Efficient Sparse-Group Bayesian Feature Selection for Gene Network Reconstruction

Edgar Steiger

Dissertation zur Erlangung des Grades
eines Doktors der Naturwissenschaften (Dr. rer. nat.)
am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

Berlin, 2018

Erstgutachter: Prof. Dr. Martin Vingron
Zweitgutachter: Prof. Dr. Hanspeter Herzel
Tag der Disputation: 20. Juli 2018

Contents

List of Figures

List of Tables

1. Introduction

"Life" is defined by several traits that distinguish the animate being from the inanimate world. One of these traits is the ability to respond to external stimuli - but how does a sunflower turn towards the sun, why does a human feel tired when the alarm clock goes off in the morning and how does a yeast cell start the fermentation process in a sugar-rich, oxygen-depraved environment? These complex responses are possible since every organism contains an equally complex manual coded in the DNA (deoxyribonucleic acid), the molecular blueprint for life on earth. The DNA is a very large molecule, present in the organism's every cell. Information is coded in segments in the DNA, where the most prominent segments are the genes. Most genes are silent or inactive most of the time, but a certain stimulus to the cell from the outside might lead to a gene's expression, and this expression will in turn also act as a stimulus or impediment for other genes, and so on, setting the complex system into motion that leads to the cell's and subsequently the organism's response to the outside stimulus. This entangled system of genes reacting, activating or inhibiting other genes, within the cell or DNA, is called a gene regulatory network. And while the particular sequence of the DNA and most genes for many organisms are identified, the knowledge about the higher level network structure of these genes is often incomplete. Carefully designed biological experiments can lead to evidence for the existence or absence of single interactions between genes in these networks, but the problem of discovering the connections between many genes at once from large-scale experimental data is known as reverse engineering of gene regulatory networks or gene network reconstruction. Knowledge about the particular gene network structure for different organisms is important, since this knowledge allows to learn about the evolution and development of life, predict responses to new stimuli or infer targets for drugs. For these reasons, gene network reconstruction is an important part in the basic research of bioinformatics, at the

intersection of molecular genetics, statistical inference and computer science.

For the most part, existing methods for gene network reconstruction comprise of methods that draw a network graph, where genes are represented by nodes and edges between nodes correspond to interactions between genes. Methods for network reconstruction need to validate these edges from given experimental data, in most cases this data consists of measurements of the genes' expressions, where the expressions themselves as well as their measurements are inherently stochastic. Furthermore, the number of possible connections between the considered genes is vast, while the actual number of true interactions is by orders of magnitude smaller, that is, gene network graphs are sparse in regard to the number of edges. In statistics, the identification of correct sparsity patterns is also called feature selection, where the genes are treated as features or variables and their connections are associated with parameters. The two main (not mutually exclusive) paradigms in statistics are the frequentist and the Bayesian approach, where the frequentist approach infers point estimates of the parameters, based on samples, as approximations to the universal true parameters, while the Bayesian approach treats the parameters as stochastic entities themselves, and as such gives probabilistic interpretations of parameters based on prior beliefs in the framework of Bayes' theorem. In feature selection, the frequentist approach is mostly represented by the lasso method, while a Bayesian approach to feature selection is the spike-and-slab method. In the context of feature selection for network reconstruction, the lasso has been intensively studied and applied in different forms, while the application of the Bayesian approach has been hampered by its reliance on Gibbs sampling, which is an algorithm to infer parameters within the Bayesian framework, but Gibbs sampling needs to run for long times to give reliable results. As such, the Bayesian approach was ill-suited for the task of large-scale gene network inference. An alternative to Gibbs sampling is expectation propagation, a deterministic algorithm not relying on stochastic sampling, which can potentially decrease the run-time of a Bayesian approach at the cost of a larger mathematical overhead and a less straight-forward implementation.

A further issue to consider is the following: The standard lasso and the standard spike-and-slab approach to feature selection treat the set of all features as a

whole and enforce sparsity on the features without taking advantage of structural associations between the features. One way to allow for a structure in the set of features is a grouping, where features are grouped by some defined prior information. For genes as features, this grouping could stem from co-binding affinity, clustering by similar gene expression or known functional similarity. Including this grouping information in the process of feature selection and subsequently network reconstruction should be instructional.

1.1. Research objective

This thesis presents a Bayesian framework for feature selection when features are grouped and are sparse on the between- and within-group level, along with a deterministic algorithm to calculate the parameters. We apply this new method to the problem of network reconstruction by neighborhood selection from gene expression data and also extend it to time series data. Our novel contributions are:

- a **fast and deterministic algorithm** for sparse-group Bayesian feature selection in linear regression and its **efficient implementation**,
- application of **Bayesian neighborhood selection** to the problem of (gene) network reconstruction,
- furthermore, application of **neighborhood selection with grouping information** for network reconstruction (Bayesian or lasso),
- analysis of the performance of **Bayesian and sparse-group feature selection for vector autoregressive models** and network reconstruction from temporal (oscillatory) data.

1.2. Thesis overview

This chapter gave a short introduction into the scope of this thesis. The following Chapter 2 will give a more detailed biological background for gene regulation and

gene regulatory networks. Chapter 3 presents several aspects of and approaches to the problem of feature selection in the context of (gene) network reconstruction, that is the lasso methods, Bayesian frameworks and the problem of grouped variables. Chapter 4 is at the heart of this thesis and our main contribution: We present the theoretical background for the proposed Bayesian framework and the corresponding efficient algorithm for feature selection in the presence of grouped variables. Chapter 5 presents the evaluation of this new method on simulated and experimental data with comparisons to other existing methods. The following Chapter 6 extends this framework to time series data and the vector autoregressive model and includes an analysis of oscillations, too. Finally we will conclude and discuss our findings and potential open questions in Chapter 7.

2. Biological Context

This chapter introduces shortly the biological background that is needed to understand the motivation and application of the methods presented in Chapters 3 to 6.

2.1. Genes and gene expression

The DNA (deoxyribonucleic acid, [92]) of a living organism is the long-stranded sequence of a combination of the four nucleotides A, C, G and T (which are named after their respective bases adenine, cytosine, guanine and thymine). A gene is a specific region within this DNA sequence [2]. The process of transcription transcribes the sequence on the DNA associated with a particular gene to Pre-mRNA (Pre-messenger ribonucleic acid), which is in structure similar to the DNA (but with an uracil nucleotide instead of thymine). After some modifications the mature mRNA (messenger RNA) will be translated by the process of translation to the final protein product which the gene on the DNA codes for. The protein might be subject to additional modifications such as folding or binding to other proteins. This very short description of the central dogma of molecular biology [17] or gene expression as **gene transcribed to RNA, RNA translated to protein** is by no means complete and its details are still subject to ongoing research.

2.2. "Gene-gene interaction" and the gene regulatory network

A gene regulatory network [19] is a graph model of the interactions between genes, where genes correspond to nodes (vertices) of the graph and interactions are de-

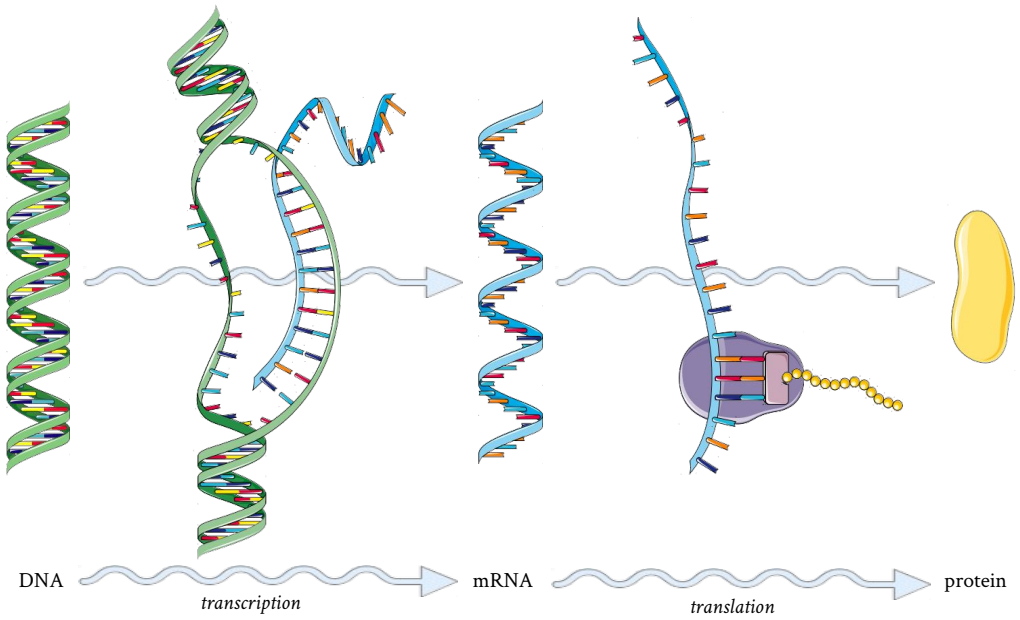

Figure 2.1.: Central dogma of molecular biology: DNA is transcribed to RNA, RNA is translated to protein. CC [52].

picted by edges between genes. In fact, there is no direct interaction between genes on the DNA, but instead the product of the gene expression of one gene (most often a protein) might regulate the transcription or translation of another gene. Thus, a gene-gene interaction in the network graph is a simplification and represents an elaborate interaction machinery of different molecules. As such the gene regulatory network puts a focus on the gene level and hides all other ligands involved. Furthermore, since most often experimental procedures are restricted to one particular aspect of the gene expression process (e.g. measurements of mRNA abundance), it makes sense to restrict the reconstruction of the interaction network on the gene level. Gene regulatory networks exhibit certain characteristics that are similar to network structures of other fields of scientific research such as societal systems and communication, semantics or finance, and as such the statistical study of gene regulatory networks can benefit (from) other disciplines [8].

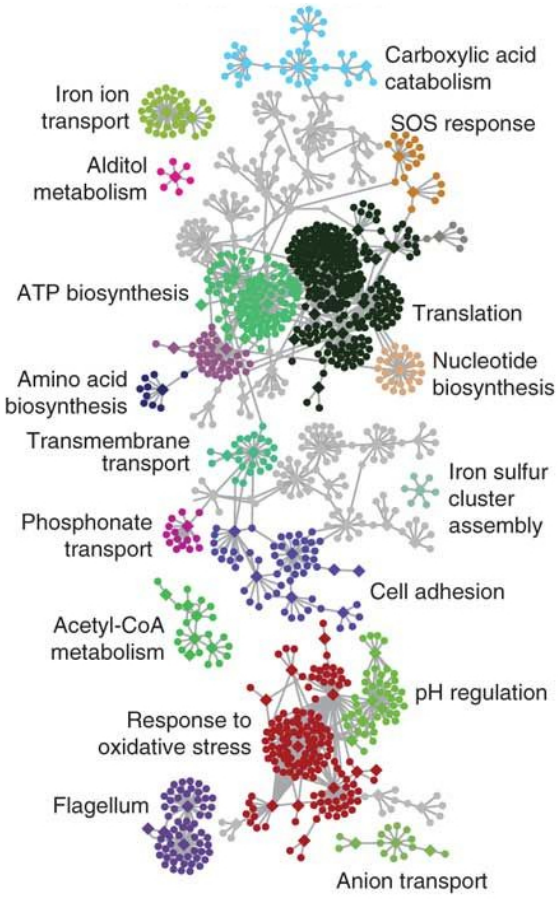

Figure 2.2.: Escherichia coli gene network, with pathway annotation. Transcription factors are depicted as diamonds, while other genes are depicted as circles. Edges correspond to interactions. Adapted from [58].

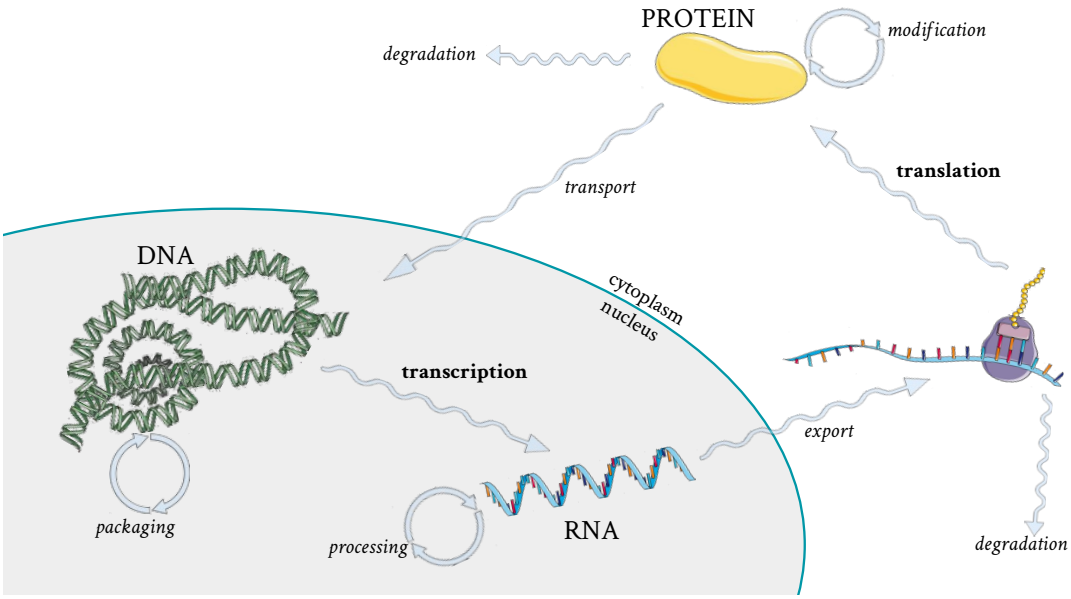

Figure 2.3.: [52] Different steps in the process of gene expression: All of the processes depicted do not happen instantaneously, as such the whole process of gene regulation is a time-delayed one.

2.3. Gene expression as a temporal process

The process of gene expression is time-dependent [7, 79]: The stages of transcription, translation, modifications and degradation at different stages, formation of compounds, transport of ligands between different parts of the cell or even between cells do not happen instantaneously, but rather on a certain time scale which can differ in units of seconds, minutes or even hours [e.g. 20, 29]. Some experimental setups are able to capture these time scales. Thus one is able to see how the expression of a certain gene regulates the expression of a (different) gene at a later time point. This gap is called a delay. Sometimes it is assumed that these delays are gene-specific, since the machinery described above takes roughly the same time for a specific gene [49].

2.4. Experimental methods to assess gene expression

The principal index for the expression of a gene is its RNA abundance within a sample. A gene is called active if a detectable amount of its mRNA can be found, and inactive if the abundance of its mRNA is low relative to the general mRNA abundance. There are two main experimental procedures for measuring the abundance of RNA as a proxy for gene expression, these are microarray and RNA-Seq (see Figure 2.4).

A microarray (or here more correctly DNA microarray) experiment [75] is an experiment to measure the expression level of thousands of genes simultaneously (also called a high-throughput experiment). First the mRNA is extracted from a sample (a sample being a bulk of cells), then reverse-transcribed to cDNA: Reverse-transcription is the process of making a complementary DNA, a DNA strand synthesized from an RNA molecule by an enzyme called the reverse transcriptase, where the cDNA is like a mirror of the original mRNA. Afterwards the cDNA is labeled by attaching special fluorescent molecules (dye) to the cDNA molecules. These labeled cDNA molecules are hybridized on an array (chip) with pinholes for the different genes, meaning that every cDNA molecule should bind to the predefined hole that complements its sequence. Afterwards the array can be scanned with a filter and laser to measure the intensity of the dye for every gene.

RNA-Seq [59] is a more recent sequencing method. Similar to microarrays, first the RNA is extracted from the sample and reverse-transcribed to cDNA. After an amplification step where the number of all cDNA molecules is artificially increased, the cDNA molecules are sequenced, that is their sequence of bases is determined directly within a "sequencing machine" (instead of binding the molecules to predefined sequences like in the microarray approach). This gives a sequence read (a particular sequence of the four bases A, C, G and T) for every single cDNA molecule after the amplification step. All of the reads will be mapped to a reference genome, that is their recovered sequences are aligned with the whole consensus sequence of the organism's DNA, and every match in the alignment will be counted.

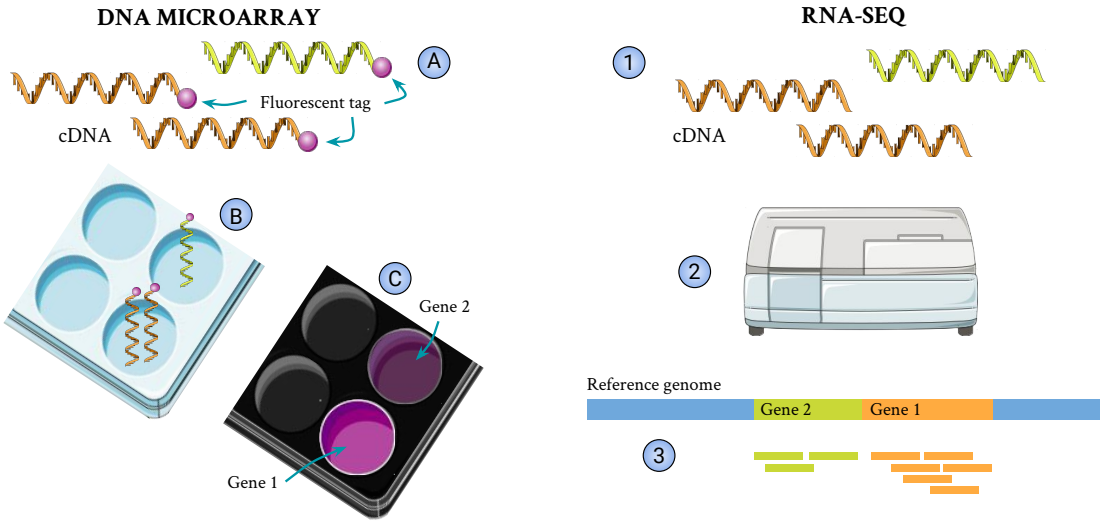

Figure 2.4.: [52] Experimental techniques to measure gene expression levels. Microarray: (A) The RNA is extracted, converted to complementary DNA (cDNA), and labeled with fluorescent tags. (B) The labeled cDNA hybridizes with nucleic acids on the array and is sorted this way. (C) The fluorescence levels (which correspond to the gene expression) are measured by scanning the array. RNA-Seq: (1) RNA is fragmented and converted to cDNA. (2) The cDNA is fed into a sequencing machine. (3) The reads from the sequencing are mapped to a reference genome and quantified per gene, giving the gene expression.

Since the original sample consisted of multiple copies of some mRNAs which were furthermore amplified, many sequences will map to the same regions (genes) on the reference genome. This gives a count of matched reads of that gene's mRNA, respectively the expression level of the gene.

3. Feature Selection Methods in the Context of Gene Network Reconstruction

First, the problem of (gene) network reconstruction is presented as a problem of feature selection, that is, neighborhood selection (Sections 3.1 and 3.2). We introduce the grouping of features in a formal way in Section 3.3. There follows a short review of the lasso, group lasso and sparse-group lasso as possible solutions to this problem of network reconstruction with neighborhood selection in Section 3.4. An alternative to the lasso methods is the spike-and-slab approach, a Bayesian framework for feature selection (Section 3.5). The most common algorithm to derive the parameters within Bayesian frameworks such as the spike-and-slab model is the stochastic method called Gibbs sampling, which is explained shortly in Section 3.5.1. Finally, the expectation propagation algorithm (as a deterministic and faster alternative to Gibbs Sampling) is reviewed in a general manner (Section 3.5.2) before we refine it for the sparse-group spike-and-slab in Chapter 4.

3.1. Gene network reconstruction and the Gaussian graphical model

In a typical gene regulatory network reconstruction procedure we consider the network as a graphical model [21, 47, 60], where the genes correspond to nodes and an interaction between two genes is represented by an edge between the respective nodes. This graphical model is also called a Markov graph [36, pp. 627]: Two nodes are not connected by an edge if they are conditionally independent given

all other nodes. The network graph is undirected, that is, edges do not indicate a causal direction (which would be represented as an arrow), but rather a mutual interdependence. For the study of directed (causal) graphs, see for example the work of Pearl [71].

If we assume that the data (that is, the M observations of P random variables $\mathbf{X} = (X_n)_{n=1}^{P}$) is generated from a multivariate normal distribution with covariance matrix Σ, the corresponding graphical model is called a Gaussian graphical model [11, p. 370]:

$$\mathbf{X} \sim \mathcal{N}_P(\mu, \Sigma).$$

The inverse of the covariance matrix $\Omega = \Sigma^{-1}$ is called the concentration matrix or precision matrix [16]. The entries ω_{ij} of the precision matrix Ω are closely related to the partial correlations ρ_{ij} between any two variables X_i and X_j:

$$\rho_{ij} = -\frac{\omega_{ij}}{\sqrt{\omega_{ii} \cdot \omega_{jj}}}.$$

The partial correlation ρ_{ij} is a measure for the (linear) dependency between the variables X_i and X_j given all other variables X_k, $k \neq i, j$. The partial correlation is zero if the two variables are conditionally independent of each other. If ρ_{ij} (or equivalently ω_{ij}) is different from zero, the variables X_i and X_j are dependent on each other and we draw an edge between the two nodes that correspond to these variables. Thus, if and only if $\omega_{ij} = 0$ there is no edge between nodes i and j within the network graph, and the precision matrix describes the network graph.

Reconstruction of gene regulatory networks with methods based on the framework of Gaussian graphical networks is a well-studied aspect of bioinformatics, see for example the reviews by Markowetz and Spang [60] and Huang et al. [45]. In the most common scenario the data consists of gene expression measurements stemming from microarray or RNA-Seq experiments.

Thus the data structure is a (big) matrix with numerical values (non-negative real numbers for microarray, non-negative integers for RNA-Seq), where every column $1, \ldots, P$ corresponds to a gene and every row $1, \ldots, M$ corresponds to an observation or experiment. Oftentimes the raw data needs to be normalized, and normalization is a difficult endeavor in itself (see for example [73]), which is

out of the scope for this work. We assume "reasonably" normalized data for the remainder, that is, where experimental biases are corrected and features live on the same scale.

Therefore, if we are given some data matrix $X = (x_{mn})$ with $m = 1, \ldots, M$ and $n = 1, \ldots, P$ we can try to reconstruct the precision matrix from the sample covariance matrix

$$\hat{\Sigma} = \frac{1}{M-1} X^T X \tag{3.1}$$

(assuming the columns of X have been centered around their column means first), which is an estimate of the true covariance matrix ($\hat{\Sigma} \approx \Sigma$).

3.2. Estimating the precision matrix and network topology with neighborhood selection

One can always estimate the sample covariance matrix $\hat{\Sigma}$ from the data matrix X using the formula (3.1). But if the number of features P is large and the number of observations M is small, inverting the sample covariance matrix $\hat{\Sigma}$ to recover the network structure from the sample precision matrix $\hat{\Omega}$ will most likely fail since the sample covariance matrix might not be of full rank and invertible [35, 76]. Many alternative approaches have been suggested to estimate the sample covariance matrix and the sample precision matrix in the "small M, large P" scenario, see the comprehensive review by Fan et al. [24].

For gene regulatory network graphs (and graph models in many other disciplines), it is very reasonable to assume that the precision matrix has many entries equal to zero, since a fully populated or dense precision matrix would correspond to a (close to) fully connected network graph. Therefore we assume a sparse estimate of the precision matrix and enforce sparsity restrictions when reconstructing the precision matrix and the network graph. There are approaches to directly find an approximation to the precision matrix in these sparse cases, e.g. the works of Yuan and Lin [97] and the graphical lasso from Friedman et al. [25]. Schäfer and Strimmer [77] proposed an alternative by assuming a certain target structure of the precision matrix.

In this work we follow a different and complementary approach proposed as "neighborhood selection" by Meinshausen and Bühlmann [62]. Instead of estimating the sparse precision matrix as a whole, we break down the problem into P independent sparse problems, one for every column (feature) of X, which corresponds to one particular node in the network graph. We determine this node's neighborhood of connected nodes, that is, find among the remaining $P - 1 = N$ variables the neighbors such that the current node is conditionally independent of all other variables given its neighborhood. In the network graph there are edges between the chosen node and all its neighborhood nodes, and no edges between nodes that are not neighbors. This means we perform $P = N + 1$ instances of the sparse feature selection method of choice, where every instance consists of removing one column i of the data matrix and using it as a response (dependent) variable $y = X_i$ that needs to be explained by the remaining N variables (features, $X_{-i} = (X_j)_{j \neq i}$) as a standard regression problem:

$$X_i \equiv y = X_{-i}\beta_i + \varepsilon_i \tag{3.2}$$

$$\text{or more general: } y = X\beta + \varepsilon. \tag{3.3}$$

X (or X_{-i}) is an $(M \times N)$-matrix, y (or X_i) a vector with M entries, β (or β_i) a vector of coefficients with N entries and ε (or ε_i) a vector of M entries that describes the stochastic error. Note that X describes depending on context either the full data matrix of dimensions $(M \times P)$ or a data matrix of dimensions $(M \times N)$. Equation (3.2) is the matrix notation of the equivalent formulation as element-wise linear sums:

$$x_{mi} \equiv y_m = \sum_{j \neq i} \beta_{ij} \cdot x_{mj} + \varepsilon_{mi}, \ m = 1, \ldots, M.$$

Why is it possible to run multiple linear regressions (which determine regression coefficients) instead of calculating the entries of the precision matrix? Within the Gaussian model, these are equivalent problems, the reasons are explained by Cox and Wermuth [16, pp. 68] or Hastie et al. [36, pp. 630], and lead to the simple connection between the entries of the precision matrix and the regression coefficients:

$$\beta_{ij} = -\frac{\omega_{ij}}{\omega_{ii}}.$$

The consistency of the neighborhood selection approach with the inversion of the covariance matrix is very much dependent on the sparsity assumption, for details and a proof of consistency see [62].

Thus, when finding solutions to equation (3.3) we need to assume that most entries of the vectors β_i are actually zero, which is called feature selection or sparse regression. There are different approaches on how to discern the zero-coefficients from the non-zero ones in the sparse linear regression scenario, the main ones being the classic lasso (introduced by Tibshirani [88]), its successor elastic net regularization (introduced by Zou and Hastie [100]) or Bayesian approaches (a particular Bayesian framework will be described in Section 3.5).

In summary, there are two ways to reconstruct a network graph under the Gaussian graphical model based on the precision matrix: Either find a sparse estimate of the precision matrix directly or do multiple linear regressions with feature selection for every feature separately (neighborhood selection). We choose to pursue the neighborhood selection approach since it allows us to include grouping information in a straightforward manner (see the following section). Since we are reconstructing undirected network graphs, we need to decide what to do if either β_{ij} or β_{ji} is zero while the other one is not (this problem does not arise if we reconstruct a proper symmetric precision matrix directly, since $\omega_{ij} = \omega_{ji}$). In the remainder, we will use the less conservative and more permissive OR rule [36, 62] for all methods considered: If β_{ij} **or** β_{ji} is different from zero (or both are different from zero), we draw an edge in the resulting network graph. There is no edge if $\beta_{ij} = \beta_{ji} = 0$.

3.3. Grouping of features

A grouping of the features or equivalent, a map $\mathcal{G} : \{1, \ldots, N\} \to \{1, \ldots, G\}$ with G the number of groups, is an additional information about the structure of the data. The map $\mathcal{G}$ assigns to every feature (respectively its index n) a group (respectively group index g): $\mathcal{G}(n) = g$. If $\mathcal{G}$ is injective, every feature is mapped to its own group and thus the mapping reduces to a non-informative grouping. For a given group g we denote by $\mathcal{G}^{-1}(g)$ the set of indices $\mathcal{G}^{-1}(g) \subseteq \{1, \ldots, N\}$ which corresponds to all the features in group g. A grouping of features may stem

from any prior information about the data, e.g. which transcription factors tend to regulate the same genes or which genes belong to the same pathways, but also the grouping could be derived from the data itself, e.g. genes that show similar expression patterns and are grouped by clustering. The idea of grouped features in a regression scheme was first formulated in the thesis of Bakin [6] and subsequently built upon by Yuan and Lin [96] and Friedman et al. [27]. As a novel idea, we use grouping information for the problem of (gene) network reconstruction.

Sparse regression in the presence of grouped features needs to distinguish two different levels of group sparsity: *between-group* and *within-group sparsity*. The first level is between-group sparsity: Given different groups of features, the (regression) algorithm needs to decide which groups to use and which groups to discard. This means that all the coefficients of features within one group are collectively zero or collectively different from zero. While Bakin [6] and Yuan and Lin [96] considered between-group sparsity only, Friedman et al. [27] discussed and introduced a second level of sparsity which is within-group sparsity: Looking at the features within a group one needs to decide which features to use (non-zero coefficients) and which features to neglect (zero coefficients). Thus, within-group sparsity should be implemented on top of between-group sparsity, if it is reasonable to assume that some features within a non-zero group should still be zero.

3.4. Lasso, group lasso and sparse-group lasso

When talking about sparsity in regression schemes, in most cases the method of choice is the "lasso" introduced by Tibshirani [88]. With the notation from equation (3.3), the lasso corresponds to the following objective [36, p. 68]:

$$\min_{\hat{\beta}}\{\frac{1}{2}||y - X\hat{\beta}||_2^2 + \lambda||\hat{\beta}||_1\}. \tag{3.4}$$

The first term is the squared error between the response and the approximation to it using an estimate $\hat{\beta}$ of the true coefficient vector β. The second term enforces the sparsity on $\hat{\beta}$ by putting a penalty on the sum of absolute values of $\hat{\beta}$. Using $||\cdot||_0$ instead of $||\cdot||_1$ would be more desirable, since the 0-norm gives the number of coefficients different from zero. Equation (3.4) with 0-norm instead of the 1-norm

is equivalent to best-subset selection [36, pp. 57], but solving the problem under the 0-norm constraint has been shown to be NP-hard [68]. The lasso approach substitutes the 0-norm with the 1-norm (sum of absolute values of coefficients) which also enforces sparsity and approximates the optimal 0-norm solution. If the 1-norm and 2-norm (euclidean distance) are combined we have the elastic net regularization introduced by [100]:

$$\min_{\hat{\beta}}\{\frac{1}{2}||y - X\hat{\beta}||_2^2 + \lambda\left(\alpha||\hat{\beta}||_1 + (1-\alpha)||\hat{\beta}||_2^2\right)\},\ \alpha \in [0;1],$$

which for $\alpha = 0.95$ is the standard within most implementations of elastic net, and with $\alpha = 1$ it reduces to the lasso regularization. For $\alpha = 0$ it is equivalent to "ridge regression" [36, pp. 61], which does not enforce sparsity in the number of non-zero coefficients directly but rather shrinks the size of the regression coefficients.

The standard lasso considers all variables equally without taking any structure between the features into account, and thus it is not a good candidate algorithm if variables are grouped.

The "group lasso" from Yuan and Lin [96] acknowledges grouped variables in a between-group sparsity approach, but it can only choose whole groups without taking advantage of within-group sparsity. Given a map $\mathcal{G}$, the objective of the group lasso is:

$$\min_{\hat{\beta}}\{\frac{1}{2}||y - X\hat{\beta}||_2^2 + \lambda\sum_{g=1}^{G}\sqrt{p_g}||\hat{\beta}_{\mathcal{G}^{-1}(g)}||_2\},\ \text{where } p_g = |\mathcal{G}^{-1}(g)|, \tag{3.5}$$

with $\hat{\beta}_{\mathcal{G}^{-1}(g)}$ being the vector of coefficients that corresponds to the features within the g-th group. It follows that we have a group-wise euclidean vector norm as a penalty in equation (3.5) and as such the objective imposes sparsity on the between-group level, but only restricts the values of the coefficients within one group without enforcing true sparsity on the within-group level. To overcome this drawback, Simon et al. [80] proposed the "sparse-group lasso" as a framework for the described two-fold sparsity which tries to model the within- and between-group sparsity. Simply put, they add a 1-norm penalty term to the objective of the group lasso:

$$\min_{\hat{\beta}}\{\frac{1}{2M}||y - X\hat{\beta}||_2^2 + \lambda\left((1-\alpha)\sum_{g=1}^{G}\sqrt{p_g}||\hat{\beta}_{\mathcal{G}^{-1}(g)}||_2 + \alpha||\hat{\beta}||_1\right)\},\ \text{with } \alpha \in [0;1],$$

where α is the mixture between the standard lasso and the group lasso. Within one non-zero group, the objective is equivalent to the elastic net regularization.

There are multiple methods to find the solutions to the given (sparse-group or group) lasso objectives, the most prominent ones being least-angle regression or LARS [22] and different gradient-based techniques [80, 88, 96]. Because of the considerable research on lasso methods, many (efficient) implementations are available, for example packages within the statistical programming environment R [74].

3.5. Spike-and-slab

An alternative to these lasso methods in general is a Bayesian approach with spike-and-slab feature selection [67]. There are Bayesian formulations of the lasso, group lasso and sparse-group lasso respectively: Park and Casella [70] introduced the Bayesian lasso, while Kyung et al. [51] expanded the Bayesian formulation to the group lasso. Xu and Ghosh [94] added a Bayesian formulation for the sparse-group lasso as well as a non-lasso Bayesian framework for the between-group and within-group sparsity scenario (which they call bi-level selection). All of these approaches in the work of Park and Casella [70], Kyung et al. [51] and Xu and Ghosh [94] (along with the similar work of Zhang et al. [98]) use Gibbs sampling (see Section 3.5.1) to derive the parameters.

The spike-and-slab is an approach to Bayesian variable selection first introduced by Mitchell and Beauchamp [67] and subsequently Geweke [31], but in our work we follow the definition of George and McCulloch [30] since it is more easily interpretable. A similar framework is formulated in [50].

In general, the relationship between the observed y and the predictors X as well as β and the error variance σ_0 can be described by the conditional probability

$$\mathbb{P}(y|\beta, X) = \mathcal{N}(y|X\beta, \sigma_0^2 I). \tag{3.6}$$

If we assume $\varepsilon \sim \mathcal{N}(0, \sigma_0 I)$, this is a reformulation of equation (3.3) where we dropped all indices. Furthermore, if we assume that $\beta = (\beta_1, \ldots, \beta_N)$ is sparse and some β_n are zero, we can introduce an auxiliary variable $Z = (Z_1, \ldots, Z_N) \in$

$\{0,1\}^N$ with $Z_n = 0$ indicating that $\beta_n = 0$ and from $Z_n = 1$ follows $\beta_n \neq 0$. We can give a prior distribution over Z:

$$\mathbb{P}(Z) = \prod_{n=1}^{N} \text{Bern}(Z_n|p_n) = \prod_{n=1}^{N} p_n^{Z_n} \cdot (1-p_n)^{1-Z_n}, \tag{3.7}$$

where p_n is the belief (probability) that the corresponding coefficient β_n should be different from zero. This leads to a definition of the conditional probability $\mathbb{P}(\beta|Z)$ that is at the heart of the spike-and-slab approach:

$$\mathbb{P}(\beta|Z) = \prod_{n=1}^{N} \left(Z_n \cdot \mathcal{N}(\beta_n|0,\sigma^2_{\text{slab}}) + (1-Z_n) \cdot \delta(\beta_n)\right). \tag{3.8}$$

If $Z_n = 1$, the second summand disappears, and we draw β_n from a normal distribution centered around zero but with a large variance parameter σ_{slab}, thus $\beta_n \neq 0$. If $Z_n = 0$, the first summand disappears, and we force β_n to be equal to zero: $\delta(\cdot)$ is the distribution that puts probability mass 1 to zero. Instead of $\delta(\cdot)$ one could also use a normal distribution $\mathcal{N}(\cdot|0,\sigma^2_{\text{spike}})$ with a very small σ_{spike} (e.g. in [30]). For an illustration of the two distributions $\mathcal{N}(\cdot|0,\sigma^2_{\text{slab}})$ and $\delta(\cdot)$ see Figure 3.1.

These representations (equations (3.6), (3.7) and (3.8)) allow for the following Bayesian framework to describe the linear regression scheme from equation (3.3):

$$\mathbb{P}(\beta, Z|y, X) = \frac{\mathbb{P}(y|\beta, X) \cdot \mathbb{P}(\beta|Z) \cdot \mathbb{P}(Z)}{\mathbb{P}(y|X)}. \tag{3.9}$$

Given the data (y, X), the task at hand is now to find the corresponding β and Z (and probabilities p_n, $n = 1, \ldots, N$). One approach is to use Markov Chain Monte Carlo simulations respectively Gibbs sampling (e.g. the solution of George and McCulloch [30], see also the next Section 3.5.1), but in this work we consider an alternative non-stochastic approach called "expectation propagation" (see Section 3.5.2).

Note that we do not consider a prior on σ_0 in this work, which means this parameter needs to be specified beforehand.

3.5.1. Gibbs sampling

Gibbs sampling (a Markov Chain Monte Carlo algorithm, MCMC [11, pp. 537]) is a prominent method to derive the parameters for Bayesian frameworks. Here

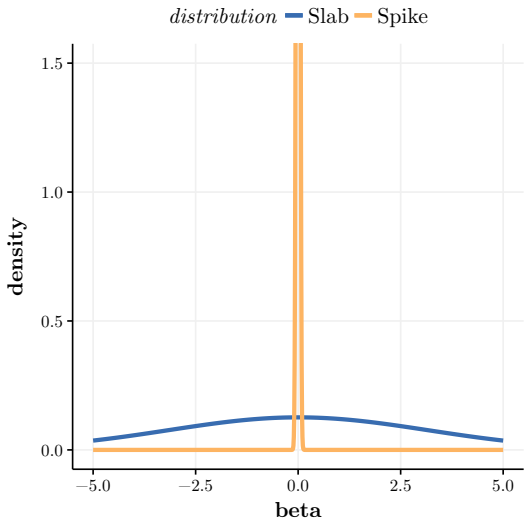

Figure 3.1.: Density function curves of spike and slab distributions: Both the spike and the slab are depicted as normal distributions $\mathcal{N}(\beta|0,\sigma)$, with $\sigma_{\text{slab}} = 10$ and $\sigma_{\text{spike}} = 0.001$.

explain shortly the general idea of Gibbs sampling and discuss its shortcomings.

Gibbs sampling as a MCMC algorithm does not solve the Bayesian formula $\mathbb{P}(\theta|D) = 1/\mathbb{P}(D) \cdot \mathbb{P}(D|\theta) \cdot \mathbb{P}(\theta)$ for some data D and parameter vector θ directly, but instead generates samples from the posterior by constructing a Markov chain that has the target posterior distribution as its equilibrium distribution. For Gibbs sampling, this corresponds to the following procedure:

1. initialize θ with random values,
2. choose some θ_i and hold all other θ_j, $j \neq i$, fixed,
3. sample a new value for θ_i from the conditional distribution $\mathbb{P}(\theta_i|D, \theta_{-i})$,
4. go back to 2 and repeat for all θ_i, then start again, and keep doing this for a total of B iterations, where B is very large (e.g. $B = 10000$).

It can be shown that this procedure leads to sample values of θ from the true joint distribution $\mathbb{P}(\theta|D)$ [11, pp. 542]. Often it is assumed that the first batch of samples does not approximate the target distribution well and is as such discarded

(burn-in period). Also, subsequent samples might not be truly independent, and as such one chooses samples of the values with a certain iteration gap between them. On top, most Gibbs samplers run for several thousand iterations. To get a meaningful point estimate for the parameters in question, one could choose the median or mean of the chosen samples from the Gibbs sampling procedure.

While Gibbs sampling gives consistent results if the number of available observations is appropriately large, Gibbs sampling for feature selection suffers in the case of few observations and many features. In addition, because of the stochastic nature of Gibbs sampling, the runtime of a Gibbs sampling procedure is quite long, thus a numerical algorithm should be preferred if available.

There is little work on Bayesian frameworks that neither depend on the lasso nor on Gibbs sampling. An iterative but deterministic algorithm called expectation propagation for Bayesian feature selection (without grouping of the features) was introduced by Seeger [78] (and another version by Hernández-Lobato et al. [43]). Hernández-Lobato et al. [41] proposed a grouped version of the spike-and-slab approach for the problem of the between-group sparsity (analogous to the group lasso by Yuan and Lin [96]) along with expectation propagation to derive the appropriate parameters.

3.5.2. Expectation propagation

All distributions we will consider in the remainder are members of the class of exponential family distributions. This means that their respective probability densities can be written in the form

$$f(x) = h(x) \cdot g(\eta) \cdot \exp(\eta^T T(x)),$$

where η is called "natural parameter" and $T(x)$ is the "sufficient statistic" (see for example Bishop [11, pp. 113-117]). More importantly, for all exponential family distributions the product (and to some extent the quotient) of two densities from an exponential family is again a density from an exponential family with updated parameters (up to a scaling factor depending on the parameters only). In our implementation we need these properties for the Bernoulli distribution and (multivariate) normal distribution, see Appendix A and B.

The expectation propagation algorithm (introduced by Minka [64]) approximates a true (complicated) posterior distribution $\mathcal{P}$ with a (simpler) approximating distribution $\mathcal{Q}$ by iteratively minimizing the Kullback-Leibler divergence $\mathrm{KL}(\mathcal{P}||\mathcal{Q})$. As such, the expectation propagation algorithm is an example of deterministic approximate inference [11, pp. 461]: the true posterior is too complex such that expectations are not analytically tractable since the resulting integrals do not have closed-form solutions and the dimensions of the parameter space prohibit numerical integration.

If $\mathcal{Q}$ is an exponential family distribution (and thus $\mathcal{Q}(x) = h(x) \cdot g(\eta) \cdot \exp(\eta^T T(x))$), the following holds true (see Bishop [11, p. 505]):

$$\begin{aligned}
\mathrm{KL}(\mathcal{P}||\mathcal{Q}) &= \int \mathcal{Q}(x) \cdot \log \frac{\mathcal{P}(x)}{\mathcal{Q}(x)} \, \mathrm{d}x \\
&= \int \mathcal{P}(x) \cdot \left(\log \mathcal{P}(x) - \log h(x) - \log g(\eta) - \eta^T T(x) \right) \, \mathrm{d}x \\
&= -\log g(\eta) - \eta^T \cdot \int T(x) \cdot \mathcal{P}(x) \, \mathrm{d}x + C(x) \\
&= -\log g(\eta) - \eta^T \cdot \mathbb{E}_{\mathcal{P}} \left[T(x) \right] + C(x),
\end{aligned}$$

where the "constant" term $C(x)$ does not depend on the natural parameter η. To minimize $\mathrm{KL}(\mathcal{P}||\mathcal{Q})$ in η we set the gradient equal to zero and get

$$-\nabla \log g(\eta) = \mathbb{E}_{\mathcal{P}} \left[T(x) \right]. \tag{3.10}$$

On the other hand, since $\mathcal{Q}(x) = h(x) \cdot g(\eta) \cdot \exp(\eta^T T(x))$ is a probability distribution, we have [11, p. 116]:

$$\begin{aligned}
\int \mathcal{Q}(x) \, \mathrm{d}x &= 1 \\
\Leftrightarrow g(\eta) \int h(x) \cdot \exp(\eta^T T(x)) \, \mathrm{d}x &= 1.
\end{aligned} \tag{3.11}$$

Taking the gradient on both sides of equation (3.11):

$$\begin{aligned}
\nabla g(\eta) \int h(x) \cdot \exp(\eta^T T(x)) \, \mathrm{d}x + g(\eta) \int h(x) \cdot \exp(\eta^T T(x)) \cdot T(x) \, \mathrm{d}x &= 0 \\
\overset{(3.11)}{\Leftrightarrow} \frac{\nabla g(\eta)}{g(\eta)} \cdot 1 + \int g(\eta) \cdot h(x) \cdot \exp(\eta^T T(x)) \cdot T(x) \, \mathrm{d}x &= 0 \\
\Leftrightarrow \mathbb{E}_{\mathcal{Q}} \left[T(x) \right] &= -\nabla \log g(\eta).
\end{aligned} \tag{3.12}$$

Putting equations (3.10) and (3.12) together gives us

$$\mathbb{E}_{\mathcal{P}}\left[T(x)\right] = \mathbb{E}_{\mathcal{Q}}\left[T(x)\right], \tag{3.13}$$

that is, to minimize the divergence between $\mathcal{P}$ and $\mathcal{Q}$ we need to match the expectations under $\mathcal{P}$ and $\mathcal{Q}$ of the sufficient statistic of $\mathcal{Q}$.

The matching under the complete distributions $\mathcal{P}$ and $\mathcal{Q}$ can be very hard, but it gets easier if the distributions $\mathcal{P}$ and $\mathcal{Q}$ are factored, for example if $\mathcal{P}$ is a posterior distribution for some data $\mathcal{D}$:

$$\mathcal{P} = \frac{1}{\mathbb{P}(\mathcal{D})}\prod_i f_i, \tag{3.14}$$

$$\mathcal{Q} = \frac{1}{\mathcal{Z}}\prod_i \tilde{f}_i, \tag{3.15}$$

where the factors $\tilde{f}_i$ belong to the exponential family of distributions, but they do not need to integrate to 1. That is why $\mathcal{Z} = \int \prod_i \tilde{f}_i$ is the normalization constant needed such that $\mathcal{Q}$ is a proper probability distribution which integrates to 1 (and $\mathcal{Z}$ approximates $\mathbb{P}(\mathcal{D})$).

Expectation propagation approximates $\mathrm{KL}(\mathcal{P}||\mathcal{Q})$ by first initializing the parameters of the $\tilde{f}_i$ and then cycling through the paired factors f_i and $\tilde{f}_i$ one at a time (see Bishop [11, pp. 506-510]) and updating the corresponding parameters. Suppose we want to update factor $\tilde{f}_i$. First we remove $\tilde{f}_i$ from $\mathcal{Q}$:

$$\mathcal{Q}^{\setminus i} = \frac{\mathcal{Q}}{\tilde{f}_i},$$

where $\mathcal{Q}^{\setminus i}$ is a probability distribution up to a normalization constant. Now we want to update $\tilde{f}_i \to \tilde{f}_i^{\text{new}}$ such that $\tilde{f}_i^{\text{new}} \cdot \mathcal{Q}^{\setminus i}$ is very close to $f_i \cdot \mathcal{Q}^{\setminus i}$, or with $\mathcal{Q}^{\text{new}}$ the normalized version of $\tilde{f}_i^{\text{new}} \cdot \mathcal{Q}^{\setminus i}$, we minimize

$$\mathrm{KL}(\frac{1}{\mathcal{Z}_i} f_i \cdot \mathcal{Q}^{\setminus i} || \mathcal{Q}^{\text{new}})$$

for some (unimportant) normalization constant $\mathcal{Z}_i$. Equation (3.13) tells us we need to match the sufficient statistics of these two distributions, where calculus and arithmetic yield the updated parameters of $\mathcal{Q}^{\text{new}}$.

After we have found $\mathcal{Q}^{\text{new}}$ we get the parameters of our updated $\tilde{f}_i^{\text{new}}$ from

$$\tilde{f}_i^{\text{new}} = \mathcal{Z}_i \frac{\mathcal{Q}^{\text{new}}}{\mathcal{Q}^{\setminus i}},$$

where the value of $\mathcal{Z}_i$ is not needed to update the parameters.

In Hernández-Lobato et al. [43] and Hernández-Lobato et al. [41] the expectation propagation framework is applied to the general spike-and-slab framework as well as the spike-and-slab for grouped variables with sparsity between groups only. In the next Chapter 4 we will expand the spike-and-slab framework with expectation propagation to the two-level sparsity on the between- and within-group levels.

3.6. Cross-validation

Every lasso method depends on a penalty parameter λ that drives the sparsity of the model. Smaller values of this parameter correspond to a smaller number of features included in the final model. Most implementations of lasso methods give a decreasing sequence of λ values along with the corresponding increasing set of chosen features with their corresponding coefficients. Cross-validation [11, pp. 51] is a way to determine the "optimal" value of this parameter λ. For k-fold cross-validation (most often $k = 10$) we divide the observations into k distinct batches of roughly equal size and run the method of choice k times, where we hold back one of the batches and run the method on the remaining $k - 1$ batches combined. Each time this gives us a sequence of feature sets with increasing size of the sets and we determine the prediction errors on the held-out batch of observations for each such set. We add the prediction errors of all k runs (thus every observation will be predicted exactly once) over the same sizes of the feature sets and obtain the cross-validated error curve $\text{CVE}(\lambda)$ along the λ sequence/size of the feature sets. The optimal size of the feature set (or the optimal penalty parameter $\lambda_{\min}$ in the case of the lasso) is the one that minimizes this cross-validated error $\text{CVE}(\lambda_{\min}) = \min_\lambda \text{CVE}(\lambda)$. Alternatively, to avoid over-fitting, one can use the 1se-rule ("within one standard error of the minimum", Hastie et al. [36, pp. 241]) to determine the optimal value: find the minimum $\text{CVE}(\lambda_{\min})$ and its corresponding standard deviation $\text{sd}(\lambda_{\min})$ and choose the lesser value

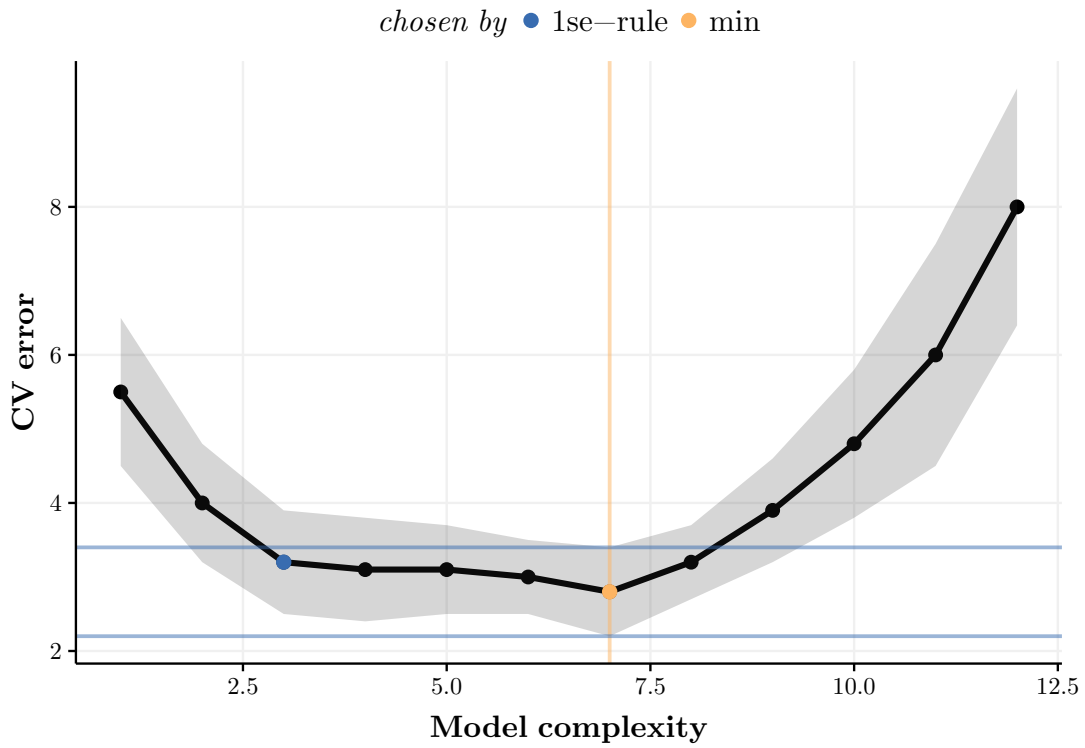

Figure 3.2.: This is an example of how to choose the proper model complexity with cross-validation: Either one goes for the global minimum (orange) or the smallest model complexity within one standard error of the minimum (blue). The gray shaded area gives the standard error of the cross-validated error.

λ_{1se} such that $\text{CVE}(\lambda_{1se})$ lies within one standard deviation of this minimum: $\lambda_{1se} = \min\{\lambda : \text{CVE}(\lambda) \leq \text{CVE}(\lambda_{\min}) + \text{sd}(\lambda_{\min})\}$. Figure 3.2 illustrates the concepts explained in this section.

Finally we run the method of choice a final $(k+1)$-th time on the whole set of observations and choose the subset of features that corresponds in size to the one chosen from the first k runs (for lasso methods, depending on $\lambda_{\min}$ or λ_{1se}).

While cross-validation is very effective to determine the optimal model complexity and runs fast for the lasso methods, it does not consider the inclusion of features on their own, but rather only the total number of features to be included.

In the worst case this means that a feature could be chosen to be in the final set because it ended up as a chosen feature in the final $(k+1)$-th run of the method, while it was never chosen in the first k runs of the cross-validation.

On the other hand, cross-validation is also a good way to choose model parameters that do not affect the complexity of the model directly, for example the σ_{slab} and σ_0 parameters of the spike-and-slab approach, by pre-defining a grid of these parameters and running the algorithm k times for each of the parameter sets along the grid, choosing the combination with the lowest cross-validated error for the final model.

In the remainder we use cross-validation with the 1se-rule for the lasso methods if a specific subset needs to be chosen: The k-fold cross-validation ($k = 10$) determines the optimal cutoff value λ_{1se} and we return the vector of coefficients $\beta_{\lambda_{\text{1se}}}$ that corresponds to this cutoff from the $(k+1)$-th run. Furthermore, we use a slightly modified cross-validation for the expectation propagation-based Bayesian methods which takes advantage of the different output from these methods. While lasso methods give a different vector of coefficients for every λ value considered, the Bayesian methods give one single vector of coefficients β and an accompanying vector of probabilities $\vec{p} = (p_n)_{n=1,\ldots,N}$ in each of the k cross-validation runs. As such we take the mean vectors over the coefficient vectors and probability vectors from these runs instead of running the method a $(k+1)$-th time, while we decide for the optimal cutoff value p_{1se} for the probabilities with the 1se-rule like in the lasso methods. The final output is the mean vector of probabilities $\vec{p}_{CV}$ and the mean vector β_{CV} of coefficients with zeros at the entries $\beta_{CV,n}$ where $p_{CV,n} < p_{\text{1se}}$.

4. Feature Selection with Sparse-Group Spike-and-Slab and Expectation Propagation

We introduce a model for Bayesian feature selection that accounts for between- and within group sparsity (Sections 4.1 and 4.2). We derive the closed-form solutions of the corresponding expectation propagation updates of the model parameters (Sections 4.3 and 4.4). Finally, we identify the arising numerical obstacles and present our implementation (Sections 4.5 and 4.6).

4.1. Feature selection with grouped features

Consider the linear regression setting from equation (3.3). Without loss of generality we set $i = P$ in the following, with $y = X_P$ and $X = (X_1, \ldots, X_N) = (x_{mn})$, $m = 1, \ldots, M$, $n = 1, \ldots, N$. As such, the linear model in this chapter will reduce to:

$$y_m = \sum_{n=1}^{N} \beta_n \cdot x_{mn} + \varepsilon_m, \tag{4.1}$$

where the β_n are the regression coefficients and ε_m are independent, identically distributed (iid) variables (errors) distributed according to $\mathcal{N}(0, \sigma_0^2)$. Equation (4.1) is equivalent to the matrix notation

$$y = X \cdot \beta + \varepsilon, \tag{4.2}$$

with vectors y, β, ε and data matrix X. If we assume that most β_n are effectively equal to zero, we have the setting of sparse regression. Furthermore, we consider

that there is an inherent grouping or map $\mathcal{G}$ of the variables $(X_1, \ldots, X_N)$ such that in every group g with corresponding features $(X_n)_{n \in \mathcal{G}^{-1}(g)}$, there are only a few variables with a pronounced influence on y while the other variables from this group can be neglected (or there is no influence from group g on y). This corresponds to the following regression scheme:

$$y_m = \sum_{g=1}^{G} \sum_{n \in \mathcal{G}^{-1}(g)} \beta_n \cdot x_{mn} + \varepsilon_m, \tag{4.3}$$

where for every group g some or none of the regression coefficients $(\beta_n)_{n \in \mathcal{G}^{-1}(g)}$ are different from zero. Of course, equation (4.3) is the same as equation (4.1) but with a different ordering of the features, so it is up to the feature selection algorithm to make use of the grouping information.

We do not consider the intercept in the regression explicitly. There are two ways to deal with the intercept: Either one centers the response y around its mean $(y \leftarrow y - \bar{y})$ in the beginning, or one adds a column $X_0 = (1, \ldots, 1)$ to the data matrix and a coefficient β_0 to the vector of coefficients, this way modeling the intercept as an additional feature (within its own group of size one).

The grouping of the variables can be understood as separating the columns of the feature matrix X into different blocks, with a corresponding separation of the vector of coefficients. This is illustrated in Figure 4.1 as a cartoon of the equation $y = X\beta + \varepsilon$ with grouping into six groups.

4.2. Spike-and-slab for between- and within-group sparsity

We will expand the Bayesian spike-and-slab framework from the last chapter (see equation (3.9)) to suit the regression scheme from equation (4.3). To this end, we introduce another auxiliary variable $\Gamma = (\Gamma_1, \ldots, \Gamma_G) \in \{0, 1\}^G$ which handles the between-group sparsity while the Z variable handles the within-group sparsity, given the value of Γ.

Given the data (y, X) and a grouping $\mathcal{G}$ of the features in analogy to Section 3.5, the following factorization of the true posterior distribution $\mathcal{P}(\beta, Z, \Gamma) =$

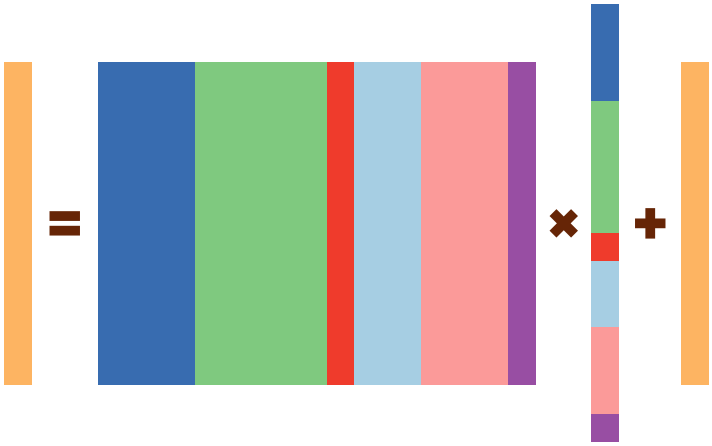

Figure 4.1.: Representation of $y = X\beta + \varepsilon$ with grouping as color-coded rectangles.

$\mathbb{P}(\beta, Z, \Gamma | y, X)$ holds true (see also equation (3.14)):

$$\mathbb{P}(\beta, Z, \Gamma | y, X) = \frac{\mathbb{P}(y|\beta, X) \cdot \mathbb{P}(\beta|Z) \cdot \mathbb{P}(Z|\Gamma) \cdot \mathbb{P}(\Gamma)}{\mathbb{P}(y|X)} = \frac{1}{\mathbb{P}(y|X)} \prod_{i=1}^{4} f_i(\beta, Z, \Gamma), \tag{4.4}$$

$$\mathbb{P}(y|\beta, X) = f_1(\beta, Z, \Gamma) = \mathcal{N}(y|X\beta, \sigma_0^2 I), \tag{4.5}$$

$$\mathbb{P}(\beta|Z) = f_2(\beta, Z, \Gamma) = \prod_{n=1}^{N} \left(Z_n \cdot \mathcal{N}(\beta_n|0, \sigma_{\text{slab}}^2) + (1 - Z_n) \cdot \delta(\beta_n) \right), \tag{4.6}$$

$$\mathbb{P}(Z|\Gamma) = f_3(\beta, Z, \Gamma) = \prod_{n=1}^{N} \left(\Gamma_{\mathcal{G}(n)} \cdot \text{Bern}(Z_n|p_n) + (1 - \Gamma_{\mathcal{G}(n)}) \cdot \delta(Z_n) \right), \tag{4.7}$$

$$\mathbb{P}(\Gamma) = f_4(\beta, Z, \Gamma) = \prod_{g=1}^{G} \text{Bern}(\Gamma_g|\pi_g). \tag{4.8}$$

The two-fold group sparsity assumption is realized by a refined spike-and-slab (equations (4.6) and (4.7)). If $\Gamma_{\mathcal{G}(n)} = 1$ for a certain n **and** $Z_n = 1$, then the corresponding β_n is different from zero, realized by β_n drawn from the distribution $\mathcal{N}(0, \sigma_{\text{slab}}^2)$ with a large σ_{slab}, while if $\Gamma_{\mathcal{G}(n)} = 1$ but $Z_n = 0$ **or** $\Gamma_{\mathcal{G}(n)} = 0$ and subsequently $Z_n = 0$ we have $\beta_n = 0$ with β_n shrunken to zero by $\delta(\cdot)$. This way the variables $\Gamma_1, \ldots, \Gamma_G$ code for the between-group sparsity while for a fixed g the variables $(Z_n)_{n \in \mathcal{G}^{-1}(g)}$ code for the within-group sparsity.

4.3. Expectation propagation for sparse-group spike-and-slab

Given the data (y, X) and the grouping of features $\mathcal{G}$, we want to find an estimate of the sparse coefficient vector β (and probability vectors p and π). To this end, we apply the algorithmic framework called expectation propagation (EP, presented in Section 3.5.2).

In our implementation of the EP algorithm, the procedure is based on the following approximation between the factorized true posterior distribution $\mathcal{P}$ and the likewise factorized approximate posterior distribution $\mathcal{Q}$ (see equations (3.14) and (3.15)):

$$\mathcal{P}(\beta, Z, \Gamma) \approx \mathcal{Q}(\beta, Z, \Gamma),$$

$$\text{where } \mathcal{P}(\beta, Z, \Gamma) = \frac{1}{\mathbb{P}(y|X)} \prod_{i=1}^{4} f_i(\beta, Z, \Gamma) \text{ (see Eq. (4.4))}, \tag{4.9}$$

$$\text{and } \mathcal{Q}(\beta, Z, \Gamma) = \frac{1}{\mathcal{Z}} \cdot \prod_{i=1}^{4} \tilde{f}_i(\beta, Z, \Gamma), \tag{4.10}$$

$$\text{with } \tilde{f}_1(\beta, Z, \Gamma) = \tilde{s}_1 \cdot \mathcal{N}(\beta|\tilde{m}_1, \tilde{V}_1), \tag{4.11}$$

$$\tilde{f}_2(\beta, Z, \Gamma) = \tilde{s}_2 \cdot \prod_{n=1}^{N} \mathcal{N}(\beta_n|\tilde{m}_{2,n}, \tilde{V}_{2,n}) \cdot \text{Bern}(Z_n|\tilde{p}_{2,n}), \tag{4.12}$$

$$\tilde{f}_3(\beta, Z, \Gamma) = \tilde{s}_3 \cdot \prod_{n=1}^{N} \text{Bern}(Z_n|\tilde{p}_{3,n}) \cdot \text{Bern}(\Gamma_{g(n)}|\tilde{\pi}_{3,n}), \tag{4.13}$$

$$\tilde{f}_4(\beta, Z, \Gamma) = \tilde{s}_4 \cdot \prod_{g=1}^{G} \text{Bern}(\Gamma_g|\tilde{\pi}_{4,g}), \tag{4.14}$$

$$\text{and thus } \mathcal{Q}(\beta, Z, \Gamma) = \mathcal{N}(\beta|\tilde{m}, \tilde{V}) \cdot \prod_{g=1}^{G} \text{Bern}(\Gamma_g|\tilde{\pi}_g) \cdot \prod_{n=1}^{N} \text{Bern}(Z_n|\tilde{p}_n). \tag{4.15}$$

The parameters $\tilde{s}_1$ to $\tilde{s}_4$ ensure that $\tilde{f}_i\mathcal{Q}^{\setminus i}$ and $f_i\mathcal{Q}^{\setminus i}$ integrate up to the same value. The particular form of $\mathcal{Q}$ in equation (4.15) as a product of the terms in equations (4.11) to (4.14) (divided by the normalization factor $\mathcal{Z}$) follows from the above mentioned properties of exponential family distributions (see Appendix A and B).

The EP algorithm starts with an initial guess for the parameters $\tilde{m}$, $\tilde{V}$, $\tilde{p}$ and $\tilde{\pi}$ of $\mathcal{Q}$ and the parameters of the functions $\tilde{f}_i$ and iteratively matches the expectations under $\mathcal{Q}$ and $\mathcal{P}$ of the sufficient statistics of $\mathcal{Q}$ until convergence in the parameters is reached:

$$\mathbb{E}_{\mathcal{Q}}\left[(\beta, \beta\beta^T, Z, \Gamma)^T\right] \stackrel{!}{=} \mathbb{E}_{\mathcal{P}}\left[(\beta, \beta\beta^T, Z, \Gamma)^T\right]. \tag{4.16}$$

The sufficient statistic of a normal distribution $\mathcal{N}(x|\mu, \sigma^2)$ is (x, x^2), while the sufficient statistic of a (N-dimensional) multivariate normal distribution $\mathcal{N}(x|\mu, \Sigma)$ is (x, xx^T) with $x \in \mathbb{R}^N$ and $xx^T \in \mathbb{R}^{N\times N}$. The sufficient statistic of a Bernoulli distribution $\text{Bern}(x|p)$ is just x (see Appendix A and Appendix B).

Since $\mathcal{P}$ and $\mathcal{Q}$ are factored (equations (4.9) and (4.10)), we update the parameters in expectation propagation by taking turns updating $\tilde{f}_1$, $\tilde{f}_2$, $\tilde{f}_3$ and $\tilde{f}_4$. The update operations are derived in the next section.

4.4. Expectation propagation update operations

We need to initialize all parameters of the respective normal and Bernoulli distributions before the start of the expectation propagation algorithm. It is an advantage of a Bayesian framework that one can include prior information into these initial parameters and give the algorithm a "head start". This is especially useful for the prior probabilities for feature inclusion (p_0) and/or group inclusion π_0. If initial probabilities are not provided, we set the default parameters $p_{0,n} = \pi_{0,g} = \frac{1}{2}$, $n = 1, \ldots, N$, $g = 1, \ldots, G$. Also, the parameter σ_0 for the random noise needs to be chosen. George and McCulloch [30] include the estimation of σ_0 by adding another prior for this parameter that then needs to be updated along with the other probabilities (they use an inverse-gamma prior distribution). In this work, to reduce the mathematical and numerical overhead, we initialize this parameter in the beginning and keep it untouched during the algorithm. Our simulations do not indicate that using an inverse-gamma prior distribution is advantageous (Section 5.1.1). Finally we need to initialize the parameter σ_{slab}, we discuss and analyze this parameter in Section 5.1.1.

An important aspect is numerical stability. Since we multiply many (small) probabilities, we do not use the probability parameters of the Bernoulli distribu-

tions directly, but instead their logit-transformed versions: $r = \text{logit}(p) = \log \frac{p}{1-p}$ and $\varrho = \text{logit}(\pi)$, as was done by Hernández-Lobato [42]. Further numerical issues are discussed in Section 4.5.

In the expectation propagation framework from Section 3.5.2, more precisely the factorizations in equations (3.14) and (3.15), we do not consider the factored distributions $\tilde{f}_i$, $i = 1, \ldots, 4$ directly, but rather we use the fully factorized [11, p. 516] feature specific distributions $\tilde{f}_{i,n}$, $i = 1, \ldots, 4$, $n = 1, \ldots, N$. In the case of $\tilde{f}_{2,n}$ this translates to $\tilde{V}_2$ being a diagonal matrix (instead of a dense matrix) and the feature-wise variances are updated independently of each other. We capture the dependent effects with the initialization of the $\tilde{V}_1$ matrix and those effects are translated into the $\tilde{V}$ parameter matrix in each iteration of the algorithm. Thus we do not match the expectations under $\mathcal{Q}$ and $\mathcal{P}$ of the original sufficient statistic $(\beta, \beta\beta^T, Z, \Gamma)$, but rather element-wise $(\beta_n, \beta_n^2, Z_n, \Gamma_n)$ separately.

4.4.1. Initialization of the parameters

The values of the $\tilde{f}_1$ parameters are easy to obtain and do not need to be re-estimated by the algorithm since $\tilde{f}_1$ and f_1 have the same form (a multivariate normal distribution), thus we derive the exact approximations for $\tilde{m}_1$ and $\tilde{V}_1$ directly from the ordinary least squares estimate $\beta \approx \hat{\beta} = (X^T X)^{-1} X^T y$. It is more convenient to save $\tilde{V}_1^{-1}$ and $\tilde{V}_1^{-1}\tilde{m}_1$ for the remaining operations of the algorithm. Also, $\tilde{V}_1^{-1}$ might not be of full rank and thus $\tilde{V}_1$ might not exist and in turn $\tilde{m}_1$ would not be unique. Thus, the initial and final estimates for the parameters of $\tilde{f}_1$ are given by:

$$\tilde{V}_1^{-1} = \frac{1}{\sigma_0^2} X^T X,$$
$$\tilde{V}_1^{-1}\tilde{m}_1 = \frac{1}{\sigma_0^2} X^T y.$$

The initial guess for the variance $\tilde{V}_{2,n}$ in $\tilde{f}_{2,n}$ is the prior probability of choosing feature n multiplied by the variance of the slab. The initial estimate for $\tilde{m}_{2,n}$ is the non-informative $\vec{0}$ and the initial values for $\tilde{p}_{2,n}$ as well as $\tilde{p}_{3,n}$ and $\tilde{\pi}_{3,n}$ will

default to p_0 and π_0 (more precisely, their logit versions).

$$\begin{aligned}\tilde{V}_2 &= \sigma^2_{\text{slab}} \cdot p_0,\\ \tilde{V}_2^{-1}\tilde{m}_2 &= \vec{0},\\ \tilde{r}_2 &= r_0,\\ \tilde{r}_3 &= r_0,\\ \tilde{\varrho}_3 &= \varrho_0.\end{aligned}$$

For the parameters of $\tilde{f}_4$ we have the same situation like $\tilde{f}_1$: Both $\tilde{f}_4$ and f_4 have the same form (a product of G Bernoulli distributions) and thus we set $\tilde{\pi}_4$ equal to π_0 (respectively the logit counterparts) and do not touch these parameters for the rest of the algorithm:

$$\tilde{\varrho}_4 = \varrho_0.$$

With all these parameters of the factor distributions initialized we can finally initialize the parameters of the product distribution $\mathcal{Q}$, too. This is done by using the properties of the product of normal and Bernoulli distributions given in the Appendix A and B. Thus the initialization for $\mathcal{Q}$ is given by:

$$\begin{aligned}\tilde{V} &= \left(\tilde{V}_1^{-1} + \tilde{V}_2^{-1}\right)^{-1},\\ \tilde{m} &= \tilde{V}\left(\tilde{V}_1^{-1}\tilde{m}_1 + \tilde{V}_2^{-1}\tilde{m}_2\right),\\ \tilde{r} &= r_0,\\ \tilde{\varrho} &= \varrho_0.\end{aligned}$$

4.4.2. Iterative updates of the parameters

After initializing all the parameters we update iteratively the parameters of the $\tilde{f}_i$ distributions. We present the iterative updates with the following theorem:

Theorem 1. *Let $\mathcal{P}(\beta, Z, \Gamma)$ given by equations (4.4)-(4.8) be the true posterior distribution and let $\mathcal{Q}(\beta, Z, \Gamma)$ given by equations (4.10)-(4.15) be an approximation to $\mathcal{P}$. Then we can apply the expectation propagation algorithm and the iterative*

updates for the parameters of $\mathcal{Q}$ are given by:

$$\begin{aligned}
\tilde{V} &= \left(\tilde{V}_1^{-1} + \tilde{V}_2^{-1}\right)^{-1}, \\
\tilde{m} &= \tilde{V}\left(\tilde{V}_1^{-1}\tilde{m}_1 + \tilde{V}_2^{-1}\tilde{m}_2\right), \\
\tilde{r}_n &= \tilde{r}_{2,n} + \tilde{r}_{3,n}, \\
\tilde{\varrho}_g &= \tilde{\varrho}_{4,g} + \sum_{l:\mathcal{G}(l)=g} \tilde{\varrho}_{3,l},
\end{aligned}$$

where the iterative updates for the parameters $\tilde{V}_1$, $\tilde{m}_1$, $\tilde{V}_2$, $\tilde{m}_2$, $\tilde{r}_2$, $\tilde{r}_3$, $\tilde{\varrho}_3$ and $\tilde{\varrho}_4$ will be given in the proof.

Proof. The expectation propagation algorithm matches expectations under $\mathcal{P}$ and $\mathcal{Q}$ of the sufficient statistics of $\mathcal{Q}$ (equation (4.16)) by iteratively minimizing the Kullback-Leibler divergence $\mathrm{KL}(\mathcal{P}||\mathcal{Q})$. Since $\mathcal{P}$ and $\mathcal{Q}$ are factored (equations (4.9) and (4.10)), this is done by iterating through the factors $\tilde{f}_i$, $i = 1, \ldots, 4$ and updating their respective parameters and $\mathcal{Q}$ in turns. In fact, only $i = 2, 3$ are considered, since f_1 and $\tilde{f}_1$ respectively f_4 and $\tilde{f}_4$ have the same form by choice. As such, the values for $\tilde{V}_1$ and $\tilde{m}_1$ (respectively $\tilde{V}_1^{-1}$ and $\tilde{V}_1^{-1}\tilde{m}_1$) as well as $\tilde{\varrho}_4$ do not need to be updated, see Section 4.4.1, and thus their respective initial and final values are given by:

$$\begin{aligned}
\tilde{V}_1^{-1} &= \frac{1}{\sigma_0^2} X^T X, \\
\tilde{V}_1^{-1}\tilde{m}_1 &= \frac{1}{\sigma_0^2} X^T y, \\
\tilde{\varrho}_4 &= \varrho_0.
\end{aligned}$$

This leaves us with the updates for $\tilde{f}_2$ and $\tilde{f}_3$. First, we cycle through the $\tilde{f}_{2,n}$ and find the parameters of the updated $\tilde{f}_{2,n}^{\text{new}}$ via finding $\mathcal{Q}^{\setminus 2,n}$ as described in Section 4.3, then updating $\mathcal{Q}$ with the rules for the product of exponential family distributions given in the Appendix A and B. Second, the same is repeated for $\tilde{f}_3$ by cycling through the factors $\tilde{f}_{3,n}$ and finding the parameters of the updated $\tilde{f}_{3,n}^{\text{new}}$ via finding $\mathcal{Q}^{\setminus 3,n}$ and afterwards updating $\mathcal{Q}$ like before.

Updates for $\tilde{f}_{2,n}$: The parameters $\tilde{V}_n^{\setminus 2,n}$, $\tilde{m}_n^{\setminus 2,n}$ and $\tilde{r}_n^{\setminus 2,n}$ of $\mathcal{Q}^{\setminus 2,n} \propto \mathcal{Q}/\tilde{f}_{2,n}$ are

derived by using the rules for quotients of Bernoulli or normal distributions:

$$\begin{aligned}
\tilde{V}_n^{\setminus 2,n} &= \left(\tilde{V}_{nn}^{-1} - \tilde{V}_{2,n}^{-1}\right)^{-1}, \\
\tilde{m}_n^{\setminus 2,n} &= \tilde{V}_n^{\setminus 2,n} \cdot \left(\tilde{V}_{nn}^{-1} \cdot \tilde{m}_n - \tilde{V}_{2,n}^{-1} \cdot \tilde{m}_{2,n}\right), \\
\tilde{r}_n^{\setminus 2,n} &= \tilde{r}_n - \tilde{r}_{2,n}.
\end{aligned}$$

We find the updated $\tilde{f}_{2,n}^{\text{new}}$ by minimizing the Kullback-Leibler divergence between $f_{2,n} \cdot \mathcal{Q}^{\setminus 2,n}$ and $\tilde{f}_{2,n}^{\text{new}} \cdot \mathcal{Q}^{\setminus 2,n}$:

$$\mathrm{KL}(f_{2,n} \cdot \mathcal{Q}^{\setminus 2,n} \cdot \frac{1}{\mathsf{N}} || \tilde{f}_{2,n}^{\text{new}} \cdot \mathcal{Q}^{\setminus 2,n} \cdot \frac{1}{\tilde{\mathsf{N}}}),$$

where N and $\tilde{\mathsf{N}}$ are the appropriate normalizing constants.

Since we are only updating the marginal parameters $\tilde{r}_{2,n}^{\text{new}}$, $\tilde{V}_{2,n}^{\text{new}}$ and $\tilde{m}_{2,n}^{\text{new}}$, we factorize $\mathcal{Q}^{\setminus 2,n}$ and N in respect to n and will thus minimize

$$\mathrm{KL}(f_{2,n} \cdot \mathcal{Q}_n^{\setminus 2,n} \cdot \frac{1}{\mathsf{N}_n} || \tilde{f}_{2,n}^{\text{new}} \cdot \mathcal{Q}_n^{\setminus 2,n} \cdot \frac{1}{\tilde{\mathsf{N}}_n}) = \mathrm{KL}(\hat{\mathcal{P}}_n || \mathcal{Q}_n^{\text{new}}), \tag{4.17}$$

$$\begin{aligned}
\text{where } f_{2,n} \cdot \mathcal{Q}_n^{\setminus 2,n} &= \left(Z_n \cdot \mathcal{N}(\beta_n | 0, \sigma_{\text{slab}}^2) + (1 - Z_n) \cdot \delta(\beta_n)\right) \cdot \mathcal{N}(\beta_n | \tilde{m}_n^{\setminus 2,n}, \tilde{V}_n^{\setminus 2,n}) \\
&\quad \cdot \mathrm{Bern}(Z_n | \tilde{p}_n^{\setminus 2,n}) \cdot \mathrm{Bern}(\Gamma_{\mathcal{G}(n)} | \tilde{\pi}_{\mathcal{G}(n)}) \\
\text{and thus } \mathsf{N}_n &= \int_{-\infty}^{+\infty} \sum_{Z_n=0,1} \sum_{\Gamma_{\mathcal{G}(n)}=0,1} f_{2,n} \cdot \mathcal{Q}_n^{\setminus 2,n} \mathrm{d}\beta \\
&= \tilde{p}_n^{\setminus 2,n} \cdot \mathcal{N}(\tilde{m}_n^{\setminus 2,n}, \tilde{V}_n^{\setminus 2,n} + \sigma_{\text{slab}}^2) + (1 - \tilde{p}_n^{\setminus 2,n}) \cdot \mathcal{N}(\tilde{m}_n^{\setminus 2,n}, \tilde{V}_n^{\setminus 2,n}).
\end{aligned}$$

Minimizing (4.17) is the same as matching the expectations of the sufficient statistics Z_n, β_n and β_n^2 under the probabilities from $\hat{\mathcal{P}}_n$ and $\mathcal{Q}_n^{\text{new}}$, this gives us the updated parameters $\hat{p}_n^{\text{new}}$, $\hat{m}_n^{\text{new}}$ and $\hat{V}_n^{\text{new}}$ of $\mathcal{Q}_n^{\text{new}}$ (here we dropped all subscripts $_n$ or $_{2,n}$ and superscripts $^{\setminus 2,n}$ for better readability):

$$\begin{aligned}
\hat{p}^{\text{new}} \equiv \mathbb{E}_{\mathcal{Q}^{\text{new}}}[Z] \overset{!}{=} \mathbb{E}_{\hat{\mathcal{P}}}[Z] &= \sum_{Z=0,1} \int_{-\infty}^{\infty} \sum_{\Gamma=0,1} Z \cdot \hat{\mathcal{P}}(\beta, Z, \Gamma) \mathrm{d}\beta \\
&= 0 + \frac{1}{\mathsf{N}} \mathcal{N}(\tilde{m}, \tilde{V} + \sigma_{\text{slab}}^2) \cdot \tilde{p} \cdot \tilde{\pi} + \frac{1}{\mathsf{N}} \mathcal{N}(\tilde{m}, \tilde{V} + \sigma_{\text{slab}}^2) \cdot \tilde{p} \cdot (1 - \tilde{\pi}) \\
&= \frac{\tilde{p}}{\mathsf{N}} \cdot \mathcal{N}(\tilde{m}, \tilde{V} + \sigma_{\text{slab}}^2),
\end{aligned}$$

$$\hat{m}^{\text{new}} \equiv \mathbb{E}_{\mathcal{Q}^{\text{new}}}[\beta] \overset{!}{=} \mathbb{E}_{\hat{\mathcal{P}}}[\beta] = \tilde{m} + \tilde{V} \cdot \frac{\partial}{\partial \tilde{m}} \mathsf{N}, \tag{4.18}$$

$$\hat{V}^{\text{new}} \equiv \mathbb{E}_{\mathcal{Q}^{\text{new}}}[\beta^2] - \mathbb{E}_{\mathcal{Q}^{\text{new}}}[\beta]^2 \overset{!}{=} \mathbb{E}_{\hat{\mathcal{P}}}[\beta^2] - \mathbb{E}_{\hat{\mathcal{P}}}[\beta]^2 = \tilde{V} - \tilde{V}^2 \cdot \left((\frac{\partial}{\partial \tilde{m}} \mathsf{N})^2 - 2 \cdot \frac{\partial}{\partial \tilde{V}} \mathsf{N} \right). \tag{4.19}$$

Equalities (4.18) and (4.19) are from Minka [65, p. 15].

Remember, these are the new parameters of $\mathcal{Q}_n^{\text{new}}$. To find the updated parameters of $\tilde{f}_{2,n}^{\text{new}}$, we need to divide $\mathcal{Q}_n^{\text{new}}$ by $\mathcal{Q}_n^{\setminus 2,n}$ and use the rules for the quotient of Bernoulli or normal distributions (see Appendix A and B):

$$\begin{aligned}
\tilde{r}_{2,n}^{\text{new}} &= \hat{r}_n^{\text{new}} - \tilde{r}_n^{\setminus 2,n}, \\
\tilde{V}_{2,n}^{\text{new}} &= (1/\hat{V}_n^{\text{new}} - 1/\tilde{V}_n^{\setminus 2,n})^{-1}, \\
\tilde{m}_{2,n}^{\text{new}} &= \tilde{V}_{2,n}^{\text{new}} \cdot (\hat{m}_n^{\text{new}}/\hat{V}_n^{\text{new}} - \tilde{m}_n^{\setminus 2,n}/\tilde{V}_n^{\setminus 2,n}).
\end{aligned}$$

After some calculus and arithmetic (find the derivatives of N_n with respect to $\tilde{m}_n^{\setminus 2,n}$ and $\tilde{V}_n^{\setminus 2,n}$, then plug-in $\hat{p}_n^{\text{new}}$, $\hat{m}_n^{\text{new}}$ and $\hat{V}_n^{\text{new}}$ and rearrange) we get the final analytical parameter updates of $\tilde{r}_{2,n}^{\text{new}}$, $\tilde{V}_{2,n}^{\text{new}}$ and $\tilde{m}_{2,n}^{\text{new}}$:

$$\tilde{r}_{2,n}^{\text{new}} = \frac{1}{2} \cdot \left(\log \left(\frac{\tilde{V}_n^{\setminus 2,n}}{\tilde{V}_n^{\setminus 2,n} + \sigma_{\text{slab}}^2} \right) + (\tilde{m}_n^{\setminus 2,n})^2 \cdot \left(1/\tilde{V}_n^{\setminus 2,n} - 1/(\tilde{V}_n^{\setminus 2,n} + \sigma_{\text{slab}}^2) \right) \right), \tag{4.20}$$

$$\tilde{V}_{2,n}^{\text{new}} = \frac{1}{a_n^2 - b_n} - \tilde{V}_n^{\setminus 2,n}, \tag{4.21}$$

$$\tilde{m}_{2,n}^{\text{new}} = \tilde{m}_n^{\setminus 2,n} - a_n \cdot (\tilde{V}_{2,n}^{\text{new}} + \tilde{V}_n^{\setminus 2,n}), \tag{4.22}$$

$$\begin{aligned}
\text{with } a_n &= p_n^{\text{aux}} \cdot \frac{\tilde{m}_n^{\setminus 2,n}}{\tilde{V}_n^{\setminus 2,n} + \sigma_{\text{slab}}^2} + (1 - p_n^{\text{aux}}) \cdot \frac{\tilde{m}_n^{\setminus 2,n}}{\tilde{V}_n^{\setminus 2,n}}, \\
b_n &= p_n^{\text{aux}} \cdot \frac{(\tilde{m}_n^{\setminus 2,n})^2 - \tilde{V}_n^{\setminus 2,n} - \sigma_{\text{slab}}^2}{(\tilde{V}_n^{\setminus 2,n} + \sigma_{\text{slab}}^2)^2} + (1 - p_n^{\text{aux}}) \cdot \frac{(\tilde{m}_n^{\setminus 2,n})^2 - \tilde{V}_n^{\setminus 2,n}}{(\tilde{V}_n^{\setminus 2,n})^2} \\
\text{and } p_n^{\text{aux}} &= \text{sigmoid}(\tilde{r}_{2,n}^{\text{new}} + \tilde{r}_n^{\setminus 2,n}).
\end{aligned}$$

These do not rely on the parameters of $\mathcal{Q}_n^{\text{new}}$ anymore, so in practice one calculates the updates of $\tilde{f}_{2,n}^{\text{new}}$ directly from the parameters of $\mathcal{Q}^{\setminus 2,n}$.

The updates for $\mathcal{Q}$ after updating $\tilde{f}_{2,n}^{\text{new}}$ are derived from the rules for the product

of Bernoulli and normal distributions:

$$\begin{aligned}
\tilde{V} &= \left(\tilde{V}_1^{-1} + \left(\tilde{V}_2^{\text{new}}\right)^{-1}\right)^{-1}, \\
\tilde{m} &= \tilde{V}\left(\tilde{V}_1^{-1}\tilde{m}_1 + \left(\tilde{V}_2^{\text{new}}\right)^{-1}\tilde{m}_2^{\text{new}}\right), \\
\tilde{r}_n &= \tilde{r}_{2,n}^{\text{new}} + \tilde{r}_{3,n}, \\
&\tilde{\varrho} \text{ does not change.}
\end{aligned} \tag{4.23}$$

Updates for $\tilde{f}_{3,n}$: The parameters $\tilde{r}_n^{\setminus 3,n}$ and $\tilde{\varrho}_n^{\setminus 3,n}$ of $\mathcal{Q}^{\setminus 3,n} \propto \mathcal{Q}/\tilde{f}_{3,n}$ are derived by using the rules for quotients of Bernoulli distributions:

$$\begin{aligned}
\tilde{\varrho}_n^{\setminus 3,n} &= \tilde{\varrho}_{\mathcal{G}(n)} - \tilde{\varrho}_{3,n}, \\
\tilde{r}_n^{\setminus 3,n} &= \tilde{r}_n - \tilde{r}_{3,n}.
\end{aligned}$$

We find the updated $\tilde{f}_{3,n}^{\text{new}}$ by minimizing the Kullback-Leibler divergence between $f_{3,n} \cdot \mathcal{Q}^{\setminus 3,n}$ and $\tilde{f}_{3,n}^{\text{new}} \cdot \mathcal{Q}^{\setminus 3,n}$:

$$\mathrm{KL}(f_{3,n} \cdot \mathcal{Q}^{\setminus 3,n} \cdot \frac{1}{\mathsf{N}} || \tilde{f}_{3,n}^{\text{new}} \cdot \mathcal{Q}^{\setminus 3,n} \cdot \frac{1}{\tilde{\mathsf{N}}}),$$

where N and $\tilde{\mathsf{N}}$ are the appropriate normalizing constants.

Since we are only updating the marginal parameters $\tilde{r}_{3,n}^{\text{new}}$ and $\tilde{\varrho}_{3,n}^{\text{new}}$, we factorize $\mathcal{Q}^{\setminus 3,n}$ and N with respect to n and will thus minimize

$$\mathrm{KL}(f_{3,n} \cdot \mathcal{Q}_n^{\setminus 3,n} \cdot \frac{1}{\mathsf{N}_n} || \tilde{f}_{3,n}^{\text{new}} \cdot \mathcal{Q}_n^{\setminus 3,n} \cdot \frac{1}{\tilde{\mathsf{N}}_n}) = \mathrm{KL}(\hat{\mathcal{P}}_n || \mathcal{Q}_n^{\text{new}}), \tag{4.24}$$

$$\begin{aligned}
\text{where } f_{3,n} \cdot \mathcal{Q}_n^{\setminus 3,n} &= \left(\Gamma_{\mathcal{G}(n)} \cdot \mathrm{Bern}(Z_n | p_{0,n}) + (1 - \Gamma_{\mathcal{G}(n)}) \cdot \delta(Z_n)\right) \cdot \mathcal{N}(\beta_n | \tilde{m}_n^{\setminus 3,n}, \tilde{V}_n^{\setminus 3,n}) \\
&\quad \cdot \mathrm{Bern}(Z_n | \tilde{p}_n^{\setminus 3,n}) \cdot \mathrm{Bern}(\Gamma_{\mathcal{G}(n)} | \tilde{\pi}_n^{\setminus 3,n}) \\
\text{and thus } \mathsf{N}_n &= \int_{-\infty}^{+\infty} \sum_{Z_n=0,1} \sum_{\Gamma_{\mathcal{G}(n)}=0,1} f_{3,n} \cdot \mathcal{Q}_n^{\setminus 3,n} \mathrm{d}\beta \\
&= \tilde{\pi}_n^{\setminus 3,n} \cdot \left(\tilde{p}_n^{\setminus 3,n} p_{0,n} + (1 - \tilde{p}_n^{\setminus 3,n})(1 - p_{0,n})\right) + (1 - \tilde{\pi}_n^{\setminus 3,n}) \cdot (1 - \tilde{p}_n^{\setminus 3,n}).
\end{aligned}$$

Minimizing (4.24) is the same as matching the expectations of the sufficient statistics Z_n, β_n and β_n^2 under the probabilities from $\hat{\mathcal{P}}_n$ and $\mathcal{Q}_n^{\text{new}}$, this gives us the updated parameters $\hat{p}_n^{\text{new}}$ and $\hat{\pi}_n^{\text{new}}$ of $\mathcal{Q}_n^{\text{new}}$ (here we drop all subscripts $_n$ or $_{3,n}$

and superscripts $^{\setminus 3,n}$ for better readability):

$$\begin{aligned}
\hat{p}^{\text{new}} \equiv \mathbb{E}_{\mathcal{Q}^{\text{new}}}[Z] \overset{!}{=} \mathbb{E}_{\hat{\mathcal{P}}}[Z] &= \sum_{Z=0,1} \int_{-\infty}^{\infty} \sum_{\Gamma=0,1} Z \cdot \hat{\mathcal{P}}(\beta, Z, \Gamma) \mathrm{d}\beta \\
&= \frac{\tilde{p}\tilde{\pi} p_0}{\mathsf{N}} \\
\hat{\pi}^{\text{new}} \equiv \mathbb{E}_{\mathcal{Q}^{\text{new}}}[\Gamma] \overset{!}{=} \mathbb{E}_{\hat{\mathcal{P}}}[\Gamma] &= \sum_{Z=0,1} \int_{-\infty}^{\infty} \sum_{\Gamma=0,1} Z \cdot \hat{\mathcal{P}}(\beta, Z, \Gamma) \mathrm{d}\beta \\
&= \frac{\tilde{p}\tilde{\pi} p_0 + (1-\tilde{p})(1-p_0)\tilde{\pi}}{\mathsf{N}}.
\end{aligned}$$

These are the new parameters of $\mathcal{Q}_n^{\text{new}}$. To find the updated parameters of $\tilde{f}_{3,n}^{\text{new}}$, we need to divide $\mathcal{Q}_n^{\text{new}}$ by $\mathcal{Q}_n^{\setminus 3,n}$ and use the rules for the quotient of Bernoulli distributions (see Appendix A):

$$\begin{aligned}
\tilde{r}_{3,n}^{\text{new}} &= \hat{r}_n^{\text{new}} - \tilde{r}_n^{\setminus 3,n}, \\
\tilde{\varrho}_{3,n}^{\text{new}} &= \hat{\varrho}_n^{\text{new}} - \tilde{\varrho}_n^{\setminus 3,n}.
\end{aligned}$$

After some arithmetic operations we get the final analytical parameter updates of $\tilde{\varrho}_{3,n}^{\text{new}}$ and $\tilde{r}_{3,n}^{\text{new}}$:

$$\begin{aligned}
\tilde{\varrho}_{3,n}^{\text{new}} &= -\log(1-\tilde{p}_n^{\setminus 3,n}) + \log(\tilde{p}_n^{\setminus 3,n} \cdot p_{0,n} + (1-\tilde{p}_n^{\setminus 3,n}) \cdot (1-p_{0,n})) \\
&= \log(1 + p_{0,n} \cdot (\exp(\tilde{r}_n^{\setminus 3,n}) - 1)), \qquad (4.25) \\
\tilde{r}_{3,n}^{\text{new}} &= \operatorname{logit}(\tilde{\pi}_n^{\setminus 3,n} \cdot p_{0,n}) \\
&= \log p_{0,n} - \log(1 - p_{0,n} + \exp(-\tilde{\varrho}_n^{\setminus 3,n})). \qquad (4.26)
\end{aligned}$$

These do not rely on the parameters of $\mathcal{Q}_n^{\text{new}}$, so in practice one calculates the updates of $\tilde{f}_{3,n}^{\text{new}}$ directly.

The updates for $\mathcal{Q}$ after updating $\tilde{f}_{3,n}$ are derived from the rules for the product of Bernoulli distributions:

$$\begin{aligned}
\tilde{V} &\text{ does not change,} \\
\tilde{m} &\text{ does not change,} \\
\tilde{\varrho}_{\mathcal{G}(n)} &= \tilde{\varrho}_{4,\mathcal{G}(n)} + \sum_{l:\mathcal{G}(l)=\mathcal{G}(n)} \tilde{\varrho}_{3,l}^{\text{new}}, \\
\tilde{r}_n &= \tilde{r}_{2,n} + \tilde{r}_{3,n}^{\text{new}}.
\end{aligned}$$

□

Corollary 1. *If $p_{0,n} = 0.5$, the update operations for $\tilde{\varrho}_{3,n}^{\text{new}}$ and $\tilde{r}_{3,n}^{\text{new}}$ are given by:*

$$\begin{aligned}\tilde{\varrho}_{3,n}^{\text{new}} &= \log(0.5) + \log(1 + \exp(\tilde{r}_n^{\setminus 3,n})),\\ \tilde{r}_{3,n}^{\text{new}} &= -\log(1 + 2\exp(-\tilde{\varrho}_n^{\setminus 3,n})).\end{aligned}$$

Thus, the (reasonable) choice of $p_{0,n} = 0.5$ is numerically advantageous, too, since it allows for the efficient use of `log1p` and `logsumexp` functions in the implementation of the update operations.

4.5. Numerical issues and solutions

Note that the calculation of $\tilde{V}$ in equation (4.23) is the bottleneck of the algorithm's complexity. In the case of $N > M$ (more features than observations) there is a more efficient way to invert the matrix from equation (4.23), that is the Woodbury formula (see Hernández-Lobato et al. [41]):

$$\begin{aligned}\tilde{V} &= \left(\tilde{V}_1^{-1} + \tilde{V}_2^{-1}\right)^{-1}\\ &= \tilde{V}_2 - \tilde{V}_2 X^T \left(\sigma_0^2 I + X\tilde{V}_2 X^T\right)^{-1} X\tilde{V}_2\end{aligned}$$

Instead of inverting an $(N \times N)$-matrix, because of the choice of $\tilde{V}_1^{-1} = \frac{1}{\sigma_0^2} X^T X$ the Woodbury matrix identity allows us to invert an $(M \times M)$-matrix instead.

Rarely the variance $\tilde{V}_n^{\setminus 2,n}$ might be negative before the update operation, in this case we do not perform an update operation on $\tilde{f}_{2,n}$. In addition, the variance $\tilde{V}_{2,n}^{\text{new}}$ might be negative after the corresponding update operation, which is a well known problem [65]. This arises as a compensation for errors in the first factor $\tilde{f}_1$, but hampers the ability of the expectation propagation algorithm to converge [78]. To improve convergence, we follow the observations of [42] and apply the constraint of $\tilde{V}_{2,n}^{\text{new}} > 0$, such that we replace its value by a large constant (100) whenever it turns out to be negative.

Minimizing the Kullback-Leibler divergence is an optimization problem with a single global optimum which can be found by matching the sufficient statistics as described above (Section 3.5.2, Bishop [11]). The expectation propagation algorithm is not guaranteed to converge to this global solution, but often converges to

a fixed point [64]. Furthermore, Minka and Lafferty [66] introduce damping of the updated factors to secure convergence of expectation propagation:

$$\tilde{f}_{i,n} = \left(\tilde{f}_{i,n}^{\text{new}}\right)^{\alpha} \cdot \left(\tilde{f}_{i,n}^{\text{old}}\right)^{1-\alpha}, \text{ where } \alpha \in [0,1].$$

By setting $\alpha = 0.9$ and decaying it by 1% in every step of the algorithm we follow the advice of [41]. The updated parameters $(\tilde{V}_{2,n}^{\text{new}})^{-1}$, $(\tilde{V}_{2,n}^{\text{new}})^{-1}\tilde{m}_{2,n}^{\text{new}}$, $\tilde{r}_{2,n}^{\text{new}}$, $\tilde{r}_{3,n}^{\text{new}}$ and $\tilde{\varrho}_{3,n}^{\text{new}}$ before updating the parameters of $\mathcal{Q}$ are then given by:

$$\begin{aligned}
(\tilde{V}_{2,n}^{\text{new}})^{-1} &\leftarrow \alpha \cdot (\tilde{V}_{2,n}^{\text{new}})^{-1} + (1-\alpha) \cdot (\tilde{V}_{2,n}^{\text{old}})^{-1}, \\
(\tilde{V}_{2,n}^{\text{new}})^{-1}\tilde{m}_{2,n}^{\text{new}} &\leftarrow \alpha \cdot (\tilde{V}_{2,n}^{\text{new}})^{-1}\tilde{m}_{2,n}^{\text{new}} + (1-\alpha) \cdot (\tilde{V}_{2,n}^{\text{old}})^{-1}\tilde{m}_{2,n}^{\text{old}}, \\
\tilde{r}_{2,n}^{\text{new}} &\leftarrow \alpha \cdot \tilde{r}_{2,n}^{\text{new}} + (1-\alpha) \cdot \tilde{r}_{2,n}^{\text{old}}, \\
\tilde{r}_{3,n}^{\text{new}} &\leftarrow \alpha \cdot \tilde{r}_{3,n}^{\text{new}} + (1-\alpha) \cdot \tilde{r}_{3,n}^{\text{old}}, \\
\tilde{\varrho}_{3,n}^{\text{new}} &\leftarrow \alpha \cdot \tilde{\varrho}_{3,n}^{\text{new}} + (1-\alpha) \cdot \tilde{\varrho}_{3,n}^{\text{old}}.
\end{aligned}$$

4.6. Implementation

In this chapter, we derived a new method for Bayesian sparse-group feature selection with expectation propagation. Our implementation of the method within the statistical programming language and environment R [74] is available as a package ("`dogss`"). The computationally most demanding operations (the Woodbury formula along with matrix inversion and matrix multiplication) were implemented efficiently in C++ and called via the R package `RcppEigen` [9]. The function call within R is `dogss(X, Y, G)` with input data matrix `X`, response `Y` and a vector of group memberships `G`. If `G=NULL`, the standard spike-and-slab without grouping is called. Further parameters that can be specified are the noise variance σ_0 (`sigma_0`, default 1), the slab variance σ_{slab} (`sigma_slab`, default 2), the damping parameter α (`damping`, default 0.9), whether to scale the data beforehand (`standardize`, default `FALSE`), whether to include an intercept (`intercept`, default `FALSE`), the machine precision to determine convergence (`tol`, default 10^{-5}) and the maximum number of iterations if no convergence is reached before (`iter.max`, default 100). The function initializes all parameters, runs through the iterations until convergence or the maximum number of iterations is reached

and returns $\tilde{m}$ as an estimate of the vector of coefficients β (`m_final`) as well as the feature- and group-wise probabilities $\tilde{p}$ and $\tilde{\pi}$ (`p_features` and `p_groups`). Furthermore, the cross-validation method `cv_dogss()` with the same parameters like `dogss()` and the additional parameter `nfolds` (default $k = 10$) runs k-fold cross-validation and returns the cross-validated estimates of the parameters.

4.7. Summary

We introduced a model for Bayesian feature selection that accounts for between- and within group sparsity. We derived the closed-form solutions of the corresponding expectation propagation updates of the model parameters to obtain a fast and deterministic algorithm for this framework. Though the procedure gives rise to some numerical issues, we presented solutions to these problems. Thus we were able to provide a new and fast sparse-group Bayesian feature selection method along with an efficient implementation.

5. Evaluation

In this chapter we compare our Bayesian model and algorithmic framework to alternative methods for the task of feature selection with sparsity on the between- and within-group level. To this end, we study signal recovery and network reconstruction by means of neighborhood selection. We apply the methods to simulated and experimental data.

5.1. Results on simulated data

We study two simulation settings:

1. **signal recovery** (Section 5.1.1): Simulations from $y = X\beta + \varepsilon$ with known β and noise ε, where we try to recover β from the observations y and X, make predictions on held-out data and measure computing time,

2. **network reconstruction** (Section 5.1.2): Simulated Gaussian graphical networks with random observations based on known precision matrices, where we want to recover the correct networks and make predictions on held-out data.

We compare six different methods:

(i) `dogss`: Our implementation of our new method (see Chapter 4), the sparse-group Bayesian feature selection with expectation propagation. `dogss` is an abbreviation for **do**uble **g**roup-sparse **s**pike-and-**s**lab.

(ii) `ssep`: Our implementation (as a special case of `dogss` without grouping of features) of the standard **s**pike-and-**s**lab (see Chapter 3) with **e**xpectation **p**ropagation.

(iii) `sgl`: The sparse-group lasso from Simon et al. [80], implemented within their R package `SGL`.

(iv) `gglasso`: The group lasso from Yuan and Lin [96], implemented by Yang and Zou [95] within the R package `gglasso`.

(v) `lasso`: The standard lasso from Tibshirani [88], implemented by Friedman et al. [28] within the R package `glmnet`.

(vi) `bsgsss`: An implementation of a different sparse-group Bayesian feature selection with Gibbs sampling from Xu and Ghosh [94], implemented by Liquet et al. [55] within the R package `MBSGS`.

All simulations, implementations, top-level method calls and calculations for evaluation of the results were done in R, a statistical programming language and environment [74].

One way to visually evaluate the performance of feature selection methods is a needle plot. The needle plot is a bar plot where every feature corresponds to one bar, aligned by index n along the x-axis. The height of the bar (relative to the y-axis) corresponds to the value of the regression coefficient β_n of the n-th feature. The "original" needle plot shows the true coefficients β, while the needle plots corresponding to the different feature selection methods show the retrieved estimates $\hat{\beta}$. Thus one can inspect how well a method recovers the true signal and identify advantages/disadvantages of different methods.

The needle plot shows only one final estimate of a method. If a ranking of retrieved coefficients of the features is provided (which is the case for all methods considered), one can measure the performance of a method in respect to correctly identified non-zero coefficients (without considering the actual value of the coefficients) along the ranking. This is done by evaluating the receiver operating characteristics (ROC) and precision-recall (PR) curves. These curves are based on the measures of true-positive rate (TPR, sensitivity or recall), false-positive rate (FPR or 1-specificity) and precision (Prec). For a given gold standard of positive and negative labels (for signal recovery: the true non-zero and zero elements of β, where the number of non-zero elements is k, for network reconstruction: the number k of true edges within the network graph), we assess the performance of an

algorithm to retrieve these labels along a tuning parameter λ by counting the number of true positives TP (the algorithm retrieves a correct non-zero element) and false positives FP (the algorithm retrieves an element as non-zero that is actually zero):

$$\begin{aligned}\mathrm{TPR}(\lambda) &= \frac{\mathrm{TP}(\lambda)}{k},\\ \mathrm{FPR}(\lambda) &= \frac{\mathrm{FP}(\lambda)}{N^* - k},\\ \mathrm{Prec}(\lambda) &= \frac{\mathrm{TP}(\lambda)}{\mathrm{TP}(\lambda) + \mathrm{FP}(\lambda)},\end{aligned}$$

where N^* is here the number of all features for the problem of signal recovery ($N^* = N$) or the number of all possible edges $N^* = (P^2 - P)/2$ in the undirected network graph of P nodes. The ROC curve plots the false-positive rate against the true-positive rate along the tuning parameter λ, while the precision-recall curve plots the true-positive rate against the precision. Thus every point on the plotted line gives the TPR and FPR (or TPR and precision) for a certain value of the tuning parameter. The "best" value of the tuning parameter would be a point close to the upper-left corner of the ROC curve plot and close to the upper-right corner of the PR curve plot. If we do not wish to give a certain $\lambda^\star$ as cutoff, but rather judge the performance of an algorithm regardless of the choice of the cutoff, we can assess the performance by calculating the area under the (ROC or PR) curve, or AUROC respectively AUPR. An estimator that chooses positive and negative labels randomly has AUROC $= 0.5$ and AUPR $= k/N^*$.

The ranking of the recovered features for the Bayesian approaches (`dogss`, `ssep` and `bsgsss`) is done via the probabilities associated with every feature. The ranking of features for the lasso methods (`lasso`, `gglasso` and `sgl`) is along the parameter λ. Figure 5.1 shows example ROC curves and PR curves for different methods applied to the signal reconstruction problem described in Section 5.1.1, while Table 5.1 gives the corresponding AUROC and AUPR values.

We also compare running times of the algorithms for the problem of signal recovery, and to this end we set the machine precision tolerance of all methods to the same value ($10^{-5} = 0.00001$) and allowed a maximum of 1000 iterations each. On top of this we provided the same λ sequence (of 100 λ values) to all lasso

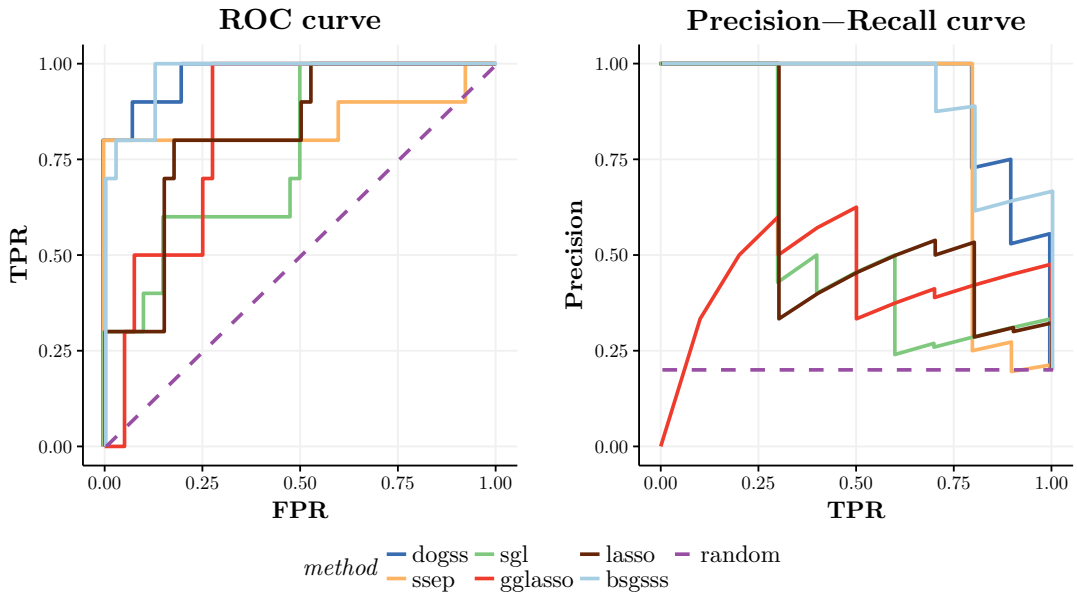

Figure 5.1.: Examples for ROC and Precision-Recall curves for the results of different feature selection methods, along with a random estimator. The parameters for the simulated data are $(M, N, G, k, \sigma_0) = (30, 50, 10, 10, 1)$.

method	dogss	ssep	sgl	gglasso	lasso	bsgsss	random
AUROC	0.97	0.85	0.76	0.84	0.82	0.97	0.5
AUPR	0.93	0.85	0.55	0.43	0.59	0.92	0.2

Table 5.1.: AUROC and AUPR values for the example ROC and PR curves in Figure 5.1.

methods, which is the one calculated by the implementation within the `glmnet` package of the standard lasso.

As a complement to the ROC/PR curve analysis, we evaluate the prediction errors of the different methods. For every scenario, we generate 100 additional observations as a test set. We calculate relative prediction errors on the test set using the retrieved parameters from the training data. In signal recovery, this corresponds to the following relative residual sum of squared errors E for test data $y_{\text{test}} = (y_m)_{m=1}^{100}$ and $X_{\text{test}} = (x_{mn})$, $m = 1, \ldots, 100$, $n = 1, \ldots, N$:

$$E = \frac{\sum_{m=1}^{100}(y_m - \sum_{n=1}^{N} \hat{\beta}_n x_{mn})^2}{\sum_{m=1}^{100} y_m^2}.$$

In network reconstruction, we have test data $X_{\text{test}} = (x_{mp})$, $m = 1, \ldots, 100$, $p = 1, \ldots, P$ and retrieve a regression coefficient matrix $\hat{B} = (\hat{\beta}_{p_1 p_2})$, $p_1, p_2 = 1, \ldots, P$ with $\hat{\beta}_{pp} = 0$, from the neighborhood selection framework:

$$E = \frac{\sum_{m=1}^{100} \sum_{p_1=1}^{P} (x_{mp_1} - \sum_{\substack{p_2=1 \\ p_2 \neq p_1}}^{P} \beta_{p_1 p_2} x_{mp_2})^2}{\sum_{m=1}^{100} \sum_{p=1}^{P} x_{mp}^2}.$$

In both cases, we choose the values $\hat{\beta}$ or $\hat{B}$ for prediction by cross-validation (see Section 3.6).

5.1.1. Simulations: Signal recovery

In this section we will show the capability of our model and algorithm to extract meaningful results from data with grouped features. The usefulness of our proposed algorithm and implementation should be measured on three aspects:

- its ability to choose the correct/important features,
- its ability to recover the correct coefficients of these features,
- computing time.

To this end, we generate M random observations of N features, each drawn from a normal distribution with mean 0 and variance 1, this gives the data matrix X. For now, the features are drawn independently of each other. The features are

divided into G groups by sampling independently for every feature a group index from $\{1, \ldots, G\}$, thus we have different numbers of features $(N_1, \ldots, N_G)$ in every group. The resulting $(M \times N)$-matrix X of observations is multiplied with a k-sparse coefficient vector β of length N (k out of the N coefficients are different from zero, drawn independently from a uniform distribution on $[-5, 5]$). The k non-zero coefficients are are only chosen within 3 random groups, thus we have sparsity on the group level with 3 non-zero groups and $G - 3$ groups with all zero coefficients. Finally we add some noise $\varepsilon \sim \mathcal{N}(0, \sigma_0^2)$ and obtain the response vector $y = X\beta + \varepsilon$.

We feed the matrix X, the response y and the group indices into the different algorithms and compare the performance on the resulting estimates $\hat{\beta}$ of the vector of coefficients. See an example of the reconstructed coefficient vectors as a needle plot in Figure 5.2 for the setting of $(M, N, G, k, \sigma_0) = (30, 50, 10, 10, 1)$. In this simulation we can already see that the Bayesian approaches (`dogss`, `ssep`, `bsgsss`) do a very good job of reconstructing the correct signals. The group lasso `gglasso` chooses whole groups without within-group sparsity and as such has many false positives. The standard `lasso` and the sparse-group lasso `sgl` choose mostly correct coefficients, but underestimate their values.

Influence of M, N, G and k

First we analyze the performance on three conceptually different sets of parameters (M, N, G, k) with fixed $\sigma_0 = 1$ and columns of X drawn independently.

We evaluate the results of the methods on the simulated data on these aspects:

- boxplots of the AUROC and AUPR from 100 simulations with the same set of parameters,
- boxplots of the relative prediction error on 100 additionally generated observations (test sample).

The choice of parameters (M, N, G, k, σ_0) for the different settings of this section and the following ones is given in Table 5.2.

The simulation setting "small" of $(M, N, G, k, \sigma_0) = (30, 30, 5, 5, 1)$ describes a scenario where we actually have enough observations at hand $(M = N)$ for pre-

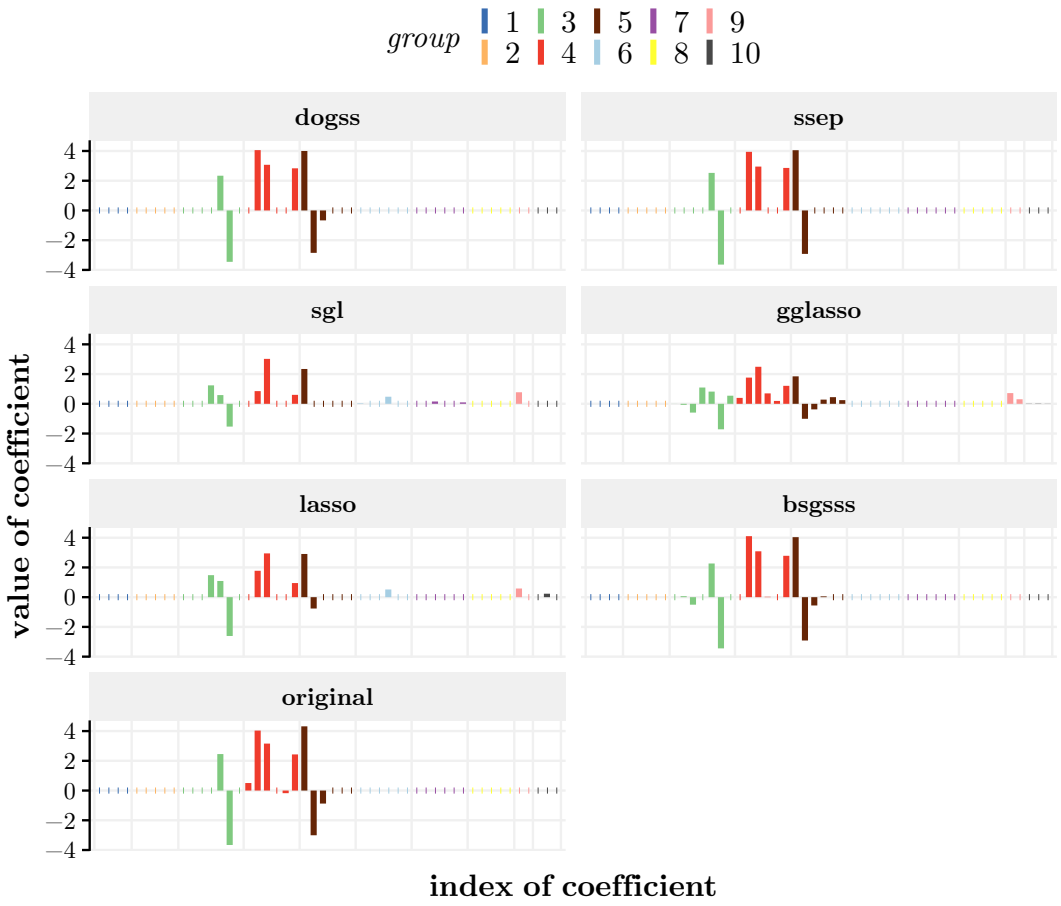

Figure 5.2.: Needle plot for signal recovery: Results from one simulation with parameters $(M, N, G, k, \sigma_0) = (30, 50, 10, 10, 1)$. Features are aligned by indices along the x-axis, the height of bars corresponds to the value of the coefficient. The "original" box shows the true signal, while the other six boxes show the retrieved coefficients from six different methods.

	M	N	G	k	σ_0	corr. structure
small	30	30	5	5	1	independent
medium	30	100	20	10	1	independent
large	100	1000	100	10	1	independent
noise	30	100	20	10	0 0.1 1 3 5	independent
correlation	30	100	20	10	1	independent pairwise groupwise

Table 5.2.: Choice of parameters for the different simulation settings in signal recovery.

dicting all of the coefficients, but we still have sparsity on the between-group and within-group level ($G = 5$, $k = 5$). Figure 5.3 shows the results, aggregated from 100 simulations. The AUROC for all methods except the group lasso is very high (almost 1 in many cases). The Bayesian methods perform a bit better than the standard lasso and sparse-group lasso. In regard of the AUPR measure, this trend is more remarkable, while the group lasso performs even worse. The group lasso does not take within-group sparsity into account and as such chooses too many coefficients which end up as false positives, we will see this behavior in all subsequent simulation scenarios. On the prediction error measure, we can see the same trends, with the Bayesian methods showing best results, while sparse-group lasso and lasso are close and group lasso performing worst. In this "small" simulation setting there is almost no difference if we include the grouping information or not: comparison of `dogss` versus `ssep` and `sgl` versus `lasso` shows basically no difference. We see important differences in the computing time: the Gibbs sampling approach `bsgsss` takes roughly 1000 times as long as the expectation propagation based calculations and the group lasso or standard lasso. The sparse-group lasso is approximately 100 times slower than the standard lasso.

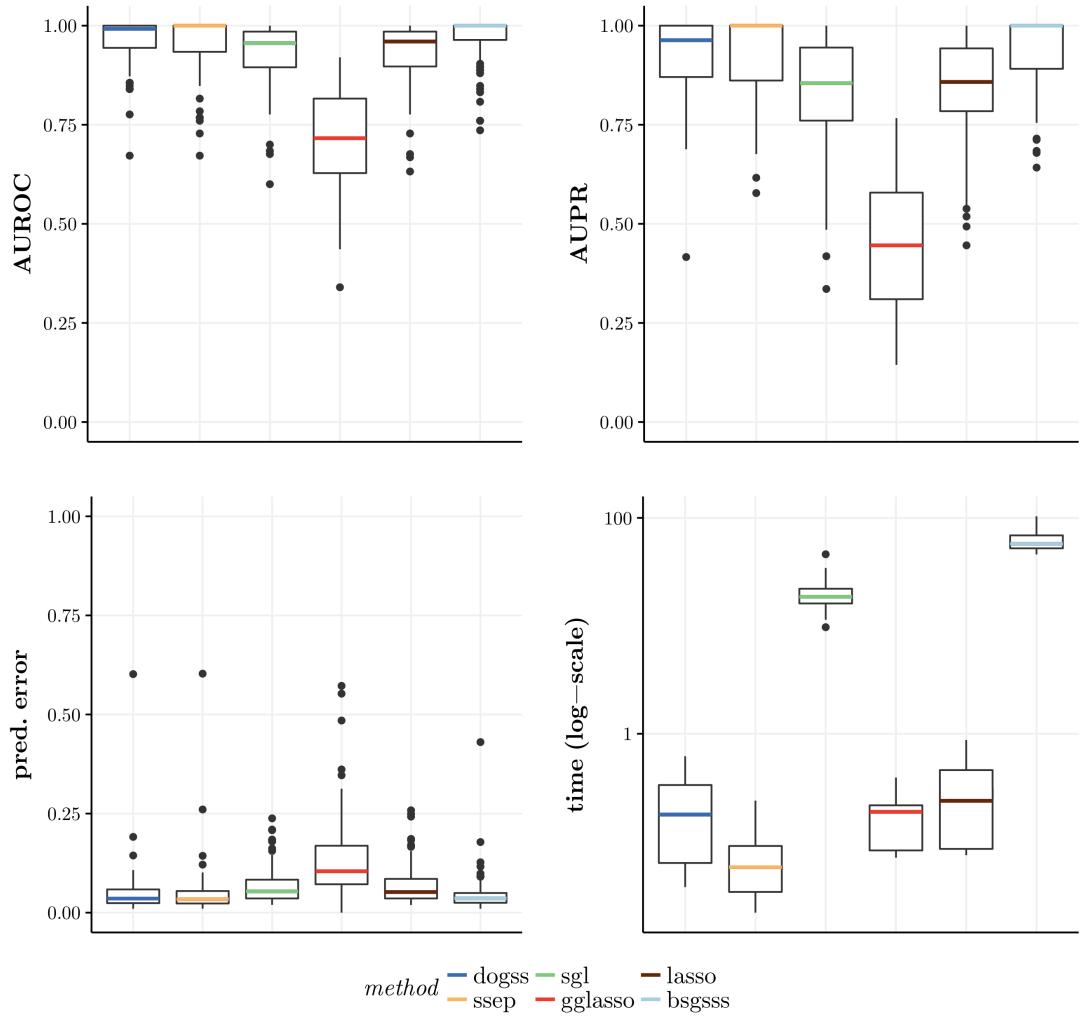

Figure 5.3.: Boxplots of AUROC, AUPR, prediction error and computing time for scenario 1: results for $(M, N, G, k, \sigma_0) = (30, 30, 5, 5, 1)$ over 100 runs.

The cutoff values for the prediction for all methods except `bsgsss` are derived by 10-fold cross validation with the 1se-rule. The predicting coefficients for the `bsgsss` method are recovered as the median values of the coefficients from the MCMC simulations.

The simulation setting "medium" of $(M, N, G, k, \sigma_0) = (30, 100, 20, 10, 1)$ describes a scenario where we have more features than observations. We can see some important differences between the methods in regards of the AUROC, AUPR and prediction error measures: The two Bayesian approaches with two levels of group sparsity (`dogss` and `bsgsss`) perform the best and nearly identical. The lasso methods perform worse on the AUROC and AUPR measures than the Bayesian approaches, but the sparse-group lasso actually has a low prediction error (while not as low as the prediction errors of `dogss` and `bsgsss`). Again, the two expectation propagation based methods (`dogss` and `ssep`) as well the group lasso and standard lasso have very low run times, while the sparse-group lasso `sgl` and the Gibbs sampling based approach `bsgsss` take about three orders of magnitude longer. There is also a wide range of the prediction errors with outliers on the prediction error boxplots for all methods.

The simulation setting "large" of $(M, N, G, k, \sigma_0) = (100, 1000, 100, 10, 1)$ describes a scenario where we have more features than observations and the number of features is actually quite high, while the signal is very sparse ($N = 1000$, $k = 10$). The trends from scenario "medium" regarding the AUROC and AUPR are carried forward: Our approach `dogss` and the Gibbs sampling `bsgsss` perform best, with the standard spike-and-slab `ssep` in second place, followed by sparse-group lasso and standard lasso close up, whereas the group lasso performs really bad. AUROC measures are very high for all methods, but differences are more pronounced on the AUPR measure. The better results of our method `dogss` come at the price of increased run time compared to standard lasso (approximately two orders of magnitude), but it is still faster than the Gibbs sampling approach or the sparse-group lasso, which are in turn about two orders of magnitude slower than the expectation propagation based methods. The sparse-group method `dogss` performs slightly better than `ssep` (which does not take grouping information into account), but the effect is not as big as in the "medium" sized scenario.

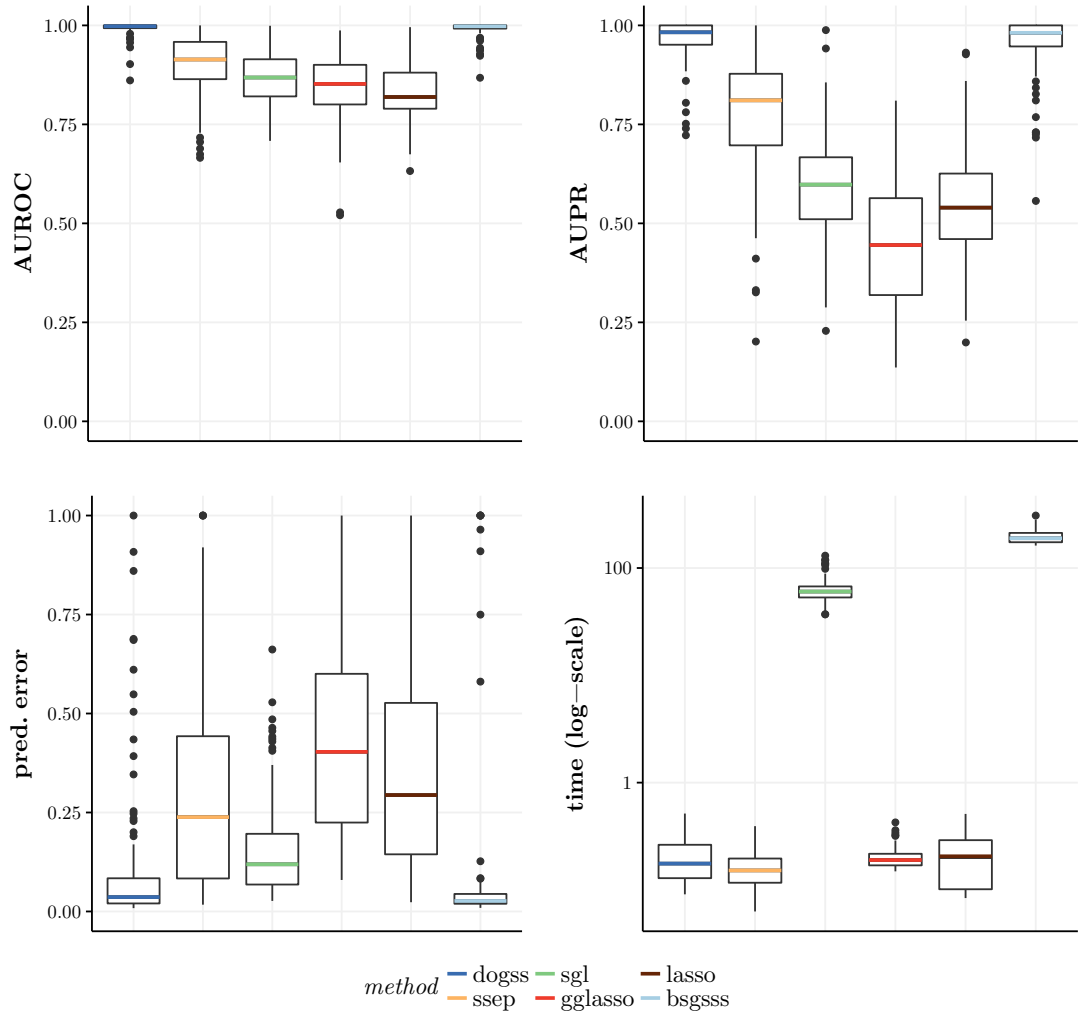

Figure 5.4.: Boxplots of AUROC, AUPR, prediction error and computing time for scenario 2: results for $(M, N, G, k, \sigma_0) = (30, 100, 20, 10, 1)$ over 100 runs.

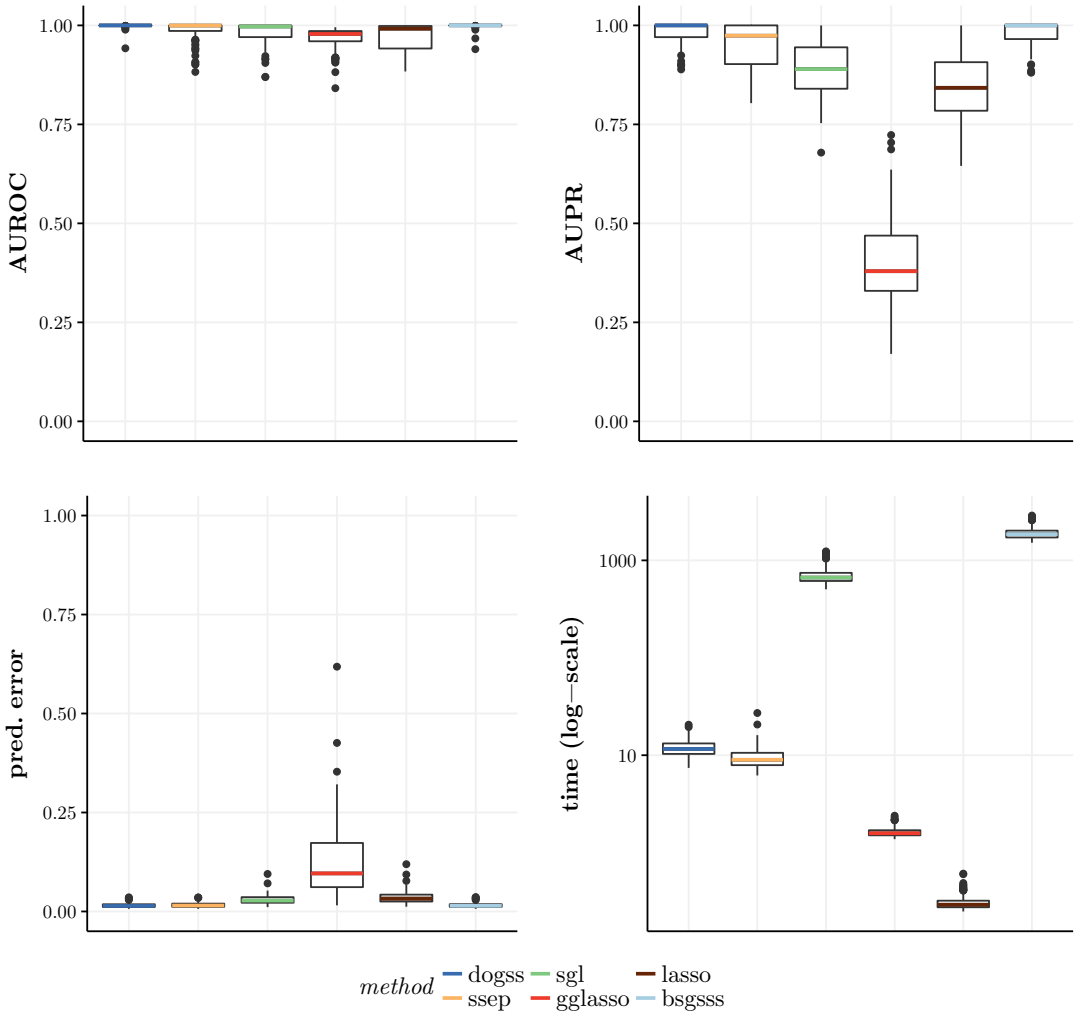

Figure 5.5.: Boxplots of AUROC, AUPR, prediction error and computing time for scenario 3: results for $(M, N, G, k, \sigma_0) = (100, 1000, 100, 10, 1)$ over 100 runs.

Influence of noise

Second we evaluate the influence of σ_0 on a fixed set of parameters. To this end, we simulated 100 data sets like in the previous section with $(M, N, G, k) = (30, 100, 20, 10)$ (the "medium" sized scenario), added noise with $\sigma_0 \in \{0; 0.1; 1; 3; 5\}$ and compared the performance of the six methods. The input parameter for σ_0 for our proposed method and the standard spike-and-slab was set to 1 in all cases. For every data set, we measured AUROC, AUPR, prediction error and run time like before, we calculated the median values and show these in Figure 5.6.

We see that the correct specification of the noise parameter is important for our proposed algorithm and the standard spike-and-slab, too. If the provided noise parameter is higher than or equal to the actual one, our proposed algorithm gives good results. If the actual noise level is too high, the expectation propagation based methods suffer considerably (which can be seen most clearly on the AUPR measure). The lasso methods do not depend on the specification of the noise parameter, but their performance deteriorates for increasing noise levels, too, but not as steep as the Bayesian approaches. The grouped spike-and-slab implementation (`bsgsss`) with Gibbs sampling does not depend on the specification of the noise parameter either (since it samples this parameter from the data), but our simulations give a surprising result: In regards of the AUROC/AUPR measures, the Gibbs sampling performs equally to our proposed method, while the prediction error is actually worse for the `bsgsss` method for higher noise levels. The run time increases slightly for the expectation propagation based methods (`dogss` and `ssep`) with increasing levels of noise.

Influence of correlated features

Third we assess the influence of the correlation structure between features within the data matrix X. The first structure we have already seen above, with columns of the data matrix X drawn independently. The second structure "pairwise" refers to an overall pairwise correlation between features of 0.5. The third structure "groupwise" is correlation on the group level: Features within a group have a pairwise correlation of 0.5, but every two features from different groups are independent. The parameters $(M, N, G, k, \sigma_0) = (30, 100, 20, 10, 1)$ are chosen like in scenario

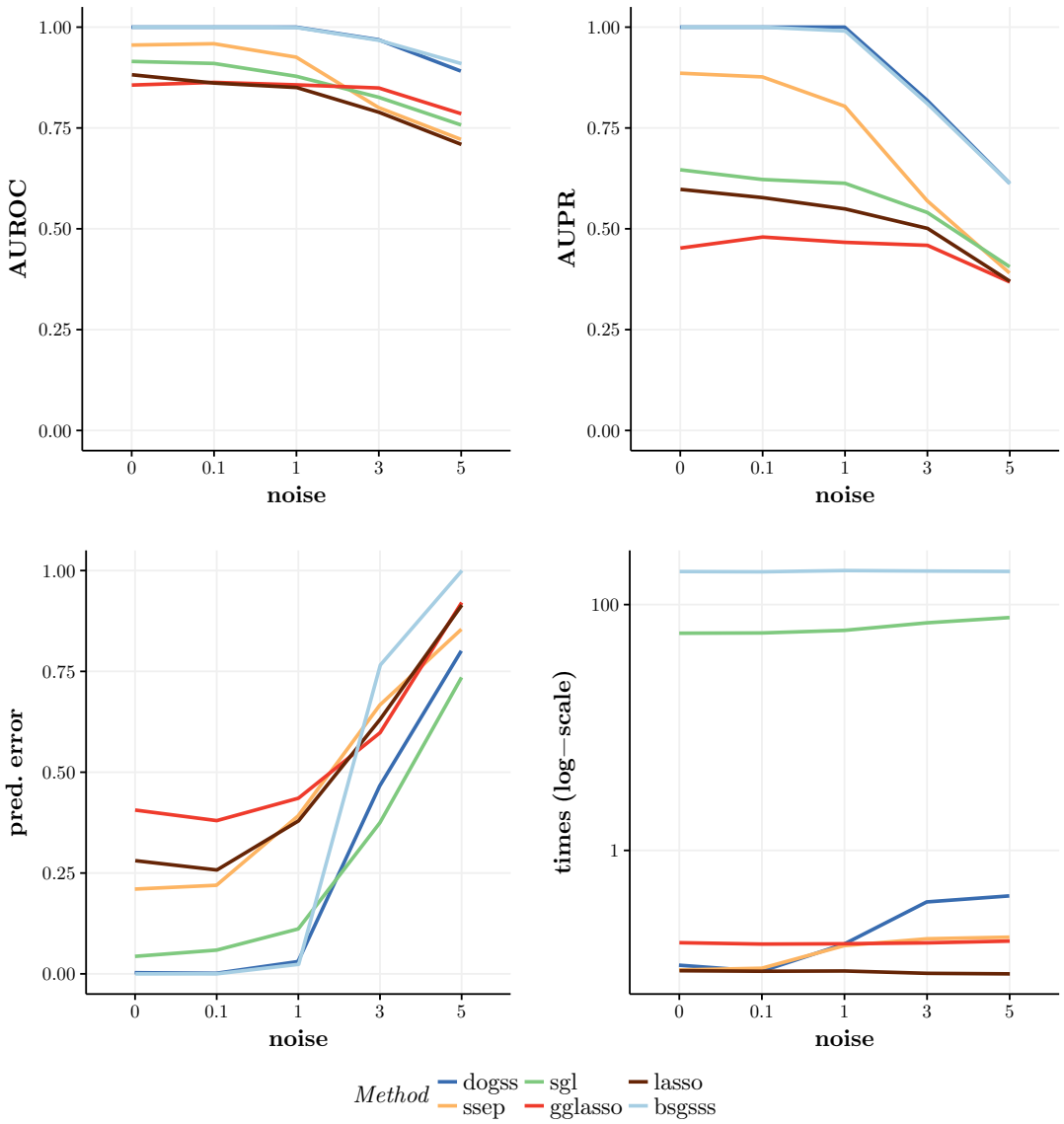

Figure 5.6.: AUROC, AUPR, prediction error and computing time for different levels of σ_0: median values over 100 runs for different methods with $(M, N, G, k) = (30, 100, 20, 10)$.

"medium". The results of these simulations are aggregated in Figure 5.7.

We cannot see many important differences between the performance in regard to the correlation structure. We see a small drop in AUROC/AUPR performance for all methods when features are pairwise correlated, while performance is roughly the same for independent or groupwise correlated features. Run times are different between correlation structures for the `dogss`, `sgl` and standard `lasso`.

Influence of slab parameter

Last, we evaluate the influence of σ_{slab} on a fixed set of (M, N, G, k). Like the penalty parameter λ for the lasso methods, the slab parameter for the spike-and-slab methods is a crucial value that needs to be chosen beforehand. But unlike the λ parameter, which models the sparsity of the model directly and thus can and should be determined by cross-validation, the slab parameter rather puts a value to the expected size of the non-zero coefficients.

We simulated 100 data sets like in the previous sections with $(M, N, G, k, \sigma_0) = (30, 100, 20, 10, 1)$, that is the "medium" scenario, and compared the performance of our proposed approach and the standard spike-and-slab procedure. The input parameter for σ_{slab} for our proposed method and the standard spike-and-slab was chosen from $\{0.1; 1; 2; 5; 10; 100\}$. For every data set, we measured AUROC, AUPR, prediction error and run time like before, we calculated the median values and show these in Figure 5.8.

The results for the AUROC and AUPR values are quite stable for our proposed method, with better results for higher values of the slab parameter. The prediction error is lowest for the values 2, 5 and 10. The run time increases a great deal for the value of 100. These results indicate that the slab parameter should be chosen reasonably in the range of the absolute size of the anticipated coefficients (β, here: the coefficients were drawn uniform-randomly from $[-5; 5]$). We note that in this simulation scenario the standard spike-and-slab benefits from a rather high slab parameter (at the cost of increased run time).

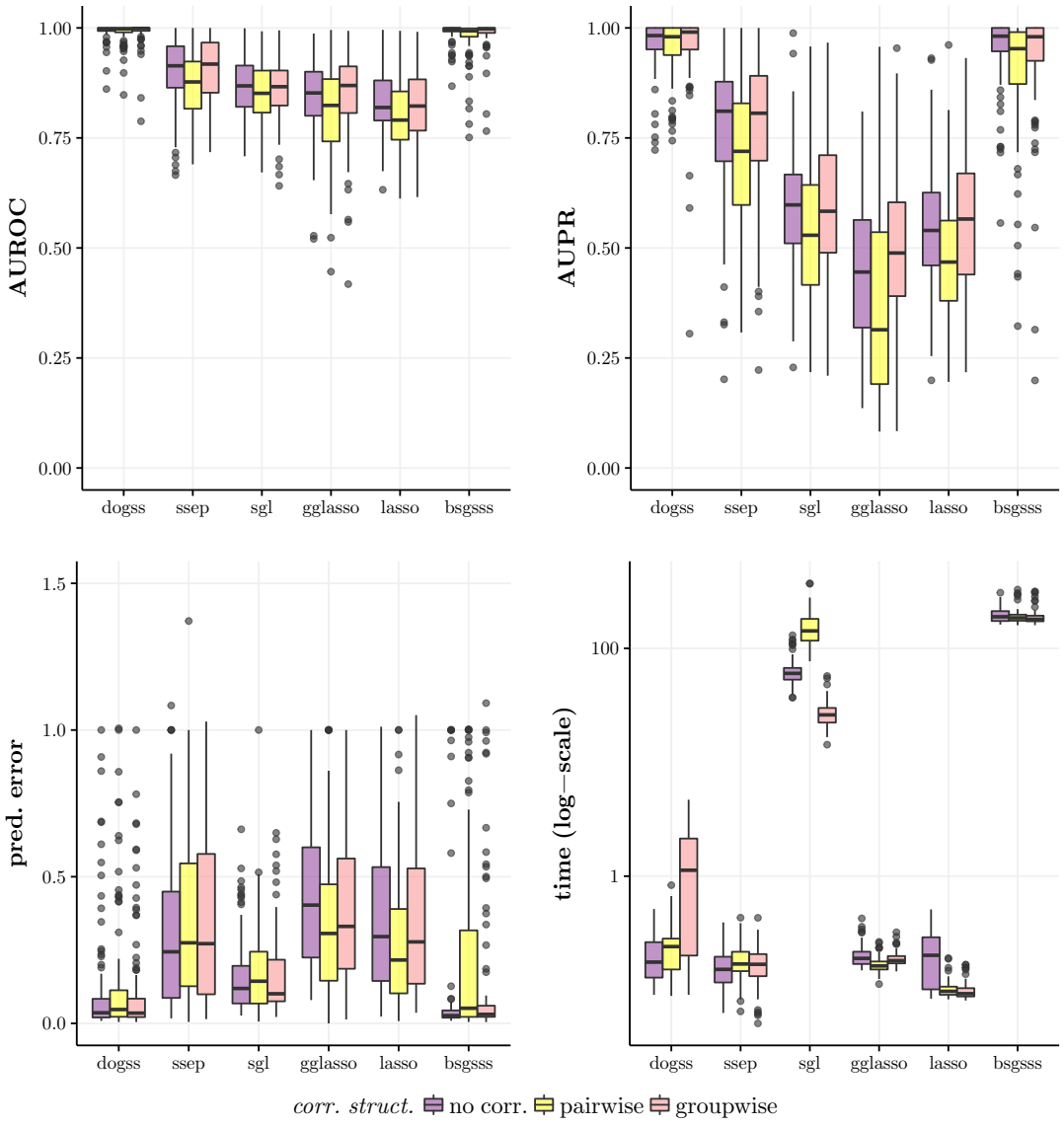

Figure 5.7.: Boxplots of AUROC, AUPR, prediction error and computing time for different correlation structures (uncorrelated, pairwise correlated or groupwise correlated features): Results from 100 runs for different methods with $(M, N, G, k, \sigma_0) = (30, 100, 20, 10, 1)$.

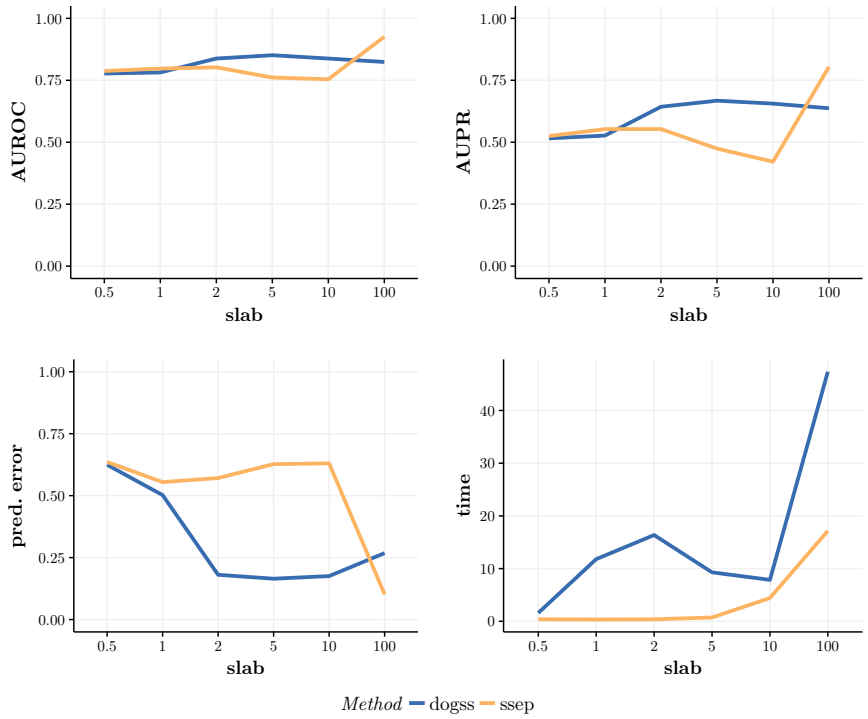

Figure 5.8.: Line plots of average AUROC, AUPR, prediction error and computing time for different σ_{slab} parameters: Results aggregated (median) over 100 runs for different methods with $(M, N, G, k, \sigma_0) = (30, 100, 20, 10, 1)$.

5.1.2. Simulations: Network reconstruction

To study the ability of our algorithm to reconstruct gene regulatory networks, we generate random network graphs with a known structure as Gaussian graphical models (see Section 3.2), which we compare as a gold standard to the results of our algorithms. A network graph where edges are just drawn discreetly-uniform and independent from the set of all possible edges is not a good representation of a biological network (given the number of nodes and the number of edges, this is the Erdős–Rényi model [23]). A better model for a gene regulatory network is a scale-free network, where some genes (transcription factors or hubs) are connected to many other genes, while most genes are connected to only very few other genes. That is, the out-degree of a node in this network follows a power law $\mathbb{P}(k) = k^{-\gamma}$, where k is the number of edges going out of a node and γ is some positive constant. In our simulations we also allow for a hub to consist of multiple genes, which is a good representation of the behavior of biologically similar transcription factors. Gene regulatory networks (and many other networks arising in different contexts) show evidence for this scale-free topology structure (see for example [15] and [4]).

We simulate our own (approximately scale-free) network graphs based on four parameters (the number of nodes P, the number of groups G, the number of hub nodes H and a random (Erdős–Rényi-like) edge probability q) from the following procedure:

1. we draw G values $(p_1^{gs}, \ldots, p_G^{gs})$ from $\mathcal{U}(0,1)$, these will represent the different group sizes (normalized such that $1 = \sum_g p_g^{gs}$),
2. we assign H hub nodes discreetly-uniform to all groups $\{1, \ldots, G\}$,
3. for every node n:
   a) If it is not a hub node itself: sample a hub node from a group drawn from a categorical distribution $\mathrm{Cat}(p_1^{gs}, \ldots, p_G^{gs})$ on $\{1, \ldots, G\}$ and draw an edge between node n and the hub node.
   b) If there are other hub nodes in this group, we draw additional edges to these hub nodes with probability 0.5 each.

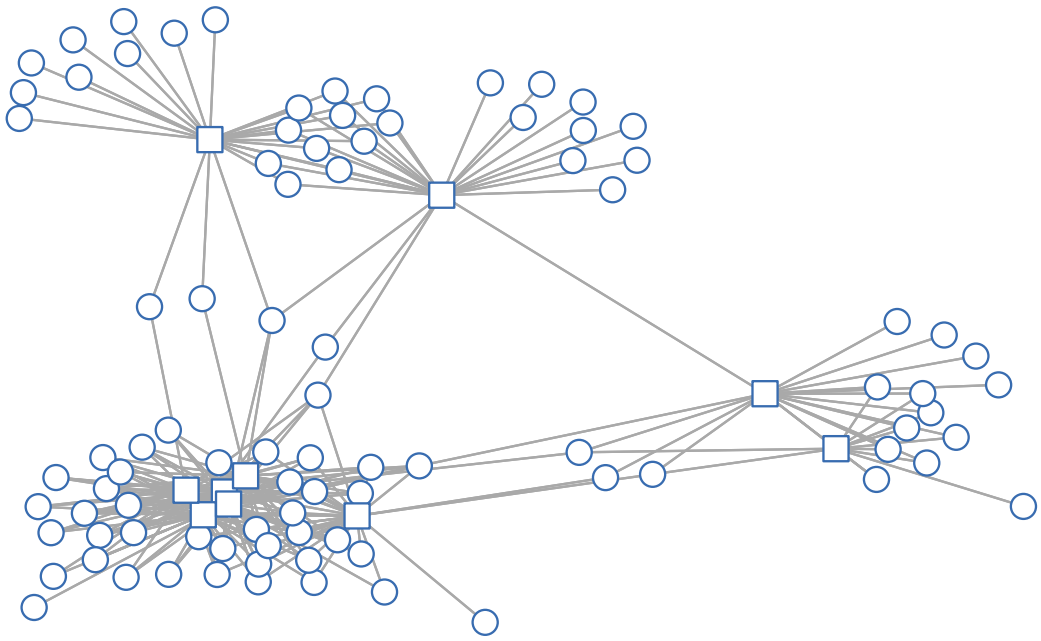

Figure 5.9.: An exemplary network generated from parameters $(P, G, H, q) = (100, 3, 10, 0.01)$. Squares correspond to hub nodes/transcription factors.

c) Finally we add random edges to any other hub node in the whole graph with (low) probability q.

The larger q, the more similar to a random Erdős–Rényi model the network gets, while with $q = 0$ groups of nodes around their respective hubs are perfectly separated from each other.

An example network graph for a randomly generated network with parameters $(P, G, H, q) = (100, 3, 10, 0.01)$ can be seen in Figure 5.9.

We ran simulations on two different network scenarios which differ in size, the parameters can be seen in table 5.3. For each scenario, we generated 100 random network graphs according to the procedure above. For each graph, we generated 100 random observations as a training set and an additional 100 observations as a test set with the `qpgraph` package from Castelo and Roverato [14] in R, which also provides a precision matrix corresponding to the graph and observations. For each graph, we applied four different setups for the network reconstruction regarding

	P	G	H	q
small	100	3	10	0.01
large	1000	20	100	0.001

Table 5.3.: Choice of parameters for the different simulation settings in network reconstruction.

the size of X and the mapping $\mathcal{G}$:

1. only hub nodes (transcription factors) are considered as features, with original grouping provided as in the graph generation,

2. all nodes (genes) are considered as features, with original grouping of the transcription factors as in the graph generation and additional groups for the non-hub nodes,

3. all nodes considered as features with a completely random grouping of the features.

We compare the same methods like in the preceding Section 5.1.1 with the exception of `bsgsss`: The Gibbs sampling approach takes too much computing time to be implemented in a feasible way for network reconstruction. We could include the `sgl` method by allowing its implementation to use its default values (threshold for convergence of 10^{-3}, maximum number of iterations 1000 and a λ sequence of just 20 values). The `gglasso` and `lasso` method use the same threshold and maximum number of iterations, but with the default length 100 of the λ sequence. Our implementations `dogss` and `ssep` use the same threshold for convergence of 10^{-3}, too, and a maximum of 100 iterations.

Figure 5.10 shows the aggregated results for the small networks. Regarding the AUROC and AUPR measures, we see that the group lasso is clearly outperformed by all other methods. Furthermore, the differences between the methods without grouping information (`ssep` and `lasso`) are marginal, but also the `sgl` method which explicitly models sparsity on the between- and within- group level does not show any advantage over the `ssep` and the regular `lasso`. Our new method `dogss`

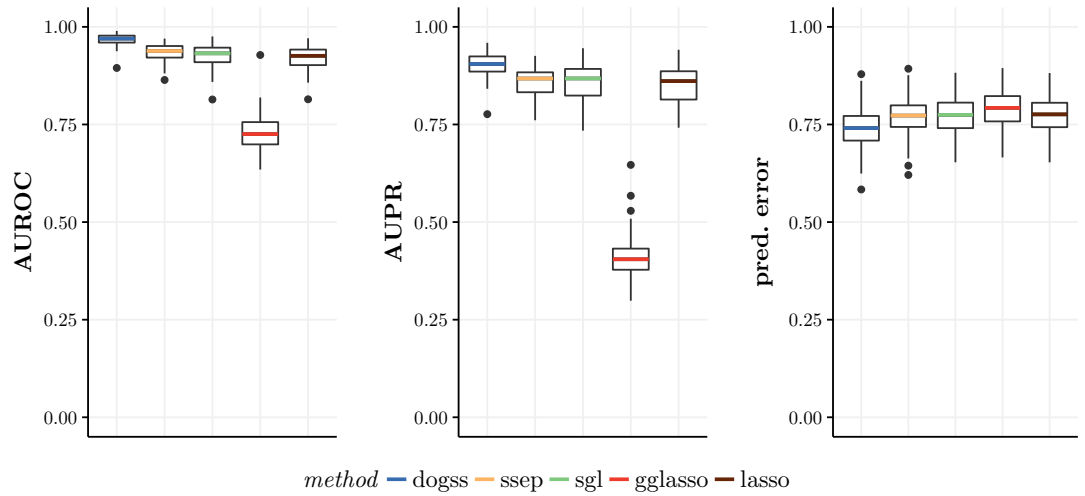

Figure 5.10.: Boxplots of AUROC/AUPR/prediction error for different methods running on 100 simulated small networks, only hub features considered.

performs best by a narrow margin. The same holds true for the prediction error measure on the held-out data.

The results for the large networks are more diverse, see Figure 5.11. Again, the `gglasso` method suffers greatly on the AUROC and AUPR measures compared to all other methods, but it does not fare too bad on the prediction error measure. The comparison of the methods without grouping information ends in a tie: While `ssep` does better on the AUROC measure, `lasso` performs better with prediction error (and both have approximately equal AUPR). The sparse-group lasso `sgl` does a bit better than the standard `lasso`, but is clearly outperformed by our method `dogss` on all measures.

In the next step we compare the performance of the methods when we include all features for the neighborhood selection approach, and as such increase the sparsity. Additionally we compare the performance when features are grouped randomly opposed to the original grouping.

Let us first consider the small network reconstruction problem, see Figure 5.12. With the original grouping, our method `dogss` outperforms all other methods on all three measures of AUROC, AUPR and prediction error. The sparse-group lasso

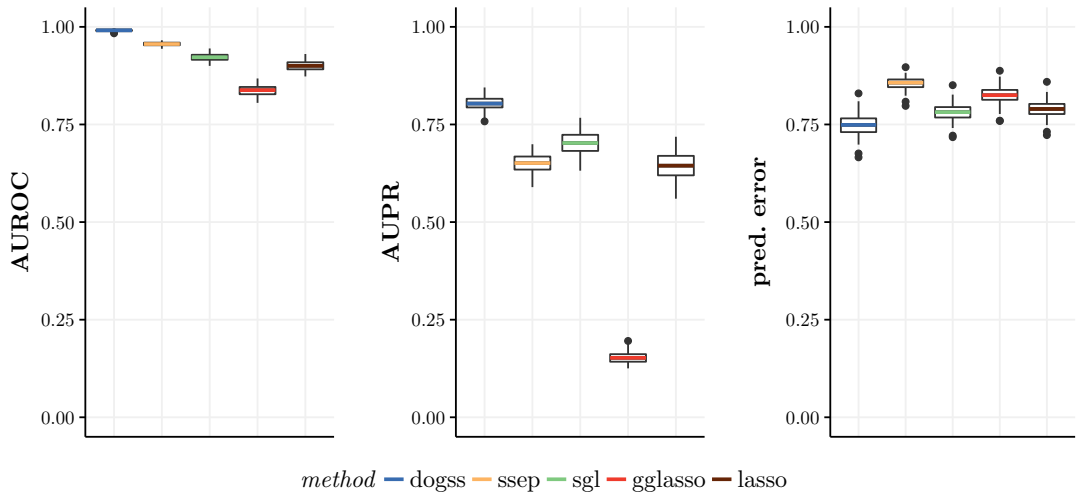

Figure 5.11.: Boxplots of AUROC/AUPR/prediction error for different methods running on 100 simulated large networks, only hub features considered.

`sgl` outperforms the standard `lasso`, but the Bayesian method without grouping information `ssep` appears as a reasonable choice, too. Results for our method `dogss` deteriorate greatly for the random grouping, while the `ssep` method becomes the better choice with performance still better than `sgl`, which is in turn on par with the standard `lasso`.

We see the same results from the small networks for the results on the large networks (Figure 5.13), but more pronounced. Our methods `dogss` performs best, but only if the grouping information is actually helpful. Otherwise, the Bayesian approach without grouping information `ssep` should be preferred.

5.2. Results on biological data

Here we assess the algorithm's capability to reconstruct gene regulatory networks from real experimental data using prior information about the grouping of the variables. The DREAM5 gene network inference challenge [58] provides extensive data along with a gold standard for the underlying networks. In the Escherichia coli challenge, 805 microarray experiments were given with measurements of gene

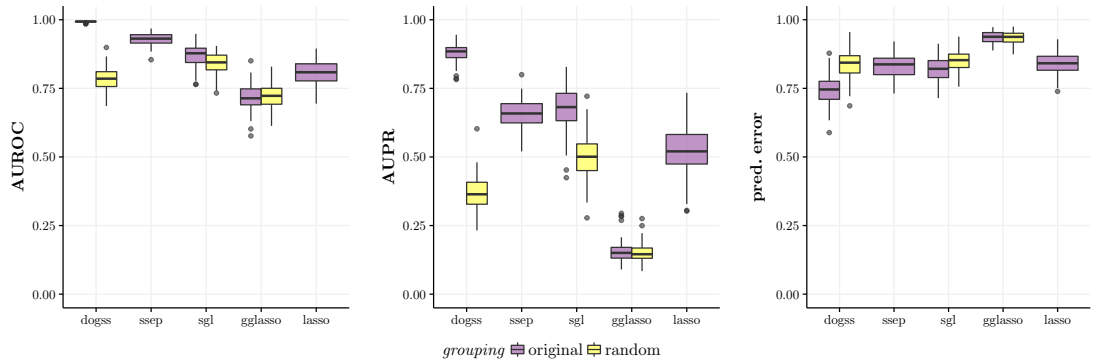

Figure 5.12.: Boxplots of AUROC/AUPR/prediction error for different methods running on 100 simulated networks (small), all features considered, with original grouping or random grouping.

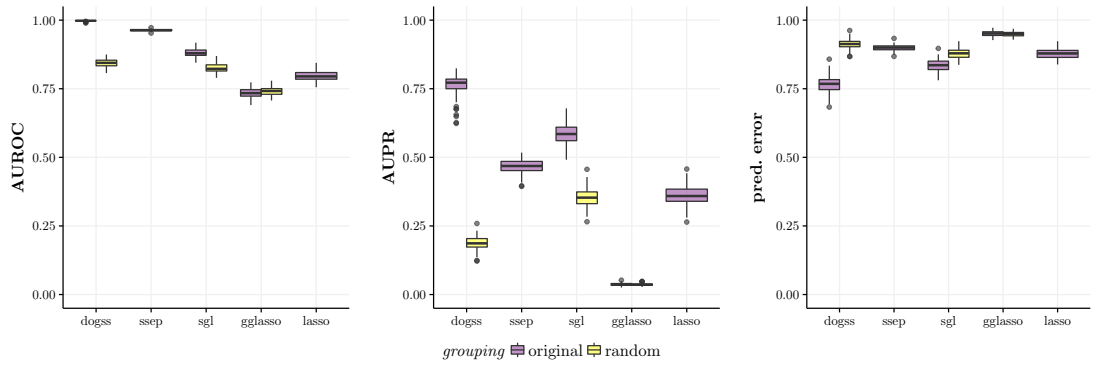

Figure 5.13.: Boxplots of AUROC/AUPR/prediction error for different methods running on 100 simulated networks (large), all features considered, with original grouping or random grouping.

expression for 4511 genes, of these 334 are transcription factors and as such candidate regulators for the genes. The gold standard network which was revealed after the challenge comprises of 2055 interactions within the network. To test the ability of our new algorithm (and the competing lasso approaches), we first derived a meaningful grouping of the transcription factors - this was not given to the participants of the challenge and as such our results are not directly comparable to the official results of the challenge.

We grouped the transcription factors by clustering them such that transcription factors belong to the same group if they predominantly bind to the same genes (thus, we identified the hubs), we will refer to this grouping as "co-binding". For comparison we also tested two additional groupings which do not contain any prior information and are as such purely data driven: Grouping of features by simple kmeans clustering of gene expression and a random grouping of features. Finally we included a grouping derived from the supplementary files of Marbach et al. [58], where a GO term analysis [86] gave groups of functional modules to all genes which we filtered for the transcriptions factors.

The total number of observations (805) is very large and thus we decided to run the different methods on a (training) subset of randomly chosen 300 observations. This also leaves 505 observations as a test set for evaluation of the prediction error. We repeated the sampling of the training/test set 10 times and averaged the results.

On each training set, we employed the neighborhood selection approach described in Section 3.2, that is, we ran every method for a total of 4511 times (each gene once as a dependent response). We assess the quality of the network reconstruction based on the gold standard by deriving the true positive rate, false positive rate and precision along the ranked list of edges. The resulting lists were averaged over the ten runs and we plot the resulting average receiver operating characteristic (ROC) curve as well as the average precision-recall curve for the different algorithms (Figure 5.14). We also calculate the respective (average) area under the curve scores (Table 5.4).

The increase in the AUROC measure for our approach is immense, compared to alternative methods. That is, the prior information about the grouping of features

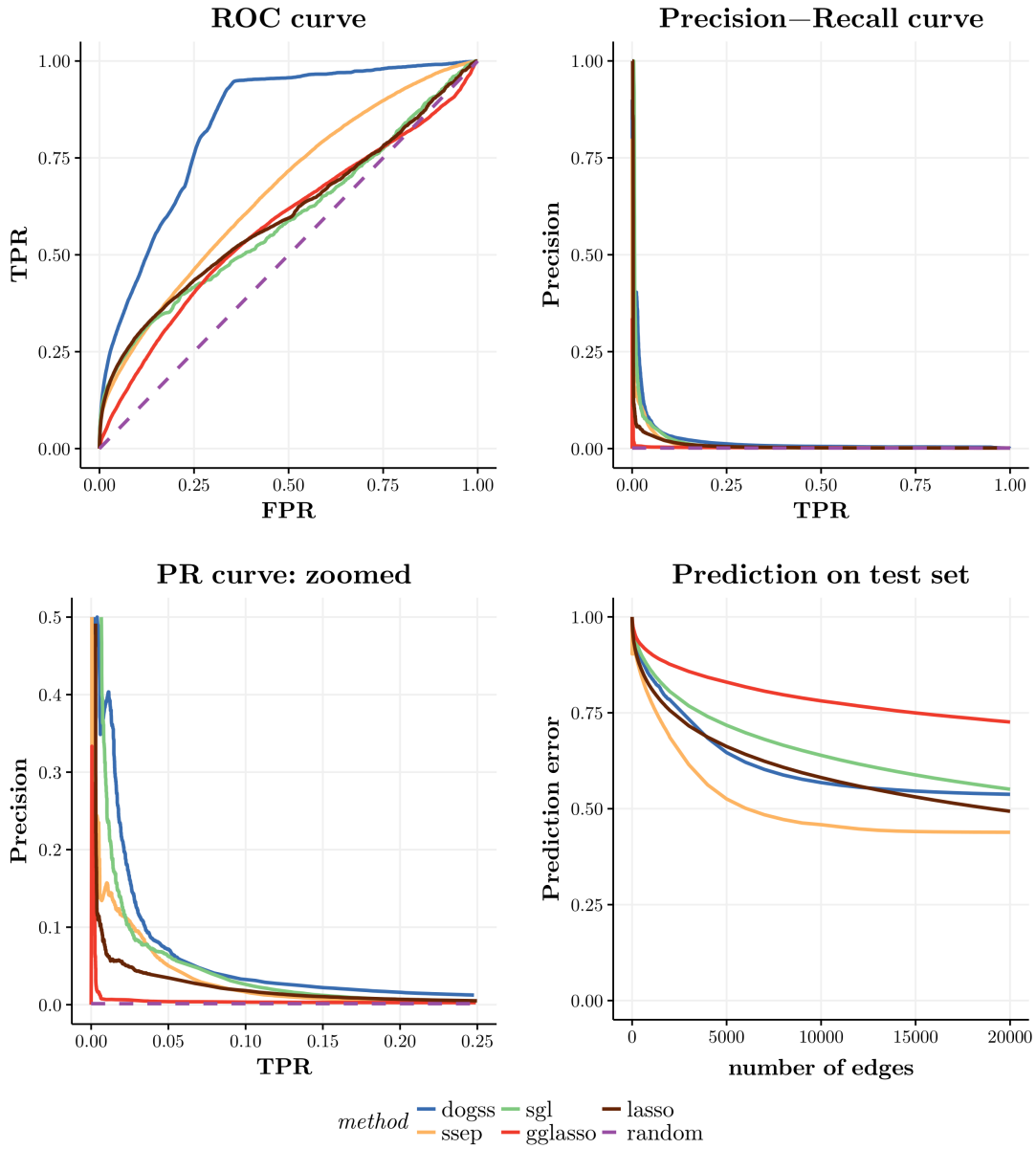

Figure 5.14.: ROC and Prec-Recall curves, prediction error on E. coli DREAM5 data. The prediction error is given along the 20 000 top ranked edges for every method.

method	dogss	ssep	sgl	gglasso	lasso	random
AUROC	0.84	0.67	0.59	0.58	0.6	0.5
AUPR	0.02	0.01	0.016	0.003	0.009	0.0014

Table 5.4.: AUROC and AUPR values for ROC and PR curves of the DREAM5 E. coli network reconstruction, see Figure 5.14.

is very useful to identify more true positive edges without choosing too many false positive interactions. The second best method is the standard spike-and-slab, which does not use any grouping information. The results from the standard lasso and sparse-group lasso are barely distinguishable, indicating that the sparse-group lasso cannot take full advantage of the available grouping information. The group lasso is not a good candidate algorithm for this network reconstruction problem, its performance on the AUPR measure is quite poor.

The AUPR measure for all methods is far from close to the theoretical optimal value of 1 (which is a common observation for "real" biological network reconstruction problems, see [58]). All methods yield very low AUPR values, indicating that the number of correctly identified interactions is in a bad proportion to the total number of called interactions. Our approach and the sparse-group lasso yield the highest AUPR values, and the standard spike-and-slab performs a little bit better than the standard lasso.

Furthermore we tested all algorithms on their predictive performance on this data set. We retrieved the regression coefficients from the neighborhood selection, chosen by cross-validation. For every method, we have a ranking of all possible edges along with a regression coefficient with every edge. This way we derive a prediction error curve along the ranked list: For rank r, we predict the expression of all 4177 genes that are not transcription factors by using the expression levels of the transcription factors in the held-out data. We sorted along the top r edges respectively regression coefficients and set all other coefficients to 0. We measured the predictive performance via the relative sum of squared errors along the ranks

group	1	2	3	4	5	6	7	8	9
#TFs	193	39	7	6	11	4	7	10	22
group	10	11	12	13	14	15	16	17	18
#TFs	4	4	4	4	5	5	3	3	3

Table 5.5.: Original grouping (co-binding) of DREAM5 E. coli transcription factors.

r, that is

$$E(r) = \frac{\sum_{m=1}^{505} \sum_{n=1}^{4177} (y_{mn} - x_m \cdot \beta_n(r))^2}{\sum_{m=1}^{505} \sum_{n=1}^{4177} y_{mn}^2}.$$

Figure 5.14 shows the results for the different algorithms for the top 20 000 edges, averaged over the ten runs: We see that the standard spike-and-slab method makes the best prediction on the test data in th relevant realm of the top ranked edges (the number of true interactions is 2055). Our sparse-group Bayesian approach and the standard lasso perform roughly the same prediction-wise, while the sparse-group lasso eventually does better predictions than our approach, but only after approximately 20 000 included edges. The group lasso is not useful for prediction in this setting.

Furthermore we compared our approach with the co-binding-based grouping of the transcription factors to a random grouping, a k-means based grouping, and a functional grouping [58]. The grouping dubbed as co-binding was derived by grouping transcription factors that bind most often to the same genes, this led to 18 groups with very different numbers of transcription factors (see Table 5.5 for the co-binding grouping). Now we assigned a random grouping to the transcription factors where group sizes are the same like in the co-binding grouping, that is we permuted the group memberships. Finally we grouped transcription factors by their gene expression similarity via k-means with $k = 18$. We ran our algorithm again with these three new groupings and compare the results to the original grouping via the ROC and Precision-Recall curves, see Figure 5.15.

We observe that the co-binding grouping performs best in terms of ROC and Precision-Recall measure. Our sparse-group Bayesian method, the sparse-group

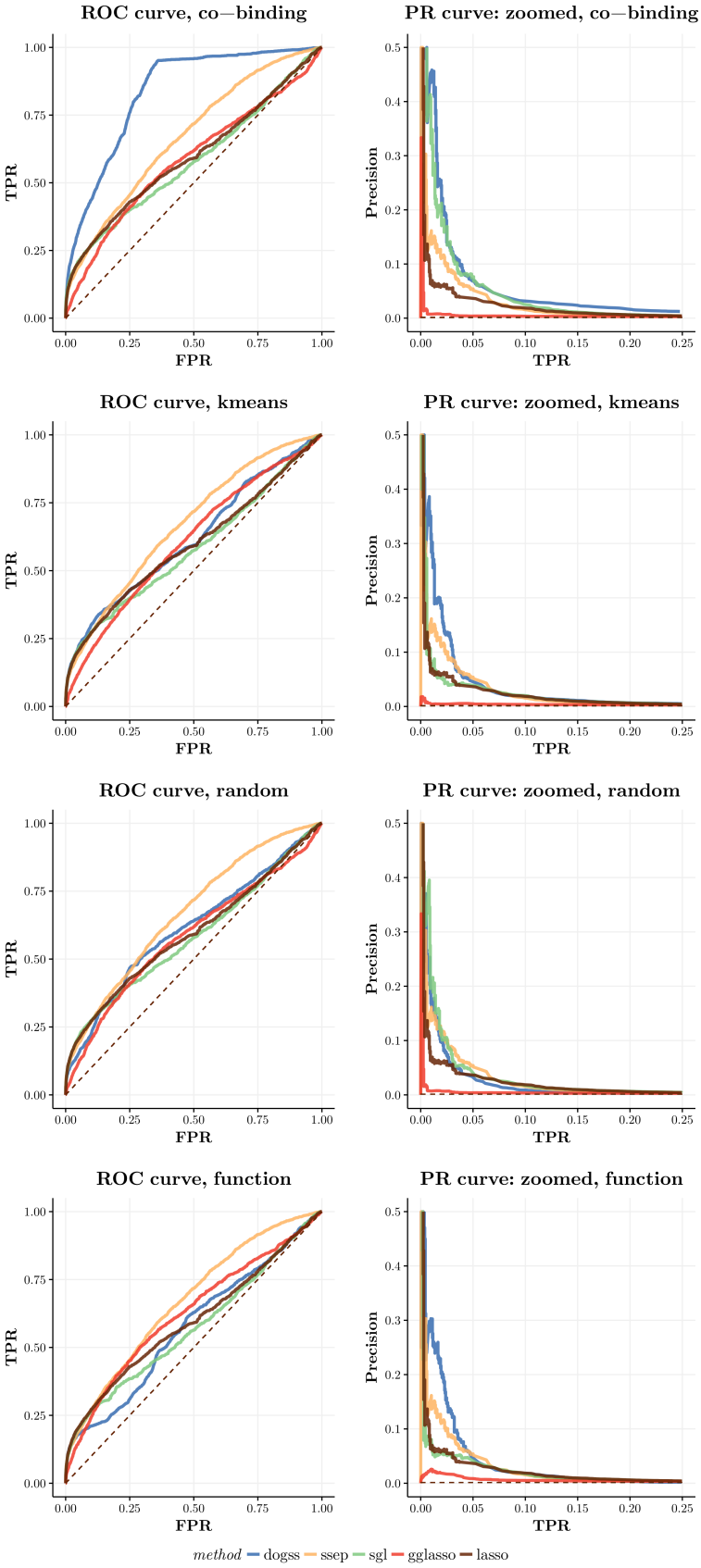

Figure 5.15.: ROC and Prec-Recall curves on E. coli DREAM5 data for different groupings: co-binding, clustering by kmeans, random, function.

lasso and the group lasso all perform worse than the standard spike-and-slab (with no grouping information) in regard of ROC curves for the kmeans, random and functional grouping. For the kmeans and functional grouping, our sparse-group Bayesian method performs best on the AUPR measure, while the sparse-group lasso performs worse than the standard spike-and-slab and approximately the same like the standard lasso. For the random grouping, sparse-group lasso, standard spike-and-slab and our method perform about the same.

5.3. Summary

We compared our new method to alternative methods on simulated and experimental data. For the task of signal recovery, the Bayesian methods perform better than lasso methods and the sparse-group Bayesian approach performs best in the presence of grouping information. The speed-up in computing time of the expectation propagation-based methods compared to the alternative Gibbs sampling is enormous, while the standard lasso remains the fastest method available. For the task of network reconstruction, we first presented a simulation procedure to generate scale-free networks that are structurally similar to gene regulatory networks. Our new sparse-group Bayesian feature selection method performs best unless for the randomized grouping information. On experimental data with grouping information from transcription factor co-binding, our new method leads to a huge increase in the AUROC measure and the highest AUPR value, too. The standard spike-and-slab has the lowest prediction error and performs reasonably well.

6. Bayesian (Sparse-Group) Feature Selection for Time Series Data

This chapter applies the Bayesian framework with and without grouping to a time series setting, where the grouping of the variables stems from a Granger model. We apply the method for the reconstruction of gene regulatory networks from gene expression time series data.

6.1. The Granger model and the $\mathrm{VAR}(p)$ process

The Granger approach to model a linear regression scheme for time series is based on the notion of Granger causality [33]: Causality between two variables X and Y can be tested by measuring the prediction ability of X for **future** values of Y using **past** values of X. E.g., if a spike in X at time t precedes a spike in Y at time $t+p$, we say that X Granger-causes Y (with delay p). Of course, it is not true in general that two events are causally linked just because they happen one after the other. But Granger causality or "predictive causality" is still a good measure of a possible causal association.

The original definition of Granger causality for two time series of variables X and Y tests whether the future values of Y can be predicted significantly better with the inclusion of knowledge about past values of X or if the prediction is about the same if X is not included. Put into mathematical terms for some maximum delay p:

$$\begin{aligned} & y_t = a_0 + a_1 y_{t-1} + \ldots + a_p y_{t-p} + b_1 x_{t-1} + \ldots + b_p x_{t-p} + \varepsilon_t \\ \text{vs. } & y_t = a'_0 + a'_1 y_{t-1} + \ldots + a'_p y_{t-p} + \varepsilon'_t \end{aligned}$$

The test of significance is an F-test on the respective residuals ε_t and ε'_t over the observations for every t.

Here we use a similar approach without hypothesis testing, that is, inference with the VAR(p) model (vector autoregressive model of order p, see [57, p. 13] and Appendix C). Suppose we have random variables $Y = (Y_1, , \ldots, Y_G)$ and time series observations of these variables $y_t = (y_{1,t}, y_{2,t}, \ldots, y_{G,t})$ for $t = 1, \ldots, T$, which are generated by the following scheme:

$$y_t = \nu + A_1 y_{t-1} + A_2 y_{t-2} + \ldots + A_p y_{t-p} + \varepsilon_t \tag{6.1}$$

for some maximum delay $p \in \mathbb{N}\backslash\{0\}$, ν the intercept vector and ε_t a zero-mean error term with finite variance where the sequence of ε_t are independently identically distributed random G-vectors. Not every choice of matrices A_i, $i = 1, \ldots, p$, leads to a stable multivariate time series (that is, the values of some of the variables might be diverging with time, see also the Appendix C).

The matrix/vector notation of equation (6.1) already shows a certain grouping of the variables by the delays 1 to p. Re-writing the equation for a single entry $y_{h,t}$ of y_t and rearranging the summands according to variables rather than delays gives us the following equation:

$$y_{h,t} = \nu + \sum_{g=1}^{G} (a_{1,gh} y_{g,t-1} + a_{2,gh} y_{g,t-2} + \ldots + a_{p,gh} y_{g,t-p}) + \varepsilon_{h,t}, \tag{6.2}$$

where the coefficients $a_{i,gh}$ are the entries of the matrices $A_i = (a_{i,gh})_{g,h=1}^{G}$ from equation (6.1). Equations (6.1) and (6.2) are equivalent, but equation (6.2) sorts the variables first by gene g and then by delay.

The VAR(p) model implies certain assumptions and requirements. E.g., the observations need to be available with constant spacing in time (this can be sidestepped by interpolating between unequally spaced observations). Furthermore, there should be no trend [57, part II] other than the influence from the included variables. Sometimes it is possible to clean the data of some trends before the analysis. The most restrictive assumption of the VAR(p) model is the linear assumption. This means that all effects within the data are linear, which might not be true for many biological time series (e.g., see [32]).

6.1.1. The Granger approach as a problem of sparse-group feature selection

There is an inherent grouping to the coefficients in equation (6.2): For G variables and a maximum delay of p, we have G groups each of size p, where each group consists of the coefficients corresponding to the different delays for a fixed variable. In many cases, the number of coefficients in equation (6.2) exceeds the number of available observations, since it grows quickly as the product of the number of variables times the maximum delay allowed in the equation. Furthermore, it is a reasonable assumption that some variables do not have influence on the response, thus setting the coefficients of this group of coefficients corresponding to this variable to zero (between-group sparsity). We can also assume that there are variable-specific delays to the effects, so eliminating for a given variable most of the delay-specific coefficients corresponds to within-group sparsity. Therefore, equation (6.2) is another example for a sparse feature selection problem with sparsity on two levels (between- and within-group), and we can apply the sparse-group feature selection model that was presented in Chapter 4.

6.2. Simulations: Time series data

We test the capability of our proposed algorithm to reconstruct networks from time series data, generated according to the VAR(p) model. To this end, we vary the following parameters: G (the number of variables/nodes/genes), p (the maximum delay), T (the length of the time series), k (the sparsity within the model/network) and σ_0 (the variance of the error). The simulation setting is as follows:

1. generate random values for ν iid from $\mathcal{U}(0, 5)$ and set start values $y_{-p+1}, \ldots, y_{-1}, y_0$ each equal to ν plus some error $\varepsilon_{-p+1}, \ldots, \varepsilon_{-1}, \varepsilon_0 \sim \mathcal{N}(0, \sigma_0)$,

2. generate random matrices $A_1, \ldots, A_p$ with all zero entries except for a total of randomly chosen k entries across these matrices, which are drawn from $\mathcal{U}(-2, 2)$,

3. to secure a stable VAR(p) process, we normalize the matrices A_1 to A_p according to Proposition 1 in Appendix C,

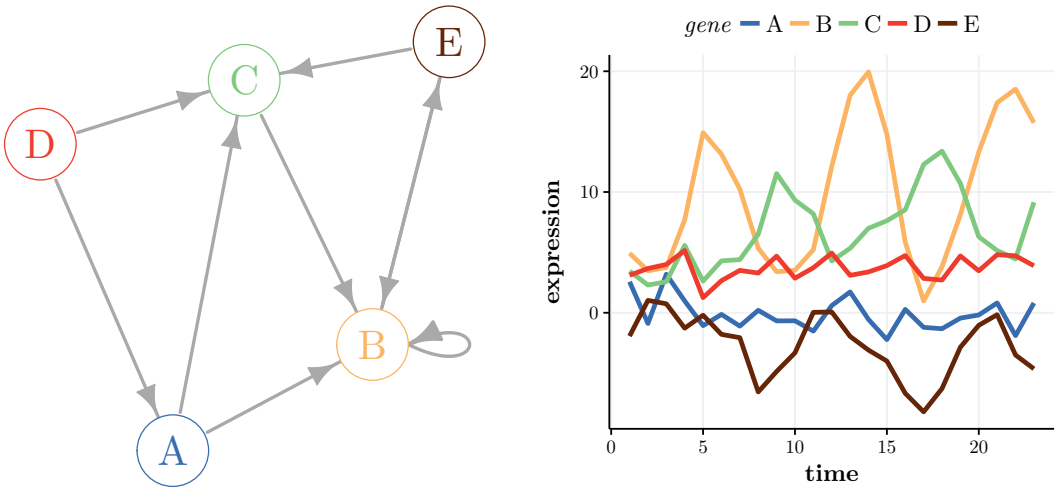

Figure 6.1.: Example of network and simulated time course data from the vector autoregressive model for parameters $(G, p, T, k, \sigma_0) = (5, 3, 20, 10, 1)$.

4. we generate values y_t sequentially for $t = 1$ to T with random error $\varepsilon_t \sim \mathcal{N}(0, \sigma_0)$ from equation (6.1) for training and additionally values y_t for $t = T + 1$ to $T + 10$ for testing the prediction.

For illustrative purposes, see an example network graph and the corresponding time series plot generated from this scheme with parameters $(G, p, T, k, \sigma_0) = (5, 3, 20, 7, 0.5)$ in Figure 6.1.

We test the capabilities of our and competing algorithms on 3 different network scenarios, which differ in the size and structure of the network. To this end, we simulate 100 random networks for each scenario, and compare the AUROC/AUPR measures based on the true (gold-standard) known network structure.

The results are shown in Figures 6.2, 6.3 and 6.4. In any scenario, the sparse-group lasso performs slightly worse than the standard lasso, and the group lasso even worse than the sparse-group lasso. The sparse-group Bayesian method and the standard spike-and-slab perform almost the same, the latter having a slight advantage over the former. Both Bayesian methods are superior to the lasso methods. We conclude that for the reconstruction of networks from time series data within the Granger model, one should choose a Bayesian approach, while it is not helpful to include the grouping-by-delay information from the Granger model.

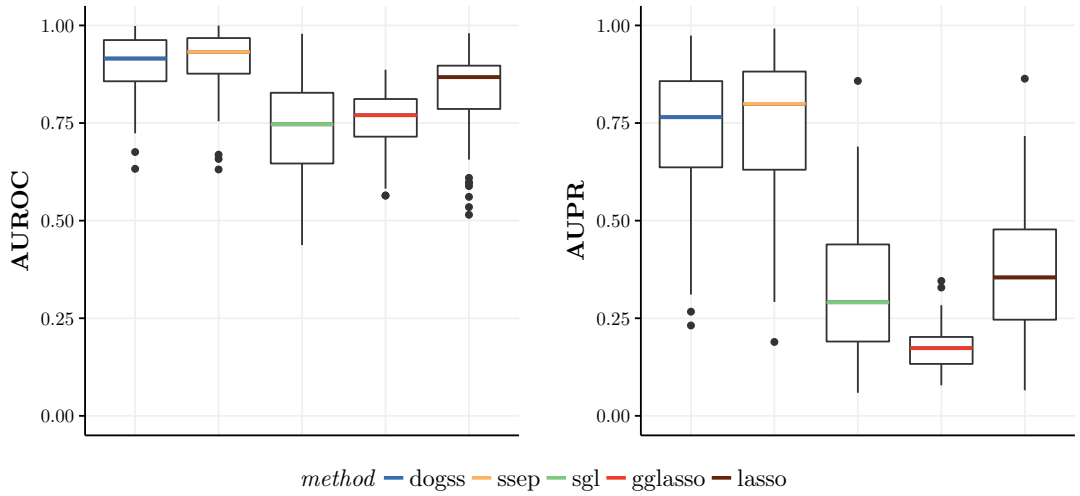

Figure 6.2.: Boxplots of AUROC and AUPR for network scenario 1 with parameters $(G, p, T, k, \sigma_0) = (10, 3, 50, 15, 1)$.

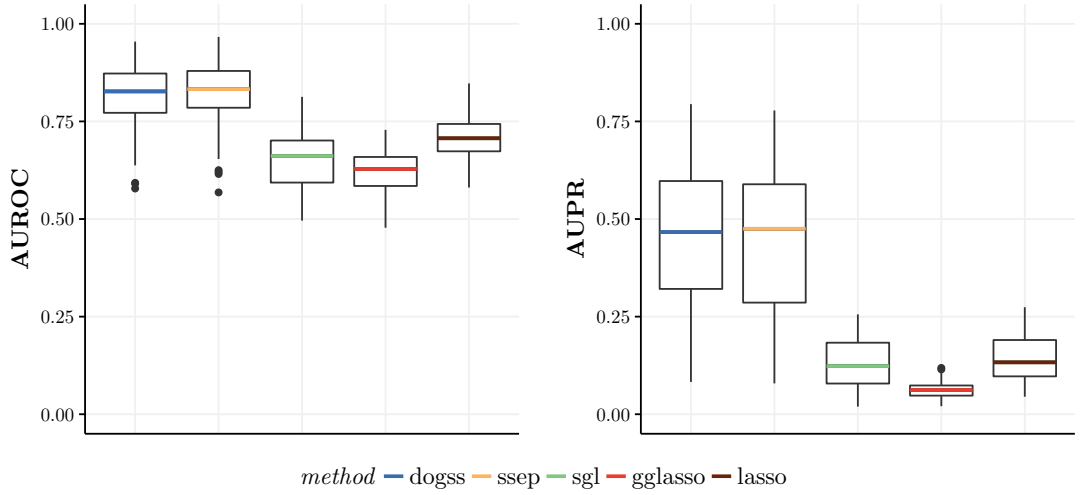

Figure 6.3.: Boxplots of AUROC and AUPR for network scenario 2 with parameters $(G, p, T, k, \sigma_0) = (30, 3, 20, 50, 1)$.

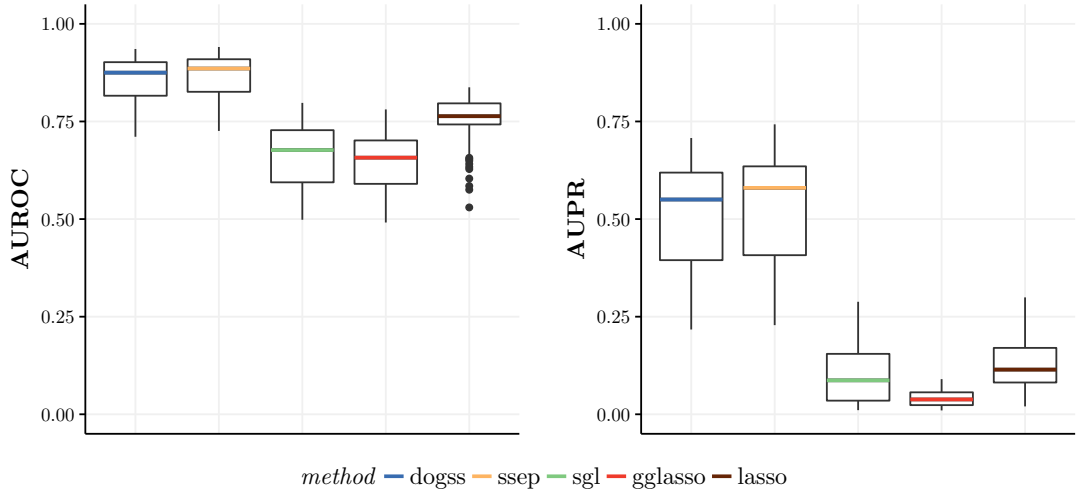

Figure 6.4.: Boxplots of AUROC and AUPR for network scenario 3 with parameters $(G, p, T, k, \sigma_0) = (100, 3, 50, 300, 1)$.

6.3. Experimental data

We assess the capability of the VAR(p) model and our proposed algorithm to reconstruct networks from experimental time series data. To this end we study two known networks from the literature: The synthetic network IRMA [13] and a classic data set from the yeast cell cycle [81]. We give a short introduction to the data sets, show the "gold standard" networks and present the resulting ROC and Precision-Recall curves for different methods. The lasso, group lasso and sparse-group lasso were used with standard parameter settings, but the intercept and standardize flags were set to false, since data was scaled beforehand. Our approach used the standard parameter settings, too (especially $\sigma_0 = 1$ and $\sigma_{\text{slab}} = 2$). All algorithms used a maximum delay of 3, thus we estimated a VAR(3) model.

6.3.1. IRMA network

The IRMA network is a synthetic network built into the yeast Saccharomyces cerevisiae, specifically designed to test and benchmark reverse-engineering and modeling approaches [13]. The five genes within the network regulate each other and are affected only negligibly by native yeast genes. We study the "switch on"

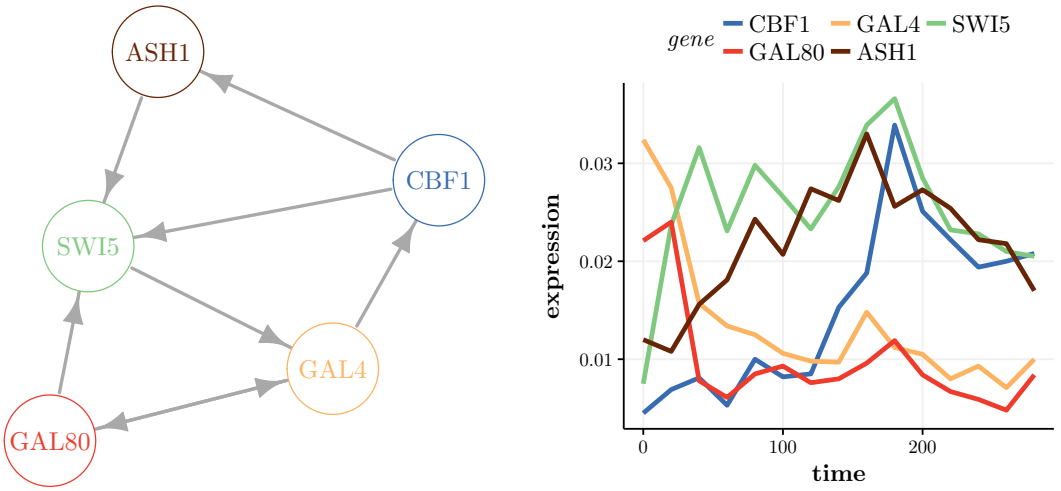

Figure 6.5.: True IRMA network and switch-on time series data.

method	dogss	ssep	sgl	gglasso	lasso	random
AUROC	0.54	0.54	0.43	0.46	0.36	0.5
AUPR	0.41	0.41	0.27	0.28	0.25	0.32

Table 6.1.: AUROC and AUPR values for the reconstruction of the IRMA network.

dataset which comprises of gene expression measurements over $T = 15$ equally spaced time points (measurements were taken in intervals of 20 minutes) after the cell culture was shifted from a glucose- to a galactose-containing medium. The gene expression levels were averaged over three experiments. Figure 6.5 shows the network of the five genes and the average expression levels from the switch-on experiment.

Figure 6.6 and Table 6.1 present the results of applying the different methods for feature selection. While the Bayesian methods perform better both in terms of ROC- and Precision-recall analysis than the lasso methods, there is still much to be desired. All methods are not significantly better than the random estimator regarding the ROC curve, and only the Bayesian methods are slightly better than a random estimator regarding the AUPR measure.

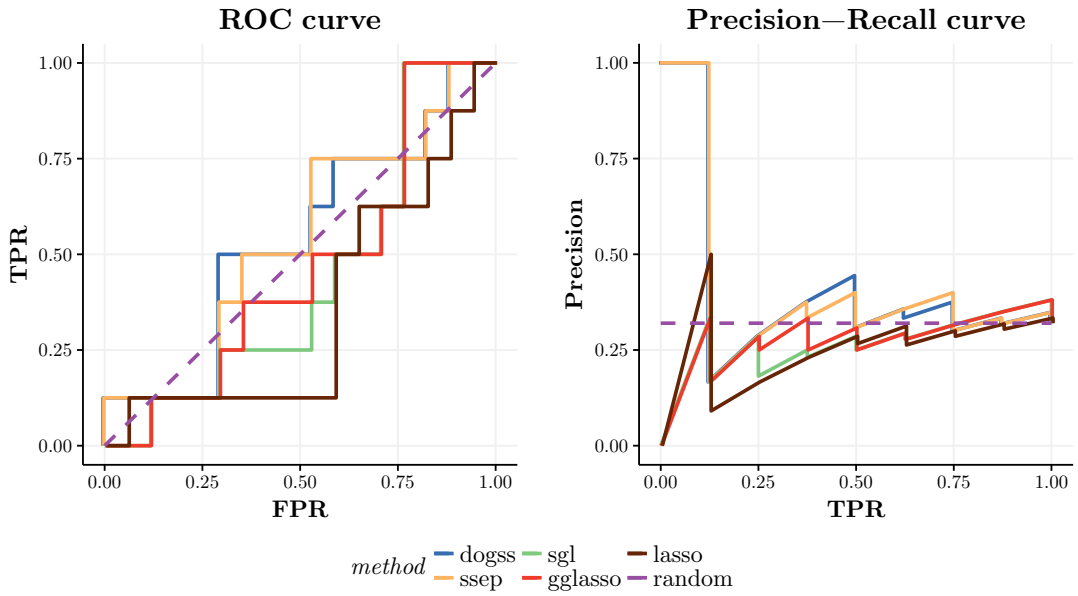

Figure 6.6.: ROC and PR curves for the reconstruction of the IRMA network, see table 6.1 for AUROC and AUPR values.

6.3.2. Yeast cell cycle

We analyze the yeast cell cycle pathway as given by the KEGG database [46]. We analyze the expression data of 11 genes which are part of the G1 step of the yeast cell cycle from a classic microarray dataset [81]. The chosen dataset comprises of $T = 17$ time points of measurements that were taken in intervals of 10 minutes. Figure 6.7 shows the KEGG (gold standard) network of the eleven genes.

Figure 6.8 and Table 6.2 present the results of our analysis, which are more convincing than the results of the IRMA data. The Bayesian methods perform better than the lasso methods, which are closer to the random estimator. The sparse-group Bayesian feature selection gives slightly better results than the Bayesian feature selection without grouping.

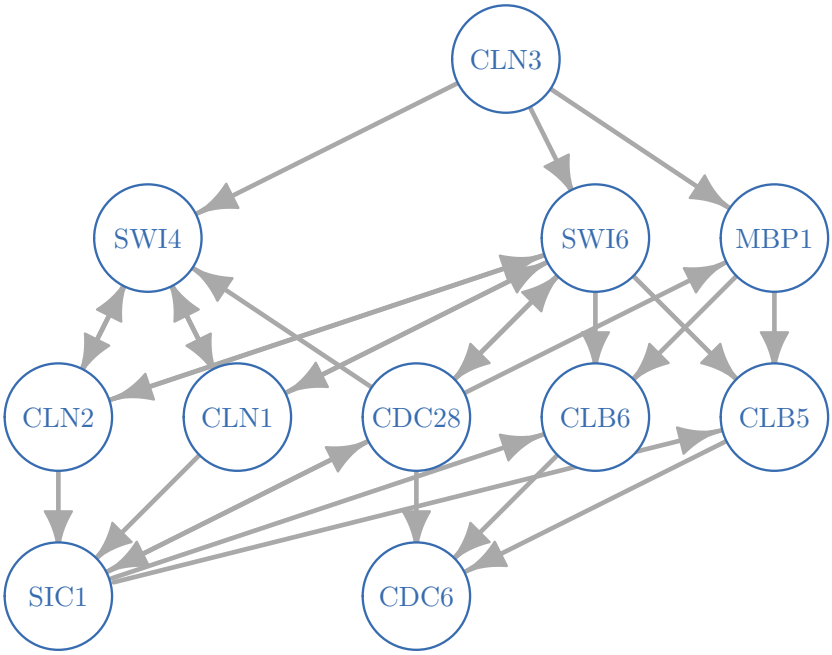

Figure 6.7.: Part of the yeast cell cycle network from KEGG [46].

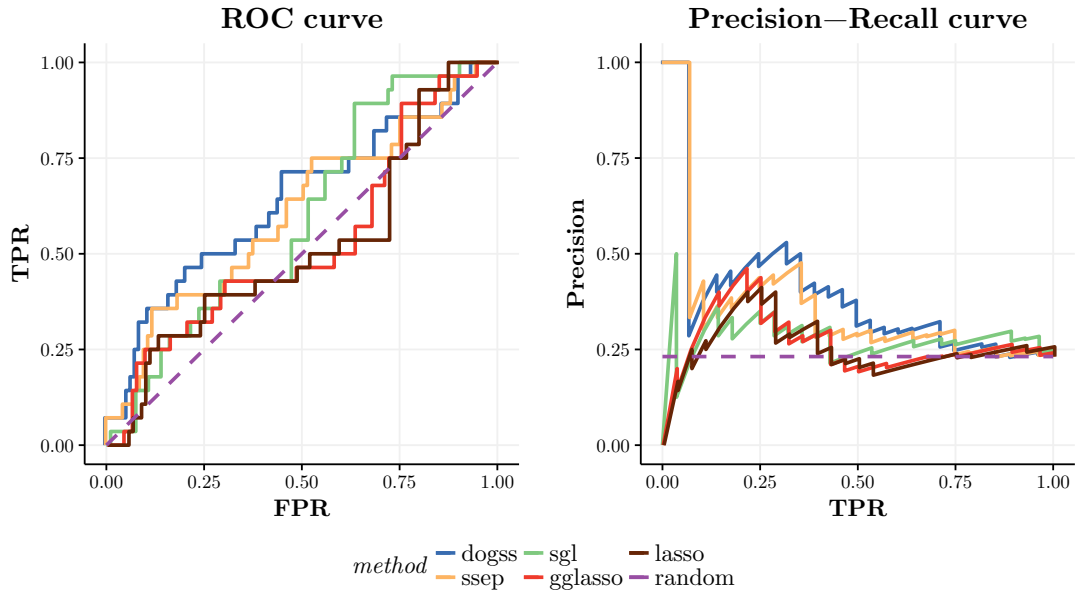

Figure 6.8.: ROC and PR curves for the reconstruction of the yeast cell cycle network, see table 6.2 for AUROC and AUPR values.

method	dogss	ssep	sgl	gglasso	lasso	random
AUROC	0.64	0.61	0.59	0.53	0.52	0.5
AUPR	0.39	0.36	0.27	0.26	0.25	0.231

Table 6.2.: AUROC and AUPR values for the reconstruction of the yeast cell cycle network.

6.4. Oscillations

The VAR(p) model is able to generate oscillating time series, and as such we discuss some general aspects of oscillating VAR(p) models and test if our feature selection approach can reconstruct network models that show oscillating behavior.

In Appendix C we discuss the link between the eigenproblem of A and the stability of the VAR(p) process defined by A. Furthermore, the eigenvalues and eigenvectors of A give us information on the periodicity of the multivariate time series [69, 90], this is called principal oscillation pattern analysis [90]. Suppose we have a VAR(p) model of G variables, then the companion $(Gp \times Gp)$-matrix A has Gp eigenvalues and matching eigenvectors. Each eigenvalue-eigenvector pair (λ_i, v^i), $i = 1, \ldots, Gp$, is associated with a certain period Π_i and damping time Θ_i (both measured in time units according to the spacing between measurements). The period describes the duration of time of one complete cycle in the oscillation, while the damping time is a measure for the decay of the (damped) oscillation. The period and damping time can be calculated as follows:

$$\Pi_i = \frac{2\pi}{|\operatorname{Arg}(\lambda_i)|},$$
$$\Theta_i = -\frac{1}{\log(|\lambda_i|)},$$

where $\operatorname{Arg}(z)$ for $z \in \mathbb{C}$ is the argument of z, $-\pi \leq z \leq \pi$ (simply put the angle between the vector z and the positive real axis in the complex plane). If $\operatorname{Arg}(\lambda_i) = 0$, that is, $\lambda_i \in \mathbb{R}^+$, the corresponding period $\Pi_i = \infty$ and the corresponding eigenvector v^i is called a relaxator. Otherwise, if $\operatorname{Arg}(\lambda_i) \neq 0$ and as such $\lambda_i \in \mathbb{R}^-$ or $\operatorname{Im}(\lambda_i) \neq 0$, the corresponding eigenvector v^i is called an oscillator. Relaxators and oscillators can be interpreted as components of the system that impede (relax-

ator) or support (oscillator) the overall oscillations. The eigenvectors (relaxators or oscillators) do not correspond directly to the features of the VAR(p) model.

Neumaier and Schneider [69] show that the original state vectors y_t and noise vectors ε_t can be represented as linear combinations of the last G entries of the length-Gp eigenvectors v^i:

$$y_t = \sum_{i=1}^{Gp} \beta_{i,t}^{y} \cdot (v_j^i)_{j=G(p-1)+1,\ldots,Gp},$$
$$\varepsilon_t = \sum_{i=1}^{Gp} \beta_{i,t}^{\varepsilon} \cdot (v_j^i)_{j=G(p-1)+1,\ldots,Gp}.$$

According to von Storch et al. [90], the eigenvectors corresponding to the eigenvalues with the largest absolute value have the most influence on the actual time series variables. While Neumaier and Schneider [69] argue that this is not true in general, our simulations did not provide any indication against this rule of thumb. Put together, we can observe oscillations in a VAR(p)-generated time series if the VAR(p) process is stable or equivalently $\rho(\mathsf{A}) < 1$ (at least approximately) and the largest eigenvalues are in absolute value close to 1 with a relevant imaginary part.

6.4.1. Simulated networks with oscillations

We employ the same simulation scheme like in Section 6.2 with $(p, k, T, N, \sigma_0) = (6, 30, 100, 5, 0.5)$, but between step 3 and 4 we check the oscillation properties of the matrix A derived from A_1 to A_p: We calculate the eigenvalues, their absolute values and the corresponding damping times and periods. Looking at the top two eigenvalues (in terms of their respective absolute values), if their damping times are larger than T and periods larger than 5, we keep A and continue by simulating the data Y. Otherwise we discard the matrix and start anew at step 1. This way we proceed until we have 100 data sets to test the reconstruction power of the algorithms.

The results can be seen in Figure 6.9 and are very similar to the results in Section 6.2. We conclude that for the network reconstruction it does not matter

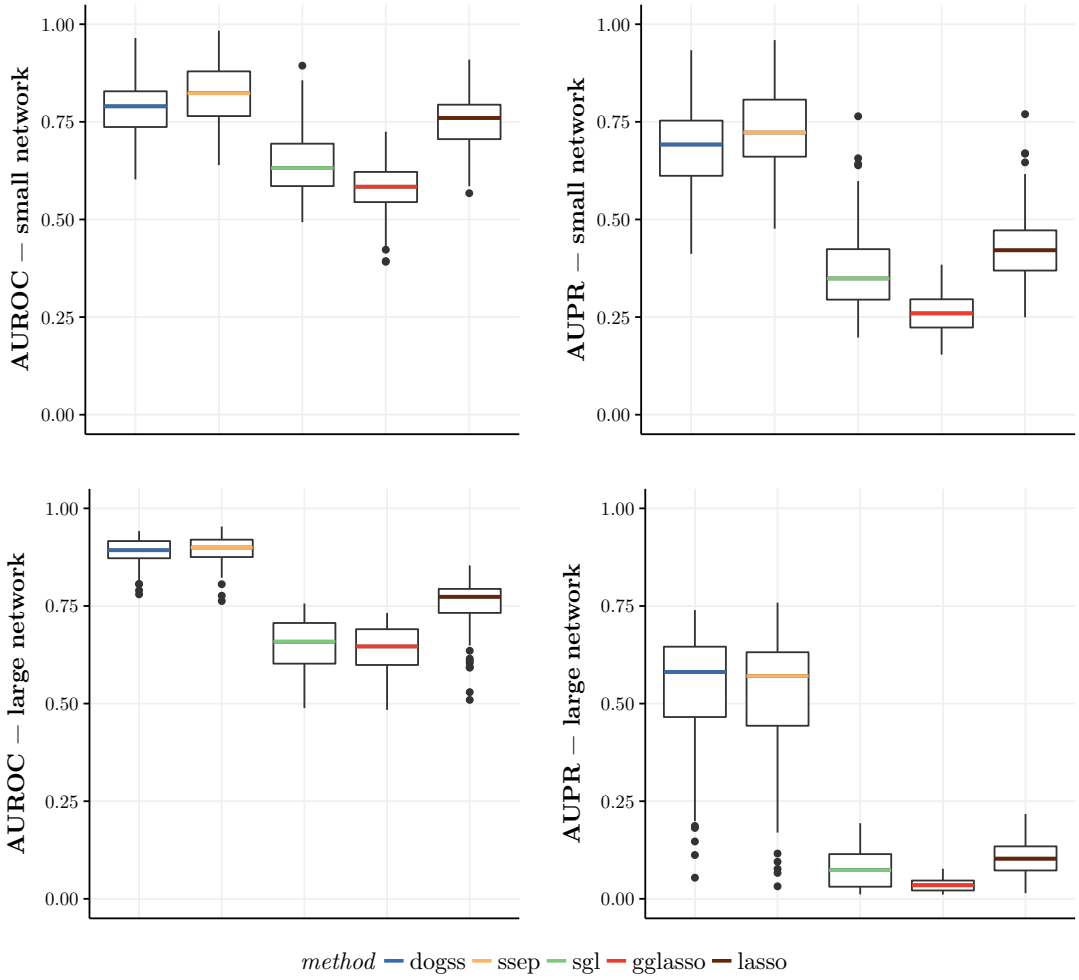

Figure 6.9.: Boxplots of AUROC/AUPR for small and large oscillating networks.

if the data is oscillating or not, as long as the assumption of the VAR(p) model is true.

6.4.2. Oscillations of circadian genes

We analyzed a dataset of multiple oscillating time series data that represent the oscillations of five genes in different mammalian tissues from Zhang et al. [99]. More specifically, the data comprises of gene expression levels (measured with RNA-Seq) of the genes BMAL1, DBP, Cry1, Rev-Erb-α and Per2, considered to be the "core clock" genes in mammals [49, 72]. Of the original 12 tissues we included nine in our analysis and excluded three because of missing values. The original time series consist of 24 measurements with an equidistant spacing of 2 hours, we linearly interpolated values such that the final data comprises of 47 time points respectively, spanning two cycles of day and night and a reasonable spacing for the VAR(p) model.

We ran our proposed method on every dataset separately multiple times over a grid-based parameter space: $p \in \{1, \dots 8\}$, $\sigma_0 \in \{0.1, 0.2, 0.5, 1, 2\}$ and $\sigma_{\text{slab}} \in \{1, 2, 3, 5, 10, 100\}$. For every instance of the algorithm, the cut-off probability for the non-zero coefficients was chosen to be 0.6. For every dataset we choose as a final model the parameters that minimized the prediction error on the second half of the dataset, that is time points 25 to 47. Figures 6.10 and 6.11 show the normalized original data, the recovered network graphs and the reconstructed time series. Overall, the reconstructed time series are a very close fit to the original data, with accurate capture of the 24 hour period and a correct succession of the peaking gene expression. Only for the skeletal muscle dataset the reconstruction failed with too much damping and a constant expression of gene BMAL1/ARNTL. Put together we can conclude that the VAR(p) model is able to find network structures and coefficients that fit the observed patterns well. But, the reconstructed networks look very different for every tissue, even though the original data differs only slightly between tissues. No common patterns can be observed in the reconstructed networks. This leads to the conclusion that the VAR(p) model is not sufficient to reconstruct the circadian core clock gene network. Reasons for this will be discussed in Chapter 7.

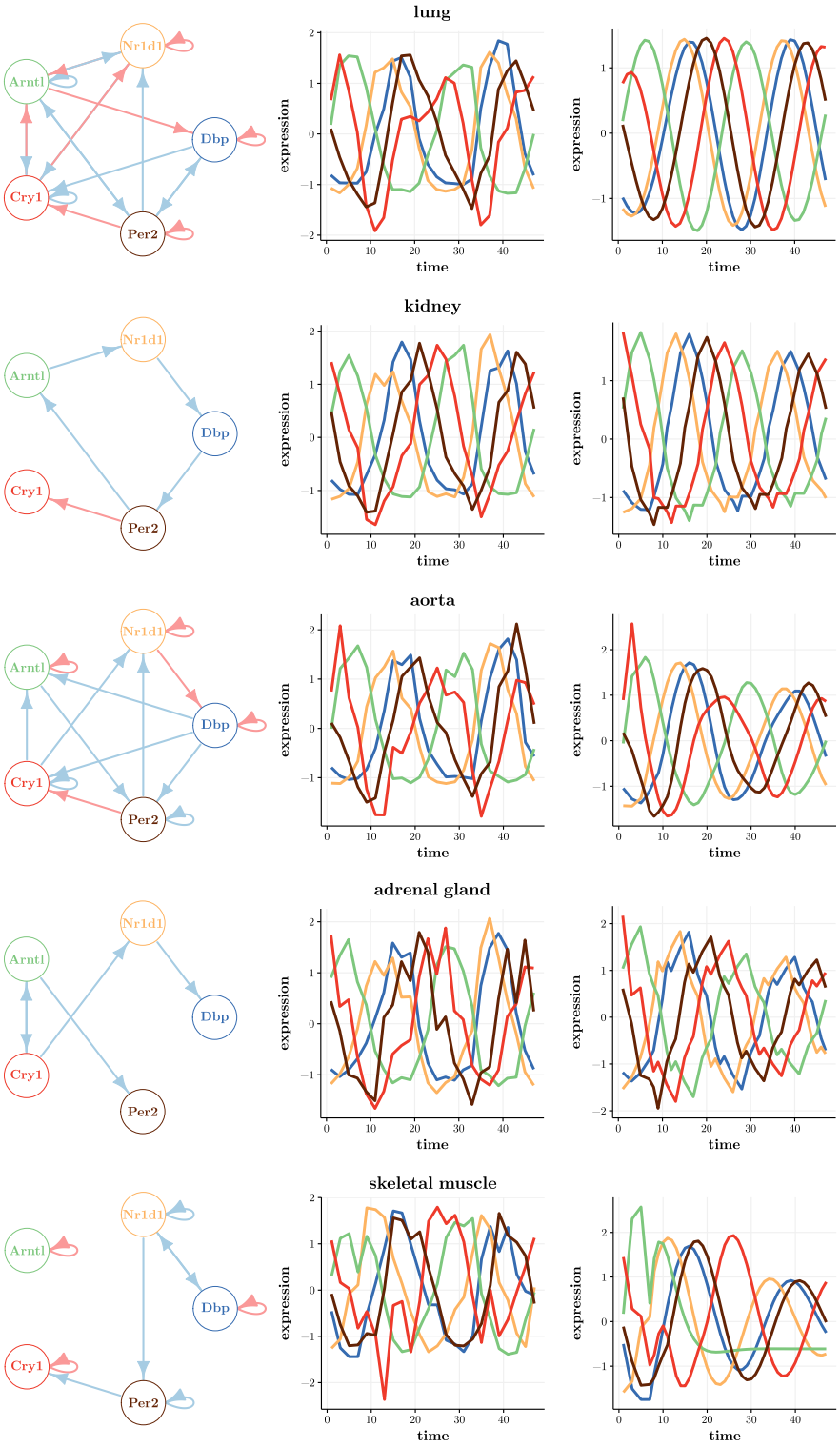

Figure 6.10.: Network reconstruction and oscillation fit on Zhang data for tissues lung, kidney, aorta, adrenal gland and skeletal muscle. Continued in Figure 6.11.

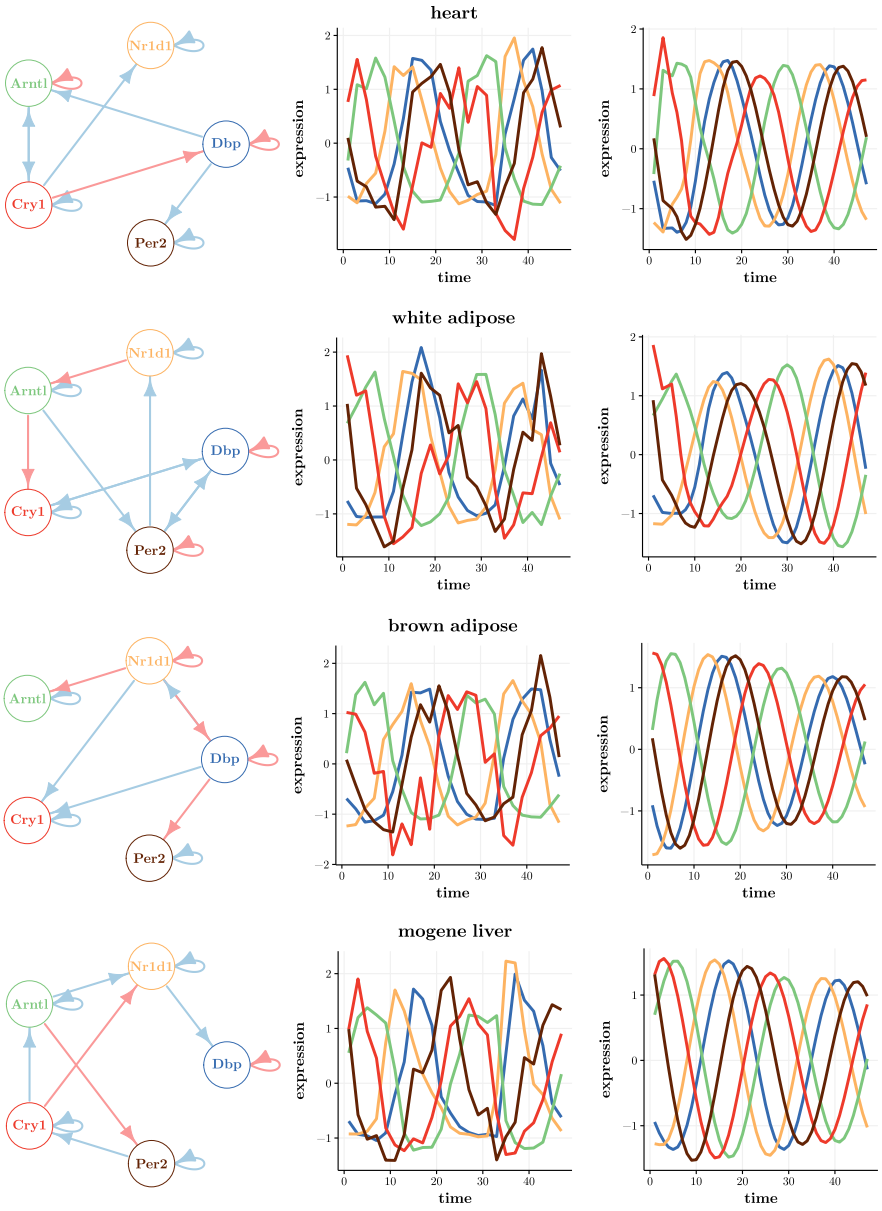

Figure 6.11.: Network reconstruction and oscillation fit on Zhang data for tissues heart, white adipose, brown adipose and mogene liver. Recovered networks on the left (blue arrows indicate activation, whereas red arrows indicate inhibition), original expression data in the middle and recovered expression patterns on the right.

6.5. Summary

We applied the (sparse-group) Bayesian feature selection method to the vector autoregressive model of order p, both on simulated and experimental data, to reconstruct gene networks from time series data. The simulations revealed that we do not gain power from including the grouping information. The Bayesian feature selection proves to be much more powerful than the lasso methods. Results on experimental data were mixed, with acceptable results on the yeast cell cycle data. We showed with simulations that the Bayesian feature selection methods are able to reconstruct networks from oscillating data, but application to an extensive experimental data set revealed a profound problem: Different network topologies are able to give rise to the same oscillation patterns.

7. Discussion

In this thesis we discussed the problem of (large-scale) gene network reconstruction from gene expression data, where we applied feature selection methods with grouping information.

To this end, we derived and applied a Bayesian framework that enforces sparsity on the between- and within-level for groups of features in tandem with a deterministic algorithm (expectation propagation) to derive the parameters of the framework. The Bayesian approach proved to be more reliable for feature selection with two-fold group sparsity than the lasso methods. The expectation propagation algorithm as an alternative to the conventional Gibbs sampling is much faster but as accurate as Gibbs sampling, and as such feasible to apply to large network inference problems. To the best of our knowledge, this is the first application of (Bayesian) neighborhood selection with grouping information to the problem of network reconstruction.

The Bayesian approach performed best on large networks regarding three measures: Correct features retrieved, prediction of expression on new data and most value for computing time spent.

The Bayesian framework is applicable to time series data with the VAR(p) model, too, and performed well on simulated time series. The results were mixed on data from temporal experiments. Incorporating the sparse-group approach did not yield advantages over the standard methods, but the Bayesian methods were much better suited for temporal data.

We compared the Bayesian approach with and without grouping information to different lasso methods. The standard Bayesian approach without grouping information is a reliable choice under any circumstances. In the presence of helpful grouping information, our new approach of sparse-group Bayesian feature selection with expectation propagation outperforms all other methods. The standard lasso

is the fastest method, but is outperformed by the Bayesian approaches and the sparse-group lasso. The group lasso suffers greatly in the presence of within-group sparsity. While the sparse-group lasso shows a slight advantage over the standard lasso in our analysis, this is by a very narrow margin and not comparable to the gain of performance that our sparse-group Bayesian approach shows. Huang [44] suggested some amendments to the sparse-group lasso, but their results are not available in the public domain.

Feature selection with grouping information is well studied (e.g. [26, 37, 56, 80, 89, 93, 94, 96, 98]), but it is rarely applied to the problem of (gene) network reconstruction. In the original sparse-group lasso paper [80] the only experimental dataset it was applied to was for classification (with logistic regression), also in [89]. Xu and Ghosh [94] only applied their methods to simulated datasets and Huang [44] used an economics dataset only. Our work highlights the advantages of employing the sparse-group Bayesian framework on genetic data and network reconstruction. This was made possible only by inclusion of the expectation propagation algorithm, yielding faster run time compared to the much slower Gibbs sampling. While the expectation propagation algorithm was already applied to a standard (no grouping, [43]) and between-group sparse model [41], it has not been extended to the two-level sparse group model yet. Our simulations and applications clearly show the superiority of this approach to detect correct sparsity patterns on a between- and within-group level.

Li and Zhang [53], Lin et al. [54] and Atchadé [3] used Bayesian approaches for neighborhood selection, but all three works suffer from the use of Markov Chain Monte Carlo/Gibbs sampling algorithms, while not including grouping information either.

Often the grouping of features arises naturally from the observed data, and here we discuss different kinds of groupings: By correlation or clustering, by network-hub membership or co-binding affinities, by biological modules, just random grouping or in a different model (vector autoregression) by time-delay information. We observe an increase of recovered interactions in network reconstruction when using available grouping information.

Of course, there is a distinction between a recovered edge in a network and a true

causal relationship (for a deeper reading into this topic consider the works of Pearl [71] and Spirtes et al. [82]). While true causal relationships cannot be recovered in most cases, recovered edges are at least a strong indication for possible causalities. The same holds true for the temporal model of vector autoregression (VAR(p)) and the underlying model of Granger causality: Granger causality does not imply "true" causality, but it is an indicator for a causal relationship. Granger himself mentioned this problem in his Nobel prize acceptance speech [34].

The standard lasso is a fast algorithm and does not depend on the specification of a noise parameter like our Bayesian approach. The problem of specifying the noise parameter in the Bayesian framework can be alleviated by cross-validation methods or plug-in estimators for the noise, although our simulations indicated that the specification of the noise gets problematic only in the realm of highly noisy data and the lasso methods suffer as much as the Bayesian methods in the presence of high noise.

Another advantage of the Bayesian approach over the lasso methods is that probabilities are provided along with the features, which makes it more easily applicable to network reconstruction (since the probabilities give a natural and comparable ranking over different instances of the algorithm). Meinshausen and Bühlmann [63] and subsequently Haury et al. [38] presented ways to assign probabilities to features within the lasso framework which rely on repeating the algorithm multiple times which increases the computational burden. We note that our algorithm can be used as a plug-in for the lasso methods within the same frameworks, too, and further research should study if this is useful and how different feature selection methods perform with stability selection.

While the Bayesian approach in general delivers good results on simulated time series data, the results on experimental time series were less conclusive, especially for the oscillation data. We argue that this is due to the violation of two assumptions of the VAR(p) model: linear dependence on previous values and furthermore the propagation of the noise throughout the time series. The latter fact is violated since the experimental procedure (different individuals and even bulk data for different time points) does not allow to capture the propagation of the noise. There exist alternatives to the (linear) autoregressive model, for example

delay-differential equations. There have been successful applications of a delay-differential equation system to oscillatory data, see for example Korenčič et al. [49].

The lasso approach (and as such the sparse-group lasso, too) are applicable in a straight-forward manner to logistic regression and thus for the task of classification [80, 89]. This is not true for the Bayesian model and the expectation propagation algorithm, since the form of the probability distribution functions changes and the direct application of the results for the linear case is not possible. Damien et al. [18, chapter 23] applied a Bayesian framework with Laplace priors to the problem of logistic regression (with Gibbs sampling).

When considering the hub model of gene regulatory networks, we need to assume some knowledge about the network structure beforehand. First, the scale-free property is implied, which is still up to debate for biological data [1, 48]. Second, if the hubs are unknown, they need to be identified beforehand. E.g., this can be done by identifying transcription factors and their binding sites/motifs [5, 87], using additional data such as DNA sequence or ChIP-seq profiles on top of the expression data. An alternative is an expression data-driven approach to identify hubs [83]. Tan et al. [84] and McGillivray [61] combined the hub identification with the graphical lasso. Hub identification could prove to be even more useful for the analysis of protein-protein interaction networks [39]. Another possibility to identify grouped features/genes within the gene regulatory network inference framework is by pathways or biological function [10, 86].

These groupings and their impact on the performance of grouped feature selection methods should be studied further. Also it would be useful to include a prior distribution for the noise level and adapt the expectation propagation algorithm to deal with this prior, too. Another open question is how to determine the slab parameter beforehand and while this can be considered with cross-validation, (testable) slab parameter estimators need to be studied. Furthermore, "fuzzy" group memberships of features or features belonging to different groups at the same time are a natural property arising in biological data and it would be interesting to include this into the grouped feature model.

We pursued the approach of neighborhood selection for the reconstruction of net-

works (based on the Gaussian graphical model and the precision matrix). Another complementary sparsity enforcing approach is the graphical lasso by Friedman et al. [25], which estimates the precision matrix directly. A Bayesian framework was applied to the graphical lasso by Wang [91], but it suffers from the use of Gibbs sampling and it does not include grouping information. Friedman et al. [26] used the graphical lasso in conjunction with ideas from the group lasso, but only to enforce symmetry on the precision matrix. Tao et al. [85] considered groups of features while estimating the precision matrix, but with a different scheme of overlapping submatrices. As such it would be fruitful to study a proper sparse-group graphical lasso or a deterministic (maybe expectation propagation-based) algorithm for the Bayesian graphical lasso.

We note that our proposed method is not restricted to gene regulatory network inference or even on biological data for that matter. The framework is very general and can be applied to any scenario where grouping information is available. Two examples are economics (stock markets) and neuroscience (fMRI studies).

Put together, we advise to use our method on any problem of feature selection where features are grouped with sparsity on a between- and within-group level. Our method is superior to the two conceptually similar alternatives currently available: the sparse-group lasso (which is slower and less reliable, and even if grouping information is more noisy, the standard spike-and-slab is a better option) and the similar Bayesian framework with Gibbs sampling (which is too slow to apply it to large-scale problems). Furthermore, Bayesian feature selection performs better than lasso methods for the task of gene regulatory network reconstruction with neighborhood selection.

Bibliography

[1] Réka Albert. Scale-free networks in cell biology. *Journal of Cell Science*, 118 (21):4947–4957, 2005.

[2] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Molecular biology of the cell*. Garland Science, New York, NY, 4th edition, 2002.

[3] Yves Atchadé. A scalable quasi-Bayesian framework for Gaussian graphical models. 2015. arXiv: 1512.07934.

[4] M. Madan Babu, Nicholas M. Luscombe, L. Aravind, Mark Gerstein, and Sarah A. Teichmann. Structure and evolution of transcriptional regulatory networks. *Current Opinion in Structural Biology*, 14(3):283–291, 2004.

[5] Timothy L. Bailey, Mikael Boden, Fabian A. Buske, Martin Frith, Charles E. Grant, Luca Clementi, Jingyuan Ren, Wilfred W. Li, and William S. Noble. MEME Suite: tools for motif discovery and searching. *Nucleic Acids Research*, 37(Web Server issue):W202–W208, 2009.

[6] Sergey Bakin. *Adaptive regression and model selection in data mining problems*. PhD thesis, Australian National University, Canberra, Australia, 1999. http://hdl.handle.net/1885/9449.

[7] Ziv Bar-Joseph, Anthony Gitter, and Itamar Simon. Studying and modelling dynamic biological processes using time-series gene expression data. *Nature Reviews Genetics*, 13(8):552–564, 2012.

[8] Albert-László Barabási and Zoltán N. Oltvai. Network biology: understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2):101–113, 2004.

[9] Douglas Bates and Dirk Eddelbuettel. Fast and elegant numerical linear algebra using the RcppEigen package. *Journal of Statistical Software*, 52(5): 1–24, 2013.

[10] Sebastian Bauer, Steffen Grossmann, Martin Vingron, and Peter N. Robinson. Ontologizer 2.0—a multifunctional tool for GO term enrichment analysis and data exploration. *Bioinformatics*, 24(14):1650–1651, 2008.

[11] Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, New York, NY, 2006.

[12] Paul Bromiley. Products and convolutions of gaussian probability density functions. *Tina-Vision Memo*, 3(4), 2003.

[13] Irene Cantone, Lucia Marucci, Francesco Iorio, Maria Aurelia Ricci, Vincenzo Belcastro, Mukesh Bansal, Stefania Santini, Mario di Bernardo, Diego di Bernardo, and Maria Pia Cosma. A yeast synthetic network for in vivo assessment of reverse-engineering and modeling approaches. *Cell*, 137(1): 172–181, 2009.

[14] Robert Castelo and Alberto Roverato. Reverse engineering molecular regulatory networks from microarray data with qp-graphs. *Journal of Computational Biology*, 16(2):213–227, 2009.

[15] Aaron Clauset, Cosma Rohilla Shalizi, and Mark E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, 2009. arXiv: 0706.1062.

[16] David Roxbee Cox and Nanny Wermuth. *Multivariate dependencies: models, analysis and interpretation*. Number 67 in Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, Boca Raton, FL, 1998.

[17] Francis H. C. Crick. Central dogma of molecular biology. *Nature*, 227(5258): 561–563, 1970.

[18] Paul Damien, Petros Dellaportas, Nicholas G. Polson, and David A. Stephens, editors. *Bayesian theory and applications*. Oxford University Press, Oxford, U.K., reprint edition, 2015.

[19] Eric Davidson and Michael Levin. Gene regulatory networks. *Proceedings of the National Academy of Sciences of the United States of America*, 102(14): 4935, 2005.

[20] Yonathan Lissanu Deribe, Tony Pawson, and Ivan Dikic. Post-translational modifications in signal integration. *Nature Structural & Molecular Biology*, 17(6):666–672, 2010.

[21] Patrik D'haeseleer, Shoudan Liang, and Roland Somogyi. Genetic network inference: from co-expression clustering to reverse engineering. *Bioinformatics*, 16(8):707–726, 2000.

[22] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

[23] Paul Erdős and Alfréd Rényi. On random graphs I. *Publ. Math. Debrecen*, 6:290–297, 1959.

[24] Jianqing Fan, Yuan Liao, and Han Liu. An overview on the estimation of large covariance and precision matrices. *The Econometrics Journal*, 19(1): 46, 2016. arXiv:1504.02995.

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

[26] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Applications of the lasso and grouped lasso to the estimation of sparse graphical models. *Technical report, Stanford University*, 2010.

[27] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A note on the group lasso and a sparse group lasso. *arXiv preprint*, 2010. arXiv:1001.0736.

[28] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 2010.

[29] Nicole L. Garneau, Jeffrey Wilusz, and Carol J. Wilusz. The highways and byways of mRNA decay. *Nature Reviews Molecular Cell Biology*, 8(2):113–126, 2007.

[30] Edward I. George and Robert E. McCulloch. Approaches for Bayesian variable selection. *Statistica Sinica*, 7(2):339–373, 1997.

[31] John F. Geweke. Variable selection and model comparison in regression. Working Paper 539, University of Minnesota and Federal Reserve Bank of Minneapolis, Minneapolis, MN, 1994. retrieved from https://www.minneapolisfed.org/research/wp/wp539.pdf.

[32] Ary L. Goldberger and Bruce J. West. Fractals in physiology and medicine. *The Yale Journal of Biology and Medicine*, 60(5):421–435, 1987.

[33] Clive W. J. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3):424–438, 1969.

[34] Clive W. J. Granger. Time series analysis, cointegration, and applications. *The American Economic Review*, 94(3):421–425, 2004.

[35] Trevor Hastie and Robert Tibshirani. Efficient quadratic regularization for expression arrays. *Biostatistics*, 5(3):329–340, 2004.

[36] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Series in Statistics. Springer, New York, NY, 2nd edition, 2009.

[37] Stefan Haufe, Klaus-Robert Müller, Guido Nolte, and Nicole Krämer. Sparse causal discovery in multivariate time series. In *Proceedings of the 2008th*

International Conference on Causality: Objectives and Assessment - Volume 6, pages 97–106, Whistler, Canada, 2008. JMLR.org. arXiv: 0901.2234.

[38] Anne-Claire Haury, Fantine Mordelet, Paola Vera-Licona, and Jean-Philippe Vert. TIGRESS: trustful inference of gene regulation using stability selection. *BMC Systems Biology*, 6(1):145, 2012.

[39] Xionglei He and Jianzhi Zhang. Why do hubs tend to be essential in protein networks? *PLOS Genetics*, 2(6):e88, 2006.

[40] Daniel Hernández-Lobato. *Prediction based on averages over automatically induced learners: Ensemble methods and Bayesian techniques*. PhD thesis, Universidad Autónoma de Madrid, Madrid, Spain, 2009. https://repositorio.uam.es/handle/10486/4161.

[41] Daniel Hernández-Lobato, José Miguel Hernández-Lobato, and Pierre Dupont. Generalized spike-and-slab priors for Bayesian group feature selection using expectation propagation. *The Journal of Machine Learning Research*, 14(1):1891–1945, 2013.

[42] José Miguel Hernández-Lobato. *Balancing flexibility and robustness in machine learning: Semi-parametric methods and sparse linear models*. PhD thesis, Universidad Autónoma de Madrid, Madrid, Spain, 2010. https://repositorio.uam.es/handle/10486/6080.

[43] José Miguel Hernández-Lobato, Daniel Hernández-Lobato, and Alberto Suárez. Expectation propagation in linear regression models with spike-and-slab priors. *Machine Learning*, 99(3):437–487, 2015.

[44] Xue Huang. *Sparse feature and element selection in high-dimensional vector autoregressive models*. PhD thesis, Florida State University, Tallahassee, FL, 2016. http://diginole.lib.fsu.edu/islandora/object/fsu%3A405587.

[45] Yufei Huang, Isabel M. Tienda-Luna, and Yufeng Wang. Reverse engineering gene regulatory networks. *IEEE Signal Processing Magazine*, 26(1):76–97, 2009.

[46] Minoru Kanehisa, Miho Furumichi, Mao Tanabe, Yoko Sato, and Kanae Morishima. KEGG: new perspectives on genomes, pathways, diseases and drugs. *Nucleic Acids Research*, 45(D1):D353–D361, 2017.

[47] Guy Karlebach and Ron Shamir. Modelling and analysis of gene regulatory networks. *Nature Reviews Molecular Cell Biology*, 9(10):770–780, 2008.

[48] Raya Khanin and Ernst Wit. How scale-free are biological networks. *Journal of Computational Biology*, 13(3):810–818, 2006.

[49] Anja Korenčič, Rok Košir, Grigory Bordyugov, Robert Lehmann, Damjana Rozman, and Hanspeter Herzel. Timing of circadian genes in mammalian tissues. *Scientific Reports*, 4:5782, 2014.

[50] Lynn Kuo and Bani Mallick. Variable selection for regression models. *Sankhyā: The Indian Journal of Statistics, Series B (1960-2002)*, 60(1): 65–81, 1998.

[51] Minjung Kyung, Jeff Gill, Malay Ghosh, and George Casella. Penalized regression, standard errors, and Bayesian lassos. *Bayesian Analysis*, 5(2): 369–411, 2010.

[52] Les Laboratoires Servier. Servier Medical Art, CC 3.0 https://creativecommons.org/licenses/by/3.0/, changes were made, original material available at https://smart.servier.com/, 2018.

[53] Fan-qun Li and Xin-sheng Zhang. Bayesian lasso with neighborhood regression method for Gaussian graphical model. *Acta Mathematicae Applicatae Sinica, English Series*, 33(2):485–496, 2017.

[54] Zhixiang Lin, Tao Wang, Can Yang, and Hongyu Zhao. On joint estimation of Gaussian graphical models for spatial and temporal data. *Biometrics*, 73 (3):769–779, 2017.

[55] Benoit Liquet, Kerrie Mengersen, Anthony N. Pettitt, and Matthew Sutton. Bayesian variable selection regression of multivariate responses for group data. *Bayesian Analysis*, 12(4):1039–1067, 2017.

[56] Aurélie C. Lozano, Naoki Abe, Yan Liu, and Saharon Rosset. Grouped graphical Granger modeling for gene expression regulatory networks discovery. *Bioinformatics*, 25(12):i110–i118, 2009.

[57] Helmut Lütkepohl. *New introduction to multiple time series analysis*. Springer, Berlin, Germany; New York, NY, 2005.

[58] Daniel Marbach, James C. Costello, Robert Küffner, Nicole M. Vega, Robert J. Prill, Diogo M. Camacho, Kyle R. Allison, The DREAM5 Consortium, Manolis Kellis, James J. Collins, and Gustavo Stolovitzky. Wisdom of crowds for robust gene network inference. *Nature Methods*, 9(8):796–804, 2012.

[59] John C. Marioni, Christopher E. Mason, Shrikant M. Mane, Matthew Stephens, and Yoav Gilad. RNA-seq: An assessment of technical reproducibility and comparison with gene expression arrays. *Genome Research*, 18(9):1509–1517, 2008.

[60] Florian Markowetz and Rainer Spang. Inferring cellular networks – a review. *BMC Bioinformatics*, 8(Suppl 6):S5, 2007.

[61] Annaliza McGillivray. *Sparse estimation of structured inverse covariance matrices*. PhD thesis, McGill University, Montreal, Canada, 2016. retrieved from http://digitool.library.mcgill.ca/R/?func=dbin-jump-full&object_id=143691.

[62] Nicolai Meinshausen and Peter Bühlmann. High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, 34(3):1436–1462, 2006.

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

[64] Thomas Peter Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 362–369. Morgan Kaufmann Publishers Inc., 2001.

[65] Thomas Peter Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2001. https://dspace.mit.edu/handle/1721.1/86583.

[66] Thomas Peter Minka and John Lafferty. Expectation-propagation for the generative aspect model. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pages 352–359, San Francisco, CA, 2002. Morgan Kaufmann Publishers Inc.

[67] Toby J. Mitchell and John J. Beauchamp. Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404): 1023–1032, 1988.

[68] Balas Kausik Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234, 1995.

[69] Arnold Neumaier and Tapio Schneider. Estimation of parameters and eigenmodes of multivariate autoregressive models. *ACM Trans. Math. Softw.*, 27 (1):27–57, 2001.

[70] Trevor Park and George Casella. The Bayesian lasso. *Journal of the American Statistical Association*, 103(482):681–686, 2008.

[71] Judea Pearl. *Causality: models, reasoning, and inference*. Cambridge University Press, Cambridge, U.K.; New York, NY, 2000.

[72] J. Patrick Pett, Anja Korenčič, Felix Wesener, Achim Kramer, and Hanspeter Herzel. Feedback loops of the mammalian circadian clock constitute repressilator. *PLOS Computational Biology*, 12(12):e1005266, 2016.

[73] John Quackenbush. Microarray data normalization and transformation. *Nature Genetics*, 32(Supp):496–501, 2002.

[74] R Core Team. R: A Language and Environment for Statistical Computing, 2017.

[75] Mark Schena, Dari Shalon, Ronald W. Davis, and Patrick O. Brown. Quantitative monitoring of gene expression patterns with a complementary dna microarray. *Science*, 270(5235):467–470, 1995.

[76] Juliane Schäfer and Korbinian Strimmer. An empirical Bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21(6):754–764, 2005.

[77] Juliane Schäfer and Korbinian Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical Applications in Genetics and Molecular Biology*, 4(1):32, 2005.

[78] Matthias W. Seeger. Bayesian inference and optimal design for the sparse linear model. *Journal of Machine Learning Research*, 9(Apr):759–813, 2008.

[79] Adrian Silvescu and Vasant Honavar. Temporal Boolean network models of genetic networks and their inference from gene expression time series. *Complex Systems*, 13(1):61–78, 2001.

[80] Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22 (2):231–245, 2013.

[81] Paul T. Spellman, Gavin Sherlock, Michael Q. Zhang, Vishwanath R. Iyer, Kirk Anders, Michael B. Eisen, Patrick O. Brown, David Botstein, and Bruce Futcher. Comprehensive identification of cell cycle–regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, 9(12):3273–3297, 1998.

[82] Peter Spirtes, Clark N. Glymour, and Richard Scheines. *Causation, prediction, and search*. MIT press, Cambridge, MA, 2nd edition, 2001.

[83] Sriganesh Srihari, Hoong Kee Ng, Kang Ning, and Hon Wai Leong. Detecting hubs and quasi cliques in scale-free networks. In *2008 19th International Conference on Pattern Recognition*, pages 1–4, 2008.

[84] Kean Ming Tan, Palma London, Karthik Mohan, Su-In Lee, Maryam Fazel, and Daniela Witten. Learning graphical models with hubs. 2014. arXiv: 1402.7349.

[85] Shaozhe Tao, Yifan Sun, and Daniel Boley. Inverse covariance estimation with structured groups. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2836–2842, Melbourne, Australia, 2017. AAAI Press.

[86] The Gene Ontology Consortium. Expansion of the Gene Ontology knowledgebase and resources. *Nucleic Acids Research*, 45(D1):D331–D338, 2017.

[87] Morgane Thomas-Chollier, Andrew Hufton, Matthias Heinig, Sean O'Keeffe, Nassim El Masri, Helge G. Roider, Thomas Manke, and Martin Vingron. Transcription factor binding predictions using TRAP for the analysis of ChIP-seq data and regulatory SNPs. *Nature Protocols*, 6(12):1860–1869, 2011.

[88] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[89] Martin Vincent and Niels Richard Hansen. Sparse group lasso and high dimensional multinomial classification. *Computational Statistics & Data Analysis*, 71:771–786, 2014.

[90] Hans von Storch, Gerd Bürger, Reiner Schnur, and Jin-Song von Storch. Principal oscillation patterns: a review. *Journal of Climate*, 8(3):377–400, 1995.

[91] Hao Wang. Bayesian graphical lasso models and efficient posterior computation. *Bayesian Analysis*, 7(4):867–886, 2012.

[92] James D. Watson and Francis H. C. Crick. Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–738, 1953.

[93] Ines Wilms and Christophe Croux. An algorithm for the multivariate group lasso with covariance estimation. *Journal of Applied Statistics*, 45(4):668–681, 2018.

[94] Xiaofan Xu and Malay Ghosh. Bayesian variable selection and estimation for group lasso. *Bayesian Analysis*, 10(4):909–936, 2015.

[95] Yi Yang and Hui Zou. A Fast Unified Algorithm for Solving Group-lasso Penalize Learning Problems. *Statistics and Computing*, 25(6):1129–1141, 2015.

[96] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[97] Ming Yuan and Yi Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

[98] Lin Zhang, Veerabhadran Baladandayuthapani, Bani K. Mallick, Ganiraju C. Manyam, Patricia A. Thompson, Melissa L. Bondy, and Kim-Anh Do. Bayesian hierarchical structured variable selection methods with application to molecular inversion probe studies in breast cancer. *Journal of the Royal Statistical Society. Series C, Applied statistics*, 63(4):595–620, 2014.

[99] Ray Zhang, Nicholas F. Lahens, Heather I. Ballance, Michael E. Hughes, and John B. Hogenesch. A circadian gene expression atlas in mammals: Implications for biology and medicine. *Proceedings of the National Academy of Sciences of the United States of America*, 111(45):16219–16224, 2014.

[100] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 67(2):301–320, 2005.

A. The Bernoulli Distribution

The Bernoulli distribution is one of the most simple and important distributions in probability theory. It is a discreet distribution that describes a random variable X with only two outcomes: Either $X = 1$ ("success") or $X = 0$ ("failure"), where the probability for a success is equal to p.

The Bernoulli distribution is an exponential family distribution, its probability density $\text{Bern}(x|p)$, sufficient statistic $T(x)$ and natural parameter η are given below:

$$\begin{aligned} \text{Bern}(x|p) &= p^x \cdot (1-p)^{1-x} \\ T(x) &= x \\ \eta &= \log \frac{p}{1-p} \\ g(\eta) &= \frac{1}{1+\exp \eta} = 1-p \\ h(x) &= 1. \end{aligned}$$

A.1. Product and quotient of Bernoulli distributions

Since the Bernoulli distribution belongs to the exponential family of distributions, the product of two Bernoulli distributions is again a Bernoulli distribution up to a normalization constant, and the parameter of the new Bernoulli distribution can be calculated as follows (see also Hernández-Lobato [40, Appendix A.1]):

$$\begin{aligned} \text{Bern}(x|p_1) \cdot \text{Bern}(x|p_2) &\propto \text{Bern}(x|p) \\ \Rightarrow p &= \frac{p_1 p_2}{p_1 p_2 + (1-p_1)(1-p_2)}. \end{aligned}$$

But with $r = \log \frac{p}{1-p} = \text{logit}(p)$ we have:

$$r = r_1 + r_2,$$

which is very helpful from a numerical point of view.

Similar results hold true for the quotient of Bernoulli distributions:

$$\begin{aligned}
&\operatorname{Bern}(x|p_1)/\operatorname{Bern}(x|p_2) \propto \operatorname{Bern}(x|p) \\
&\Rightarrow p = \frac{p_1/p_2}{p_1/p_2 + (1-p_1)/(1-p_2)} \\
&\Rightarrow r = r_1 - r_2.
\end{aligned}$$

B. The (Multivariate) Normal Distribution

The normal distribution (also "Gaussian" distribution) is probably the most well-known and important continuous probability distribution. It arises in many different scenarios, e.g. the central limit theorem and the modeling of error terms. The normal distribution describes a random variable X on a continuous (real) scale, whose values are centered around some mean μ in a bell-shaped manner with a variance σ^2. It belongs to the exponential family of distributions, and its distribution function $\mathcal{N}(x|\mu, \sigma^2)$, sufficient statistic $T(x)$ and natural parameter η are given below for the univariate case:

$$\begin{aligned}
\mathcal{N}(x|\mu, \sigma^2) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) \\
T(x) &= (x, x^2) \\
\eta &= (\frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2}) \\
g(\eta) &= \exp\frac{\eta_1^2}{4\eta_2} \cdot \sqrt{-2\eta_2} = \frac{1}{\sigma}\exp\frac{\mu^2}{-2\sigma^2} \\
h(x) &= \frac{1}{\sqrt{2\pi}}.
\end{aligned}$$

The normal distribution can be defined for a multivariate random variable $X \in \mathbb{R}^d$, too:

$$\begin{aligned}
\mathcal{N}(x|\mu, \Sigma) &= \frac{1}{\sqrt{(2\pi)^d \det\Sigma}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) \\
T(x) &= (x, xx^T) \\
\eta &= (\Sigma^{-1}\mu, -\frac{1}{2}\Sigma^{-1}) \\
g(\eta) &= \exp\left(\frac{1}{4}\eta_1^T\eta_2^{-1}\eta_1 + \frac{1}{2}\log\det(-2\eta_2)\right) = \sqrt{\frac{\det(\Sigma^{-1})}{\exp\mu^T\Sigma^{-1}\mu}} \\
h(x) &= (2\pi)^{-\frac{k}{2}}.
\end{aligned}$$

B.1. Product and quotient of normal distributions

The product or quotient of two normal distribution functions is again a normal distribution function, up to a normalization constant. The parameters μ and Σ can be calculated from the parameters of the factor/denominator/numerator distributions as follows:

$$\begin{aligned}\mathcal{N}(x|\mu_1,\Sigma_1)\cdot\mathcal{N}(x|\mu_2,\Sigma_2) &\propto \mathcal{N}(x|\mu,\Sigma)\\ \Rightarrow \Sigma &= (\Sigma_1^{-1}+\Sigma_2^{-1})^{-1},\\ \mu &= \Sigma(\Sigma_1^{-1}\mu_1+\Sigma_2^{-1}\mu_2).\end{aligned}$$

$$\begin{aligned}\mathcal{N}(x|\mu_1,\Sigma_1)/\mathcal{N}(x|\mu_2,\Sigma_2) &\propto \mathcal{N}(x|\mu,\Sigma)\\ \Rightarrow \Sigma &= (\Sigma_1^{-1}-\Sigma_2^{-1})^{-1},\\ \mu &= \Sigma(\Sigma_1^{-1}\mu_1-\Sigma_2^{-1}\mu_2).\end{aligned}$$

The results for a univariate normal distribution are just the same with $\Sigma = \sigma^2$.

B.2. Integral of product of two normal distributions

We have a useful identity for the integral over the product of two normal distributions [12], which we need for the calculation of the updated parameters in the expectation propagation algorithm:

$$\begin{aligned}&\int \mathcal{N}(\beta|m_1,V_1)\cdot\mathcal{N}(\beta|m_2,V_2)\mathrm{d}\beta\\ =&\mathcal{N}(m_2|m_1,V_1+V_2)\\ =&\mathcal{N}(m_1|m_2,V_1+V_2).\end{aligned}$$

C. Stability of the $\mathrm{VAR}(p)$ Process

We will take a closer look at equation (6.1) and study the properties of the matrices A_i and their influence on the stability of the time series model.

First we will consider the case $p = 1$ or the VAR(1) process where the maximum delay in the model is just one step in the past and equation (6.1) reduces to

$$y_t = \nu + Ay_{t-1} + \varepsilon_t$$

with $A_1 = A$. Let $\rho(A)$ be the maximum absolute value of the eigenvalues of A (that is, $\rho(A)$ is the spectral radius of A). If $\rho(A) < 1$ the VAR(1) process is stable [57, chapter 2]. In a simulation setting, if we want to simulate a stable multivariate time series with a randomly chosen coefficient matrix, we need to make sure its spectral radius is smaller than 1. We can achieve this by taking any random matrix A' and multiply it by the reciprocal of its spectral radius plus a very small constant δ: $A = \frac{1}{\rho(A')+\delta}A'$, this way A defines a stable VAR(1) process with $\rho(A) < 1$.

Moving on to the general VAR(p) process we note that every VAR(p) process (with notation from equation (6.1)) can be written as a VAR(1) process, too [57]:

$$\mathsf{y}_t = \boldsymbol{\nu} + \mathsf{A}\mathsf{y}_{t-1} + \boldsymbol{\epsilon}_t$$

with

$$\begin{aligned}
\mathsf{y}_t &= (y_t, y_{t-1}, \ldots, y_{t-p+1})^T, \\
\boldsymbol{\nu} &= (\nu, 0, \ldots, 0)^T, \\
\boldsymbol{\epsilon}_t &= (\varepsilon_t, 0, \ldots, 0)^T, \\
\mathsf{A} &= \begin{pmatrix} A_1 & A_2 & \ldots & A_{p-1} & A_p \\ I & 0 & \ldots & 0 & 0 \\ 0 & I & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & I & 0 \end{pmatrix}.
\end{aligned} \tag{C.1}$$

C. Stability of the VAR(p) *Process*

Again, the VAR(p) process is stable if $\rho(\mathsf{A}) < 1$. But this time, if we want to simulate a stable multivariate time series from random matrices $A'_1, \ldots, A'_p$, we cannot use the "divide by $\rho(\mathsf{A}')$" trick like in the VAR(1) case, since we need to leave the identity matrices in the lower part of A' untouched. This leads to the following proposition:

Proposition 1. *Let* A' *be a* VAR(p) *process-defining matrix like in equation* (C.1) *from random matrices* $A'_1, \ldots, A'_p$*, and* δ *a small positive constant. Define* $\rho = \rho(\mathsf{A}') + \delta$ *and* $A_i = \frac{1}{\rho^i} A'_i$*, and* A *the matrix defined by the matrices* $A_1, \ldots, A_p$ *according to equation* (C.1)*. Then the* VAR(p) *process defined by* A *defines a stable multivariate time series.*

Proof. The eigenvalues of the matrix A' can be derived by finding the roots of the characteristic equation:

$$\begin{aligned} 0 &\overset{!}{=} \det(\mathsf{A}' - \lambda' I) \\ \Leftrightarrow 0 &= \det(\sum_{i=1}^{p} \lambda'^{p-i} A'_i - \lambda'^{p} I). \end{aligned}$$

Likewise, the eigenvalues of the matrix A can be derived by finding the roots of another characteristic equation:

$$\begin{aligned} 0 &\overset{!}{=} \det(\mathsf{A} - \lambda I) \\ \Leftrightarrow 0 &= \det(\sum_{i=1}^{p} \lambda^{p-i} \rho^{-i} A_i - \lambda^p I) \\ &= \det(\sum_{i=1}^{p} \lambda^{p-i} \frac{\rho^{p-i}}{\rho^p} A_i - \lambda^p I) \\ &= \left(\frac{1}{\rho^p}\right)^n \det(\sum_{i=1}^{p} (\lambda\rho)^{p-i} A_i - (\lambda\rho)^p I) \\ \Rightarrow \lambda' = \lambda\rho \text{ and } |\lambda| &= |\frac{\lambda'}{\rho}| < 1. \end{aligned}$$

Thus the VAR(p) process defined by A defines a stable multivariate time series. □

D. Abstract

All the genes of an organism's genome build up an intricate network of connections between them. Many of these connections are unknown, but knowing about the structure of the network is important for e.g. medical applications. This leads to the problem of reverse engineering the (large-scale) gene regulatory network from gene expression data. Gene network reconstruction can be formulated as a problem of feature selection in a linear regression framework, and we include additional information (like co-binding of transcription factors) about the network with a grouping of features. Available methods for feature selection in the presence of grouping information have different short-comings: Lasso methods underestimate the regression coefficients and do not make good use of the grouping information, and Bayesian approaches often rely on the stochastic and slow Gibbs sampling procedure to recover the parameters, which makes them infeasible for gene network reconstruction.

Here we present a Bayesian method for feature selection with grouping information (with sparsity on the between- and within group level), where the parameters are recovered by a deterministic algorithm (expectation propagation). This sparse-group framework is applied to (large-scale) gene network reconstruction from gene expression data and extended to the vector autoregressive model for time series data.

We prove (on simulated and experimental data) that the Bayesian approach is the best choice for network reconstruction for three reasons: Highest number of correctly selected features, best prediction on new data and reasonable computing time.

We show that a Bayesian approach to feature selection is superior to lasso methods on time series data. Results on experimental temporal data are inconclusive for the linear model.

Finally we note that the presented method is very fundamental and not restricted to the reconstruction of gene regulatory networks, but can be applied to any feature selection problem with grouped features.

E. Zusammenfassung

Die Gesamtheit der Gene eines Organismus ist verwoben in einem ausgeklügelten Netzwerk von Interaktionen. Viele dieser Interaktionen sind unbekannt, aber das Wissen um die genaue Gennetzwerkstruktur ist unter anderem wichtig für medizinische Anwendungen. Das unterstreicht die Dringlichkeit, aus experimentellen Genexpressionsdaten das zugrundeliegende Gennetzwerk zu rekonstruieren, auch für sehr große Netzwerke mit vielen Genen. Gennetzwerkrekonstruktion kann als ein Problem von Variablenselektion in linearer Regression aufgefasst werden. Wir nehmen als zusätzliche Information über das Netzwerk (wie z.B. das gemeinsame Binden von Transkriptionsfaktoren) eine Gruppierung der Variablen hinzu. Die bisher verfügbaren Methoden für Variablenselektion mit Gruppierung haben verschiedene Nachteile: „Lasso“ und seine Abwandlungen setzen die Regressionskoeffizienten zu gering an und nutzen die Gruppierungsinformation nicht voll aus, Bayes'sche Ansätze benutzen meist das langsame Gibbs-Sampling, um Parameter zu bestimmen, dies verhindert ihren Einsatz für die Gennetzwerkrekonstruktion.

Wir präsentieren hier eine Bayes'sche Methode für Variablenselektion mit Gruppierungsinformation, die Spärlichkeit in den Koeffizienten zwischen und innerhalb von Gruppen durchsetzt, und außerdem die Parameter mit einem deterministischen und schnellen Algorithmus bestimmt („Expectation Propagation“). Wir wenden unsere neue Methode für die Gennetzwerkrekonstruktion an und erweitern sie auch auf das vektorautoregressive Modell für Zeitreihendaten.

Wir zeigen auf simulierten und experimentellen Daten, dass aus drei Gründen der Bayes'sche Ansatz die beste Wahl für Netzwerkrekonstruktion ist: die höchste Zahl an korrekt identifizierten Variablen, beste Voraussagekraft auf neuen Daten und eine angemessene Rechendauer.

Weiterhin zeigen wir, dass auch auf Zeitreihendaten der Bayes'sche Ansatz den Lasso-Methoden überlegen ist, wobei die Resultate mit einem linearen Modell auf experimentellen Zeitreihendaten generell weniger belastbar sind.

Darüber hinaus ist unsere neue Methode nicht nur auf die Rekonstruktion von Gennetzwerken beschränkt, sondern kann auf jedes Variablenselektionsproblem angewendet werden, bei dem eine Gruppierung der Variablen vorliegt.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Diese Arbeit oder Teile davon sind noch nicht in einem früheren Promotionsverfahren eingereicht worden.

Berlin, den

. . . . . .

(Edgar Steiger)