UNIVERSITY OF TARTU

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Institute of Computer Science

**Nikita Shipilov**

# Detecting influential transcription factors using linear models

**Master's thesis (30 EAP)**

Supervisor: Konstantin Tretyakov, M.Sc.

Autor: …… "….." mai 2010

Juhendaja: ……. "….." mai 2010

TARTU 2010

# Contents

# Introduction

Biological cell is the fundamental unit of life. The cell was first discovered in 1665 by Hooke, who examined cork material through a microscope. The cell theory arose in 1839 with the statement by Schleiden and Schwann that all organisms are composed of one or more cells, which perform the vital functions of an organism, and that all cells contain the hereditary information, which is necessary for regulation of these functions. Further scientific research of the cell nature has led to the generalization that it is a complex and a highly tuned mechanism able to carry out a lot of reactions. Understanding of the processes taking place in the cell underlies a set of biological sciences, such as physiology, genetics, molecular biology etc.

According to the current biological knowledge, a cell life cycle is regulated mainly by protein molecules. The structure of a certain protein determines the function it performs. Proteins are encoded by the particular coding regions (genes) of the DNA molecules stored in the cell. Interestingly, every cell in an organism has the same DNA, but different cell types of an organism produce different sets of proteins. This happens because of the fact that different genes are active among the differently specialized groups of cells. In particular, there are proteins the function of which consists in influencing the production of other proteins. Such proteins are known as transcription factors. They are part of the cell regulation machinery which makes decisions concerning when and what amount of proteins to produce.

With the recent development of the high throughput DNA microarray technology, it became possible to measure the levels of gene activity on a large scale. In a single experiment microarray technology allows one to make snapshots of all genes in a cell. It helps to identify which genes are active in the particular conditions, and this way to discover the cellular processes they are involved in. The data collected from a microarray usually requires sophisticated analysis involving biological knowledge and the application of statistical techniques.

In this work we address the problem of inferring ‘influential’ transcription factors from microarray data using linear models. Linear models are easy to understand and are able to produce interpretable solutions. However, microarray data is typically high-dimensional and contains noise and measurement errors. This makes the application of straightforward linear

modeling techniques tricky and often unsatisfactory. We observe the state-of-the-art methods for solving linear regression problems and their application to our biological problem. Besides the classical Least Squares linear regression, we consider such state-of-the-art approaches as ridge regression, the LASSO and the ElasticNet.

The work is organized as follows:

- we introduce the basic biological and statistical concepts needed to understand this work, and formulate the goal of this work in Chapter 2;
- we give an overview of the techniques we use in our research in Chapter 3;
- in Chapter 4 we analyze performance of the presented techniques on artificial data;
- in Chapter 5 we describe the test results obtained by applying the considered methods on real microarray data.

# Chapter 2

# Background

In this chapter we introduce the general biological and statistical notions that are used further in the text. We present the basics of the cell theory and describe the microarray technology in Section 2.1. We introduce the basic concepts of linear modeling in Section 2.2 and formulate the problem of our current research in Section 2.3.

## 2.1 Biological Background

The biological cell is classified to be the basic unit of life. It is the building block for all living organisms. The smallest organisms, such as bacteria, consist of a single cell, while larger organisms, such as plants and animals, are *multicellular* with the number of cells up to several hundred of trillions. Cells in a multicellular organism are differently specialized. It means they carry out different biological functions, in cooperation providing the complete lifecycle of an organism.

A group of cells specialized for a particular function is a *tissue*. There are three basic types of plant tissues: epidermis, vascular and ground; and animal tissue types are: connective, muscle, nervous and epithelial. Each type is represented by a variety of tissues, which differs depending on the classification of the organism. Besides the functional specialization, cells also differ by their physical characteristics. For example, the single nerve cell that is known as a *pseudounipolar nerve* may reach from the toe to the spinal cord in the human body, while a drop of blood has more than 10000 cells in it.

Imaging how many various organisms there are in the world, it is clear that the variety of cell types is tremendous. However, almost all of them have the same structure and they are similar in their activities. Formally, a cell is enclosed in a membrane, filled with a cytoplasm, and contains a material and mechanisms for translating genetic messages into protein molecules. The cell components are called *organelles*.

Proteins make a work of organelles possible. Being primary components in life cycle of a cell, they play many different roles. For example, proteins promote chemical reactions performed in a cell; they provide the structural and infrastructural supports holding an organism together; they produce, transform and translate energy into physical work in muscles; they act like sensors, detectors and protectors from malicious elements, creating an immune response; and have many more other functions.

All proteins are constructed from linear sequences of amino acids. The number of amino acids in a chain may vary from $100$ to $5000$. The structure of the chain determines the biochemical activity and the function of the protein. One of the major goals in molecular biology is trying to predict the properties of the protein according to its amino acid sequence. Unfortunately, this remains an unsolved problem yet.

### 2.1.1 The DNA

The exact order of amino acids in a protein is encoded by the *deoxyribonucleic acid* molecule (the *DNA*). With the *ribonucleic acid* (*RNA*) DNA form the genetic material of a cell. These are polymers of four nucleic acids, called *nucleotides*. Each nucleotide consists of a base molecule (a *purine* or a *pyrimidine*), sugar (deoxyribose in DNA and ribose in RNA) and phosphate groups. The purine nucleotides are *adenine* ($\mathrm{A}$) and *guanine* ($\mathrm{G}$); the pyrimidines are *cytosine* ($\mathrm{C}$) and *thymine* ($\mathrm{T}$). DNA molecule usually contains sequences of thousands of nucleotides, and is generally presented by the first letters of their bases, for example, $\mathrm{GATATCC}$.

DNA consists of two complementary strands of nucleotides which are bound together with phosphodiester bonds. Each nucleotide from a strand is linked to the nucleotide from the opposite strand directionally. This means there is the one-to-one mapping between the nucleotides that are at the same places in the sequences. Adenines are bound exclusively with thymines ($\mathrm{A}-\mathrm{T}$) and guanines are bound exclusively with cytosines ($\mathrm{G}-\mathrm{C}$). So, knowing the order of nucleotides in a strand, its complementary strand is totally determined. In *eukaryotic* cells (cells multicellular organisms are usually built of) bounded DNA strands form a spiral shape; in bacteria (*prokaryotic* cells) this shape is generally circular.

The primary role of nucleotides is to carry the encoding of proteins. A triplet of nucleotides, called a *codon*, corresponds to a certain amino acid. A series of subsequent codons encodes a

particular protein. For example, alanine[i] is presented in DNA by the codons GCT, GCC, GCA and GCG. There are also *start* and *stop* codons, which define regions in the DNA for coding the exact proteins. These coding regions are referred to as *genes*.

Interestingly, that every cell in the organism contains exactly the same DNA and exactly the same set of genes, respectively. However, the different specialization of cells implies they produce different proteins. Besides, under various environmental conditions the cell may change the set of proteins it synthesizes. Thus, the difference lies in the regulation of the genetic machinery. This stays the major research direction in the molecular biology to be able to predict which genes and under what circumstances are active in a cell.

### 2.1.2 Gene Expression

The process of producing proteins according to genes takes three basic steps: *transcription*, *splicing* and *translation*. The transcription begins with a molecular complex called *RNA-polymerase* binding to the segment of the DNA. The polymerase copies the portion of the DNA into the complementary RNA molecule. RNA is single-stranded and contains the *uracil* nucleotide instead of thymine, hence, the copying is performed according to the rules $(A - U)$ and $(G - C)$.

The polymerase determines where to bind on the DNA according to signals, which detect a part of the DNA near the beginning of the coding region of a protein. This region before a gene is referred to as *promoter*. The polymerase copies the DNA from the promoter until a certain stop codon is achieved. It then releases the RNA, containing exactly the same encoding as that on the corresponding segment of the DNA.

While splicing a molecular complex called *spliceosome* removes parts of the RNA, producing the 'prepared' RNA to be translated to the protein. This RNA is known as *messenger-RNA* (shortly *mRNA*). In the end of the splicing process the mRNA is transported from the nucleus (the cell kernel) to the cytoplasm, where it binds to a *ribosome*.

The *ribosome* (the complex combination of RNA and proteins) uses the mRNA as the blueprint for the production of the corresponding protein. This process is known as translation.

[i] Alanine is the α-amino acid which is refered to the group of proteinogenic amino acids – the building blocks of proteins.

The production of proteins from genes is referred to as *gene expression*. It was mentioned above that not all genes in a cell are expressed at once. This process is directed by a finely tuned molecular machinery, which is complex and is not well studied yet by the biology. However, what is known is that gene expression is influenced by the certain collection of proteins in the cell nucleus that define which genes will be expressed and which will not. These proteins are referred to as *transcription factors*.

Transcription factors attach themselves to the specific DNA segments (*motifs*) which border genes. This provokes or prevents the polymerase to bind at these segments, in turn encouraging or inhibiting the transcription of the corresponding genes. Transcription factors perform this function alone or in complex with other proteins. Promoting the polymerase activity transcription factors are known as *activators* and blocking – as *repressors*.

### 2.1.3 Microarray Technology

The *microarray* is a high throughput technology, which provides the possibility to make snapshots of expression of many genes in a cell simultaneously. Microarrays help to figure out the active genes within the cell types, to learn how their expression changes under various environmental conditions and to identify the cellular processes they are involved in.

A microarray is a glass or a polymer slide onto which DNA fragments are attached at fixed locations called *spots*. There are usually thousands of spots on the array, each containing millions of the identical fragments of DNA. The fragments of a spot should correspond to a certain gene only, however, in practice, it is not always possible to collect them so clearly due to the presence of similar genes in the DNA.

Microarray can be used to measure gene expression in different ways. The most typical application is comparing two different samples of the same cell or cell type. The approach is based on labeling the mRNA molecules extracted from each sample with fluorescent dyes. The mRNA molecules from the first sample are labeled with a green dye, and a red dye is used for the mRNA from the second sample.

All the extracted mRNA molecules are reverse-transcribed to create strands of complementary DNA (*cDNA*) . These cDNA molecules are then put on the microarray, where they form the hydrogen bonds with the matching DNA fragments. This action is referred to as *hybridization*.

After exciting the hybridized microarray with a laser, the amount of fluorescence emitted from each spot indicates the amount of the attached cDNA molecules on it. The color of the spot is green if cDNA molecules referred to the sample labeled with a green dye are in abundance, and this color is red if in abundance are cDNA molecules corresponding to the sample labeled with a red dye, respectively. The spot is of yellow color if the amounts of cDNA from different samples are equal, and if neither are present the spot do not fluoresce, hence, is black.

The color of a spot and its intensity demonstrate the expression level of the corresponding gene. The amount of cDNA molecules bound to a spot corresponds to the amount of mRNA produced in a cell for a certain gene, what in turn is the indicator of the expression of this gene.

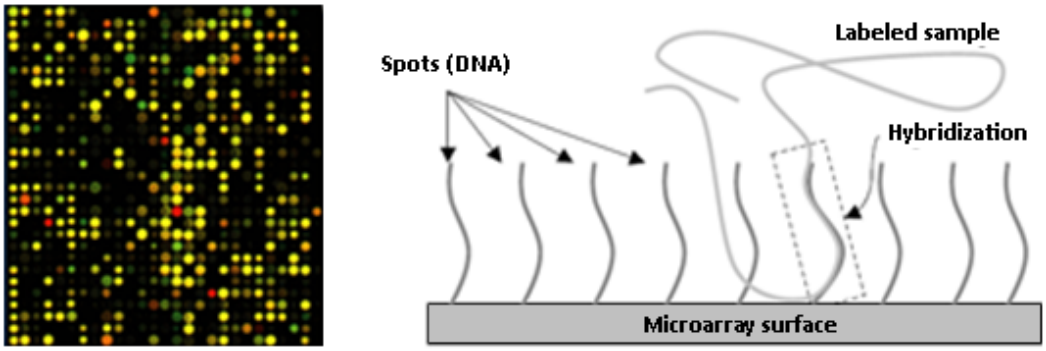


**Figure 2.1**. *A picture of hybridized microarray (left) and the schematic representation of a microarray experiment (right).*

## 2.2 Linear Model

By definition, linear model is a mathematical model in which linear equations connect random variables and parameters. Linear models underlie most of the statistical analyses that are used in the social, scientific or other types of research. They help to explain an impact of the input data on the output.

Linear models are preferred because of their simplicity. It is easier to understand and to interpret them than some other competing models. They can sometimes even outperform more powerful nonlinear models, especially in case of the small number of training cases. Or suppose there is a nonlinear function that is smooth in some region and this region is just of interest. Even then a linear model can give an adequate approximate solution to the stated problem. Moreover, some of nonlinear models can be reduced to the linear form by applying a set of appropriate transformations.

So it is obvious that linear models are the first step in many types of analyses. At least they are able to provide meaningful results that could be a good basis for constructing wider and more complex models.

### 2.2.1 Terminology

Let $x_1, x_2, \dots, x_p$ and $y$ be some quantities. The linear model has the form:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon \,, \tag{2.2.1}$$

where term $\beta_0$ is known as the *intercept*, $\beta_1, \beta_2, \dots, \beta_p$ are the *model parameters* (or *coefficients*) and $\varepsilon$ denotes the noise (or error). It is typically assumed that the error $\varepsilon$ is random and has normal distribution with zero mean and variance $\sigma^2$:

$$\varepsilon \sim N(0, \sigma^2) \,.$$

Variables $x_1, x_2, \dots, x_p$ are referred to as *explanatory variables* and $y$ is the *response variable* or simply *response* (sometimes it is called the *dependent variable*). In case there is a set of $n$ ($n \geq 1$) observations of the response and explanatory variables the linear model has the form:

$$y_i = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} + \varepsilon_i \,, \quad i = 1, \dots, n \,. \tag{2.2.2}$$

The intercept $\beta_0$ is usually included into the set of model parameters, causing an addition of a constant variable 1 to the explanatory variables. This makes possible to represent the set of $n$ equations from (2.2.2) in more compact way using matrix and vector notations[ii]:

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon}\,, \tag{2.2.3}$$

where $\boldsymbol{y}$ is a $n \times 1$ vector containing $n$ observations of the response, $\boldsymbol{X}$ is a $n \times (p+1)$ matrix containing observations of $p$ explanatory variables and a column vector of 1 values, $\boldsymbol{\beta}$ is a $(p+1) \times 1$ vector of model parameters (also containing the intercept) and $\boldsymbol{\varepsilon}$ is a $n \times 1$ vector of errors.

Vector $\boldsymbol{\varepsilon}$ elements are assumed to be pairwise independent (uncorrelated) and also independent of all the explanatory variables from $\boldsymbol{X}$. Besides, it was written before that errors are expected to have zero mean and the common variance $\sigma^2$. These assumptions can be formulated by using $E(\cdot)$ and $\mathrm{Var}(\cdot)$ notations:

$$E(\boldsymbol{\varepsilon}) = \boldsymbol{0}\,, \quad Var(\boldsymbol{\varepsilon}) = \sigma^2 \boldsymbol{I}\,,$$

where $\boldsymbol{I}$ is an $n \times n$ identity matrix. Here $E(\cdot)$ stands for the "expected value" that is the same as the "mean value" in the context of modeling (for detailed explanation see appendices A.2.1 and A.2.2 in [Wei05]). The *mean function* is defined by the notation $E(Y|X)$, which can be interpreted as "the expected value of the response $Y$ when the explanatory variables are fixed at the values of $X$". The value of the function but depends on the problem. The same also holds for the *variance function* $Var(Y|X)$, with the only difference that its value is a fixed quantity representing the expected variance.

The typical problem as it was mentioned before is to figure out how the set of explanatory variables $\boldsymbol{x}$ describes the response $y$, or, in other words, the relationship between $\boldsymbol{x}$ and $y$ is of interest (that is supposed to be linear, of course, in context of linear modeling). In terms of statistics this relationship is formulated through the mean and the variance of the response $y$ (denoted by $E(y)$ and $Var(y)$, respectively). The aim is to figure out how the mean $E(y)$ changes as $\boldsymbol{x}$ is varied, assuming that the variance $Var(y)$ stays unaffected. Since $\varepsilon$ is the only random variable in a model, its assumption about the variance also holds for the response $y$, that is, $Var(y) = \sigma^2$.

[ii] In context of this work matrices and vectors are presented in boldface. Uppercase letters are used for matrices and lowercase letters – for the representation of vectors.

Collecting the data from one or more observations of explanatory and response variables, the task is to estimate the *true* coefficients, which quantify the impact of the explanatory variables on the behavior of the response. Summarizing all the conditions described earlier, the problem can be formulated using the mean and the variance functions in the following way:

$$E(\boldsymbol{y}|\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{\beta}\,, \quad Var(\boldsymbol{y}|\boldsymbol{X}) = \sigma^2\boldsymbol{I}\,. \tag{2.2.4}$$

The representation (2.2.4) is called the *linear regression model*. Vector $\boldsymbol{\beta}$ is then referred to as vector of *regression parameters*. The term 'linear regression' itself has geometrical origin and stands for the technique of *fitting* a straight line to a set of observations of one explanatory variable (in some space). The slope of this line represents the strength of the effect of the explanatory variable on the response (see Figure 2.2).

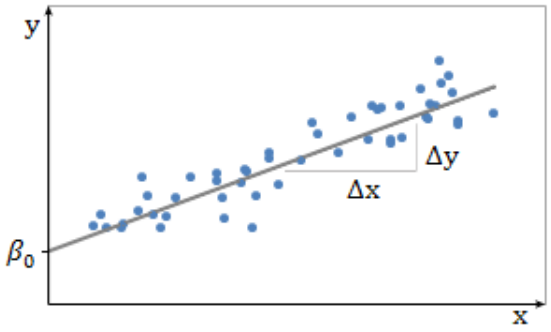


**Figure 2.2**. *Example of linear regression with one explanatory variable. The quantity* $\beta_0$ *(the* $y$ *value when* $x$ *is zero) stands for the intercept and the slope of the regression line is* $\frac{\Delta y}{\Delta x}$.

From the practical point of view, most applications of linear regression fall into two main categories:

- Prediction – given an input vector $\boldsymbol{x}$, make a good prediction of the output $y$ (the prediction is denoted by $\hat{y}$)[iii]. Elements of the vector $\boldsymbol{x}$ are then referred to as *predictor* variables or simply *predictors*.

  After *fitting the model* (obtaining the values of $\widehat{\boldsymbol{\beta}}$) using training datasets of explanatory and response variables, it is assumed that with an additional vector $\boldsymbol{x}$ the model will

[iii] 'Hat' notation is used in this work to represent the estimates. E.g. $\hat{\beta}$ is an estimate of the coefficient $\beta$ and $\hat{y}$ is an estimated prediction of the response $y$ etc.

make a prediction $\hat{y}$ of the output $y$:

$$\hat{y} = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j = \boldsymbol{x}\widehat{\boldsymbol{\beta}} \tag{2.2.5}$$

Such prediction $\hat{y}$ will have an error $\hat{\varepsilon}$ and is called the *residual*. Residual $\hat{\varepsilon} = (y - \hat{y})$ is the difference between the value of the actual response $y$ and the estimated prediction $\hat{y}$. The residual is expected to be as small as possible.

The problem known as *overfitting* occurs when the constructed model gives 'good' results using training data and makes a lot of 'mistakes' in case of the test data. Such overfitted models are meaningful in terms of prediction.

- Descriptive analysis – given a set of explanatory variables $\boldsymbol{x}$ and the response $y$ one may be interested in the quantifying the strength of the relationship between them. This means that after estimating the regression parameters, according to their absolute values it is possible to find the most influential variables – the higher is the value of the coefficient the stronger is the impact of the corresponding variable on the response. Usually, the aim is to find the smallest subset of $\boldsymbol{x}$ having the strongest influence on $y$.

In this work we are mainly interested in 'descriptive analysis'. However, as it will be seen later, the prediction ability of a model and the task to perform a descriptive analysis are highly related.

### 2.2.3 Least Squares Estimation

By far the most popular technique for obtaining the estimates of the regression parameters is the *Least Squares* method. It tries to choose the values of $\boldsymbol{\beta}$ to minimize a quantity called *Residual Sum of Squares* (*RSS*):

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_j\right)^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \, . \tag{2.2.6}$$

In the matrix notation equation (2.2.6) has the form:

$$RSS(\boldsymbol{\beta}) = (\boldsymbol{y} - \boldsymbol{X\beta})^T (\boldsymbol{y} - \boldsymbol{X\beta}) \, .$$

Formally, model parameters obtained by the Least Squares method are:

$$\widehat{\boldsymbol{\beta}}_{LS} = argmin_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X\beta}\|^2 \, ,$$

where $\|\cdot\|$ denotes the *L2-norm* (also known as *Euclidean norm*) of a vector. Vector $\widehat{\boldsymbol{\beta}}_{\boldsymbol{LS}}$ elements are typically referred to as just the *Least Squares estimates*.

$RSS(\boldsymbol{\beta})$ is the quadratic function what means that its minimum always exists, but with a possibility being non-unique. Differentiating the function with respect to $\boldsymbol{\beta}$ and setting it to zero the following set of *normal equations* is got:

$$\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) = 0 \tag{2.2.7}$$

General solution to equation (2.2.7) has the form $(\boldsymbol{X}^T\boldsymbol{X})^-\boldsymbol{X}^T\boldsymbol{y}$, where $(\boldsymbol{X}^T\boldsymbol{X})^-$ is any *generalized inverse* of $\boldsymbol{X}^T\boldsymbol{X}$. If $\boldsymbol{X}^T\boldsymbol{X}$ is nonsingular ($\boldsymbol{X}$ has full column rank), then the unique Least Squares estimates are computed as follows:

$$\widehat{\boldsymbol{\beta}}_{LS} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}\,. \tag{2.2.8}$$

**Theorem 2.1** (*Gauss-Markov theorem*). *Let* $\boldsymbol{y}$ *be a* $(n \times 1)$ *random vector with* $E(\boldsymbol{y}|\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{\beta}$ *and* $Var(\boldsymbol{y}|\boldsymbol{X}) = \sigma^2\boldsymbol{I}$, *where* $\boldsymbol{\beta}$ *is a* $(p \times 1)$ *vector of unknown parameters and* $\boldsymbol{X}$ *is a* $(n \times p)$ *matrix of unknown constants with rank* $p$, $n \geq p$. *If* $l$ *is a vector of constants such that* $\boldsymbol{\theta} = \boldsymbol{l}\boldsymbol{\beta}$ *is a linear parametric function, then Best Linear Unbiased Estimator of* $\boldsymbol{\theta}$ *is* $\widehat{\boldsymbol{\theta}} = \boldsymbol{l}\widehat{\boldsymbol{\beta}}$, *where* $\widehat{\boldsymbol{\beta}}$ *is the vector which minimizes the quantity* $S = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})$.

It follows from the Gauss-Markov theorem (Theorem 2.1) that the Least Squares method produces *unbiased* estimates of the regression parameters that have the lowest variance among all linear unbiased estimates. Consider either the prediction $\widehat{\boldsymbol{\theta}} = \boldsymbol{l}\widehat{\boldsymbol{\beta}}$ of $\boldsymbol{\theta}$. Its *mean squared error* (*MSE*) is:

$$MSE(\widehat{\boldsymbol{\theta}}) = Var(\widehat{\boldsymbol{\theta}}) + [E(\widehat{\boldsymbol{\theta}}) - \boldsymbol{\theta}]^2\,,$$

and it is implied by the theorem that the prediction provided by the Least Squares method has the lowest *MSE* of all other predictions with unbiased parameters. However, this *MSE* is not the minimal and there may exist regression models with *biased* parameters, which cause smaller *MSE*. Such models sacrifice a little of bias in order to reduce variance, what in overall improves the prediction accuracy (see Section 2.2.4).

**Multiple Least Squares.** In typical case regression analysis is applied on data having one output (*single-response*). But as it will be seen later there can be more than one response in

the considered data. Let us suppose there are $r$ ($r \geq 1$) response variables, which form the vector $\boldsymbol{y} = (y_1, y_2, \dots, y_r)$.

A linear model for each of the elements of the vector $\boldsymbol{y}$ is:

$$y_k = \beta_{0k} + \sum_{j=1}^{p} \beta_{jk} x_j + \varepsilon_k \,, \quad k = 1, \dots, r \tag{2.2.9}$$

Here $\beta_{jk}$ is the variable $x_j$ coefficient ($j = 1, \dots, p$) in case of the $k$'th response. With $n$ observations of $p$ explanatory and $r$ response variables the representation of equation (2.2.9) using matrix notation has the following form:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{B} + \boldsymbol{E} \,, \tag{2.2.10}$$

where $\boldsymbol{Y}$ is a $n \times r$ matrix containing $n$ observations of $r$ responses, $\boldsymbol{X}$ is a $n \times (p+1)$ matrix containing observations of $p$ explanatory variables and a column vector of 1 values (because of intercepts $\beta_{0k}, k = 1, \dots, r$), $\boldsymbol{B}$ is a $(p+1) \times r$ matrix of model parameters (each column vector corresponds to a certain response), and $\boldsymbol{E}$ is a $n \times r$ vector of errors.

Defined *RSS* function for $\boldsymbol{B}$ in *multiresponse* setting has then the following form:

$$RSS(\boldsymbol{B}) = \sum_{k=1}^{r} \sum_{i=1}^{n} (y_{ik} - \hat{y}_{ik})^2 = (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{B})^T (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{B}) \,.$$

Note, that there is no relation between the model parameters estimated in case of different response variables. Applying the Least Squares method on multiresponse data simultaneously is the same as computing the estimates for each of the responses separately.

### 2.2.4 Regularization

It was mentioned above that the Least Squares method provides the unique solution if $\boldsymbol{X}^T\boldsymbol{X}$ is nonsingular. But it may happen that $\boldsymbol{X}$ is not of full rank. Rank deficiency occurs if some of the matrix $\boldsymbol{X}$ columns are not linearly independent (*collinear*), or when the number of variables exceeds the number of observations ($p \gg n$ case). In both cases $\boldsymbol{X}^T\boldsymbol{X}$ is singular what makes it hard to invert.

While performing a regression analysis highly correlated variables allow the corresponding coefficients grow in the opposite directions, cancelling each other's effects: for example, if the coefficient of the first variable grows large in the positive direction, then the second coefficient of the second may grow large in the negative direction. When there are many

correlated input variables a *high variance* model is produced that is quite unrealistic (known as *multicollinearity problem*).

Some techniques like *pseudoinverse* are available to overcome the rank deficiency problem, making possible the inverse of singular matrices, but they do not resolve some other issues related to *RSS*. For example, if $\boldsymbol{X}$ is *ill-conditioned* (contains quantities that are sensitive even to small changes, resulting in different regression models) then computational routines while estimating the regression parameters may be *numerically unstable* due to e.g. rounding, cancelation and other operations.

However, despite the criticism, the Least Squares method still gives adequate solutions that are often quite satisfactory. In case of more complex models there is usually a need of better accuracy of predictions and interpretations. The technique referred to as *regularization* unlike Least Squares is used to produce *biased* estimates of regression parameters, improving but the overall result in terms of a model error.

**L2-norm Regularization.** The idea of regularization itself was developed by mathematician Tikhonov A.[iv] who was working on the solution to the numerical instability of matrix inversions. This technique became popular with a computational method proposed by Hoerl and Kennard [HK70] for improving the Least Squares solution. This method is known as *ridge regression.* The idea of ridge regression is to add a small positive constant $\lambda \geq 0$ to the diagonal of the matrix $\boldsymbol{X}^T\boldsymbol{X}$, what makes the matrix nonsingular if it is not of full rank, hence, $\boldsymbol{X}^T\boldsymbol{X}$ becomes invertible. The ridge regression estimates are then computed as follows:

$$\widehat{\boldsymbol{\beta}}_{Ridge} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y} \ , \tag{2.2.11}$$

where $\boldsymbol{I}$ is the $p \times p$ identity matrix and $\lambda \geq 0$ is referred to as *regularization parameter*. If $\lambda = 0$ then $\widehat{\boldsymbol{\beta}}_{Ridge}$ reduces to the Least Squares estimates $\widehat{\boldsymbol{\beta}}_{LS}$. And if $\boldsymbol{X}$ is orthonormal ($\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{I}$) ridge estimates are just *scaled* version of $\widehat{\boldsymbol{\beta}}_{LS}$, that is, $\widehat{\boldsymbol{\beta}}_{Ridge} = \widehat{\boldsymbol{\beta}}_{LS}/(1+\lambda)$.

Ridge regression can be characterized as the method that *penalizes RSS* function and as the method that performs the *continuous shrinkage* of the regression coefficients. Actually, this is the common process because the penalty is applied on the size of coefficients.

[iv] Andrey Tikhonov (1906-1993) is Soviet and Russian mathematician known for important contributions to topology, functional analysis, mathematical physics, and ill-posed problems.

Let us consider the penalized *RSS* function as:

$$RSS(\boldsymbol{\beta}) = (\boldsymbol{y} - \boldsymbol{X\beta})^T(\boldsymbol{y} - \boldsymbol{X\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta} \ , \qquad (2.2.12)$$

where $\lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$ is *the ridge penalty*, respectively. Differentiating equation (2.2.12) with respect to $\boldsymbol{\beta}$ and setting it to zero, the set of normal equations is got, the solution to which has exactly the same form as presented in equation (2.2.11).

The regularization parameter $\lambda$ controls the amount of shrinkage. The larger the value of $\lambda$, the greater is the amount of shrinkage: the coefficients are shrunk towards zero and each other (becoming more similar).

Let us consider the *singular value decomposition* (*SVD*) of the matrix $\boldsymbol{X}$ with $n$ rows and $p$ columns. That is, $\boldsymbol{X} = \boldsymbol{U\Sigma}^{1/2}\boldsymbol{V}^T$, where $\boldsymbol{U}$ is a $n \times n$ *unitary matrix* with the columns representing the *eigenvectors* of $\boldsymbol{XX}^T$, $\boldsymbol{V}$ is a $p \times p$ unitary matrix ($\boldsymbol{VV}^T = \boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}$) with the columns representing the eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$, and $\boldsymbol{\Sigma}$ is a *diagonal matrix* holding the ordered *eigenvalues* of $\boldsymbol{X}^T\boldsymbol{X}$. Thus, $\boldsymbol{X}^T = \boldsymbol{V\Sigma}^{1/2}\boldsymbol{U}^T$, $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{V\Sigma V}^T$ and the ridge estimates from equation (2.2.11) can be rewritten as follows:

$$\begin{aligned}\widehat{\boldsymbol{\beta}}_{Ridge} &= (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y} \\ &= (\boldsymbol{V\Sigma V}^T + \lambda\boldsymbol{VV}^T)^{-1}\boldsymbol{V\Sigma}^{1/2}\boldsymbol{U}^T\boldsymbol{y} \\ &= \boldsymbol{V}(\boldsymbol{\Sigma} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Sigma}^{1/2}\boldsymbol{U}^T\boldsymbol{y} \ ,\end{aligned}$$

where $j$'th element ($j = 1, \dots, p$) of the diagonal matrix $(\boldsymbol{\Sigma} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Sigma}^{1/2}$ is $\Sigma_j^{1/2}/(\Sigma_j + \lambda)$.

In the same way the Least Squares estimates (with some simplifications) are defined as:

$$\widehat{\boldsymbol{\beta}}_{LS} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} = (\boldsymbol{V\Sigma V}^T)^{-1}\boldsymbol{V\Sigma}^{1/2}\boldsymbol{U}^T\boldsymbol{y} = \boldsymbol{V\Sigma}^{-1/2}\boldsymbol{U}^T\boldsymbol{y} \ .$$

Now the estimated outputs $\widehat{\boldsymbol{y}} = \boldsymbol{X}\widehat{\boldsymbol{\beta}}$ for the Least Squares method and ridge regression can be rewritten as:

$$\widehat{\boldsymbol{y}}_{LS} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{UU}^T\boldsymbol{y} \ ,$$

$$\widehat{\boldsymbol{y}}_{Ridge} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{U\Sigma}^{1/2}(\boldsymbol{\Sigma} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Sigma}^{1/2}\boldsymbol{U}^T\boldsymbol{y} = \sum_{j=1}^{p}\boldsymbol{u}_j\frac{\Sigma_j}{(\Sigma_j + \lambda)}\boldsymbol{u}_j^T\boldsymbol{y}$$

If $\lambda = 0$ then the Ridge estimate $\widehat{\boldsymbol{y}}_{Ridge}$ reduces to the Least Squares estimate $\widehat{\boldsymbol{y}}_{LS}$. Otherwise, $\widehat{\boldsymbol{y}}_{Ridge}$ is regularized $\widehat{\boldsymbol{y}}_{LS}$.

Note, that the matrix of explanatory variables $\boldsymbol{X}$ in equation (2.2.11) does not contain the column vector of constants 1 for estimating the intercept. This is an important assumption before performing the regularization that the intercept must be left out from the model. Shrinking the intercept would not result in a shift of predictions by the same amount. The second reason concerns the need of standardizing the inputs before obtaining the estimates. The *centering procedure*, when all observations of explanatory variables are subtracted by their means, is used to reduce the multicollinearity problem without changing the 'significance' of the variables. Hence, it is assumed that the centering of the input matrix $\boldsymbol{X}$ has been done only in case it just contains the observations of the predictors. Then, the intercept $\beta_0$ is set to $\bar{y} = 1/n \sum_{i=1}^{n} y_i$, where $n$ is the number of observations of the response $y$, and the remaining estimates of the coefficients are obtained without the intercept.

In conclusion, we define the *ridge estimates* in the *Lagrangian form* as:

$$\widehat{\boldsymbol{\beta}}_{Ridge} = argmin_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}. \tag{2.2.13}$$

The criterion (2.2.13) using the matrix notation (without the intercept) is presented as:

$$\widehat{\boldsymbol{\beta}}_{Ridge} = argmin_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X\beta}\|^2 + \lambda \|\boldsymbol{\beta}\|^2 . \tag{2.2.14}$$

**L1-norm Regularization.** While L2-norm regularization is effective in increasing the *prediction accuracy* of regression models it does not address another quite important issue related to the Least Squares method – *interpretability problem*. With a large number of predictors usually a smaller subset that has the strongest effect on the output is of interest. But applying the L2-norm penalty on the regression parameters does not encourage their sparsity – all of them are associated with non-zero values.

Sparse models are provided by the *Subset Selection* method, but because of its discrete nature (variables are either added or removed from the model) it produces very high variance models. Promising technique has been proposed by Tibshirani [Tib96] – the *Least Absolute Shrinkage and Selection Operator* (*LASSO*). The LASSO performs the shrinkage of some number of regression coefficients while setting all the other coefficients to zero (see the graphical representation in Figure 2.3). It turns out that the LASSO performs a continuous shrinkage and a variable selection simultaneously. Theoretically, it must help to improve the

prediction accuracy like in case of ridge regression and in addition to fit more interpretable models.

The criterion of the LASSO estimates is defined as:

$$\widehat{\boldsymbol{\beta}}_{LASSO} = argmin_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}. \quad (2.2.15)$$

Leaving the intercept $\beta_0$ out, the criterion (2.2.15) in the matrix form is:

$$\widehat{\boldsymbol{\beta}}_{LASSO} = argmin_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X\beta}\|^2 + \lambda \|\boldsymbol{\beta}\|_1 , \quad (2.2.16)$$

where $\|\cdot\|_1$ denotes the *L1-norm* of a vector.

In the case of the orthogonal design the LASSO solution is:

$$\widehat{\boldsymbol{\beta}}_{LASSO} = (|\widehat{\boldsymbol{\beta}}_{LS}| - \frac{\lambda}{2})_+ sign(\widehat{\boldsymbol{\beta}}_{LS}) ,$$

where $(\cdot)_+$ denotes the positive part (only positive elements are taken from the set).

The only difference in the representations of the ridge and the LASSO estimates (equations (2.2.14) and (2.2.16), respectively) is that the L2-norm penalty $\|\boldsymbol{\beta}\|^2$ is replaced by the L1-norm penalty $\|\boldsymbol{\beta}\|_1$, referred to as *the LASSO penalty*. The *RSS* function penalized by the LASSO is not differentiable if some regression parameter from the set $\boldsymbol{\beta}$ is equal to zero. Thus, it is impossible to obtain the solution in a closed-form as in case of ridge regression. The specific methods must be used to compute the LASSO estimates.

Usually, the LASSO is preferred to ridge regression because it provides the possibility to prevent the inclusion of irrelevant variables into the model. Such possibility allows to produce more accurate models in terms of prediction that often outperform those produced using the L2-norm regularization. However, the LASSO has some fundamental limitations, which make the technique to be inapplicable in some situations. We describe these limitations in Section 3.1.4.

**Constrained formulation.** The lack of a closed-form solution to the LASSO problem leads to the *constrained optimization problem* (refer to [BV04]). An equivalent way to represent the LASSO criterion from equation (2.2.16) is:

$$\|\boldsymbol{y} - \boldsymbol{X\beta}\|^2 \text{ subject to } \|\boldsymbol{\beta}\|_1 \leq t , \quad (2.2.17)$$

where the parameter $t$ is one-to-one correspondent to the regularization parameter $\lambda$.

The objective function $\|\boldsymbol{y} - \boldsymbol{X\beta}\|^2$ is convex and the constraints define a convex set. This forms a *convex optimization problem*. From this follows that any local minimizer of the objective function subject to the constraints is also the global minimizer. The objective function is quadratic in $\boldsymbol{\beta}$, and the set of non-differentiable constraints from the criterion (2.2.17) can be converted into a set of linear constraints. Combining all this enables to interpret the problem of computing the LASSO estimates as the *quadratic programming problem* (refer to [BV04] for further explanation).

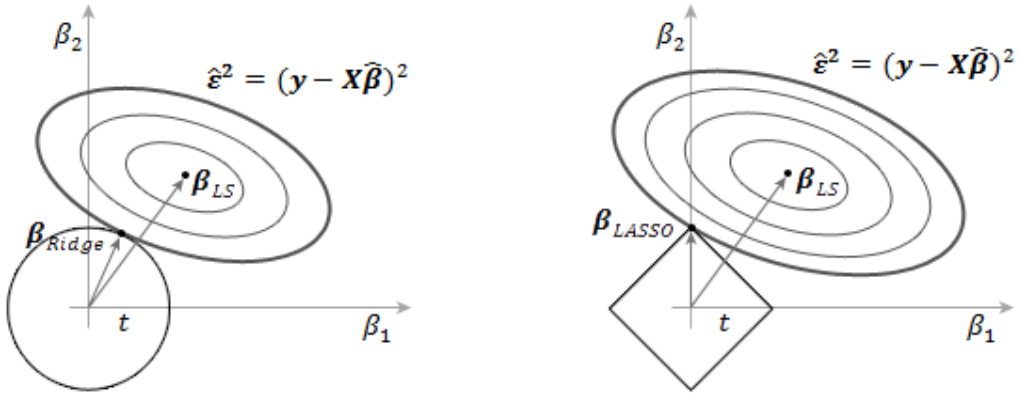


**Figure 2.3**. *Graphical representation of ridge regression (left) and the LASSO (right) in case of two explanatory variables in the model. The ellipses show the contours of the Least Squares error function. The circle is the region of the penalty function* $\beta_1^2 + \beta_2^2 \leq t$ *and the rhomb shows the boundary of the penalty function* $|\beta_1| + |\beta_2| \leq t$. *The ridge and the LASSO solutions are the points at which the innermost elliptical contours touch the circle and the rhomb, respectively.*

## 2.3 Rationale

It is believed that microarray data can provide important evidence about the functions of genes and relations among them. Unfortunately, transforming such data into knowledge is not a trivial task. Development of techniques for analyzing huge amounts of gene expression data is the task of computational biology and bioinformatics.

After hybridization (see Section 2.1.3) the microarray is scanned and the color of each spot can be converted to a single real number, representing the expression of the corresponding gene – *expression value*. Performing a series of different experiments for a certain group of genes and collecting the data from all the microarrays produced, we can represent these expression values in the matrix form with columns corresponding to genes and rows corresponding to experiments.

Suppose we are given a gene expression matrix. We know that microarray measurements represent only the amounts of mRNA for each gene. Ideally, these amounts could correspond to the amounts of proteins a cell produces. Unfortunately, this does not tend to be true. Besides translation the production process of the protein involves complex cell regulation machinery, which can modify the proportion of the protein being produced or even suppress it. However, in this work we ignore this fact because otherwise one would have to take into account all the translational issues and it could be very hard to develop and apply some universal techniques.

Thus, we consider a simplified interpretation of microarray data, which assumes that mRNA abundances correspond to protein abundances a cell produces. Then, we assume that some of the proteins are transcription factors that influence the synthesis of all the remaining proteins. More formal, we are interested in the transcription factor genes, and the influence of their expression on the expression of other genes. In context of our current work we seek to find the set of the most ‘influential’ transcription factor genes.

We are trying to achieve our aim by using a linear regression model. A linear approximation is not necessarily true in reality. However, there are several motivations for preferring a linear model over the others: it is simple enough to be fitted to data, it is interpretable and there are physical arguments supporting the use of a linear model (see [SWB02]).

Before applying linear regression on the data collected from a set of microarray experiments, we have to perform some reconstruction of the data and to describe some more assumptions.

We divide a gene expression matrix by columns into two sub-matrices, where the first sub-matrix contains expression of genes involved in the synthesis of transcription factors and the second holds the expression of the rest of the genes under the same set of experiments. For simplicity, we refer to the first matrix as the matrix of transcription factors, and to the second one as the expression matrix of genes. We denote the matrices using the letters $\boldsymbol{T}$ and $\boldsymbol{G}$, respectively.

We model the expression of each gene as a linear function of the expression of transcription factor genes as follows:

$$g_{ik} = \beta_{0k} + \sum_{j=1}^{p} \beta_{jk} t_{ij} + \varepsilon_{ik}\,, \quad i = 1, \dots, n,\ k = 1, \dots, r$$

where $n$ is the number of experiments, $r$ is the number of genes in the dataset (without the transcription factor genes), $p$ is the number of transcription factors and $\varepsilon$ is the noise that cannot be explained by the approximation.

Transcription factors are believed to have the most effect on the expression of genes from all the influencing factors. The presence of other impacts besides the transcription factors prevents us from considering the problem of inferring true regulators from a microarray data as a simple linear regression problem. Besides, this data usually contains measurement errors and noise. That is why, classical linear regression methods may have instabilities in the produced solutions. We consider some of the state-of-the-art algorithms, which are more robust and are designed to resolve the issues inherent to noisy and ill-conditioned data.

# Chapter 3

# Variable Selection Methods

Different techniques and methods are available for linear regression. Each of them has its own advantages and drawbacks, which force the methods to succeed or fail under various circumstances. For example, the typical dataset bioinformatics deals with contains a large amount of data without having enough information provided about the nature of this data. In context of linear modeling this means there are a lot of explanatory and response variables and much less observations in a dataset. In such situations straightforward methods are usually computationally inefficient and are not able to guarantee stable solutions. That is why, some more specific techniques are of interest.

We seek to find an optimal approach that could satisfy the needs described in the previous chapter. We do not prefer any class of techniques in advance, but investigate the problem from different sides, comparing the strategies. We start from the classical Forward Stepwise algorithm, proceeding to the recent achievements in the linear regression methodology, considering the *Least Angle Regression* algorithm, the *ElasticNet* penalty and others.

## 3.1 Algorithms

### 3.1.1 Forward Stepwise Selection

*Forward Stepwise* is the heuristic procedure which greedily adds predictive variables to the model according to their significance, and up to some predefined significance threshold. Forward Stepwise starts with no variables in a model. Firstly, it selects a single predictor that provides the best fit (e.g. the smallest *RSS*). Then another predictor is added that provides the best fit in combination with the first variable. Every next variable is selected by the same logic: it must produce the best fit in combination with all of its predecessors. The process continues until some stopping conditions are fulfilled. This can be either the number of predictors in the model or an insignificant reduction of the model error by appending more variables.

Usually, the influence of a variable on the prediction accuracy of a model is measured using *F-test*, but also other techniques like *t-test*, *R-square*, *Akaike Information Criterion* (*AIC*), *Bayesian Information Criterion* (*BIC*), *False Discovery Rate* (*FDR*) etc. are possible. F-test is designed to check if two sample variances are equal by comparing the ratio of these variances (*F-statistic*). As we are going to use the Forward Stepwise procedure for obtaining the Least Squares estimates we can define the significance of a regression parameter by calculating its F-statistic. The equation is the following:

$$F = \frac{RSS(\boldsymbol{\beta}_k) - RSS(\boldsymbol{\beta}_{k+1})}{RSS(\boldsymbol{\beta}_{k+1})/(n - (k+1) - 1)} \tag{3.1.1}$$

where $\boldsymbol{\beta}_k$ is the vector of $k$ regression parameters, $\boldsymbol{\beta}_{k+1}$ is the vector of size $k+1$, which contains $\boldsymbol{\beta}_k$ as the subset and the coefficient of a candidate variable for inclusion into the model; $n$ is the number of data observations. In case there are $r$ response variables in consideration, it is assumed that the F-statistic (3.1.1) has $F(r,\ r(n-(k+1)-1))$ distribution if the additional element in $\boldsymbol{\beta}_{k+1}$ does not sufficiently minimizes the *RSS* (see Figure 3.1 for example).

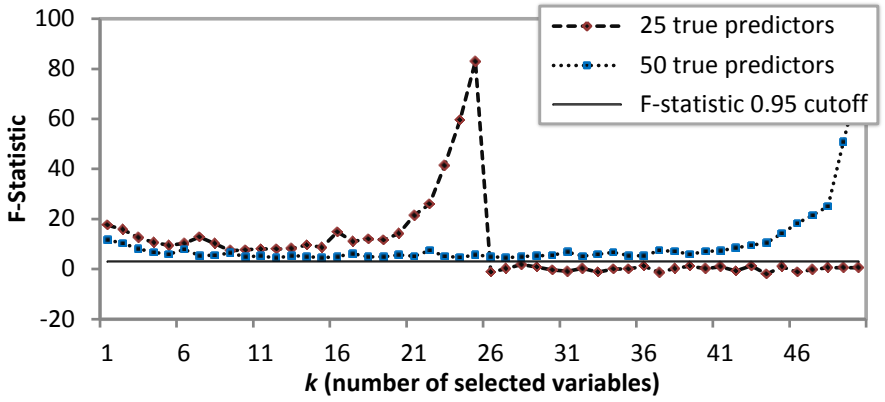


**Figure 3.1**. *Example of F-test application on two simulated regression problems* $\boldsymbol{Y} = \boldsymbol{XB} + \boldsymbol{E}$, $E \sim N(0,1)$. *In both cases there are* 100 *observations of* 50 *explanatory and* 10 *response variables. In the first case there are only* 25 *of true predictors, in the second – all of the variables are predictors. It can be seen from the figure that the F-value falls under its 0.95 significance bound on the 26'th step in case of* 25 *true predictors, and it does not fall at all when all variables are significant.*

In Section 2.2.3 it is shown how to compute the Least Squares estimates solving the set of normal equations (see equation 2.2.8). This operation requires $O(p^3 + np^2)$ computations for $n$ observations of $p$ predictors. If there are $r$ response variables the computational complexity is $O(r(p^3 + np^2))$, respectively. It is possible to decrease this value by using the *Cholesky decomposition* of $\boldsymbol{X}^T\boldsymbol{X}$ or the *QR decomposition* of $\boldsymbol{X}$. The Cholesky decomposition is computed in $O(p^3 + np^2/2)$ steps, while the *QR* decomposition requires only $O(np^2)$ operations.

The Forward Stepwise algorithm is greedy. For each new variable in a model it has to follow the whole path of computations from the beginning. It is possible to reduce computational burden by updating the Cholesky or *QR* decompositions at each step. However, this strategy works until there are less variables than the number observations in a model, because performing the decomposition is possible only in case of full-rank matrices.

The main drawback of the Forward Stepwise procedure is that it does not guarantee the best subset of variables. There may exist better combinations of variables, if they enter the model in the different order. But often even an approximate combination is sufficient, so the Forward Stepwise algorithm is still noteworthy.

### 3.1.2 Least Angle Regression

*Least Angle Regression* (*LAR*) algorithm proposed by Efron, et al. [EHJ04] relates to the class of Forward Selection methods. Being a stylized version of the *Stagewise procedure* its strategy in principle is very similar to the Forward Stepwise selection.

LAR works roughly as follows: given a set of predictors $\boldsymbol{X}$ the algorithm first selects the one most correlated with the response $\boldsymbol{y}$, say $x_{j_1}$ $(j = 1, \dots, p)$. Instead of obtaining the coefficient of the selected predictor 'completely', LAR starts to move the value of this coefficient from 0 towards its Least Squares solution (along the direction of $x_{j_1}$). As this value increases the $x_{j_1}$ correlation with the residual (denoted by $r$) decreases enabling some other predictor, say $x_{j_2}$, to have at least the same correlation with the residual $r$. At this point newly selected variable $x_{j_2}$ joins the *active set* (the set of the 'most correlated' variables). The process of moving the coefficients is then continued. Instead of continuing along the $x_{j_1}$ direction, *LAR* proceeds in the direction *equiangular* between $x_{j_1}$ and $x_{j_2}$, what makes possible to keep these

variables tight and their correlations with the residual decreasing. When the third 'most correlated' predictor $x_{j_3}$ is reached, LAR proceeds in the equiangular direction between $x_{j_1}$, $x_{j_2}$ and $x_{j_3}$, until a fourth variable enters the model, etc.

In order to proceed with the further description of the algorithm some notions must be defined first. At each step LAR adds one variable to the model, building up an estimate $\widehat{\boldsymbol{\mu}} = \boldsymbol{X}\widehat{\boldsymbol{\beta}}$. So after *k* steps there are exactly *k* elements in the vector of regression parameters, denoted by $\widehat{\boldsymbol{\beta}}_k$. Notation $\boldsymbol{u}_k$ is used to represent the unit bisector, which provides the equiangular movement between the *k* selected variables, and quantity $\gamma_k$ stands for the minimum step required along the $\boldsymbol{u}_k$, to allow some other not yet selected variable to enter the model.

The origin of the term 'least angle' comes from the geometrical interpretation of the LAR algorithm. Namely, this means that the unit bisector $\boldsymbol{u}_k$ forms the smallest angles with the selected predictors at *k*'th step (see Figure 3.2).

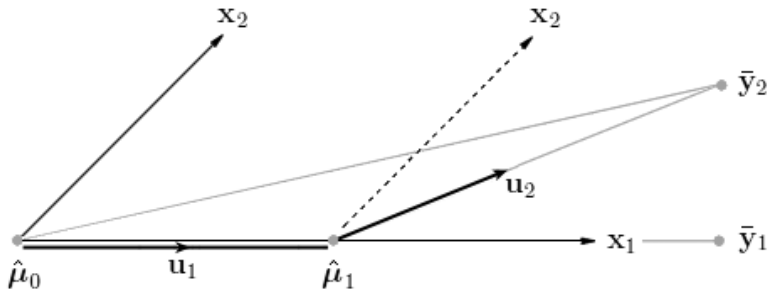


**Figure 3.2**. *The explanation of the LAR work cycles in case of two predictors. Point $\bar{y}_2$ is the Least Squares fit – the projection of y onto the subspace spanned by $x_1$ and $x_2$. Starting from the estimate $\hat{\mu}_0 = 0$, the residual $r_1 = \bar{y}_2 - \hat{\mu}_0$ has greater correlation with $x_1$ than with $x_2$. LAR proceeds in the direction to $x_1$ until $r_2 = \bar{y}_2 - \hat{\mu}_1$ bisects the angle between $x_1$ and $x_2$, where $\hat{\mu}_1 = \hat{\mu}_0 + \gamma_1 u_1$. The next LAR estimate is $\hat{\mu}_2 = \hat{\mu}_1 + \gamma_2 u_2 = \bar{y}_2$. If there are more than two input predictors, the $\hat{\mu}_2$ will not be equal to $\bar{y}_2$, because the algorithm will change the direction before the point $\bar{y}_2$ as new variable enters the model.*

Before applying LAR it is assumed that the predictor variables have been *standardized* to have zero mean and unit length, and that the response has been centered to have zero mean:

$$\sum_{i=1}^{n} y_i = 0\,, \quad \sum_{i=1}^{n} x_{ij} = 0 \quad \text{and} \quad \sum_{i=1}^{n} x_{ij}^2 = 1\,, \quad j = 1, \ldots, p\,,$$

where $p$ is the number of predictors and $n$ is the number of data observations. Further details of the rest computations performed by LAR are introduced in Algorithm 3.1(a).

---

**Algorithm 3.1(a)** *Least Angle Regression*

---

Input: $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_p)$, $\boldsymbol{y}$.

1) Start with $\widehat{\boldsymbol{\beta}}_0 = (\hat{\beta}_1, \hat{\beta}_2, \dots, \hat{\beta}_p) = 0$, $\widehat{\boldsymbol{\mu}}_0 = 0$, the active set $A = \emptyset$ ($A \subset \{1, \dots, p\}$), and the initial step $k = 0$ ;
2) While $|A| \leq p$ :
   a) $k \leftarrow k + 1$ ;
   b) Compute the correlations: $\hat{\boldsymbol{c}} \leftarrow \boldsymbol{X}^T(\boldsymbol{y} - \widehat{\boldsymbol{\mu}}_{k-1})$ ;
   c) Find the greatest absolute correlation: $\hat{C} = max_j\{|\hat{c}_j|\}$ ;
   d) Update the active set: $A \leftarrow A \cup \{j\}$ ;
   e) Define the unit vector making equal angles with the $\boldsymbol{X}_A$ : $\boldsymbol{w}_k \leftarrow A_A(\boldsymbol{X}_A^T\boldsymbol{X}_A)^{-1}1_A$ , where $A_A = [1_A^T(\boldsymbol{X}_A^T\boldsymbol{X}_A)^{-1}1_A]^{-\frac{1}{2}}$ and $1_A$ is a $|A|$-length vector of constant values 1;
   f) Compute the equiangular vector: $\boldsymbol{u}_k \leftarrow \boldsymbol{X}_A\boldsymbol{w}_k$ ;
   g) Choose the step size $\gamma_k$ along the $\boldsymbol{u}_k$:
      i. if $|A| == p$ then $\gamma_k = \hat{C}/A_A$ ;
      ii. else $\gamma_k = min_{l\notin A}^{+}\left\{\frac{\hat{C}-\hat{c}_l}{A_A-a_l}, \frac{\hat{C}+\hat{c}_l}{A_A+a_l}\right\}$ , $l \in \{1, \dots, p\}$ , where $a_l$ is an element of inner product vector $\boldsymbol{a} = \boldsymbol{X}^T\boldsymbol{u}_k$ and $min^+$ means that the minimum is taken only over positive values;
   h) Update the LAR estimate: $\widehat{\boldsymbol{\mu}}_k \leftarrow \widehat{\boldsymbol{\mu}}_{k-1} + \gamma_k\boldsymbol{u}_k$ ;
   i) Update the regression parameters: $\widehat{\boldsymbol{\beta}}_k \leftarrow \widehat{\boldsymbol{\beta}}_{k-1} + \gamma_k\boldsymbol{w}_k$ ;

Output: $\widehat{\boldsymbol{\beta}}_k = (\hat{\beta}_{k1}, \hat{\beta}_{k2}, \dots, \hat{\beta}_{kp})$.

---

If LAR obtains the coefficients for all predictor variables in a model, then these estimates are equal to the Least Squares estimates (step *2.g.i.* in Algorithm 3.1(a)). Furthermore, it has been observed (see [EHJ04]), that the LAR estimates are very similar to the coefficients

provided by the LASSO. Thus, with a simple modification the algorithm can provide the full set of LASSO solutions.

The LASSO implies that at the step $k$ the sign of any non-zero coefficient $\hat{\beta}_{kj}$ ($j = \{1, \dots, p\}$) is the same as the sign of the corresponding correlation $\hat{c}_j = \boldsymbol{x}_j^T(\boldsymbol{y} - \widehat{\boldsymbol{\mu}}_{k-1})$. Consider step *2.i* in Algorithm 3.1(a). It is clear that the $\hat{\beta}_{kj}$ is changed the sign at $\tilde{\gamma}_{kj} = -\hat{\beta}_{(k-1)j}/w_{kj}$ (for each of the estimates this value is unique). So, if there exists such the smallest $\tilde{\gamma}_{kj}$, that is less than the common step size $\gamma_k$, then the corresponding variable $\boldsymbol{x}_j$ is left out from the model and the algorithm makes just the step of size $\tilde{\gamma}_{kj}$ in the equiangular direction (refer to Algorithm 3.1(b)).

---

**Algorithm 3.1(b)** *Least Angle Regression (the LASSO modification)*

---

2) While $|A| \leq p$ :

   g) …

      iii. $\tilde{\boldsymbol{\gamma}}_k \leftarrow -\widehat{\boldsymbol{\beta}}_{k-1}/\boldsymbol{w}_k$ , $\tilde{\gamma}_{kj} = min^{+}_{j \in \{1,\dots,p\}}\{\tilde{\boldsymbol{\gamma}}_k\}$ ;

      iv. if $\tilde{\gamma}_{kj} < \gamma_k$ then $A \leftarrow A \backslash \{j\}$ , $\gamma_k \leftarrow \tilde{\gamma}_{kj}$ ;

---

LAR with the LASSO modification is renamed to *LARS* (LAR-LASSO). In the initial implementation (refer to [EHJ04]) LARS uses the Cholesky decomposition of the matrix $\boldsymbol{X}^T\boldsymbol{X}$. It is not computed completely in the beginning, but is updated at each step the algorithm performs. This provides the computational complexity $O(p^3 + np^2)$ for $n$ observations of $p$ variables included into the model. This is the cost of a Least Squares fit for the same number of variables. Besides, dropping the variables from the model (because of the LASSO modification) costs at most $O(p^2)$ operations per downdate of the decomposition matrix.

### 3.1.3 Shooting Algorithm for computing the LASSO

Moving back in time from *LARS*, it is necessary to point out one either efficient method for solving the LASSO. The method is referred to as '*Shooting*' and is developed by Fu [Fu98]. Being a special case of the Coordinate Descent procedure, 'Shooting' updates estimates of the regression parameters one-at-a-time, keeping the remaining variables fixed. The

efficiency of the algorithm is explained by its implementation simplicity and the small number of computations required to produce the solution. This makes the method more attractive in comparison with the similar strategies. Also the idea of '*warm start*' has a high effect. '*Shooting*' does not obtain the estimates of the coefficients from zero, but fixes the Least Squares estimates given as input, reducing them to the LASSO solution. The description of this procedure is provided in Algorithm 3.2.

**Algorithm 3.2** *'Shooting' algorithm for computing the LASSO*

Input: $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_p)$, $\boldsymbol{y}$, $\widehat{\boldsymbol{\beta}}_{LS}$, tuning parameter $\lambda$.

1) Start with $\widehat{\boldsymbol{\beta}}_0 = \widehat{\boldsymbol{\beta}}_{LS} = \left(\hat{\beta}_1, \hat{\beta}_2, \dots, \hat{\beta}_p\right)$ ;
2) Iterate until convergence :
   a) For each $j \in \{1, \dots, p\}$ :
      i. Compute the conventional variable $S_j$: $S_j \leftarrow -\boldsymbol{x}_j^T \boldsymbol{y} + \sum_{i \neq j} \boldsymbol{x}_j^T \boldsymbol{x}_i \hat{\beta}_i$ ;
      ii. Update the coefficient $\hat{\beta}_j$:

$$\hat{\beta}_j = \begin{cases} \dfrac{\lambda - S_j}{2\boldsymbol{x}_j^T \boldsymbol{x}_j} & if \;\; S_j > \lambda \\ \dfrac{-\lambda - S_j}{2\boldsymbol{x}_j^T \boldsymbol{x}_j} & if \;\; S_j < -\lambda \\ 0 & if \;\; |S_j| \leq \lambda \end{cases}$$

Output: $\widehat{\boldsymbol{\beta}}_{LASSO}$.

### 3.1.4 LARS-EN

Combining features from ridge regression and the LASSO, Zou and Hastie [ZH05] introduced new regularization technique called the *ElasticNet*. Similarly to the LASSO, the ElasticNet performs the simultaneous variables selection and the continuous shrinkage. It is proposed that the ElasticNet works as well as the LASSO, but overcomes some of its limitations, in overall, outperforming the latter in terms of prediction accuracy.

The fundamental drawbacks of the LASSO are described by Zou and Hastie [ZH05] with the following three scenarios.

1. The most sufficient disadvantage of the LASSO concerns its convex nature (the penalty function is convex, but not strictly convex). The LASSO is able to include at most $min(p, n)$ predictors into the model, where $p$ is the number of variables and $n$ is the number of observations in a dataset. Due to this limiting property the LASSO may be inconsistent if the number of potential predictors for inclusion into the model is much larger than the number of existing observations.

2. In the $p \gg n$ case the LASSO solution is not uniquely defined (see [OPT00] for further details). Besides, if there are pairwise correlations between variables in a dataset it has been empirically observed that ridge regression dominates the LASSO in terms of prediction performance (see [Tib96]). Usually, the high-dimensional data contains many variables that are highly correlated. Thus, the LASSO is not very suitable technique in this kind of problems as well.

3. Variables among which there are pairwise correlations form a group. Sometimes, it is useful to reveal this kind of information from the data and to consider a group as a common entity. The LASSO does not provide this possibility. It selects variables from groups without any further analysis which one to select.

The ElasticNet is supposed to overcome the problems described above. It can select more than $n$ variables in the $p \gg n$ case and can include the whole groups of correlated variables into the model (known as *grouping effect* of the ElasticNet).

The ElasticNet criterion is defined in two steps. Let us consider the so-called *naïve ElasticNet estimates* first:

$$\widehat{\boldsymbol{\beta}}_{EN_{naive}} = argmin_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda_2 \|\boldsymbol{\beta}\|^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 \ , \tag{3.1.2}$$

where the penalty function $\lambda_2 \|\boldsymbol{\beta}\|^2 + \lambda_1 \|\boldsymbol{\beta}\|_1$ is the linear combination of the ridge and the LASSO penalties. If $\lambda_1 = 0$ or $\lambda_2 = 0$ then the naive ElasticNet problem reduces to ridge regression or the LASSO, respectively. The naïve ElasticNet penalty function as in case of the LASSO is not differentiable at zero (refer to Section 2.2.4). This allows the ElasticNet to do automatic variable selection. However, unlike the LASSO the function $\lambda_2 \|\boldsymbol{\beta}\|^2 + \lambda_1 \|\boldsymbol{\beta}\|_1$ is strictly convex if $\lambda_2 > 0$. Thus, the ElasticNet solution is well defined in both the $p < n$ and the $p \gg n$ cases. Figure 3.2 represents the ElasticNet penalty function graphically.

Combining both the L1-norm and the L2-norm penalties a double amount of shrinkage occurs. Double shrinkage does not help much in reducing the variance of a regression model, but spends an extra bias of the coefficients.

In [ZH05] is suggested to use the 'corrected' ElasticNet criterion, which is defined as:

$$\widehat{\boldsymbol{\beta}}_{EN} = (1 + \lambda_2)\, \widehat{\boldsymbol{\beta}}_{EN_{naive}} \ . \tag{3.1.3}$$

The correction (3.1.3) is the simpliest way to undo double shrinkage while holding present all the other features of the introduced regularization technique. Thus, the ElasticNet estimates are just rescaled version of the naïve ElasticNet estimates.

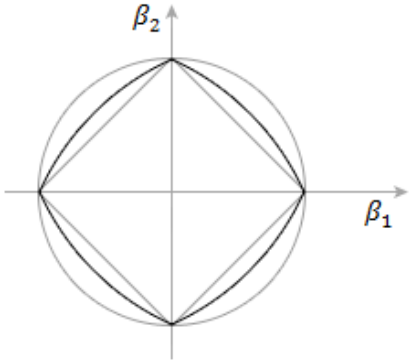


**Figure 3.2**. *The ElasticNet penalty in case of two variables in the model. The outer shape stands for the ridge penalty function and the inner one for the LASSO penalty. The ElasticNet penalty function is the black contour in between with* $\alpha = \lambda_2/(\lambda_1 + \lambda_2)$*=0.5. The α value varies from* [0,1]*; when* $\alpha = 1$ *the ElasticNet becomes ridge regression, and if* $\alpha = 0$ *the ElasticNet reduces to the LASSO.*

In conclusion, we define the ElasticNet criterion in the matrix form as follows:

$$\widehat{\boldsymbol{\beta}}_{EN} = argmin_{\boldsymbol{\beta}} \boldsymbol{\beta}^T \left( \frac{\boldsymbol{X}^T\boldsymbol{X} + \lambda_2 \boldsymbol{I}}{1+\lambda_2} \right) \boldsymbol{\beta} - 2\boldsymbol{y}^T\boldsymbol{X}\boldsymbol{\beta} \ + \lambda_1 \|\boldsymbol{\beta}\|_1 \ . \tag{3.1.4}$$

In the fashion equivalent to (3.1.4) the LASSO estimates are defined as:

$$\widehat{\boldsymbol{\beta}}_{LASSO} = argmin_{\boldsymbol{\beta}} \boldsymbol{\beta}^T (\boldsymbol{X}^T\boldsymbol{X})\boldsymbol{\beta} - 2\boldsymbol{y}^T\boldsymbol{X}\boldsymbol{\beta} \ + \lambda_1 \|\boldsymbol{\beta}\|_1 \ .$$

Hence, it can be seen that the ElasticNet solution is the regularized version of the LASSO solution (refer to [ZH05] for further explanation).

**LARS-EN.** Like in case of the LASSO computing the ElasticNet solution is a quadratic programming problem. In [ZH05] it is shown that the ElasticNet solution path is a piecewise linear function of $\lambda_1$ for each fixed $\lambda_2$. As the LAR algorithm proceeds in the piecewise liner fashion (refer to [EHJ04] ) it can be successfully used for computing the ElasticNet regularization path also. Zou and Hastie [ZH05] have developed an appropriate modification of LAR, which enables to obtain the ElasticNet estimates in efficient way. As in case of LAR it requires $O(p^3 + np^2)$ operations for computing the coefficients for $p$ predictor variables.

### 3.1.5 Group LASSO

The ElasticNet exhibits the *grouping effect*, but these groups are unknown in the beginning of analysis. In some situations predictor variables belong to the already predefined groups. The task is then to consider the members of a group as one entity, simultaneously selecting all of the members, and to shrink the corresponding coefficients together. Bakin [Bak99] proposed an extension to the LASSO for solving these purposes. This extension has been generalized by Yuan and Lin [YL06] and became known as the *Group LASSO*.

Let $\boldsymbol{X}$ be a set of $p$ predictors, which are divided into $L$ groups ($L \in \{1, \dots, p\}$). We use the notation $\boldsymbol{X}_l$ to represent the predictors of the group $l$ ($l = 1, \dots, L$) and the notation $\boldsymbol{\beta}_l$ to represent the corresponding coefficients. The criterion of the Group LASSO estimates is then defined as:

$$\widehat{\boldsymbol{\beta}}_{G-LASSO} = argmin_{\boldsymbol{\beta}} \frac{\mathbf{1}}{\mathbf{2}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda \sum_{l=1}^{L} \sqrt{|\boldsymbol{X}_l|} \|\boldsymbol{\beta}_l\| \ , \tag{3.1.5}$$

where $|\cdot|$ denotes the number of predictors in $\boldsymbol{X}_l$ and the L1-norm of the coefficients is replaced by the L2-norm ($\|\boldsymbol{\beta}_l\|$) in a difference from the ordinary LASSO. The norm $\|\boldsymbol{\beta}_l\|$ is equal to zero only if all coefficients in $\boldsymbol{\beta}_l$ are zeros, what explains the possibility of the introduced technique to produce sparse solutions on the group level. The illustration of the Group LASSO penalty function is brought in Figure 3.3.

Yuan and Lin [YL06] have developed an extension to the 'Shooting' algorithm for solving the Group LASSO. Their idea is based on the *Karush-Kuhn-Tucker conditions*[v], which state that $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \dots, \boldsymbol{\beta}_L)$ is a solution to the problem (3.1.5) if the following two

[v] In mathematics, the Karush–Kuhn–Tucker conditions are necessary for a solution in nonlinear programming to be optimal.

requirements are fulfilled:

$$\begin{aligned} &(1)\ \ -\boldsymbol{X}_l^T(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})+\frac{\lambda\sqrt{|\boldsymbol{X}_l|}\boldsymbol{\beta}_l}{\|\boldsymbol{\beta}_l\|}=\boldsymbol{0} \qquad \forall\boldsymbol{\beta}_l\neq\boldsymbol{0} \\ &(2)\ \ \|-\boldsymbol{X}_l^T(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})\|\leq\lambda\sqrt{|\boldsymbol{X}_l|} \qquad \forall\boldsymbol{\beta}_l=\boldsymbol{0} \end{aligned} \tag{3.1.6}$$

The solution to the proposition (3.1.6) is:

$$\boldsymbol{\beta}_l=(1-\frac{\lambda\sqrt{|\boldsymbol{X}_l|}}{\|\boldsymbol{S}_l\|})_+\boldsymbol{S}_l\ ,$$

where $\boldsymbol{S}_l=\boldsymbol{X}_l^T(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}_{-\boldsymbol{l}})$ and $\boldsymbol{\beta}_{-\boldsymbol{l}}=(\boldsymbol{\beta}_1,\dots,\boldsymbol{\beta}_{l-1},\boldsymbol{0},\boldsymbol{\beta}_{l+1},\dots,\boldsymbol{\beta}_L)$ . By iteratively estimating the group coefficients $\boldsymbol{\beta}_l$ $(l=1,\dots,L)$, the solution, which satisfies the criterion (3.1.5) is produced.

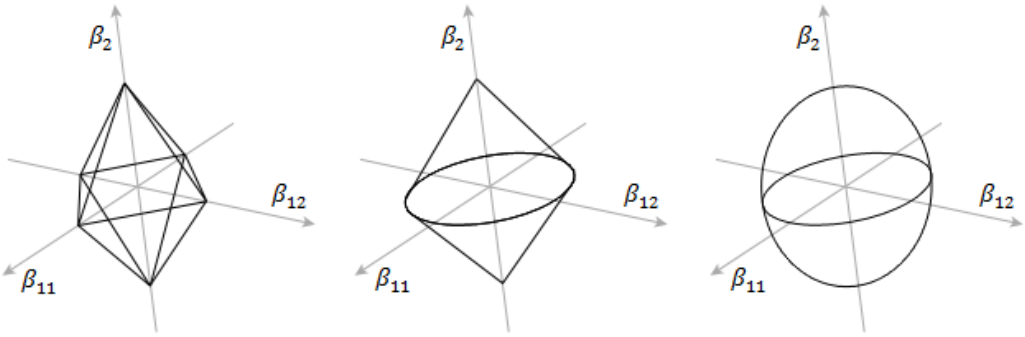


**Figure 3.3**. *The contours of the LASSO penalty (leftmost), the Group LASSO penalty (central) and the ridge penalty (rightmost) in case of three variables in the model, where two of the variables form a group. It can be seen from the central plot that the ridge penalty is applied within the group coefficients $\beta_{11}$ and $\beta_{12}$, and the LASSO penalty is applied on pairs $(\beta_{11}, \beta_2)$ and $(\beta_{12}, \beta_2)$. This provides the sparsity within the groups.*

**Group LARS.** Solving the Group LASSO based on the 'Shooting' method may require huge amount of computations as the number of predictor variables increases. Using the Group LARS algorithm proposed by Yuan and Lin [YL06] is more efficient way for obtaining the Group LASSO estimates. The modification of LAR implies that a group of variables can be included into the model only if the *averaged squared correlation* of the variables with the residual is the largest. At each step of Group LARS one group enters the model, and the coefficients of the elements from all active groups are then shrunk in the direction to the Least Squares solution (like in case of the ordinary LAR algorithm). Further implementation details are explained in [YL06].

Park and Hastie [PH06] suggest using *averaged absolute correlations* for each group instead of averaged squared correlations because this yields more variables in a group to be strongly correlated with the residual. They introduce the *Group LARS type II* algorithm that is described in [PH06].

### 3.1.6 Multiresponse Sparse Regression Algorithm

Similä and Tikka [ST05] address the multiple response problem, offering a technique that is called the *Multiresponse Sparse Regression* (*MRSR*). Its main point is based on the idea that observing all the response variables at once may influence the accuracy of the regression parameters. This, in turn, is suggested to improve the prediction in terms of the collinearity and the overfitting problems. The MRSR implementation is an extension to the LAR algorithm which allows the latter to process multiple responses at once. MRSR performs simultaneous variable selection and continuous shrinkage with the difference that all coefficients corresponding to one predictor are estimated as a group. This means that if a predictor variable is not included into the model then its coefficients for every response are all zeros, and they are all non-zeros if the variable is selected.

Let $\boldsymbol{X}$ be a $(n \times p)$ matrix and $\boldsymbol{Y}$ be a $(n \times r)$ matrix holding $n$ observations of $p$ predictors and $r$ response variables, respectively. It is assumed that the variables in $\boldsymbol{X}$ have been standardized to have zero mean and the unit length, and the variables in $\boldsymbol{Y}$ have been just centered (zero mean). The regression model has then the form $\widehat{\boldsymbol{Y}} = \boldsymbol{X}\widehat{\boldsymbol{B}}$, where $\widehat{\boldsymbol{B}}$ denotes the $(p \times r)$ matrix of regression parameters for all the responses. At each step $k$ $(k = 1, \dots, p)$ the MRSR algorithm introduces a new row of non-zero coefficients to $\widehat{\boldsymbol{B}}$, and at the step $k = p$ the matrix $\widehat{\boldsymbol{B}}$ equals to the Least Squares solution for the considered regression problem.

The vector of correlations between the predictor variables and the residuals $\widehat{\boldsymbol{E}} = \left(\boldsymbol{Y} - \widehat{\boldsymbol{Y}}\right) = = (\widehat{\boldsymbol{\varepsilon}}_1, \widehat{\boldsymbol{\varepsilon}}_2, \dots, \widehat{\boldsymbol{\varepsilon}}_r)$ at the step $k$ is denoted by $\hat{\boldsymbol{c}}_k = (\hat{c}_{k1}, \hat{c}_{k2}, \dots, \hat{c}_{kp})$. The correlation between the $j$'th predictor $(j = 1, \dots, p)$ and the residuals $(\hat{c}_{kj})$ is computed as follows:

$$\hat{c}_{kj} = \left\| \boldsymbol{x}_j^T (\boldsymbol{Y} - \widehat{\boldsymbol{Y}}) \right\|_q , \tag{3.1.7}$$

where $q \geq 1$ fixes the norm of the obtained vector. It is logical to suggest that the predictor $\boldsymbol{x}_j$, corresponding to the highest value in $\hat{\boldsymbol{c}}_k$, is the most influential in the reduction

of the residual, and, hence, can be included into the model.

Let $\hat{C}_k = max_j\{\hat{c}_{kj}\}$ be the maximum correlation at the step $k$. The active set $A_k$ is then appended by the corresponding index $j$, that is, $A_k = A_{k-1} \cup \{j\}$. The Least Squares estimates for the selected predictors $\boldsymbol{X}_A$ are $\widehat{\boldsymbol{\beta}}_{LS} = (\boldsymbol{X}_A^T\boldsymbol{X}_A)^{-1}\boldsymbol{X}_A^T\boldsymbol{Y}$. In turn, the MRSR estimates of the regression parameters and the corresponding prediction of the response are then defined as:

$$\begin{aligned} &(1)\ \ \widehat{\boldsymbol{Y}}_k = (1-\gamma_k)\widehat{\boldsymbol{Y}}_{k-1} + \gamma_k\boldsymbol{X}\widehat{\boldsymbol{\beta}}_{LS} \\ &(2)\ \ \widehat{\boldsymbol{\beta}}_k = (1-\gamma_k)\widehat{\boldsymbol{\beta}}_{k-1} + \gamma_k\widehat{\boldsymbol{\beta}}_{LS} \end{aligned} \tag{3.1.8}$$

where $\gamma_k \in (0,1]$ is the step size towards the Least Squares solution, which acts like shrinking parameter for the coefficients. Shrinking is applied only on the active variables while not yet obtained coefficients are constrained to zero. If $\gamma_k = 1$ MRSR reduces to the greedy Forward Selection algorithm producing the Least Squares fit. The task is to find the smallest positive $\gamma_k$ such that some non-active predictor $\boldsymbol{x}_j$ ($j \notin A_k$) could have the same correlation with the residuals as those from the active set.

Substituting equation (3.1.8(1)) into equation (3.1.7) and presenting the latter as the function of $\gamma$, we have:

$$\begin{aligned} &(1)\ \hat{c}_{k+1,j}(\gamma) = |1-\gamma|\hat{C}_k && \forall j \in A_k \\ &(2)\ \hat{c}_{k+1,j}(\gamma) = \left\|\boldsymbol{u}_{k,j} - \gamma\boldsymbol{v}_{k,j}\right\|_q && \forall j \notin A_k \end{aligned} \tag{3.1.9}$$

where $\boldsymbol{u}_{k,j} = (\boldsymbol{Y} - \boldsymbol{Y}_{k-1})^T\boldsymbol{x}_j$ and $\boldsymbol{v}_{k,j} = (\boldsymbol{X}\widehat{\boldsymbol{\beta}}_{LS} - \boldsymbol{Y}_{k-1})^T\boldsymbol{x}_j$. New predictor variable $\boldsymbol{x}_j$ enters the model when equations (3.1.9(1)) and (3.1.9(2)) are equal. Solving the problem (3.1.9) for all the $\boldsymbol{x}_j$ ($j \notin A_k$), the minimum value is chosen from the obtained set as the common step size: $\gamma_k = min_{j\notin A_k}\gamma_{kj}$. The computation of the $\gamma_{kj}$ value depends on the selected norm of the vector of correlations $\hat{\boldsymbol{c}}_k$. The norms and the corresponding equations are:

- $q = 1 \rightarrow$ $\quad \gamma_{kj} = max_\gamma\{\gamma : \left\|\boldsymbol{u}_{k,j} - \gamma\boldsymbol{v}_{k,j}\right\|_1 \le (1-\gamma)\hat{C}_k\}$

$$= min^+\left\{\frac{\hat{C}_k - \sum_{i=1}^r s_i u_{k,ji}}{\hat{C}_k - \sum_{i=1}^r s_i v_{k,ji}}\right\}, \text{ where } s_i = \pm 1$$

- $q = 2 \rightarrow$ $\gamma_{kj} = min_{\gamma>0}\{\gamma : \left\|\boldsymbol{u}_{k,j} - \gamma\boldsymbol{v}_{k,j}\right\|_2^2 \leq (1-\gamma)^2\hat{C}_k^2\}$

$$= min^{+}\left\{\frac{b\pm\sqrt{b^2-ac}}{a} : \begin{matrix} a=\hat{C}_k^2-\|\boldsymbol{v}_{k,j}\|_2^2 \\ b=\hat{C}_k^2-\boldsymbol{u}_{k,j}^T\boldsymbol{v}_{k,j} \\ c=\hat{C}_k^2-\|\boldsymbol{u}_{k,j}\|_2^2 \end{matrix}\right\}$$

- $q = \infty \rightarrow$ $\gamma_{kj} = max_{\gamma}\{\gamma : \left\|\boldsymbol{u}_{k,j} - \gamma\boldsymbol{v}_{k,j}\right\|_\infty \leq (1-\gamma)\hat{C}_k\}$

$$= min_{1\leq i\leq r}^{+}\left\{\frac{\hat{C}_k+u_{k,ji}}{\hat{C}_k+v_{k,ji}}, \frac{\hat{C}_k-u_{k,ji}}{\hat{C}_k-v_{k,ji}}\right\}$$

Given $\boldsymbol{u}_{k,j}$, $\boldsymbol{v}_{k,j}$ and $\hat{C}_k$ the computation of the parameter $\gamma_{kj}$ is solved in $O(2^r)$ steps in case of the L1-norm MRSR algorithm, while the computational complexity of the L2-norm and the sup-norm (∞-norm) modifications of the algorithm depends on the number of response variables linearly, that is $O(r)$. Thus, the L1-norm MRSR algorithm may be memory consuming as the number of response variables increases.

### 3.1.7 Blockwise Coordinate Descent Procedure for the Multi-task LASSO

The definition of the *Multi-task* LASSO was proposed by Zhang [Zha06], who generalized the LASSO to the multi-task setting. Suppose a dataset consists of some set of variables and the corresponding outcome. If there are several datasets sharing the common design, and the relationship between variables and outcomes across all the datasets is of interest, then this problem is referred to as the multi-task problem.

We can consider a multiresponse regression model as the *multi-task* problem with the number of tasks corresponding to the number of response variables in the model. The aim is then to find the subset of explanatory variables that simultaneously explains all the responses in the best manner.

Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be matrices of $n$ observations of $p$ explanatory and $r$ response variables, respectively. The matrix $\widehat{\boldsymbol{B}}$ is then the $(p \times r)$ matrix containing the regression parameters. Turlach, et al. [TVW05] define the '*simultaneous explanatory power*' of the *j*'th predictor $(j = 1, \dots p)$ on all $r$ responses as $\hat{\beta}_j^{max} = \max(|\hat{\beta}_{j1}|, |\hat{\beta}_{j2}|, \dots, |\hat{\beta}_{jr}|)$, and suggest to apply a penalty on the sum of these elements $\hat{\beta}_1^{max}, \hat{\beta}_2^{max}, \dots, \hat{\beta}_p^{max}$. Hence, L1-norm regularization

(the LASSO) in case of multiple responses becomes the sup-norm regularization, named by the authors as the *Simultaneous LASSO*.

The criterion of the Simultaneous LASSO estimates is formulated as:

$$\widehat{\boldsymbol{B}}_{S-LASSO} = argmin_{\boldsymbol{B}} \frac{1}{2} \|\boldsymbol{Y} - \boldsymbol{XB}\|^2 + \lambda \|\boldsymbol{\beta}^{max}\|_\infty \,. \tag{3.1.10}$$

An important remark must be made before continuing. The Simultaneous LASSO estimates (like the Multi-task LASSO ones defined later) have no any inherent meaning. They are just used for the learning which variables to include into the model. Appropriate coefficients can be estimated using other statistical methods, for example, the Least Squares method etc.

The Multi-task LASSO criterion proposed by Zhang has the following form:

$$\widehat{\boldsymbol{\beta}}^{(t)}_{MT-LASSO} = argmin_{\boldsymbol{\beta}} \frac{1}{2} \left\|\boldsymbol{y}^{(t)} - \boldsymbol{X}^{(t)}\boldsymbol{\beta}\right\|^2 + \lambda \|\boldsymbol{\beta}^{max}\|_\infty \,, \tag{3.1.11}$$

where $\boldsymbol{\beta}^{(t)}$ denotes the regression parameters in case of the task $t$ and $\boldsymbol{\beta}^{max}$ is the vector of the maximal $p$ coefficients across all $t$ tasks.

It is obvious that the criterions (3.1.10) and (3.1.11) are equivalent if the matrix $\boldsymbol{X}$ is fixed for all the tasks introduced to the Multi-task LASSO. Collecting the Multi-task LASSO estimates into matrix $\widehat{\boldsymbol{B}}$, the solution to the multiresponse regression problem is produced.

Zhang has introduced the Double Coordinate Descent procedure for computing the Multi-task LASSO (see [Zha06]), while Turlach, et al. [TVW05] use the Interior-Point method. Both of these techniques have the performance issues in the case of large number of inputs. More efficient algorithm is proposed by Liu, et al. [LPZ09], which is referred to as the *Blockwise Coordinate Descent* procedure (*BCD)*. BCD is developed to produce the Multi-task LASSO solution. However, it can be used for solving the Simultaneous LASSO as well.

The term 'blockwise' means that the regression coefficients for one predictor in case of the multiple tasks (responses) are considered as a block (the same as in case of the MRSR algorithm). BCD cycles through blocks updating them one-at-time. Let $\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\beta}}_2, \dots, \widehat{\boldsymbol{\beta}}_p$ be such blocks. Each $\widehat{\boldsymbol{\beta}}_j$ $(j = 1, \dots, p)$ is updated by the following sub-problem:

$$\widehat{\boldsymbol{\beta}}_j = argmin_{\boldsymbol{\beta}} \frac{1}{2} \left\|\boldsymbol{r}_j - \boldsymbol{x}_j\boldsymbol{\beta}\right\|^2 + \lambda |max(\boldsymbol{\beta})| \,, \tag{3.1.12}$$

where $\boldsymbol{r}_j = \boldsymbol{Y} - \sum_{i \neq j} \boldsymbol{x}_i \boldsymbol{\beta}_i$ denotes the partial residual vector.

The solution to the problem (3.1.12) can be efficiently obtained by using a *closed-form Winsorization operator* (see [LPZ09]). This operator makes possible to avoid some sort of computations performed by other methods mentioned above (Double Coordinate Descent etc.). The BCD procedure is described in detail in Algorithm 3.3.

---

**Algorithm 3.3** *Blockwise Coordinate Descent Procedure for the Multi-task LASSO*

---

Input: $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_p)$, $\boldsymbol{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_r)$, tuning parameter $\lambda$.

1) Start with $\widehat{\boldsymbol{B}}_0 = \left(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\beta}}_2, \dots, \widehat{\boldsymbol{\beta}}_p\right) = \boldsymbol{0}$, $\boldsymbol{C} = \boldsymbol{X}^T\boldsymbol{Y}$, and $\boldsymbol{D} = \boldsymbol{X}^T\boldsymbol{X}$ ;
2) Iterate until convergence:
   a) For each $j \in \{1, \dots, p\}$ :
      i. Compute $\boldsymbol{\alpha}$ : $\forall k \in \{1, \dots, r\}$, $\alpha_k \leftarrow c_{jk} - \sum_{i \neq j} \hat{\beta}_{ik} d_{ij}$ ;
      ii. If $\|\boldsymbol{\alpha}\|_1 \leq \lambda$ then $\widehat{\boldsymbol{\beta}}_j \leftarrow \boldsymbol{0}$

          else :
          1. Sort the $\boldsymbol{\alpha}$ indexes : $\left|\alpha_{k_1}\right| \geq \left|\alpha_{k_2}\right| \geq \cdots \left|\alpha_{k_r}\right|$ ;
          2. $m^* \leftarrow \max_m \left(\sum_{i=1}^{m} \left|\alpha_{k_i}\right| - \lambda\right) / m$ ;
          3. for each $i = 1, \dots, r$ :
             - if $i > m^*$ then $\hat{\beta}_{jk_i} \leftarrow \alpha_{k_i}$
             - else compute the *Winsorization operator* :

               $$\hat{\beta}_{jk_i} \leftarrow \frac{sign(\alpha_{k_i})}{m^*}\left(\sum_{l=1}^{m^*} \left|\alpha_{k_l}\right| - \lambda\right)$$

Output: $\widehat{\boldsymbol{B}}_{MT-LASSO}$.

---

The most computationally expensive step in Algorithm 3.3 is sorting (step 2.a.ii.1). It takes $O(r \log r)$ operations in case of $r$ response variables. However, if a sparse model is expected, then sorting is performed more seldom as the coefficients of a whole block are set to zero already at the first steps of the algorithm's work. This process is controlled by the tuning parameter.

## 3.2 The Explanation of Choice

We started our current research with the Forward Stepwise procedure. We used it mainly for identifying the applicability of liner models on microarray data. This algorithm is simple and can be easily implemented. We then considered the regularization techniques: ridge regression and the LASSO. These techniques help to overcome some of the computational issues of the Least Squares method and are suggested to produce more interpretable models.

We have included the LARS algorithm into the research, first of all, because of its efficiency in solving the LASSO problem. There is a set of modifications to LAR that make it possible to use the algorithm for other purposes as well. In our case the MRSR and the LARS-EN algorithms are based on LAR, and the Group LASSO solution can be obtained using Group LARS.

The Least Squares method, ridge regression, the LASSO, the ElasticNet and the Group LASSO are single-response techniques. We consider the MRSR algorithm and the Multi-task LASSO as the techniques being able to perform in the multiresponse setting.

In our analysis of the methods performance we actually do not use the Group LASSO technique directly. We do not require a method for handling the grouped variables. However, we assume that with the simple data transformation we can convert the grouping property of the Group LASSO into the property which could make it possible to include variables into the model according to the relation with all the responses simultaneously.

Suppose $\boldsymbol{X}$ and $\boldsymbol{Y}$ are matrices of $n$ observations of $p$ explanatory and $r$ response variables, respectively. We can stack the columns of $\boldsymbol{Y}$ into a $(rn \times 1)$ vector and copy $\boldsymbol{X}$ into the diagonal blocks of a new $(rn \times rp)$ matrix. Thus, input data becomes single-response. We form $p$ groups, each containing one explanatory variable $r$ times. The Group LASSO produces then the $(rp \times 1)$ vector of regression parameters $\widehat{\boldsymbol{\beta}}$. Each group of estimates from $\widehat{\boldsymbol{\beta}}$ corresponds to the one particular predictor.

# Chapter 4

# Performance Analysis

In the previous chapter we presented a number of methods for estimating regression parameters in linear models. In this chapter we experimentally analyze these methods in order to define their applicability to biological data. We are mainly interested in the variable selection ability of the methods. However, we also partially touch their predictive performance. We simulate the appropriate datasets and compare performance of the methods in various conditions.

## 4.1 Experimental Setup

For the comparison we generate artificial datasets with three different sets of true predictors. In all datasets the number of observations is less than the number of explanatory variables, and the datasets are multiresponse.

The performance metrics used in the further analysis are described in Section 4.1.2.

### 4.1.1 The Artificial Dataset

To simulate a gene expression dataset, we first generate a $(n \times p)$ matrix $\boldsymbol{T}$ representing the expression data of $p$ transcription factors collected over $n$ experiments. Next, we generate a $(p \times r)$ weights matrix $\boldsymbol{W}$ holding the regulation rules for $r$ genes. The expression values of the transcription factors and the regulation parameters are randomly chosen from the standard normal distribution $N(0, 1^2)$. In $\boldsymbol{W}$ we substitute rows $1, \dots, (p-q)$, where $q < p$, with zero vectors in order to simulate the situation when only part of the transcription factors influence the expression of genes. The $(n \times r)$ gene expression matrix $\boldsymbol{G}$ is then constructed according to the model:

$$\boldsymbol{G} = \boldsymbol{TW} + \boldsymbol{E}, \quad \text{where } \sim N(0, \sigma^2)\ . \tag{4.1.1}$$

Using the scheme (4.1.1), we design the simulation as follows:

- We generate 3 datasets consisting of expression data of 100 transcription factors and 1000 genes under 50 experiments.

  In the first dataset 10 rows in the weights matrix are non-zeros. Thus, only 10 transcription factors are true regulators. In the same way, there are 25 regulators in the second dataset, and there are 50 regulators in the third dataset.

  We use the notation $(\boldsymbol{T}_{n\times p}, \boldsymbol{G}_{n\times r}, q)$ to represent the constructed datasets. These are:

  $$(\boldsymbol{T}_{50\times 100}, \boldsymbol{G}_{50\times 1000}, 10),\ (\boldsymbol{T}_{50\times 100}, \boldsymbol{G}_{50\times 1000}, 25) \text{ and } (\boldsymbol{T}_{50\times 100}, \boldsymbol{G}_{50\times 1000}, 50)\ ,$$

  respectively. For simplicity we sometimes omit the size indications in the matrices $\boldsymbol{T}$ and $\boldsymbol{G}$, and refer to the datasets as $\boldsymbol{TG}^{(\mathbf{10})}$, $\boldsymbol{TG}^{(\mathbf{25})}$ and $\boldsymbol{TG}^{(\mathbf{50})}$.

- In order to achieve more stable results, we examine the behavior of the methods in case of 50 random datasets of a given type, computing performance metrics as averaged over all replications.

- We assess the stability of obtained test results by comparing the outputs of the methods within the datasets also. For that we generate $\boldsymbol{T}$ and $\boldsymbol{G}$ matrices containing 100 experiments, and then divide the dataset into two parts: $(\boldsymbol{T}_{50\times 100}, \boldsymbol{G}_{50\times 1000}, q)_1$ and $(\boldsymbol{T}_{50\times 100}, \boldsymbol{G}_{50\times 1000}, q)_2$. We apply each method on both $\boldsymbol{TG}_{\mathbf{1}}^{(q)}$ and $\boldsymbol{TG}_{\mathbf{2}}^{(q)}$ and examine the difference between two sets of the produced estimates (detailed explanation is provided in Section 4.3.2).

- For fitting the models we use *training* datasets and independent *test* datasets for evaluating their prediction ability. Independent *validation* datasets are used for estimating the methods' tuning parameters. This process is described in Section 4.2.1.

- In the initial experiments we generate noise-free data. We add noise to datasets and analyze the methods' dependence on it in Section 4.4.

### 4.1.2 Performance Metrics

The metrics we use for assessing the performance of the techniques being analyzed are described below. Formally, we need to examine how good the methods perform variable selection, to measure the prediction accuracy of the models produced and to investigate the

stability of the estimated model parameters. For that we use the following metrics: the *coefficient of determination* (denoted as $R^2$), the *mean squared error* (*MSE*) and the *discordance measure*.

**Coefficient of Determination.** The coefficient of determination ($R^2$) indicates how much of the total variation in the response variable is explained by the regression model. In other words, it measures the goodness of the fit and how good the model predicts. The $R^2$ statistic is calculated as follows:

$$R^2(\hat{\boldsymbol{y}}) = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where $\boldsymbol{y}$ denotes the response variable, $\bar{y}$ denotes its mean, and $\hat{\boldsymbol{y}}$ is the estimated prediction of $\boldsymbol{y}$. The $R^2$ statistic value varies in the range $[0..1]$. The value of $R^2 = 1$ means that the estimated model predicts perfectly. If the model does not predict better than just using the mean of the considered response, then $R^2 = 0$. Model that introduces noise rather than explaining the data can, in principle, have negative $R^2$.

**Mean Squared Error.** The mean squared error (*MSE*) of the estimated model parameters is the mean of the squared differences between the estimates and the true coefficients:

$$MSE(\hat{\boldsymbol{\beta}}) = \frac{1}{n}\sum_{i=1}^{n}(\beta_i - \hat{\beta}_i)^2 \ .$$

**Discordance measure.** The discordance measure presents the compliance of the solution produced by the method with the expected one. In this work we check how good the considered methods perform the variable selection quantitatively. We do this by simply counting the number of selections made incorrectly plus the number of unselected true variables, that is:

$$discordance = \#false\ positives + \#false\ negatives \ .$$

## 4.2 Methods Preparation

In Chapter 3 we present more methods than we compare here. As we are seeking for an optimal approach, in addition to the variable selection ability and the prediction accuracy we also take into account the computational complexity of the methods. The exact methods we are going to test are listed in Table 4.1.

| **Method** | **Parameters** | **Multiresponse** |
|---|---|---|
| Least Squares (Forward Stepwise) | - | - |
| Ridge regression | $\lambda$ | - |
| The LASSO (LARS) | - | - |
| The ElasticNet (LARS-EN) | $\lambda$ | - |
| The ∞-norm MRSR algorithm | - | yes |
| The L2-norm MRSR algorithm | - | yes |
| The Multi-task LASSO (BCD) | $\lambda$ | yes |

**Table 4.1**. *The methods included into the current testing.*

**LARS.** It is possible to compute the LASSO estimates with both the 'Shooting' and the LARS algorithms. In our testing we choose the latter. There are two reasons for this choice. Firstly, 'Shooting' implies a tuning parameter. The popular method for selecting tuning parameters is *k-fold cross-validation*. For performing $k$ folds the method requires the amount of computations equal to the $k$ Least Squares fits. This may sufficiently slow down the whole regression process, especially, if the number of predictor and response variables in the dataset is large. A dataset $(\boldsymbol{T}_{50\times100}, \boldsymbol{G}_{50\times1000}, q)$ may be considered as a large dataset for the purposes of this method.

Higher values of tuning parameters produce more sparse solutions. However, it does not guarantee the exact number of variables to be included into the model. Our primary aim is to compare the variable selection ability of the presented techniques. Therefore, to make it in a more fair way, we choose the LARS algorithm for obtaining the LASSO estimates, because it allows to predefine the expected number of variables in the model.

**LARS-EN.** The statements above also hold for the ElasticNet. Its tuning parameters must be cross-validated from the two-dimensional grid. However, using the LARS-EN algorithm, only one parameter is sufficient, because the ElasticNet penalty is the regularized LASSO (see Section 3.1.4). LARS produces the LASSO solution without any additional inputs, and by specifying $\lambda$ for the LARS-EN algorithm we just control the amount of the ridge penalty applied on the LASSO estimates, what results in the ElasticNet solution.

**Ridge regression.** Ridge regression is not designed to produce sparse solutions. To make its variable selection ability comparable to the other methods we do the following: first, for every response we pick up the 'best' $q$ predictors, resulting in different subsets of variables for different responses; second, we select the common subset of the 'best' $q$ predictors according to the sum of the corresponding coefficients for all the responses. In both cases we consider the absolute values of the coefficients. These two approaches are later in this work referred to as '*ridge regression (best subsets)*' and '*ridge regression (common subset)*', respectively.

### 4.2.1 The Computational Issues of the Group LASSO

The Group LASSO is not a multiresponse method. Moreover, we do not expect to have groups of correlated variables in the gene expression data simulations, because we generate variables independently. In Section 3.2 we proposed an approach for input data transformation, which makes it possible to apply the Group LASSO on datasets of type $\boldsymbol{TG}^{(q)}$.

However, the proposed approach is memory consumption. Converting the dataset $(\boldsymbol{T}_{n\times p}, \boldsymbol{G}_{n\times r}, q)$ into single-response requires $O(r^2np)$ memory space. In our case this results in a matrix $\boldsymbol{T}$ of size $(50000 \times 100000)$, what in turn increases the number of computations proportionally. In comparison to other methods, such assumption seems to be a serious drawback. That is why we have performed only a rough testing, to see if using the Group LASSO makes sense at all.

For computing the Group LASSO estimates we have implemented the 'Shooting' algorithm. We have selected the dataset $\boldsymbol{TG}^{(\mathbf{25})}$, and the optimal tuning parameter was empirically chosen to be 0.1. Then we have performed the *group-wise* test of the approach. This means, we have divided the matrix $\boldsymbol{G}$ by columns into $k$ groups ($k \in \{500, 100, 40, 20, 10\}$); for

each pair of a group and the matrix $\boldsymbol{T}$, first, the transformation procedure was applied, and then the algorithm computed the model coefficients corresponding to the currently considered genes. Thus, for a dataset we have run 5 tests, considering 2, 10, 25, 50 and 100 genes at once. The test results are presented in Figure 4.1.

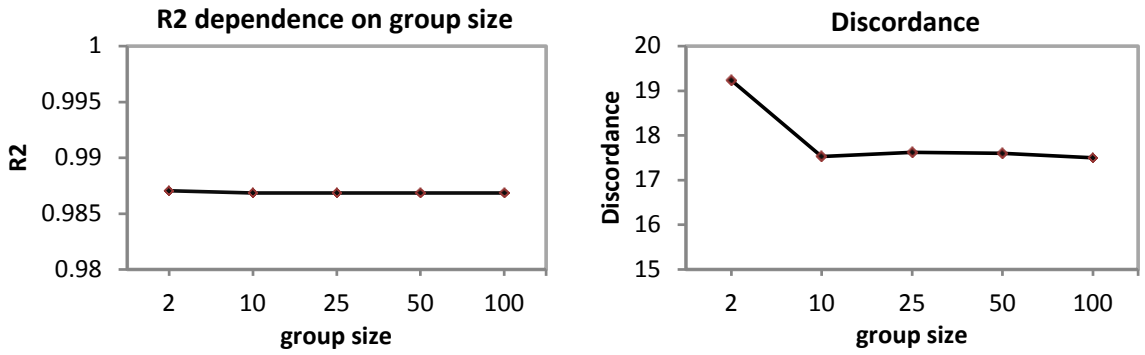


**Figure 4.1**. *The R2 values and the discordance rates corresponding to the Group LASSO while performing the group-wise test.*

The current testing has shown that the accuracy of the Group LASSO estimates does not depend on the number of simultaneously observed genes. The percentage of the explained variance stays the same for all the group sizes, and the discordance rate decreases only in 2 points within the groups of sizes 2 and 10 (see Figure 4.1). So, it turns out we can reduce the memory usage performing the group-wise analysis of a dataset. Nonetheless, the computational burden still holds. Less computations are required if one uses the Group LARS algorithm. However, it is obvious that the problem becomes more serious as the number of variables grows.

Fortunately, comparing our various test results we noticed and later found in [ST06] the confirmation of the fact that the L2-norm MRSR algorithm produces exactly the same solution as Group LARS does. In case of MRSR we do not need any reconstructions of the data, hence, can compute the estimates efficiently. As it will be seen later, these intermediate performance results of the Group LASSO are not perfect. However, we still assess the idea of grouping the variables and compare it with other approaches, but through the L2-norm MRSR technique.

### 4.2.2 Setting up Tuning Parameters

Ridge regression, LARS-EN and BCD require tuning parameters. For that, we have defined the following grid of values: $(1^{-4}, 1^{-3}, 1^{-2}, 1^{-1}, 1, 10, 100, 1000)$. For each of the $\lambda$ we have applied the methods on the datasets $\boldsymbol{TG^{(10)}}$, $\boldsymbol{TG^{(25)}}$ and $\boldsymbol{TG^{(50)}}$. Performance of the methods was assessed by calculating the corresponding coefficients of the determination ($R^2$) and the discordance rates. These metrics were calculated on the training data. Test results are depicted in Figure 4.2.

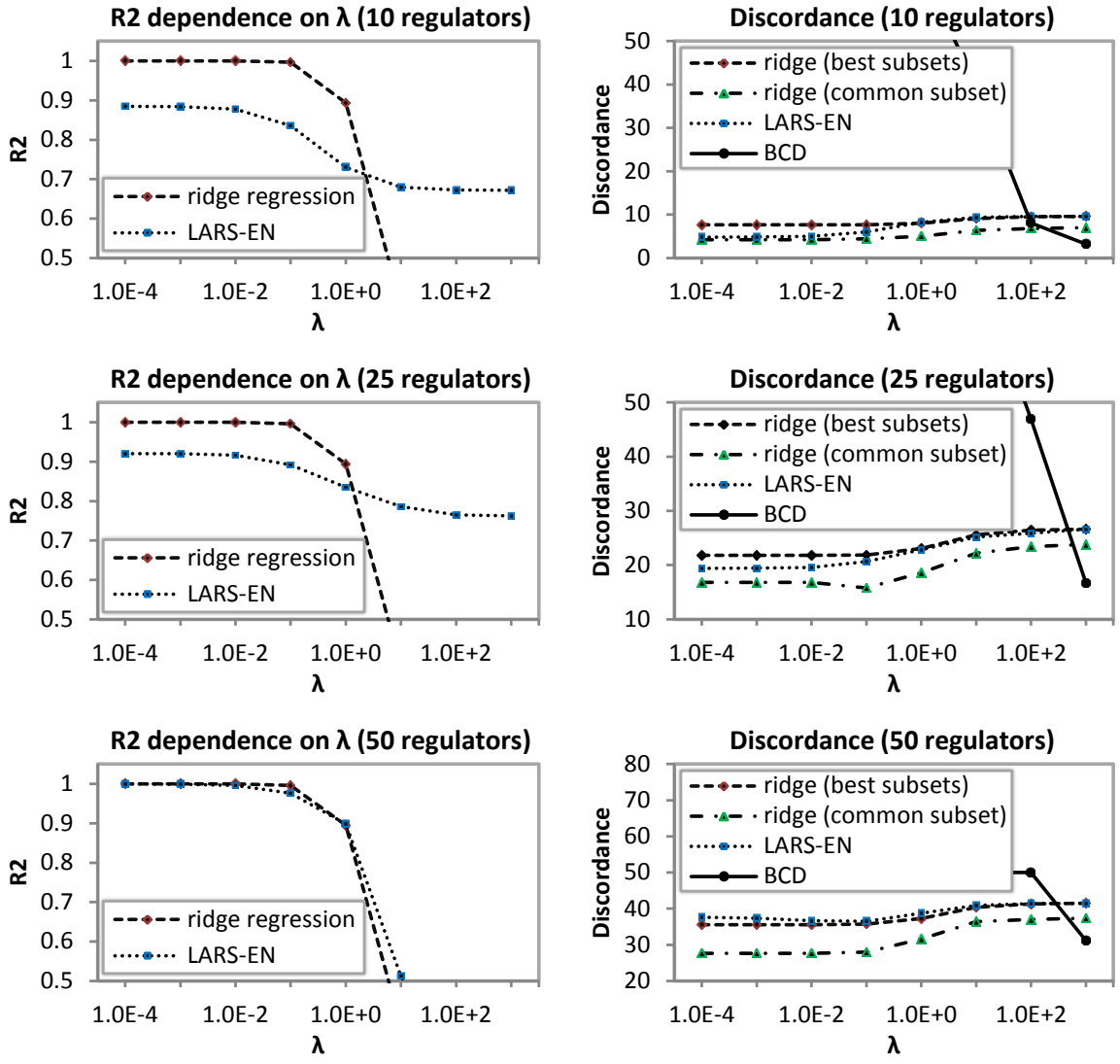


**Figure 4.2**. *The percentage of the explained variance and the discordance rates of methods for different values of λ in case of the datasets* $\boldsymbol{TG^{(10)}}$, $\boldsymbol{TG^{(25)}}$ *and* $\boldsymbol{TG^{(50)}}$.

It can be seen from Figure 4.2 that the ridge regression and the LARS-EN algorithm perform well with $\lambda$ in the range $(1^{-4}, 1^{-3}, 1^{-2}, 1^{-1})$. In order to assess the behavior of the ridge and the ElasticNet penalties, but keeping their impact on the output relatively small, in the further analysis we prefer to use $\lambda = 0.001$ as the tuning parameter for both of them.

In all three cases ($\boldsymbol{TG^{(10)}}$, $\boldsymbol{TG^{(25)}}$ and $\boldsymbol{TG^{(50)}}$) ridge regression with column-wise variable selection (common subset) shows better results than 'ridge regression (best subsets)'. However, we will not exclude the latter from the analysis, and will investigate whether performance difference for these approaches still holds under other testing conditions.

The optimal tuning parameter for the BCD procedure in our test cases seems to be always $1000$. Using the $5$-fold cross-validation technique, we have performed more accurate selection of lambda in the range $(100, 200, \ldots, 1000)$ for the datasets $\boldsymbol{TG^{(25)}}$ and $\boldsymbol{TG^{(50)}}$. The optimal parameters occurred to be $700$ and $500$, respectively. We ignore some more sensitive tunings and assume that the parameters for BCD are $\lambda_1 = 1000$, $\lambda_2 = 700$ and $\lambda_3 = 500$, accordingly.

For smaller values of $\lambda$ the BCD procedure does not provide desired sparsity within the variables. Besides, it takes a lot of steps until convergence. In this intermediate testing we had to interrupt the algorithm at the number of 10000 iterations in case of $\lambda \leq 1$. For $\lambda \in \{1000, 700, 500\}$ and according to the datasets $\boldsymbol{TG^{(10)}}$, $\boldsymbol{TG^{(25)}}$ and $\boldsymbol{TG^{(50)}}$ BCD converges in 9, 24 and 32 steps in average.

## 4.3 Performance

### 4.3.1 Prediction and Variable Selection

We have applied the presented methods on the datasets $\boldsymbol{TG^{(10)}}$, $\boldsymbol{TG^{(25)}}$ and $\boldsymbol{TG^{(50)}}$, measuring their performance over $50$ replications. For evaluation we measured the prediction accuracy of the produced regression models on the test datasets, which were generated independently from the training datasets. The corresponding discordance rates were computed for the estimated model parameters. The results are provided in Figure 4.3.

Looking at Figure 4.3 and Figure 4.4 we can say that almost all the methods perform well. The obtained results justify our expectations about the nature of the considered methods,

and prove the theoretical assumptions provided in Chapter 3.

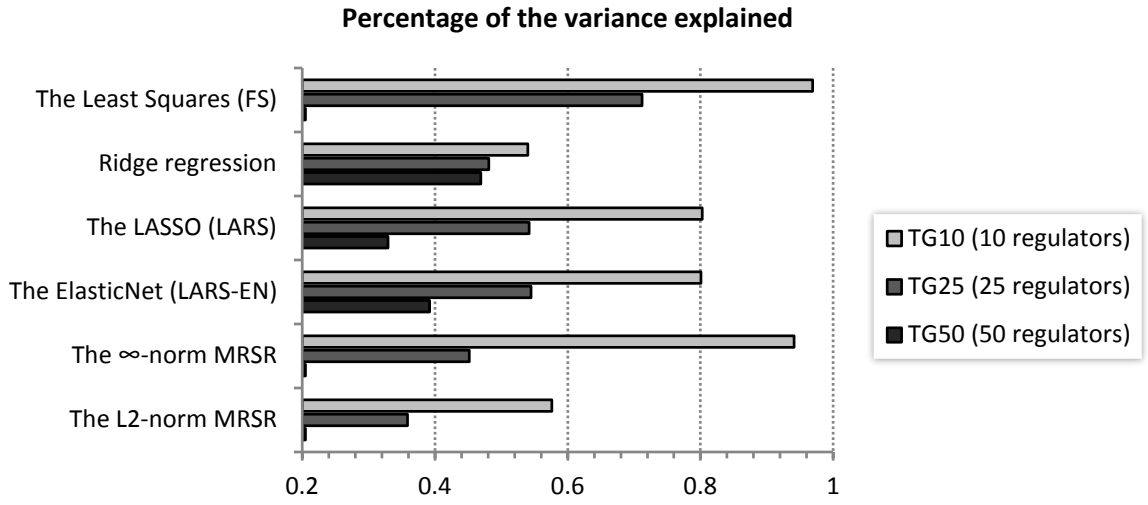


**Figure 4.3**. *Percentage of the explained variance of the methods in case of the datasets* $\boldsymbol{TG^{(10)}}$, $\boldsymbol{TG^{(25)}}$ *and* $\boldsymbol{TG^{(50)}}$.

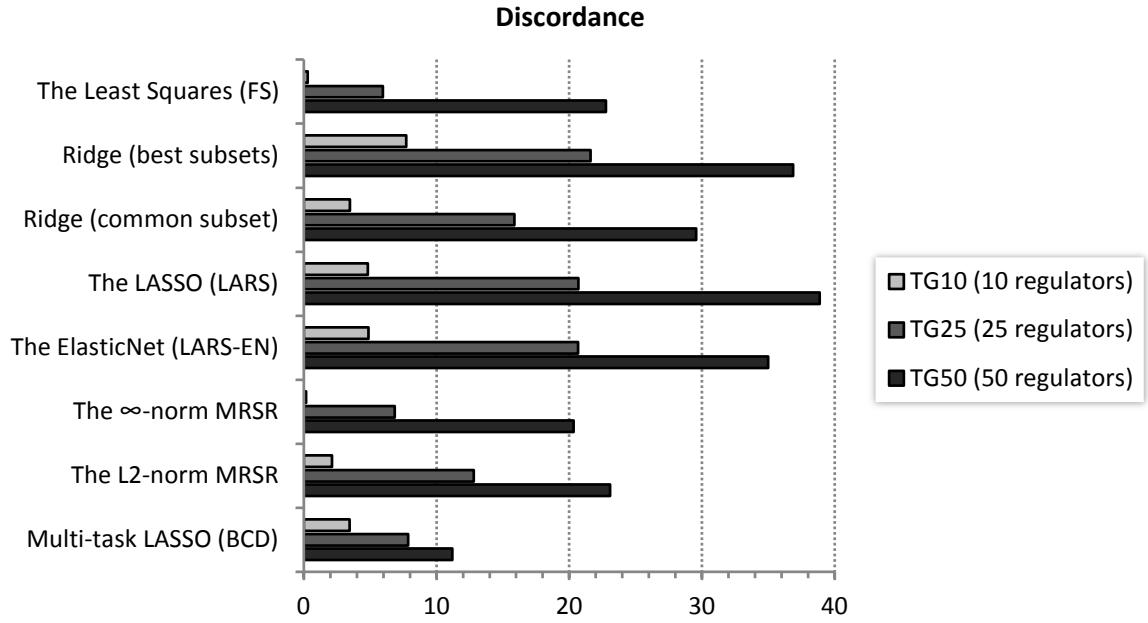


**Figure 4.4**. *Discordance rates of the methods in case of the* $\boldsymbol{TG^{(10)}}$, $\boldsymbol{TG^{(25)}}$ *and* $\boldsymbol{TG^{(50)}}$ *datasets*.

First, we see from Figure 4.3 that using the Forward Stepwise algorithm for computing the Least Squares estimates is quite satisfactory. With smaller number of true variables in the model, the Least Squares-FS approach predicts better, but in $p = n$ case (the dataset $\boldsymbol{TG^{(50)}}$) overfitting occurs. The same also holds for the MRSR algorithm, which is related to Least Squares (see Section 3.1.6). In typical bioinformatics problems, the smallest sets of significant variables are usually of interest. That is why, we can state, that the Least Squares-FS and the MRSR methods are still applicable to the data.

The models produced by ridge regression do not seem to predict perfectly. But, as it can be seen, their predictive performance is stable. This means that the proportion of the variance they explain is similar for all the datasets. And in case of 50 true regulators (the dataset $\boldsymbol{TG^{(50)}}$) the ridge regression even outperforms all the other methods.

The LASSO and the ElasticNet techniques show almost the same results until the $\boldsymbol{TG^{(50)}}$ dataset. In this case the ElasticNet predicts better, what follows from the definition (see Section 3.1.4). In comparison to ridge regression, the LASSO penalty seems to be more efficient in case of the $\boldsymbol{TG^{(10)}}$ dataset. Actually, this is the known fact that the LASSO provides the best solution when the estimated model is assumed to be very sparse.

As our primary aim in context of the current work is the methods' variable selection ability, the more important for us are the discordance rates they provide. From Figure 4.4 we see, that the variable selection the Least Squares method performs is quite adequate – the discordance rates of the produced estimates are moderate. From 10 true regulators (the dataset $\boldsymbol{TG^{(10)}}$) the method selects 9 of them correctly, and in case of 50 true regulators it is mistaken in 11 selections, what is 22%. The same also holds for the both types of the MRSR algorithm, with the difference that the ∞-norm MRSR algorithm outperforms its relatives a little bit, providing the lower discordance in case of the $\boldsymbol{TG^{(50)}}$ and giving the 100% result in case of the $\boldsymbol{TG^{(10)}}$ datasets**.**

In terms of the discordance the ElasticNet solution is again almost identical to the LASSO one, except the case of 50 true regulators – the ElasticNet makes mistakes for 4% less, what is expected. We introduced the two approaches of making the solution of ridge regression sparse in the previous section and observed that picking up the best subsets is dominated by taking the best subset from all the coefficients according to the sums of their absolute values.

Comparing the latter with the LASSO and the ElasticNet techniques, we see that its discordance rate is lower in all three cases of the $\boldsymbol{TG^{(10)}}$, $\boldsymbol{TG^{(25)}}$ and $\boldsymbol{TG^{(50)}}$ datasets. At the same time, the 'best subsets' approach by the results is the worst from all the other methods.

Note, that the exact discordance rates are not comparable among the datasets with the different number of true variables. Let us consider the results of the BCD procedure. The discordance rate when there are 10 true regulators in the dataset is ~4 and the discordance is 11 when there are 50 true regulators. This means that in the first case the method selects 8 regulators correctly out of 10, and in the second case – 44.5 out of 50. In the percentage ratio these values are 80% and 89%, respectively. Thus, it comes out that BCD performs the best, being applied on the $\boldsymbol{TG^{(50)}}$ dataset.

In conclusion, we would like to point out two superior methods in terms of the variable selection task. These are the ∞-norm MRSR algorithm and the BCD procedure. The first one gives 100% result in case of the $\boldsymbol{TG^{(10)}}$ dataset, and the second outperforms in case of the $\boldsymbol{TG^{(50)}}$ datasets. When the number of true regulators in the dataset is 25 both methods provide almost the same discordance rates. Besides, we could also assume that choosing more accurate tuning parameters for the BCD procedure could improve its performance. However, this operation in turn requires more computations.

### 4.3.2 Comparison of Estimates

We have performed the test described further in order to assess the presented techniques in terms of the stability of the solutions they produce. For that we have generated the dataset $(\boldsymbol{T}_{100\times100}, \boldsymbol{G}_{100\times1000}, 25)$ and then have split it into two parts $\boldsymbol{TG_1^{(25)}}$ and $\boldsymbol{TG_2^{(25)}}$, each one containing 50 experiments. We have applied the methods on both the sub-datasets, measuring the difference between the two sets of estimated model parameters each method produces. Actually, we expect such difference to be as small as possible.

For the purposes described above we have used the *MSE* metric in the following way: $MSE(\widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\beta}}_2) = E\left[(\widehat{\boldsymbol{\beta}}_1 - \widehat{\boldsymbol{\beta}}_2)^2\right]$, where $\widehat{\boldsymbol{\beta}}_1$ and $\widehat{\boldsymbol{\beta}}_2$ are the sets of coefficients computed using the $\boldsymbol{TG_1^{(25)}}$ and $\boldsymbol{TG_2^{(25)}}$ datasets, respectively. The current testing results are provided in Figure 4.5.

Performing the comparison of the estimates produced by each method, we see that the $MSE$ value in case of the Least Squares fit is twice higher than all the other methods explore. This, of course, leads to the overfitting problem, but we still consider this result to be acceptable. In overall, all the other methods, except the ∞-norm MRSR algorithm, perform well under the defined conditions. The ∞-norm MRSR algorithm operates but the best, providing the estimates different only in $0.055$ *MSE* value.

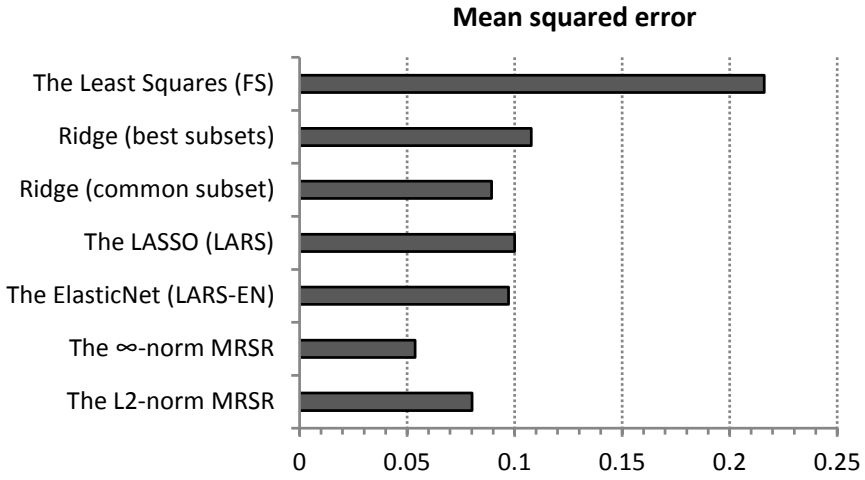


**Figure 4.5**. *The MSE of two sets of regression parameters estimated using the relative datasets* $\boldsymbol{TG_1^{(25)}}$ *and* $\boldsymbol{TG_2^{(25)}}$. *This metric shows how similar the sets of the estimates produced by the presented methods are (smaller value of MSE stands for the less difference).*

## 4.4 Varying the Noise Level

We examine roughly the methods' dependence on the external influences, that are usually unbeneficial for the performance, by adding Gaussian noise ($\varepsilon \sim N(0, \sigma^2)$) to the dataset $\boldsymbol{TG^{(25)}}$. We vary the $\sigma$ from 0.0 to 1.0 and examine the change in the coefficient of determination (R2) values and the discordance rates corresponding to the methods we analyze.

From the obtained results (see Figure 4.6) we can conclude that the prediction accuracy in case of all the methods decreases proportionally to the noise variance. The Least Squares method and the ridge regression show more quick recession, while the MRSR algorithm seems to be more resistant to noise (mainly L2-norm MRSR).

In terms of the discordance the stable rates correspond to the L2-norm MRSR algorithm, ridge regression (common subset) and the BCD procedure, what means their variable selection ability is not affected by the noise sufficiently.

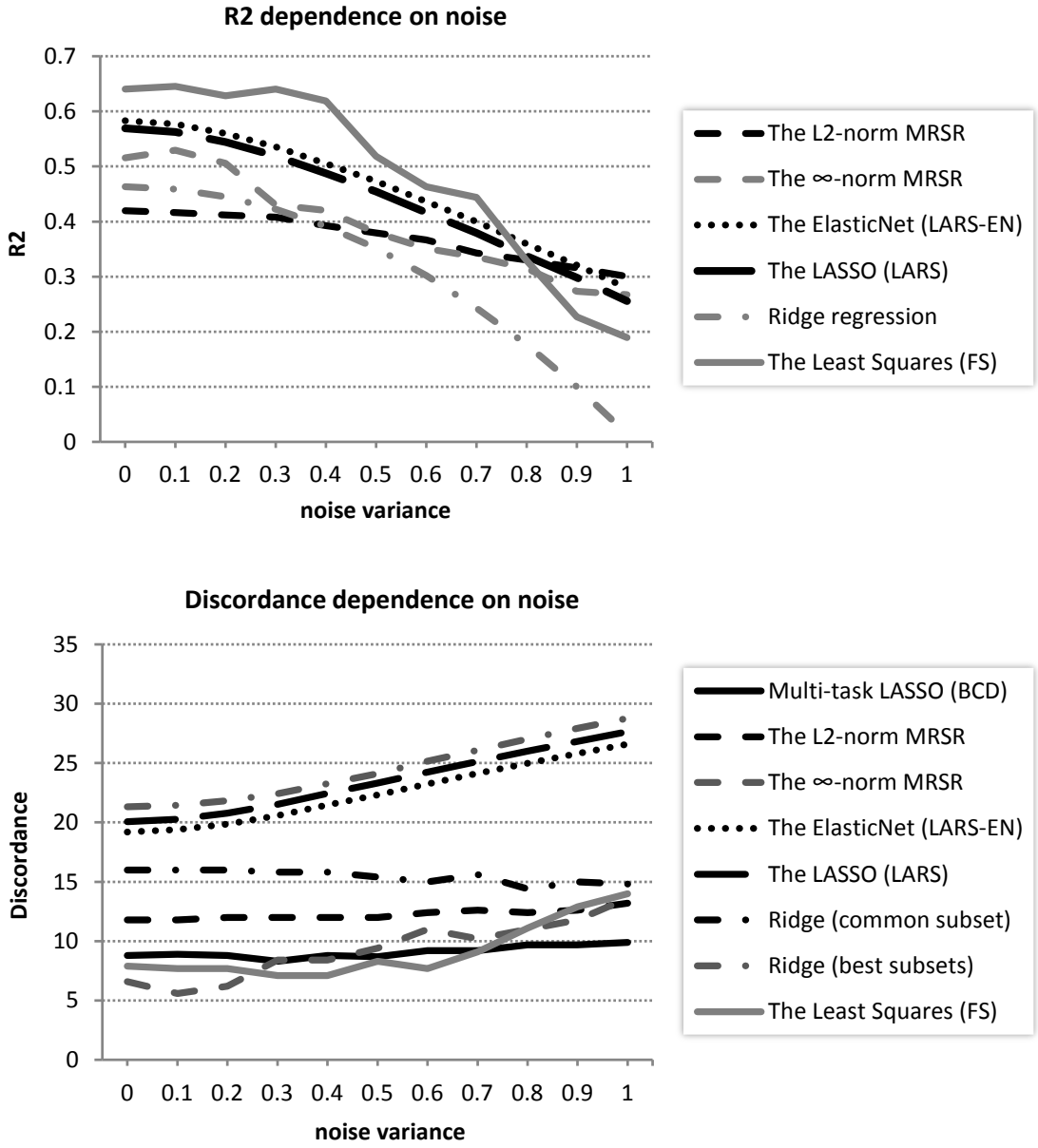


**Figure 4.6**. *The influence of Gaussian noise* $\varepsilon \sim N(0, \sigma^2)$ *on the prediction accuracy of the models produced by the methods (the upper plot) and on the discordance rates corresponding to the variable selections performed by the methods (the bottom plot).*

# Chapter 5

# Application on Real Data

In the previous chapter we examined the behavior of the methods, consider in this work, by applying them on artificial data. In order to have a more complete overview of the presented methods efficiency, we have to analyze their performance on real life problems. We consider two classical yeast cell cycle microarray datasets – the dataset by Spellman, et al. [SSZ+98] and the dataset by Gasch, et al. [GSK+00]. These datasets are thoroughly studied, and are, therefore, are often used as validation frameworks for the techniques being developed.

Unfortunately, it is not possible to measure the performance of the methods on real data in the same way as we did it for artificial data, due to the lack of ground truth to validate against. Besides, we must remember that the use of linear models for inferring regulators is just an approximation. However, we can assess the methods performance by exploring the stability of the solutions they produce. Whenever the method provides similar outputs for similar microarray datasets we count it as evidence of biological relevance of the results.

Analysis we describe below is similar to the one introduced in Section 4.3.2. We split the datasets experiment-wise into two parts; apply the algorithms on both sub-datasets; and compare the solutions produced by each method. We perform only the quantitative analysis, i.e., we just examine the variable selection ability of the techniques without taking into account their prediction accuracy.

## 5.1 The Spellman Dataset

The aim of the experiments performed by Spellman, et al. [SSZ+98] was to create a catalog of *Saccharomyces cerevisiae* yeast genes whose transcription varies periodically within the cell cycle. As a result of this project a dataset was constructed, which contains the expression values of 6178 genes measured under 77 microarray experiments. In the current analysis we use pre-processed data from this dataset.

The missing values in the dataset we use are imputed using the *KNN-impute* algorithm. The dataset contains the expression values of 6084 genes out of initial 6178 according to the sequence data availability in SGD [SGD] on July 1, 2007. 318 of these genes have *transcription regulator activity* GO annotation (on July 1, 2007). We have constructed the corresponding matrices $\boldsymbol{T}$ and $\boldsymbol{G}$, containing the expression values of 318 transcription factors and 5766 genes under 77 experiments, respectively. We have divided the matrices $\boldsymbol{T}$ and $\boldsymbol{G}$ by rows into two parts, thus obtaining two datasets: ($\boldsymbol{T}_{39\times318}, \boldsymbol{G}_{39\times5766}$) and ($\boldsymbol{T}_{38\times318}, \boldsymbol{G}_{38\times5766}$), referred to as $\boldsymbol{TG_1}$ and $\boldsymbol{TG_2}$.

We apply the Least Squares method (using the Forward Stepwise algorithm), the ridge regression method (where we pick the common subset of variables according to the sum of absolute values of the corresponding coefficients, see Section 4.2), the LARS-EN algorithm, the ∞-norm MRSR algorithm and the Blockwise Coordinate Descent procedure (BCD) on both datasets described above.

All the methods, apart from BCD, make it possible to specify the desired number of variables to be included into the model. We choose to select the 100 most 'influential' transcription factors out of 318. For that, we have empirically chosen the tuning parameter $\lambda$ for BCD to be 1700. With such $\lambda$ the procedure selects 102 transcription factors in case of the $\boldsymbol{TG_1}$ dataset and 104 – in case of the $\boldsymbol{TG_2}$ dataset. The tuning parameters for ridge regression and the ElasticNet (LARS-EN) were empirically chosen from the range ($1^{-4}, 1^{-3}, 1^{-2}, 1^{-1}, 1, 10, 100, 1000$). Both methods produce models with the lowest discordance rates using $\lambda = 100$.

After applying the methods we computed the discordance rate of both models produced by each method. According to these rates we can define the ratio of the commonly selected transcription factors by a method and the transcription factors selected differently. The corresponding ratios are presented in Figure 5.1.

From Figure 5.1 we can see that the Least Squares approach is dominated by all the other methods. The best results are shown by the ridge regression and the ElasticNet techniques. Note, that LARS-EN is the only algorithm of all which does not produce the set of model parameter estimates common for all the responses. For every gene we select the set of regulators separately. However, even then LARS-EN provides similar discordance rates in

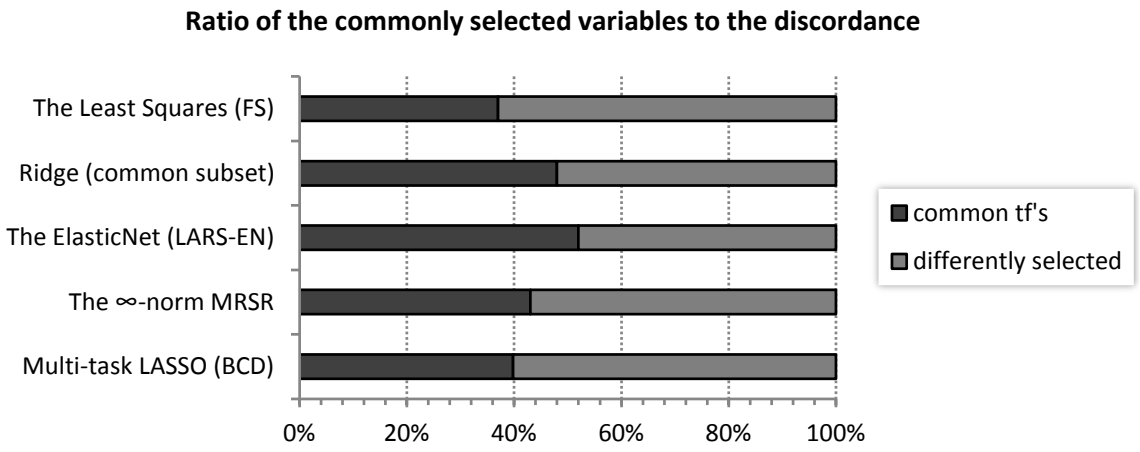


**Figure 5.1** *The percentage of the commonly selected variables by the methods (the Spellman dataset)*

comparison to other methods. For example, the discordance of the models produced by the Least Squares method in our case is 126, while the models by LARS-EN differ in 128 selections. In order to perform a more fair comparison, we have processed the LARS-EN solution in the same way as the ridge regression solution, by taking the 100 most 'influential' variables according to the sums of the absolute values of the corresponding coefficients. The discordance is then equal to 96, which means that 48 out of 100 regulators are selected by the algorithm differently and the other 52 are common.

In case of the datasets $\boldsymbol{TG_1}$ and $\boldsymbol{TG_2}$ each method selects on average 44 of the same transcription factors from both of them (see Figure 5.1). We refer to the commonly selected transcription factors as 'regulators'. There are 12 'regulators', which are defined by at least 4 algorithms simultaneously. If we consider the selection performed by at least 3 methods sufficient, then this number is 33.

We do not consider the results we have obtained by applying the presented techniques on the dataset by Spellman to be perfect, however, we see they are quite adequate, taking into account the fact that we are dealing with the $p \gg n$ case (the number of experiments in the datasets we analyze is 8 times less than the number of transcription factors). We present the list of the 5 most selectable regulators in Table 5.1.

| Protein | Description |
|---|---|
| STB1 | Protein with a role in regulation of MBF-specific transcription at Start, phosphorylated by Cln-Cdc28p kinases in vitro (ID:YNL309W) |
| ACE2 | Transcription factor that activates expression of early G1-specific genes, localizes to daughter cell nuclei after cytokinesis and delays G1 progression in daughters, localization is regulated by phosphorylation (ID:YLR131C) |
| GAT1 | Transcriptional activator of genes involved in nitrogen catabolite repression (ID:YFL021W) |
| HCM1 | Forkhead transcription factor that drives S-phase specific expression of genes involved in chromosome segregation, spindle dynamics, and budding (ID:YCR065W) |
| TEC1 | Transcription factor required for full Ty1 expression, Ty1-mediated gene activation, and haploid invasive and diploid pseudohyphal growth (ID:YBR083) |

**Table 5.1**. *Proteins correspondent to the transcription factor genes selected by the methods.*

## 5.2 The Gasch Dataset

The dataset we use for the second analysis contains microarray data provided by Gasch, et al. [GSK+00], who measured the expression of genes in the yeast *Saccharomyces cerevisiae* responding to a set of external physical impacts. The dataset contains the expression values of 53 transcription factors and 1410 genes collected under 173 experiments.

We reconstruct the Gasch dataset in the same way as described in the previous section (see the section 5.1) into the corresponding $\boldsymbol{TG_1}$ and $\boldsymbol{TG_2}$ sub-datasets. We apply the same methods on $\boldsymbol{TG_1}$ and $\boldsymbol{TG_2}$ and explore the discordance rates of the models produced. We choose to select 25 the most 'influential' transcription factors out of 53. The obtained results are presented in the Figure 5.2.

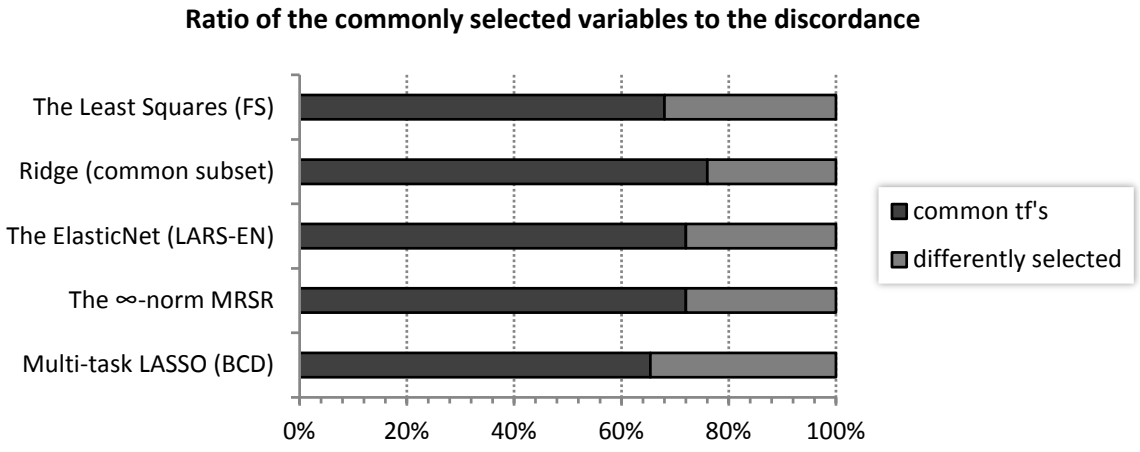


**Figure 5.2** *The percentage of the commonly selected variables by the methods (the Gasch dataset)*

The discordance rates obtained using the dataset by Gasch, et al. [GSK+00] are slightly lower than the discordance rates in case of the dataset by Spellman, et. al [SSZ+98] (see Section 5.2). The difference in performance of the presented methods can be explained by the fact that in the current case the number of experiments in the datasets $\boldsymbol{TG_1}$ and $\boldsymbol{TG_2}$ exceeds the number of transcription factors ($p < n$ case).

Ridge regression performs the best, selecting 19 'regulators' out of 25 (with the tuning parameter $\lambda = 0.0001$). The LARS-EN algorithm produced the model with the lowest discordance rate with $\lambda = 0.0001$. In case of the $\boldsymbol{TG_1}$ dataset the BCD procedure required

the tuning parameter to be 1200 in order to select 25 transcription factors, and in case of the $\boldsymbol{TG_1}$ dataset this value occurred to be 1600.

The common set of 'regulators' defined by at least 4 of the considered methods contains 9 transcription factors. This number in ratio to the number of the 'regulators' expected (25) is 5 times higher than in case of our previous analysis (Spellman dataset). The most influential selections are presented in Table 5.2.

| **Protein** | **Description** |
|---|---|
| BMH1 | 14-3-3 protein, major isoform; controls proteome at post-transcriptional level, binds proteins and DNA, involved in regulation of many processes including exocytosis, vesicle transport, Ras/MAPK signaling, and rapamycin-sensitive signaling (YER177W) |
| ATG1 | Protein serine/threonine kinase required for vesicle formation in autophagy and the cytoplasm-to-vacuole targeting (Cvt) pathway; structurally required for pre-autophagosome formation; during autophagy forms a complex with Atg13p and Atg17p (YGL180W) |
| PPT1 | Protein serine/threonine phosphatase with similarity to human phosphatase PP5; present in both the nucleus and cytoplasm; expressed during logarithmic growth; computational analyses suggest roles in phosphate metabolism and rRNA processing (YGR123C) |
| XBP1 | Transcriptional repressor that binds to promoter sequences of the cyclin genes, CYS3, and SMF2; expression is induced by stress or starvation during mitosis, and late in meiosis; member of the Swi4p/Mbp1p family; potential Cdc28p substrate (YIL101C) |
| USV1 | Putative transcription factor containing a C2H2 zinc finger; mutation affects transcriptional regulation of genes involved in protein folding, ATP binding, and cell wall biosynthesis (YPL230W) |

**Table 5.2**. *Proteins correspondent to the transcription factor genes selected by the methods.*

# Summary

Processes that take place in the biological cell are very complex and are not yet completely understood by the contemporary molecular biology. What is known is that the most of the activities in the cell are performed by proteins. The exact structure of a protein determines its function. The structure of each protein is encoded by the particular region (gene) of the DNA molecules stored in the cell. The process of mapping from a gene to the particular protein is referred to as gene expression. Despite the fact that every cell of an organism contains exactly the same copy of the DNA, and, hence, the same genes, the differently specialized cells produce different proteins. This is because the different genes are expressed among different cell types. Besides, there is a complex regulation machinery inside each cell, which defines when and what protein to produce. This machinery uses a certain group of proteins knows as transcription factors. These proteins influence the production of the other proteins.

The question of which genes encode the transcription factors is of great interest for contemporary molecular biology. With the development of the high-throughput techniques, such as the microarray technology, studying the processes underlying the cell cycle has become more convenient. Unfortunately, converting the data produced by the microarray experiments into knowledge is not a trivial task. Usually, it implies a series of biological studies with an application of statistical techniques.

In this work we addressed the problem of inferring influential transcription factor genes from microarray data. Assuming that transcription factors have the most effect on the expression of genes, we modeled the expression of each gene as a linear function of the expression of transcription factor genes. We examined the applicability of the linear model to biological data. Experiments on both the simulated and the real gene expression datasets demonstrated that the linear approximation is quite interpretable.

Microarray data is usually high-dimensional and contains noise and measurement errors. This makes the application of the classical linear regression methods complicated. In this work we described several techniques which are proposed to outperform the ordinary Least Squares approach in terms of the variable selection ability and the prediction accuracy. These are ridge regression, the LASSO, the ElasticNet, MRSR and the Multi-task LASSO.

We studied the properties and the computational complexity of the presented algorithms in order to choose the most reliable of them. We analyzed performance of the techniques on both real and artificial data. The obtained test results demonstrated that almost all of the considered methods are efficient and are able to produce biologically meaningful solutions.

# Oluliste transkriptsioonifaktorite tuvastamine lineaarsete mudelite abil

**Magistritöö (30EAP)**

**Nikita Shipilov**

**Resümee**

Mitmerakuline organism tavaliselt koosneb erinevatest rakutüüpidest, kuid kõikides rakkudes on sama DNA. Vastavalt teatud DNA lõikudele (geenidele) sünteesitakse rakus valke. Erinevat tüüpi rakud toodavad erinevaid valke vastavalt sellele, mis geenid rakus aktiivsed on. Geenide aktiivsust määravad spetsiifilised valgud – transkriptsioonifaktorid. Nad on võimelised seonduma DNA ahelaga teatud kohtades ning aktiveerima või deaktiveerima vastavaid geene.

Transkriptsioonifaktorite tuvastamine on aktuaalne probleem molekulaarbioloogias. Tänapäeval võimaldavad erinevad tehnoloogilised saavutused jälgida rakus toimuvaid protsesse, kuigi nende analüüs ei ole triviaalne ülesanne, mis vajab erinevate teaduste kaasamist. Nende hulgas on ka statistika.

Käesolevas töös kirjeldatakse lineaarsete mudelite kasutamise võimalusi oluliste transkriptsioonifaktorite tuvastamiseks mikrokiibi andmetest. Lineaarse mudeli parameetreid võib käsitleda kui transkriptsioonifaktorite olulisust määravaid näitajaid. Töös on vaadeldud erinevad lineaarregressiooni meetodid koos nende iseärasuste põhjaliku kirjeldusega ning on analüüsitud nende sobivus bioloogiliseks rakenduseks.

# References


[**Bak99**] Bakin, S. *Adaptive regression and model selection in data mining problems.* Ph.D. thesis, The Australian National University, 1999.

[**B**V04] Boyd, S. and Vandenberghe, L. *Convex Optimization.* Cambridge University Press, New York, USA, 2004.

[EHJ04] Efron, B., Hastie, T., Johnstone, I. and Tibshirani. R. Least angle regression (with discussion). *Annals of Statistics*, 32: 407-451, 2004.

[Fu98] Fu, W. J. Penalized regressions: The bridge versus the lasso. *Journal of Computational and Graphical Statistics*, 7(3): 397-416, 1998.

[HK70] Hoerl, A.E. and Kennard, R.W. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12: 55-67, 1970.

[LPZ09] Liu, H., Palatucci, M. and Zhang, J. Blockwise Coordinate Descent Procedures for the Multi-task Lasso, with Applications to Neural Semantic Basis Discovery. *International Conference on Machine Learning*, 2009.

[OPT00] Osborne, M. R., Presnell, B. and Turlach, B. A. On the LASSO and its dual. *Journal of Computational and Graphical Statistics*, 9: 319–337, 2000.

[PH06] Park, M. Y. and Hastie, T. Regularization path algorithms for detecting gene interactions. Technical report, Department of Statistics, Stanford University, 2006.

[ST06] Similä, T. and Tikka, J. Common Subset Selection of Inputs in Multiresponse Regression. *International Joint Conference on Neural Networks*, 2006.

[ST05] Similä, T. and Tikka, J. Multiresponse sparse regression with application to multidimensional scaling. *Proceedings of the 15th International Conference on Artificial Neural Networks*, vol. 3697 of *Lecture Notes in Computer Science*, pp. 97–102, 2005.

[Tib96] Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B,* 58: 267-288, 1996.

[TVW05] Turlach, B., Venables, W. N. and Wright, S. J. Simultaneous variable selection. *Technometrics,* 27: 349-363, 2005.

[Wei05] Weisberg, S. *Applied Linear Regression.* 3rd edition. Wiley, New York, USA, 2005.

[YL06] Yuan, M. and Lin., Y. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B,* 68: 49-68, 2006.

[Zha06] Zhang, J. *A probabilistic framework for multi-task learning.* Ph.D. thesis, Carnegie Mellon University, 2006.

[ZH05] Zou, H. and Hastie, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B,* 67: 301-320, 2005.

[SSZ+98] Spellman, P. T. , Sherlock, G., Zhang, M. Q., Iyer, V. R., Anders, K., Eisen, M. B., Brown, P. O., Botstein, D. and Futcher, B. Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. *Mol Biol Cell,* 9(12): 3273-3297, 1998.

[GSK+00] Gasch, A. P., Spellman, P. T., Kao, C. M., Carmel-Harel, O., Eisen, M. B., Storz, G., Botstein, D. and Brown, P. O. Genomic expression programs in the response of yeast cells to environmental changes. *Mol Biol Cell,* 11(12): 4241-4257, 2000.

[SWB02] Someren, E. P., Wessels, L. F., Backer, E. and Reinders, M. J. Genetic network modeling. *Pharmacogenomics,* 3(4):507-25, 2002.

[SGD] SGD Project. "Saccharomyces Genome Database". http://www.yeastgenome.org/