -0.3 & 0 & 0.5 & 1 & 0.5 & 0 \\ 0 & 0 & 0 & -0.5 & 0 & 0.5 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The $n = 2000$ observations are sampled from a $N(0, \Theta^{-1})$ with probability $\pi = 0.8$ and a $N(0, \Theta_{out}^{-1})$ with probability $1 - \pi$.
We use a cross validation (section 4.2) for the parameters G and ρ between the values (2,3,4,5,10,20) and (0.01,0.02,0.04,0.08,0.16,0.32) respectively, we evalute the estimations in $g = 15$ test groups, for the whole variables and using the Frobenius norm. The estimated errors are:

$$\begin{pmatrix} 24.08 & 24.25 & 23.95 & 23.86 & 23.73 & 24.19 \\ 23.88 & 23.78 & 23.92 & 23.90 & 23.85 & 24.16 \\ 24.45 & 24.29 & 24.04 & 24.21 & 23.81 & 24.30 \\ 24.37 & 24.28 & 24.13 & 23.91 & 23.92 & 24.28 \\ 24.41 & 24.44 & 24.17 & 23.55 & 23.82 & 24.18 \\ 24.80 & 24.79 & 24.80 & 24.65 & 24.27 & 24.85 \end{pmatrix}$$

where the columns are the values of penalty, the rows of groups. We have similar values, but the best one is in position (5,4), so the parameter that we will use are $(G, \rho) = (10,0.08)$ it's useful show also the optimization with parameters $(G, \rho) = (3,0.02)$, which is close to the second one lower.
The two estimated matrix (after the trasformation as correlation matrix) are

$$\widehat{\Theta}_{(10,0.08)} = \begin{pmatrix} 1.00 & -0.01 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0 \\ -0.01 & 1.00 & -0.63 & -0.01 & 0.01 & 0.00 & 0.00 & 0 \\ 0.00 & -0.63 & 1.00 & -0.01 & 0.01 & -0.01 & 0.01 & 0 \\ 0.00 & -0.01 & -0.01 & 1.00 & 0.02 & -0.01 & 0.00 & 0 \\ 0.00 & 0.01 & 0.01 & 0.02 & 1.00 & 0.28 & 0.00 & 0 \\ 0.00 & 0.00 & -0.01 & -0.01 & 0.28 & 1.00 & 0.49 & 0 \\ 0.00 & 0.00 & 0.01 & 0.00 & 0.00 & 0.49 & 1.00 & 0 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1 \end{pmatrix}$$

$$\widehat{\Theta}_{(3,0.02)} = \begin{pmatrix} 1 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0 \\ 0 & 1.00 & -0.72 & 0.00 & 0.00 & 0.00 & 0.00 & 0 \\ 0 & -0.72 & 1.00 & -0.03 & 0.00 & -0.10 & 0.03 & 0 \\ 0 & 0.00 & -0.03 & 1.00 & 0.00 & -0.04 & 0.02 & 0 \\ 0 & 0.00 & 0.00 & 0.00 & 1.00 & 0.43 & -0.03 & 0 \\ 0 & 0.00 & -0.10 & -0.04 & 0.43 & 1.00 & 0.49 & 0 \\ 0 & 0.00 & 0.03 & 0.02 & -0.03 & 0.49 & 1.00 & 0 \\ 0 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1 \end{pmatrix}$$

We can see that the second estimation may be better to estimate Θ, but is definitely worst if we are intrested to find the correlations more than estimate them.
Number of groups and penalty shrink to 0 the values, the true correlations values are shrinked to 0 more in the first case, but the bias is not enough to shrink too much the true correlations to 0, i.e. even with higher penalty we find the true correlations.
The others values with higher penalty are correctly shrinked to zero, the improvement is especially for the outliers, as we can see especially for the one in position (3,6).
Intuitively a signal in all the observations remains in all G groups, instead an erratic signal goes in different frequency to the various groups and, chosing the group which explain less, the erratic signal is skrinked more to 0 than the true. With higher G we have more unstable estimate, but we hope that the distortion to zero is higher for the outliers, so the "only" estimated correlations are real.
Cross validation maximizes the prediction with respect to some norms, if some couple of parameters has similar prediction error can be useful to choose the higher.
Of course in a real dataset we do not know Θ, so after the choice of (G, ρ) (after the cross validation), we have to suppose that these parameters are the best for the optimization.

The next simulations will be in higher dimension, so we evaluate the results with respect to the generating matrix using some matrix norms, different norms can lead to different results and we need to choose it also for the cross validation function.
There are not a right norm to use, we utilize the most common.

For the second simulation we keep $n = 2000$ observations but in dimension $p = 50$ and probability of outliers $1 - \pi = 0.2$. We sample the correlations controlling the maximum frequency (we need a sparse graph) with a parameter $s < 1$, the maximum number of edges is $e_{max} = p(p/2 - 1)$, we fix the maximum number of correlations is $s \cdot e_{max}$ for the normal values, the outliers have others additional correlations, again we set a parameter s_{out}.
Now we have $s = 0.10$ and $s_{out} = 0.30$, the normal correlations are sampled from a $U(-0.4,0.4)$ and rounded at the first decimal, the erratic correlations are from $U(-0.1,0.1)$ and are rounded to the second decimal (the precision matrix, which in our case is a correlation matrix, has to be

positive definite).
Again we have choose to the parameters by cross validation, the procedure is moderately slow, so we have slightly modified the function to give us the error evaluated with Frobenius norm and infinity norm for the same optimizations, the errors are the average of 5 full procedures, with paths for G equal to 2,4,8,16,32 and for ρ equal to 0.02,0.04,0.08,0.16,0.32, the groups in the test set are $g = 15$ and we evaluated the prediction in 25 variables sampled before than the procedure starts. The results of the errors are:

$$\begin{pmatrix} 42.24 & 41.67 & 41.46 & 41.50 & 43.31 & 47.46 \\ 42.88 & 42.09 & 41.92 & 41.34 & 43.01 & 47.04 \\ 44.13 & 43.39 & 42.64 & 41.74 & 43.11 & 46.59 \\ 44.42 & 43.58 & 42.94 & 42.81 & 43.68 & 46.28 \\ 44.80 & 44.53 & 44.22 & 44.05 & 44.27 & 46.48 \end{pmatrix} \quad \begin{pmatrix} 28.67 & 27.60 & 27.92 & 28.50 & 28.72 & 31.92 \\ 29.54 & 28.73 & 27.80 & 27.88 & 28.58 & 30.00 \\ 30.76 & 30.11 & 29.85 & 27.91 & 29.13 & 29.90 \\ 30.17 & 28.72 & 28.35 & 28.69 & 29.05 & 30.68 \\ 29.81 & 30.82 & 29.97 & 30.22 & 29.20 & 30.67 \end{pmatrix}$$

evaluated respectively with the Frobenius norm and the infinity norm. The best set of parameters with the Frobenius are $(G, \rho) = (4,0.16)$, with the infinity are $(G, \rho) = (2,0.04)$, we evaluate the results of the optimization with this two couples of parameters and the result with the maximum likelihood estimation using the matrix $\widehat{\Theta}_{MLE} = \text{corr}(S^{-1})$ (the estimation obtained with the standardized precision matrix, same as the others, we generated from a correlation matrix, so we trasform all the estimations in to that).
We use all the norm defined in section 4.2, and the results are:

$$\begin{aligned} &\|\widehat{\Theta}_{(4,0.16)} - \Theta\|_F = 1.39 && \|\widehat{\Theta}_{(2,0.04)} - \Theta\|_F = 0.70 && \|\widehat{\Theta}_{MLE} - \Theta\|_F = 1.17 \\ &\|\widehat{\Theta}_{(4,0.16)} - \Theta\|_\infty = 0.86 && \|\widehat{\Theta}_{(2,0.04)} - \Theta\|_\infty = 0.58 && \|\widehat{\Theta}_{MLE} - \Theta\|_\infty = 1.19 \\ &\|\widehat{\Theta}_{(4,0.16)} - \Theta\|_2 = 0.44 && \|\widehat{\Theta}_{(2,0.04)} - \Theta\|_2 = 0.23 && \|\widehat{\Theta}_{MLE} - \Theta\|_2 = 0.40 \\ &\|\widehat{\Theta}_{(4,0.16)} - \Theta\|_* = 8.14 && \|\widehat{\Theta}_{(2,0.04)} - \Theta\|_* = 4.04 && \|\widehat{\Theta}_{MLE} - \Theta\|_* = 6.64 \end{aligned}$$

we see that using $(G, \rho) = (2,0.04)$ we have way better results than the other cases for all norm, in this case $\widehat{\Theta}_{(2,0.04)}$ lead to a better estimation of Θ.
As we have said in the previous simulation, higher penalty and number of groups conduct to biased estimation of the correlations, so a worst estimation of Θ, but can be useful to identify better the true correlations if there are outliers. To evaluate this we fix a threshold in which the higher values are set to 0, both in the theoretical matrix Θ and in the estimates $\widehat{\Theta}$ to compare the "low" correlations between the estimates and the theoretical. We fix the threshold as 0.05 (is an arbitrary value, we haven't a procedure to set it) and define the matrix A^* where the generic element $a^*_{i,j} = a_{i,j}$ if $a_{i,j} < 0.05$, instead $a^*_{i,j} = 0$.
The evaluated norm are:

$$\begin{aligned} &\|\widehat{\Theta}^*_{(4,0.16)} - \Theta^*\|_F = 0.94 && \|\widehat{\Theta}^*_{(2,0.04)} - \Theta^*\|_F = 0.56 && \|\widehat{\Theta}^*_{MLE} - \Theta^*\|_F = 1.09 \\ &\|\widehat{\Theta}^*_{(4,0.16)} - \Theta^*\|_\infty = 0.54 && \|\widehat{\Theta}^*_{(2,0.04)} - \Theta^*\|_\infty = 0.49 && \|\widehat{\Theta}^*_{MLE} - \Theta^*\|_\infty = 1.09 \\ &\|\widehat{\Theta}^*_{(4,0.16)} - \Theta^*\|_2 = 0.30 && \|\widehat{\Theta}^*_{(2,0.04)} - \Theta^*\|_2 = 0.18 && \|\widehat{\Theta}^*_{MLE} - \Theta^*\|_2 = 0.34 \\ &\|\widehat{\Theta}^*_{(4,0.16)} - \Theta^*\|_* = 4.89 && \|\widehat{\Theta}^*_{(2,0.04)} - \Theta^*\|_* = 3.16 && \|\widehat{\Theta}^*_{MLE} - \Theta^*\|_* = 6.24 \end{aligned}$$

we see that in the low correlations $\widehat{\Theta}_{(4,0.16)}$ is better than $\widehat{\Theta}_{MLE}$, but remains worst than $\widehat{\Theta}_{(2,0.04)}$, even if the difference between the errors in the two estimators is lower.
In figure 5.1 is show the theoretical and the estimated model.

The last simulation is with $p = 200$, Θ really sparse, with normal correlations sampled in $\{-0.4, -0.3, ...,0.4\}$. After the sampling there are 81 correlations in $\{\pm 0.1, \pm 0.2, \pm 0.3, \pm 0.4\}$. Same as before there are outliers in proportion $1 - \pi = 0.20$, in this case the "outliers graph" is the same

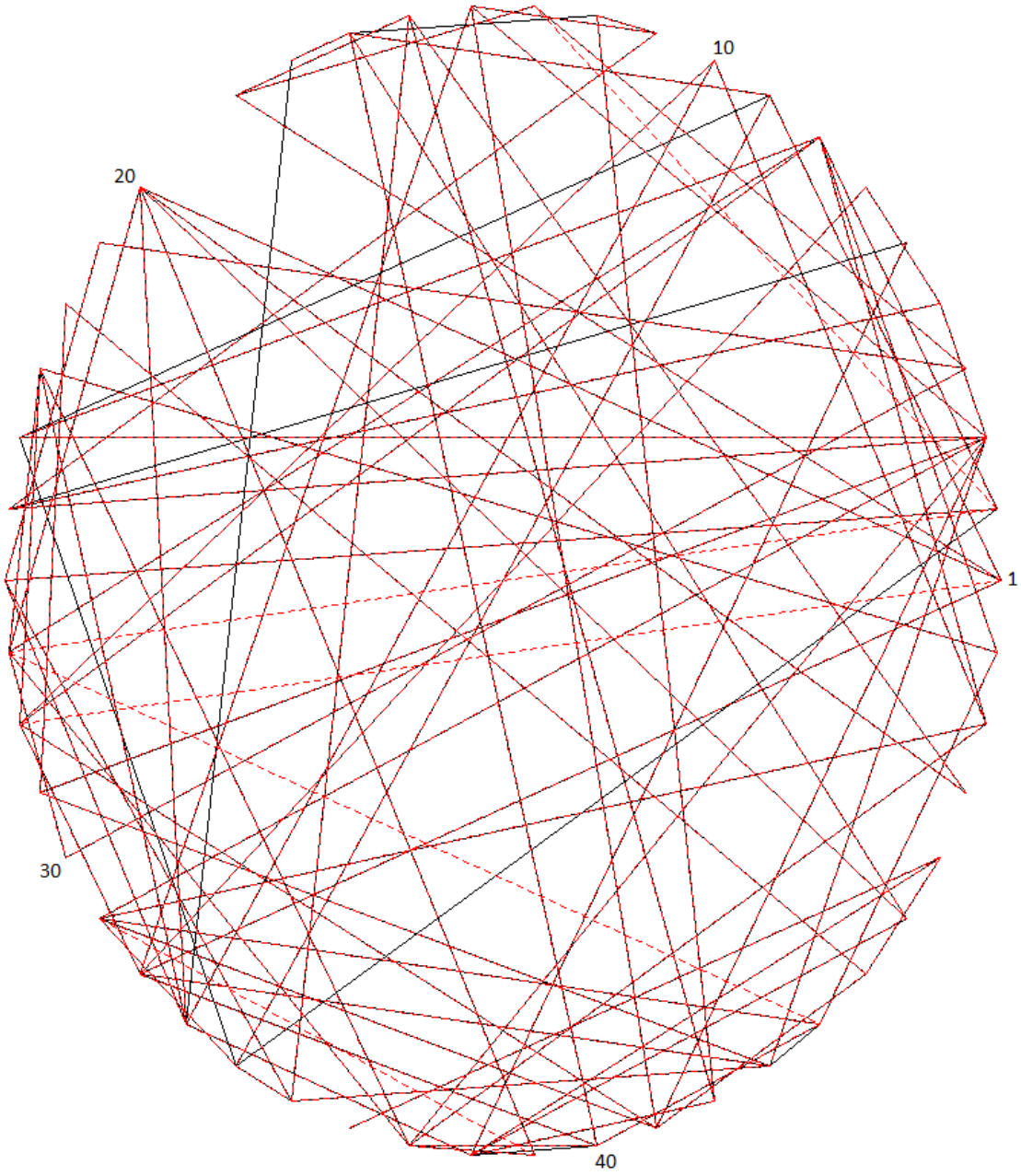


Figure 5.1. this is the theoretical and the estimated correlations for the graphical model in simulation 2. The black solid line are the edges of the theoretical graph, the red dashed lines are the estimated ones using $\widehat{\Theta}_{(2,0.04)}$ and 0.05 as threshold, i.e. there is a black solid line from variable X_i to X_j if $\Theta_{i,j} > 0$, there is a red dashed line if $[\widehat{\Theta}_{(2,0.04)}]_{i,j} \geq 0.05$. As we see, the errors in the estimate are not too much, considering that some true correlations are ± 0.1. We have 8 false negative (true dependence not estimated) in 94 correlations and 5 false positive.

as the normal with some more edges with high correlations, more precisely the outliers observations are generated from a matrix with the same 81 correlations as the normal, plus 9 high correlations, set to 0.6.
The graph is really sparse, the "normal" has 81 edges where the maximum is 19800, the "wrong" one has 90 edges.
By cross validation (the procedure is really slow), the best values for the parameters seems $(G, \rho) =$ (2,0.2). This values, for the estimation of Θ, lead to a better results than using the empirical covariance matrix, the norm ratio

$$\frac{\|\widehat{\Theta}_{(2,0.2)} - \Theta\|}{\|\widehat{\Theta}_{MLE} - \Theta\|}$$

are 0.47, 0.21, 0.81 and 0.37 respectively for Frobenius, infinity, induced l_2 and nuclear norm, but this results are because the low values (there are a lot because the matrix is really sparse) are shrinked to 0, so the numerator in the norm ratio is low because the low values balance the bias in the high values.
If we check the theoretical correlations and the estimates we see that the values where $|\Theta_{i,j}| = 0.4$ have an average estimates (all the elements are taken in absolute values) of 0.23 for $\widehat{\Theta}_{(2,0.2)}$ and 0.39 for $\widehat{\Theta}_{MLE}$, the values where $|\Theta_{i,j}| = 0.3$ has 0.12 and 0.31 respectively for $\widehat{\Theta}_{(2,0.2)}$ and $\widehat{\Theta}_{MLE}$, the values where $|\Theta_{i,j}| = 0.2$ are 0.03 and 0.20, the outliers (the original correlation was 0.6) have values 0.11 for $\widehat{\Theta}_{(2,0.2)}$ and 0.24 for $\widehat{\Theta}_{MLE}$.
In this case the maximum likelihood estimator (with an opportune threshold to cut to 0 the lower values) is better than the magging estimator to eliminate the effect of the outliers. $\widehat{\Theta}_{(2,0.2)}$ give an estimation of the erratic correlations which is similar to the estimation when the true correlations is 0.3, instead $\widehat{\Theta}_{MLE}$ shrink the outliers as an estimated correlation of 0.25 (which is the estimation when the true correlation are 0.25).
Look at as magging/maximin shrink the coefficients to 0 we see that the more the coefficient as low, the more is shrinked (0.4 is shrinked to 0.23, 0.2 is shrinked to 0.03), so this method can be useful when the outliers haven't high correlations.
The previous simulation was better for maximin estimator than the last one, the outliers correlations was more (almost three times the right one) but not elevated (were from 0 to 0.1 in absolute values, instead the right correlations were from 0.1 to 0.4 in absolute values).
This is a behaviour similar to the lasso, which is used in high dimension because shrink a lot the low coefficients and keep the moderate/high (see section 11 of Tibshirani - 1996 [17]), the penalty function doing the same with the outliers, so maximin estimation with l_1 norm works well when we have moderate/high true correlations (which are not shrinked too much) and a large number of erratic low correlations.

Chapter 6

Conclusion

In this thesis, starting from maximin effect for linear model [14], I have given a possible extension to gaussian graphical model, focusing in the correlations between the variables. This lead to a precision matrix where the (i, j)-th element is the maximin linear correlation effect between the variables X_i and X_j.
In the second chapter is given the corresponding estimation, following the approach of the graphical lasso [6] as extension to the lasso for linear model [17]. However, the optimization is computationally problematic because we have to solve it using a derivative free method, which is really slow and, in our case, has some problems to converge. I tried to solve it using an approximation, which works well in the linear model case, but definitivly not in graphical models, so I have extended the idea of magging estimation (introduced for the maximin estimation for linear model in high dimension [4]) to graphical model also in low/moderate dimension. Magging estimation works really well in our case.
The ways to implement the optimization are showed in chapter 3, where there are also a brief explanation of all the optimization methods used in the thesis.
In chapter 4 there is an improvement of the algorithm which can be used in high dimension, that follows the improvement for the graphical lasso algorithm [19]. There is also a possible way to implement cross validation in two dimension (the parameters that have to be learned from the data are the number of groups G and the penalty ρ), in case of graphical model there is not a response variable, so we evaluate the prediction ciclically, fixing a variable used as response, with the others as explicative, the method is really slow in high dimension.
Finally, in chapter 5, there are two simulations in which maximin works well, compared to the maximum likelihood estimation, and a estimation when does not.

For the future, can be evaluated the proprieties of the maximin estimation using different penalty, such as ridge or other extensions (see [8]), In fact we have seen in the simulations that maximin with l_1-norm shrink to 0 the coefficients in a similar way of the lasso estimation, which can be not good in some cases (like in the third simulation, in chapter 5).
Also, can be useful derive the function that the estimator maximize, because, if is function of the likelihood, maximin effect can be extend to generalized linear model. Moreover in a bayesian approach, as the lasso correspond to the posterior mode when the coefficients are indipendent with prior Laplace distributions, can be derived the prior information which lead to a posterior mode equal to the maximin estimator.
Find the function maximized by the estimator is useful especially in graphical model, to derive some analytically methods to evaluate the prediction error (such AIC or BIC) to replace, or at list improve the cross validation, that is really slow.

Appendix A

A.1 Proof of Theorem 3

Without loss of generally we can proof the theorem in the simpliest case, with 2 blocks.
Using the fact that the inverse of a block diagonal matrix is another block diagonal matrix, evaluated the inverse of each block,

$$\Theta = \begin{pmatrix} \Theta^{(1)} & \\ & \Theta^{(2)} \end{pmatrix} \implies \Sigma = \begin{pmatrix} \Sigma^{(1)} & \\ & \Sigma^{(2)} \end{pmatrix}$$

and the estimations of Σ are:

$$S = \begin{pmatrix} S^{(1)} & \\ & S^{(2)} \end{pmatrix}, \quad W = \begin{pmatrix} W^{(1)} & \\ & W^{(2)} \end{pmatrix}.$$

The equation

$$W_{11} \cdot \arg\min_{\beta} \left\{ \frac{1}{2} \max_{g} \left(\beta^T W_{11} \beta - 2\beta^T s_{12,g} \right) + \rho \|\beta\|_1 \right\}$$

become

$$\begin{pmatrix} W^{(1)} & \\ & W^{(2)} \end{pmatrix}_{11} \cdot \arg\min_{\beta} \left\{ \frac{1}{2} \left({\beta^{(1)}}^T {\beta^{(2)}}^T \right) \begin{pmatrix} W^{(1)} & \\ & W^{(2)} \end{pmatrix}_{11} \begin{pmatrix} \beta^{(1)} \\ \beta^{(2)} \end{pmatrix} + \right.$$
$$\left. - \min_{g} \left(\left({\beta^{(1)}}^T {\beta^{(2)}}^T \right) \begin{pmatrix} 0 \\ s^{(2)}_{12,g} \end{pmatrix} \right) + \rho \left(\|\beta^{(1)}\|_1 + \|\beta^{(2)}\|_1 \right) \right\}$$

which is equal to

$$\begin{aligned} &\begin{pmatrix} W^{(1)} & \\ & W^{(2)} \end{pmatrix}_{11} \cdot \arg\min_{\beta} \left\{ \frac{1}{2} \left({\beta^{(1)}}^T W^{(1)} \beta^{(1)} + {\beta^{(2)}}^T W^{(2)}_{11} \beta^{(2)} \right) + \min_{g} {\beta^{(2)}}^T s^{(2)}_{12,g} + \rho(...) \right\} \\ &= \begin{pmatrix} W^{(1)} & \\ & W^{(2)} \end{pmatrix}_{11} \cdot \arg\min_{\beta^{(1)}, \beta^{(2)}} \left\{ f\left(\beta^{(1)}\right) + g\left(\beta^{(2)}\right) \right\} \end{aligned} \tag{A.1}$$

where

$$\begin{aligned} f(x) &= \frac{1}{2} x^T W^{(1)} x + \rho \|x\|_1 \\ g(x) &= \frac{1}{2} x^T W^{(2)}_{11} x + \min_{g} x^T s^{(2)}_{12;g} + \rho \|x\|_1 \end{aligned}$$

The former minimization can be solved separately for $\beta^{(1)}$ and $\beta^{(2)}$:

$$\widehat{\beta^{(1)}} = \arg\min_{\beta^{(1)}} \left\{ f\left(\beta^{(1)}\right) \right\} = 0$$
$$\widehat{\beta^{(2)}} = \arg\min_{\beta^{(2)}} \left\{ g\left(\beta^{(2)}\right) \right\}$$

where the first is equal to 0 because the function f is always positive, except in 0, in which f is equal to 0. So (A.1) become

$$\begin{pmatrix} W^{(1)} & \\ & W^{(2)} \end{pmatrix}_{11} \begin{pmatrix} 0 \\ \widehat{\beta^{(2)}} \end{pmatrix} = \begin{pmatrix} 0 \\ W_{11}^{(2)} \widehat{\beta^{(2)}} \end{pmatrix}$$

this conclude the proof: the optimization for the last column, if Θ is block diagonal does not depend from the first block.

A.2 Proof of Theorem 4

the equation

$$W_{11} \cdot \arg\min_{\beta} \left\{ \frac{1}{2} \max_{g} \left(\beta^T W_{11} \beta - 2\beta^T s_{12,g} \right) + \rho \|\beta\|_1 \right\}$$

using the fact that $\min_g \beta^T s_{12;g} = \beta^T s_{12;g*}$ for some unknown group $g* \in \{1, ..., G\}$, can be write as:

$$W_{11} \cdot \arg\min_{\beta} \left\{ \frac{1}{2} \beta^T W_{11} \beta - \beta^T s_{12,g*} + \rho \|\beta\|_1 \right\}$$

this is the usual equation of the graphical lasso problem.
Theorem 2 in [19] prove that $|S_{i,j;g*}| \leq \rho \implies W_{i,j;glasso} = 0$.
Considering that $g*$ is in $\{1, ..., G\}$, if $|S_{i,j;g}| \leq \rho \; \forall g$, i.e. $\max_g |S_{i,i';g}| \leq \rho$, than $W_{i,j;maximin} = 0$.

Appendix B

In this chapter are showed some codes of the algorithms in chapter 3 and 4.
Some modifications of the codes are write as comments.

B.1 Sampling groups

If the groups of variables are unknown we sample from the dataset `X`:

```
n<-nrow(X)
G<-50
ng<-trunc(n/G)
# ng<-15
X_gr<-array(dim=c(G,ng,p))
# S_gr<-array(dim=c(G,p,p))

for(i in 1:G) {
  sample_values<-sample((1:n),size=ng)
  X_gr[i,,]<-X[sample_values,]
#  S_gr[i,,]<-t(X[sample_values,])%*%X[sample_values,]/ng
}
```

The data are organized in array, sampled without replacing inside a group, with replacing bethween groups, like in [14]. If we need also the empirical covariance matrix for the groups we can remove the comments in rows 6 and 11.
The current estimation W start as the sum of the empirical covariance S and the matrix ρI_p, because is not singular also in the case in which $n < p$:

```
S<-t(X)%*%X/n
rho<-0.3
W<-S+rho*diag(p)
```

B.2 Optimization function

Here is showed the cycle which use the algorithm in section 3.1 for the optimization, the function `nmk` is in the library `dfoptim` [18].

```
obj<-function(beta,W,X,pos=1,rho) {
  vett<-rep(NA,times=G)
  for(i in 1:G)  vett[i]<- t(beta)%*%t(X[i,,-pos])%*%X[i,,pos]
  t(beta)%*%W[-pos,-pos]%*%beta + -(2/ng)*min(vett) +rho*sum(abs(beta))
```

```
}

cycle.nmk<-function(Start,W,X,rho) {
  cl<-rep(NA,ncol(W))
  for(i in 1:p) {
    a<-nmk(par=solve(Start[-i,-i])%*%Start[-i,i], fn=function(x)
           obj(x,W,X,pos=i,rho))
    cl[i]<-a$convergence
    W[-i,i]<<-W[i,-i]<<-W[-i,-i]%*%a$par
#    print(i)
  }
  return(cl)
}
```

The optimization using l_0-norm (section 3.3) is done by the cycle

```
cycle.ire<-function(WP,X,rho,zeta=0.01,tol=.001,maxit=100) {
  G<-dim(X)[1]
  ng<-dim(X)[2]
  p<-dim(X)[3]
  n<-G*ng
  X.tot<-{}
  c.p<-array(NA,dim=c(G,p-1,p))
  conv<-rep(NA,p)
  for(i in 1:G) {
    X.tot<-rbind(X.tot,X[i,,])
    for(j in 1:p) c.p[i,,j]<-t(X[i,,-j])%*%X[i,,j]
  }
  for(j in 1:p) {
    P<-diag(n)
    par<-rep(1,p-1)
    parm1<-rep(0,p-1)
    i<-1
    while(max(abs(par-parm1))>tol & i<=maxit) {
      parm1<-par
      par<-lassoshooting(XtX=n*WP[-j,-j],XtY=t(X.tot[,-j])%*%P%*%X.tot[,j],
                         lambda=rho)$coefficients
      Pgr<-sapply(1:G,function(x) t(par)%*%c.p[x,,j]/ng)
      Pgr<-(Pgr-t(par)%*%WP[-j,-j]%*%par*rep(1,G))^(zeta-1)
      for(k in 1:n) P[k,k]<-Pgr[(k-1)%/%G+1]
      i<-i+1
    }
    conv[j]<-i
    WP[j,-j]<<-WP[-j,j]<<-WP[-j,-j]%*%par
  }
  return(conv)
}
```

For this function is needed the library `lassoshooting` [1].
The array `c.p` has all the cross products of the dataset, we create it because in this way we need to evaluate the various cross products one time.
Here we need all the datasets in a matrix, we join the groups as $(X_1^T, X_2^T, ..., X_G^T)^T$ in row 10. This is useful also in row 24: as we know that the dataset (`X.tot`) is ordered by consecutive groups we can easily put in the matrix `P` the correspondent weights in `Pgr`.
Here the vector `conv` has the number of iterations to optimize each dimension of the matrix, we

have reached convergence if all the elements in `conv` are lower than `maxit`.
Finally, the magging optimization is:

```
cycle.magging<-function(W,X,rho) {
  A<-rbind(rep(1,G),diag(1,G))
  b<-c(1,rep(0,G))
  d<-rep(0,G)
  B<-matrix(NA,p-1,G)
  for(i in 1:p) {
    for(g in 1:G) B[,g]<-as.vector(glmnet(X[g,,-i],X[g,,i],
                                family="gaussian",lambda=rho,intercept=F)$beta)
    H<-t(B)%*%W[-i,-i]%*%B+rho*diag(G)
    w<-solve.QP(H,d,t(A),b,meq=1)$solution
    W[-i,i]<<-W[i,-i]<<-W[-i,-i]%*%apply(B%*%diag(w),1,sum)
#    print(i)
  }
}
```

This function needs the libraries `glmnet` [7] and `quadprog` [3].

B.3 Estimation

Now we show for the magging how obtain the estimate W, we need to repeat the cycle until convergence

```
Wminus1<-W-.5
i<-1
while(max(abs(W-Wminus1))>0.01 & i<=100) {
  Wminus1<-W
  cycle.magging(W,X_gr,rho)
#  print(max(abs(W-Wminus1)))
  i<-i+1
}

Theta.est<-diag(diag(solve(W))^-.5)%*%solve(W)%*%diag(diag(solve(W))^-.5)
# View(round(solve(W),2))
View(round(Theta.est,2))
```

In this case the tollerance is fixed as 0.01.
After the estimation W of $\Sigma_{maximin}$, we can obtain the estimation of the precision matrix $\Theta_{maximin}$ as W^{-1}. We are intrested to the correlations between the variables, so we trasform this matrix as a correlation matrix.

B.4 Dimensionality reduction

Here there are an example of how implement the algorithm in section 4.1.

```
Atot<-partial<-A
for(i in 2:p) {
  partial<-partial%*%A
  Atot<-Atot + partial
}
Atot<-Atot+diag(p)
```

and to extract the groups from this matrix:

```
groups<-matrix(0,p,p)
remaining.var<-(1:p)

while(length(remaining.var)>0) {
  row<-min(which(apply(groups,1,sum)==0))
  gr<-which(Atot[remaining.var[1],]>0)
  groups[row,]<-c(gr,rep(0,times=(p-length(gr))))
  for(j in 1:length(gr)) {
    remaining.var<-remaining.var[-which(remaining.var==gr[j])]
  }
}
```

B.5 Cross validation

The function to implement the algorithm in section 4.2 is

```
magging.cv<-function(X,rep,rho,groups,g,pp,normpar=c(1,2),tol=0.01,
                     maxit=100) {
  n<-nrow(X)
  p<-ncol(X)
  error<-array(NA,dim=c(rep,length(groups),length(rho)))

  for(f in 1:rep) {

    Z<-sample(1:n,trunc(n/2))

    ng<-trunc(trunc(n/2)/g)
    X2<-array(NA,c(g,ng,p))
    for(i in 1:g) X2[i,,]<-X[-Z,][sample(1:trunc(n/2),ng),]

    for(G in groups) {
      nG<-trunc(trunc(n/2)/G)
      X1<-array(NA,c(G,nG,p))
      for(i in 1:G) X1[i,,]<-X[Z,][sample(1:trunc(n/2),nG),]

      for(rh in rho) {
        W<-t(X[Z,])%*%X[Z,]/trunc(n/2)+rh*diag(p)
        Wminus1<-W-.5
        j<-1
        while(max(abs(W-Wminus1))>tol & j<=maxit) {
          Wminus1<-W
          W<-cycle.magging(W,X1,rh,G)
          j<-j+1
        }
        Th<-diag(diag(solve(W))^-.5) %*% solve(W) %*% diag(diag(
            solve(W))^-.5)

        hatX<-obsX<-matrix(0,ng,p)

        for(i in pp) {
          hatbeta<- -Th[-i,i]
```

```
          gmin<-which.min(sapply(1:g,function(x) expvar(hatbeta,X2[x,,-i],
                                                        X2[x,,i])))
          hatX[,i]<-X2[gmin,,-i]%*%hatbeta
          obsX[,i]<-X2[gmin,,i]
        }

        error[f,which(groups==G),which(rho==rh)]<-matrix.norm(induced=normpar[1],
          q=normpar[2],hatX-obsX)
      }
    }

  print(f)
  }
  return(list(meanerror=apply(error,c(2,3),mean),
              maxerror=apply(error,c(2,3),max)))
}
```

where the parameters are the dataset `X`, `rep` is how many times we repeat the procedure with different training and test set, to obtain better estimations of the errors, `rho` is now a vector of possible values for the penalty ρ, `groups` is a vector of possible values for the number of groups, `g` is the number of groups from the test sample in which evaluate the explained variance, `pp` is the parameters in which evaluate the prediction error, `norm` are the parameters to choose the matrix norm to evaluate the prediction error, `tol` and `maxit` are the parameters for the convergence of the optimization cycle.
Is used a function to evaluate some possible matrix norms:

```
matrix.norm<-function(induced=1,q,A) {
  if(induced==1) {
    if(q==1) return(max(apply(abs(A),2,sum)))
    if(q=="inf") return(max(apply(abs(A),1,sum)))
    if(q==2) return(max(svd(A)$d))
  }
  else {
    if(q=="max") return(max(abs(A)))
    else return((sum(svd(A)$d^q))^(1/q))
  }
}
```

which evaluate some matrix norms defined in section 4.2, particularly the induced l_1, l_∞, l_2 - norms if the parameter `induced` is equal to 1, instead the function evaluate the Schatten l_q-norm (set `q=2` for the Frobenius norm, `q=1` for the nuclear norm). If `induced` is not equal to 1 and `q="max"` is evaluate the norm $\|A\|_{max} = \max_{i,j}(a_{ij})$ (it is not a Schatten norm).
In `magging.cv` is used also a function to evaluate the quantities (2.11):

```
expvar<-function(beta,X,Y) {
  1/nrow(X)*(2*t(beta)%*%t(X)%*%Y-t(beta)%*%t(X)%*%X%*%beta) }
```

and a function for the cycle slightly different than the one in section B.2

```
cycle.magging<-function(W,X,rho,G) {
  WW<-W
  A<-rbind(rep(1,G),diag(1,G))
  b<-c(1,rep(0,G))
  d<-rep(0,G)
  B<-matrix(NA,p-1,G)
```

```
  for(i in 1:p) {
    for(g in 1:G) B[,g]<-as.vector(glmnet(X[g,,-i],X[g,,i],
                                family="gaussian",lambda=rho,intercept=F)$beta)
    H<-t(B)%*%WW[-i,-i]%*%B+rho*diag(G)
    w<-solve.QP(H,d,t(A),b,meq=1)$solution
    WW[-i,i]<-WW[i,-i]<-WW[-i,-i]%*%apply(B%*%diag(w),1,sum)
  }
  return(WW)
}
```

Bibliography

[1] Tobias Abenius. *lassoshooting: L1 regularized regression (Lasso) solver using the Cyclic Coordinate Descent algorithm aka Lasso Shooting*. R package version 0.1.5-1. 2012. URL: `http://CRAN.R-project.org/package=lassoshooting`.

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

[3] S original by Berwin A. Turlach R port by Andreas Weingessel. *quadprog: Functions to solve Quadratic Programming Problems.* R package version 1.5-5. 2013. URL: `http://CRAN.R-project.org/package=quadprog`.

[4] Peter Bühlmann and Nicolai Meinshausen. "Magging: maximin aggregation for inhomogeneous large-scale data". In: *Annals of Statistics* (2014).

[5] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. *glasso: Graphical lasso- estimation of Gaussian graphical models*. R package version 1.8. 2014. URL: `http://CRAN.R-project.org/package=glasso`.

[6] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. "Sparse inverse covariance estimation with the graphical lasso". In: *Biostatistics* 9.3 (2008), pp. 432–441.

[7] Jerome Friedman et al. *glmnet: Lasso and Elastic-Net Regularized Generalized Linear Models*. R package version 2.0-2. 2015. URL: `http://CRAN.R-project.org/package=glmnet`.

[8] Jerome Friedman et al. "Pathwise coordinate optimization". In: *The Annals of Applied Statistics* 1.2 (2007), pp. 302–332.

[9] Alan Genz et al. *mvtnorm: Multivariate Normal and t Distributions*. R package version 1.0-3. 2015. URL: `http://CRAN.R-project.org/package=mvtnorm`.

[10] Donald Goldfarb and Ashok Idnani. "A numerically stable dual method for solving strictly convex quadratic programs". In: *Mathematical programming* 27.1 (1983), pp. 1–33.

[11] Vincent Goulet et al. *expm: Matrix Exponential*. R package version 0.999-0. 2015. URL: `http://CRAN.R-project.org/package=expm`.

[12] Harald E Krogstad. *Quadratic Programming Basics*. 2012. URL: `http://www.math.ntnu.no/~hek/Optimering2012/QPbasics2012.pdf`.

[13] Nicolai Meinshausen and Peter Bühlmann. "High-dimensional graphs and variable selection with the lasso". In: *The annals of statistics* (2006), pp. 1436–1462.

[14] Nicolai Meinshausen, Peter Bühlmann, et al. "Maximin effects in inhomogeneous large-scale data". In: *The Annals of Statistics* 43.4 (2015), pp. 1801–1830.

[15] John A Nelder and Roger Mead. "A simplex method for function minimization". In: *The computer journal* 7.4 (1965), pp. 308–313.

[16] Mee Young Park and Trevor Hastie. "L1-regularization path algorithm for generalized linear models". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 69.4 (2007), pp. 659–677.

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

[18] Ravi Varadhan et al. *dfoptim: Derivative-free Optimization*. R package version 2011.8-1. 2011. URL: `http://CRAN.R-project.org/package=dfoptim`.

[19] Daniela M Witten, Jerome H Friedman, and Noah Simon. "New insights and faster computations for the graphical lasso". In: *Journal of Computational and Graphical Statistics* 20.4 (2011), pp. 892–900.

[20] Elizabeth Lai Sum Wong. "Active-set methods for quadratic programming". UC San Diego, 2011.