Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis Econometrics and Operations Research

Analysing the Performance of ℓ_1-Regularised Regression Techniques for the Graphical Modelling of High-Dimensional Binary Data

Eefje Goes (598773)

Supervisor:	D.J.W. Touw
Second assessor:	dr. F. Frasincar
Date final version:	1st July 2024

Abstract

Analysing high-dimensional binary data is crucial in finding relations between variables in various fields, including medical research. Graphical modelling is a useful tool for such analyses, but it poses challenges due to the high-dimensional and binary nature of the data. This paper presents and evaluates alternative methods for constructing graphical models using ℓ_1-regularised regression techniques, namely Lasso and Adaptive Lasso (AL), combined with bootstrap aggregating (bagging). The performance of these methods is tested on simulated data with varying underlying dependence structures and measured using the Structural Hamming Distance (SHD) and the Youden index. Besides that, a sensitivity analysis of one of the parameters is conducted to evaluate its effect on model performance. A comparative analysis highlights the trade-offs associated with the models and their parameter settings, offering insight into their practical suitability for varying data conditions. The findings highlight the importance of context-specific model and parameter choices, balancing certain trade-offs based on the application's needs.

1 Introduction

In the field of medical research, the analysis of data is fundamental in discovering the causes, solutions, and effects of diseases and treatments. However, these medical datasets often present challenges, particularly in the form of high-dimensionality, where a large number of variables is accompanied by a relatively small number of observations. This imbalance can hinder the analysis of finding meaningful associations between variables and can lead to issues such as overfitting, multicollinearity, and model instability (Breiman, 1996b).

Graphical models emerged as a useful tool for handling complex data structures in various fields, including medical research (Tsai & Camargo Jr, 2009). Graphical models (Lauritzen, 1996) are constructed based on the conditional dependence between variables. In these graphs, the nodes represent variables, and their conditional dependent relations are represented by edges, providing a clear visualisation of the underlying structure of the data. However, traditional graphical modelling methods may struggle with overfitting and instability in high-dimensional settings, which calls for the use of ℓ_1-regularised regression techniques (Meinshausen & Bühlmann, 2006).

This paper builds on the work of Strobl, Grill and Mansmann (2012), who adapted these regularised methods for binary data. Strobl et al.'s research serves as the starting point of the research and methods in this study. While some of their methodological approaches are replicated, this paper extends their methods, applies them to various data conditions, and examines the impact of different parameter settings.

This study aims to analyse the performance of various graphical models in estimating the conditional dependency structures of variables within high-dimensional binary data. Specifically, different ℓ_1-regularised regression methods are used, including Lasso, Adaptive Lasso (AL), and their bootstrap aggregated variants. These methods perform variable selection as a result of the penalties they apply, which is useful in high-dimensional settings to discern important relations between variables and prevent overfitting (Tibshirani, 1996; Zou, 2006). This paper investigates the accuracy of the different ℓ_1-regularised methods in estimating true edge sets in various underlying data conditions, the impact of bootstrap aggregating (bagging) on model

performance and consistency, and the effect of different optimisation criteria and parameter settings.

Accurate modelling of high-dimensional binary data is essential for developing reliable predictive tools in the medical field, such as predicting patient outcomes or treatment responses. These models must be able to handle complex dependencies and data features without losing accuracy. As ℓ_1-regularised techniques are useful in high-dimensional settings due to their variable selection, it is essential to analyse how methods like Lasso and AL perform in graphical modelling under these settings. Furthermore, the impact of bagging is of interest as it can improve model accuracy and stability (Breiman, 1996a), which could lead to more reliable results in practical applications.

By comparing Lasso, AL, and their bagged variants, this study seeks to evaluate how these methods perform in different data conditions, including varying levels of dependency strength and sparsity. Applying the models to different underlying data structures will give insight into their robustness and applicability across various realistic situations. Understanding the strengths and weaknesses of each model under these varying conditions provides insight into their practical use and helps with model selection for specific types of data.

This study contributes to the field of graphical model estimation by offering a comprehensive analysis of the Lasso, AL, and their bagged versions under various data conditions. It investigates whether bagging can improve model performance and stability, specifically in high-dimensional and binary settings. This study also investigates the effect of different optimisation criteria and parameter settings and offers practical recommendations for their optimal selection as well as the optimal selection of models based on their characteristics. This paper aims to add insight into how to most effectively use graphical models for high-dimensional, binary data.

The findings underscore the importance of selecting the appropriate regularised regression techniques and tuning parameters to effectively uncover dependency structures. Especially in high-dimensional datasets, these choices affect performance and depend on the application's needs. The findings also highlight the potential advantages of bagging in enhancing model performance under various data conditions.

The rest of this paper is structured as follows: Section 2 reviews relevant literature, providing an overview of previous research and this paper's contribution to the literature. Section 3 details the methodology employed in this study. Section 4 outlines the data-generating process and methods for the different data conditions. Section 5 presents the findings from the analysis, offering insight into the different models and their characteristics. Finally, Section 6 discusses the key findings and implications and provides suggestions for future research.

2 Literature Review

Graphical models have been widely used and studied for many different fields and applications. Initially focusing on Gaussian graphical models (GMMs) for continuous multivariate data with a multivariate normal distribution, the models have evolved to accommodate high-dimensional datasets and binary data, addressing the limitations of traditional statistical methods.

Dempster (1972) introduced covariance selection for GMMs, which refers to the estimation of conditional independence through the inverse covariance matrix. The zero entries of this matrix

indicate conditional independence between variables, allowing for the construction of graphical models. However, his proposed method, which iteratively estimates the inverse covariance matrix using conditional independence tests, is computationally inefficient for high-dimensional data. To address this, Wong, Carter and Kohn (2003) propose a more efficient Bayesian method for Gaussian-distributed data, improving the applicability of GMMs in high-dimensional settings. Despite these advancements, there is a need for methods that extend beyond Gaussian data and are applicable to other data types, such as binary data.

The introduction of the Least Absolute Shrinkage and Selection Operator (Lasso) by Tibshirani (1996) has been a significant contribution to the analysis of high-dimensional data. Lasso's ℓ_1-penalty provides variable selection by shrinking some coefficients to zero, which is valuable for constructing sparse graphical models in high-dimensional contexts. This method has been extended to graphical models by Meinshausen and Bühlmann (2006), who propose using Lasso for neighbourhood selection. The neighbourhood of a variable refers to the set of variables directly connected to it by edges indicating all the direct conditional dependencies. By fitting Lasso regressions for every variable and using all others as predictors, the non-zero entries in the inverse covariance matrix were efficiently and consistently identified, even in high-dimensional settings. This method provides an improvement in computational efficiency over traditional selection techniques.

Further enhancements to the Lasso have been developed to address its limitations and expand its applicability. The Adaptive Lasso (AL), introduced by Zou (2006), is a two-step procedure where a Lasso regression is run with penalty weights determined by the estimated coefficients of an initial regression. This method was created to address the inconsistency of the Lasso in certain situations and its bias in even simple regression settings (Fan & Li, 2001). Unlike the Lasso, the AL possesses oracle properties under certain conditions, meaning it can estimate the true model as if it were known in advance. Zhou, van de Geer and Bühlmann (2009) show that the AL is consistent in modelling high-dimensional GMMs.

Yuan and Lin (2007) develop a penalised-likelihood method for GMMs that employs an ℓ_1-penalty along with constraints for positive definiteness and symmetry of the precision matrix. They propose using the BIC for parameter tuning, a selection criterion also used in this study. Their results highlight the trade-off between sensitivity and specificity, controlled by the choice of parameters. The finding on this trade-off provides motivation to investigate the sensitivity and specificity of the models explored in this paper, along with the effects of parameter settings on these values.

Surprisingly, Friedman, Hastie and Tibshirani (2010a) found that in sparse settings, simple estimation methods perform as well as or even better than more complex methods for estimating the true edge set. This finding highlights the importance of investigating the use of different models and levels of model complexity and their performance.

Another method of describing the conditional relations between variables is to estimate the partial correlation matrix. Krämer, Schäfer and Boulesteix (2009) propose to use the AL to estimate this matrix, alongside Ridge and Partial Least Squares regressions. Although this method of estimation is not used in this paper, it does show the use of ℓ_1-penalties in identifying conditional relations.

All previously mentioned literature apply their models and methods to continuous data. However, Strobl et al. (2012) adapt the method of Meinshausen and Bühlmann (2006) for binary data and investigate the performance of a logistic Lasso and bootstrap aggregated Lasso (Bolasso) on high-dimensional data. Their findings suggest that bootstrap aggregating can improve the performance of graphical models in high-dimensional data. The Bolasso was introduced by Bach (2008), where it is shown that this method provides consistent variable selection and improves upon the variable selection of Lasso. Strobl et al.'s methods provide a foundation for the approaches used in this paper. Furthermore, both Strobl et al.'s and Bach's results motivate the use of bagging in this study to evaluate whether it can improve model consistency under the investigated data conditions.

Additionally, Ravikumar, Wainwright and Lafferty (2010) propose a method of neighbourhood estimation based on ℓ_1-regularised logistic regression. They analyse their method for high-dimensional binary data and find that it succeeds at consistent and simultaneous neighbourhood estimation, reaffirming the potential of ℓ_1-regularised methods for estimating complex dependencies in binary datasets.

While this paper follows the general direction and methods of previous literature, it provides a comprehensive analysis of ℓ_1-regularised regression methods, including Lasso, AL, and their bootstrap aggregated variants in high-dimensional binary data. This study contributes to existing literature by offering insight into the performance of these graphical models in different data conditions and provides practical recommendations for selecting the most effective model and parameter settings.

3 Methodology

In this section, the methodologies employed to analyse the conditional dependencies among high-dimensional binary data using ℓ_1-regularised regression techniques are described. These methodologies are based on the work of Strobl et al. (2012).

3.1 Graphical Models

Graphical models provide a useful tool for presenting and analysing the dependencies among variables. In this paper, the focus is specifically on binary random variables and how their conditional dependencies can be modelled using graph-based methods.

Consider a p-dimensional vector containing binary random variables

$$\mathbf{X} = (X_1, \ldots, X_p),$$

where each $X_i \in \{0, 1\}$ for $i = 1, \ldots, p$. The dependencies between these variables can be displayed in a graph $G = (V, E)$ with $V = \{1, \ldots, p\}$ the set of nodes representing variables, and E the set of edges representing conditionally dependent relations.

Two variables are conditionally independent when their relationship remains unchanged when conditioning on a third variable. Mathematically, variables X and Y are conditionally independent given Z if $P(X|Y, Z) = P(X|Z)$. In other words, Y does not contribute any additional information about X once Z is known. This relation can be written as $X \perp\!\!\!\perp Y \mid Z$.

In the graph, this conditional independence between two variables implies the absence of an edge between the two corresponding nodes. Mathematically, for two variables X_a and X_b:

$$(a, b) \notin E \iff X_a \perp\!\!\!\perp X_b \,|\, X_{V \setminus \{a,b\}},$$

where $a, b \in V$.

The neighbourhood ne_a of a node a then consists of all nodes b that are directly connected to a by an edge. Formally, $ne_a = \{b \in V \setminus \{a\} : (a, b) \in E\}$. This means that X_a is conditionally independent of all remaining variables given those in its neighbourhood ne_a. This can be expressed as:

$$X_a \perp\!\!\!\perp \{X_i : \forall X_i \in V \setminus ne_a\} \,|\, ne_a.$$

To estimate the neighbourhoods of all nodes, ℓ_1-regularised regression techniques are used. With this approach, each variable X_a is regressed on all remaining variables $X_{V \setminus \{a\}}$. The non-zero coefficients in the regression then indicate which variables are conditionally dependent on X_a. Let β_a denote the vector of coefficients obtained from regressing X_a on all the other variables. The neighbourhood of node a can then be defined as

$$ne_a = \{b \in V : \beta_{a,b} \neq 0\}. \tag{1}$$

This means that if $\beta_{a,b}$ is non-zero, there is an edge between nodes a and b, indicating a conditional dependent relation between variables X_a and X_b.

The edge set E is then constructed based on these estimated neighbourhoods. In this paper, only undirected edges are considered, which represent a two-way relation between variables without a specific direction of influence.

Two definitions can be used to set up this undirected edge set based on the neighbourhoods, namely the AND-rule and the OR-rule. The AND-rule specifies that an edge is only present in the edge set when two nodes are in both of the other's neighbourhood:

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

The OR-rule is more lenient and one-sided, where edges are present when at least one of the nodes is in the other's neighbourhood:

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

3.2 Regularised Regression Techniques

Regularised regression methods like Lasso and AL are useful in high-dimensional settings as they can select relevant variables while avoiding overfitting. These methods incorporate ℓ_1-penalties into their regression to shrink some coefficients towards zero, performing variable selection. This subsection explains these methods and discusses their implementation and parameter optimisation.

3.2.1 Lasso

Given a binary variable Y and a set of independent variables X, the logistic regression expresses the log-odds of Y as a linear combination of the predictors:

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_p x_{p,i},$$

where $\pi_i = P(Y_i = 1|X_i)$ denotes the probability that Y_i equals 1 given the predictors, and $(\beta_0, \beta_1, \ldots, \beta_p)$ are the coefficients to be estimated. This can be rewritten as:

$$\pi_i = \frac{\exp(\beta_0 + \beta_1 x_{1,i} + \cdots + \beta_p x_{p,i})}{1 + \exp(\beta_0 + \beta_1 x_{1,i} + \cdots + \beta_p x_{p,i})}.$$

To estimate the parameters, typically the likelihood function is maximised. This is translated to minimising the negative log likelihood function:

$$l(\beta) = -\sum_{i=1}^{N} y_i \log(\pi_i) + (1-y_i)\log(1-\pi_i).$$

Lasso (Tibshirani, 1996) adds a penalty to this equal to the regularisation parameter λ times the absolute values of the coefficients β_j. Thus, to obtain estimates for β, the following is minimised for Lasso:

$$-\sum_{i=1}^{N} y_i \log(\pi_i) + (1-y_i)\log(1-\pi_i) + \lambda \sum_{j=1}^{p} |\beta_j|,$$

where λ is the regularisation parameter that controls the strength of the penalty. As λ increases, more coefficients are driven towards zero, selecting a smaller subset of variables and therefore controlling the size of the neighbourhood. This variable selection of the Lasso makes it suitable for constructing neighbourhoods in graphical models by identifying important relations among variables.

3.2.2 Adaptive Lasso

While Lasso presents a robust tool for variable selection, it may not always yield optimal results, especially when dealing with variable relations that have different levels of strength. Adaptive Lasso (Zou, 2006) addresses this by incorporating weighted ℓ_1-penalties based on the estimated coefficients from an initial regression. These weights are added to the negative log likelihood function, and then β is estimated by minimising

$$-\sum_{i=1}^{N} y_i \log(\pi_i) + (1-y_i)\log(1-\pi_i) + \lambda \sum_{j=1}^{p} w_j |\beta_j|,$$

with w_j the weights defined as $w_j = 1/\left|\hat{\beta}_{j,MLE}\right|^{\gamma}$. These weights use the estimated coefficients $\hat{\beta}_{j,MLE}$ from an initial regression run with MLE, and $\gamma > 0$ adjusts the weights. In this paper, the choice of γ is equal to 1.

3.2.3 Optimal Regularisation Parameter

The performance of regularised regression methods heavily depends on the choice of the regularisation parameter λ. Selecting an optimal λ is crucial to balancing model complexity and goodness-of-fit. This optimisation is done using several methods, including cross-validation (CV), the Akaike information criterion (AIC), and the Bayesian information criterion (BIC). These optimisation techniques have been previously used and suggested in works such as Gao, Pu, Wu and Xu (2009), Strobl et al. (2012), and Yuan and Lin (2007).

Specifically, k-fold cross-validation is used where the data is split into k subsets and subsequently trained and tested on $k-1$ and 1 subset, respectively. This process is repeated k times, where each subset is used as the test set once and the optimal λ is chosen based on the average performance metric across all folds. In this paper, the choice of k is equal to 10. This method provides robust estimates of model performance and helps prevent overfitting.

AIC evaluates the models based on their likelihood and a penalty for the number of parameters. The formula for the AIC is as follows:

$$\text{AIC} = 2k - 2\log(L),$$

where k is the number of parameters and L is the maximised likelihood of the model. AIC balances goodness-of-fit with model complexity.

BIC is similar to AIC but applies a stronger penalty for the number of parameters. It is computed as:

$$\text{BIC} = k\log(n) - 2\log(L),$$

where n is the sample size. BIC places more emphasis on model simplicity, often leading to the selection of more parsimonious models than AIC.

The R package `glmnet` is used for the implementation of these regularised regression methods (Friedman, Hastie & Tibshirani, 2010b)(v4.1-8). This package provides efficient algorithms for fitting Lasso and AL models and includes tools for parameter tuning using CV.

3.3 Bootstrap Aggregating

Bootstrap aggregating, also known as bagging, is a method that aims to enhance model performance by reducing variance and improving accuracy and stability (Breiman, 1996a). It works by creating multiple subsets of the original data by sampling with replacement and running the regression methods on each of these subsets. The final model prediction is then an aggregate of the predictions from the different subsets.

In the context of graphical models, bagging can be used to improve the stability of neighbourhood estimation from the ℓ_1-regularised regression methods. A similar approach to Strobl et al. (2012) is used for bagging to construct neighbourhoods. However, they only use Lasso, and this paper extends their methodology by including AL. The algorithm is as follows:

1. Generate B bootstrap samples from the original data by sampling with replacement. Each sample is the same size as the original dataset, but observations may appear more than once.

2. For every bootstrap sample, fit ℓ_1-regularised regression models for each variable using all others as predictors with CV-optimised λ and obtain estimated coefficients β for each model.

3. For each variable X_a, identify its neighbourhood based on the non-zero coefficients β from the regression. Specifically, for a variable X_a, the neighbourhood is defined as in equation (1) from Section 3.1.

4. Calculate the fraction $\mu_{a,b}$ for each pair of nodes (a, b) representing how often node b is in the neighbourhood of node a across the B bootstrap samples.

5. Define the neighbourhood of node a as $ne_a = \{b : \mu_{a,b} \geq \pi_{cut}\}$, where π_{cut} is a certain cut-off value. This threshold determines the minimum frequency with which a node must appear in the bootstrap neighbourhoods to be included in the final neighbourhood.

6. Using the estimated neighbourhoods as defined above, construct the edge set E as described in Section 3.1. The edges then represent the estimated conditional dependencies between variables.

This process involves two parameters, namely the number of bootstrap samples B and the cut-off value π_{cut}. As Strobl et al. (2012) found that the number of bootstrap samples B has the least amount of impact on model performance with a slight increase in performance with higher B, the only value used for B in this study is 200.

Unlike Strobl et al. (2012), this paper treats π_{cut} as a parameter to be tuned. Due to time constraints, rather than performing cross-validation, the optimal π_{cut} is chosen by iteratively going through π_{cut} values and selecting those that attain the highest Youden index for a given dataset and edge set definition (AND-rule or OR-rule). For this optimisation, all values between 0 and 1 with steps of 0.05 are used as candidates. These same values are used for a sensitivity analysis to evaluate the impact of π_{cut} on model performance.

To implement this bagging algorithm, the R package `boot` is used (Davison & Hinkley, 1997)(v1.3-30).

3.4 Performance Measures

To evaluate the performance of models in identifying the correct structure of a graphical model, the estimated edges are compared to a true edge set constructed based on the known underlying conditional dependence structure. This comparison uses the concepts of positives and negatives, along with their true and false classifications, which are necessary to assess model accuracy.

In the context of graphical models, there are positive and negative edges. Positives are edges that represent a conditionally dependent relation between variables in the true graph. Negatives are absent edges that represent conditional independence between variables.

When estimating these true edges and the underlying dependence structure, there are four possible outcomes:

- **True positive**: an edge that is present in both the true graph and the estimated graph.

- **False positive**: an edge that is incorrectly included in the estimated graph but is not present in the true graph.
- **True negative**: an absent edge that is correctly not present in the estimated graph and the true graph.
- **False negative**: an edge that exists in the true graph but is not included in the estimated graph.

Understanding these outcomes is important in evaluating the accuracy of the estimated edge set compared to the true one. To assess this accuracy, this paper uses the same two performance measures as Strobl et al. (2012), namely the Structural Hamming Distance (Tsamardinos, Brown & Aliferis, 2006) and the Youden index (Youden, 1950).

The Structural Hamming Distance (SHD) measures the number of edge transformations needed to transform the estimated graph into the true graph. These edge transformations include insertions and deletions. Therefore, this measure counts the number of false negatives and false positives, and a lower score is preferred. However, the SHD alone might not provide a complete picture, especially in sparse graphs where there are many absent edges, and models that predict fewer edges still achieve a relatively low SHD by avoiding false positives but miss many true edges.

To complement SHD and get a more balanced evaluation, the Youden index is also used. This index combines sensitivity and specificity to provide a more complete view of model performance.

Sensitivity (true positive rate) is defined as the ratio of positives that are correctly identified by the model. It measures the model's ability to detect true edges and is calculated as follows:

$$\textbf{Sensitivity} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}.$$

Specificity (true negative rate) is the ratio of negatives that are correctly not identified by the model. It measures the model's ability to detect true absent edges and avoid false edges and is calculated as

$$\textbf{Specificity} = \frac{\text{true negatives}}{\text{true negatives} + \text{false positives}}.$$

The Youden Index (J) then combines these two metrics and provides a single measure:

$$J = \text{sensitivity} + \text{specificity} - 1.$$

The Youden index ranges from -1 to 1, where 1 signifies a perfect estimation, 0 suggests the model's performance is not better than random selection, and -1 implies the model performs worse than random selection. The Youden index is valuable as it measures the trade-off between sensitivity and specificity, offering insight into the model's ability to correctly identify both positives and negatives.

4 Data

This section describes the datasets used to evaluate the performance of the ℓ_1-regularised regression models. First, a basic data-generating process (DGP) is set up to examine the models' performance under ideal conditions. Subsequently, a simulation study is conducted to explore the models' robustness and effectiveness in more complex, high-dimensional scenarios with varying levels of sparsity and dependency strength.

4.1 Basic DGP

To evaluate the performance of the different models, they are first applied to a generated dataset with a simple, predefined correlation structure. The aim is to validate the models proposed by Strobl et al. (2012) and to investigate whether the models work as expected under ideal circumstances before applying them to more complex data structures.

At this point, the data is binary but not yet high-dimensional. This simplicity of the correlation and data structure ensures that the nature and accuracy of the models can be observed without the effects of high-dimensionality and more complicated dependency structures.

A correlation matrix is defined to represent the relations among variables. This matrix ensures clear and interpretable conditional relations that form the true underlying graphical model. Based on this predefined correlation matrix, several binary datasets are generated.

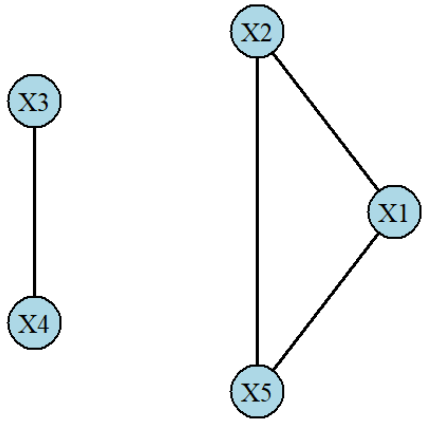

Figure 1: The true graphical model for the basic DGP. This graph is defined by the inverse covariance matrix which is constructed based on the correlation matrix given in Table 2 in Appendix B.

The true graphical model corresponding to the used correlation matrix can be seen in Figure 1. In this graph, the nodes represent the variables, and the edges represent the conditional dependencies between these variables. This is a clear visualisation of the ideal structures the models are expected to estimate.

To evaluate the consistency of the models' performance, 100 datasets are generated from this same correlation matrix, with each dataset containing 5 variables and 1000 observations. With this high ratio of observations to variables, the model performance can be reliably assessed for ideal conditions.

4.2 Simulation Study

To see how the models perform under less ideal conditions and how they react to different dependence structures, a simulation study is conducted. This time, the data is binary and high-dimensional, with varying levels of sparsity and strength of dependencies.

As mentioned earlier, the inverse covariance matrix, often referred to as the precision matrix, is used to identify the conditional independent relations among variables. Given a set of random variables $X_1, \ldots, X_p$, the covariance matrix Σ presents the covariances between variable pairs. The inverse of this matrix, Σ^{-1}, provides insight into conditional dependencies. Specifically, if the (i, j)-th element of Σ^{-1} is zero, then the corresponding variables X_i and X_j are conditionally independent given all other variables. These zero entries in the precision matrix translate to an absence of edges in the graphical model, which is why this matrix is essential in graphical modelling.

To explore how the different models respond to varying dependence structures, datasets are generated based on predefined precision matrices with certain characteristics. These datasets have a relatively small number of observations compared to the number of variables to imitate the often-found high-dimensionality of medical data.

The level of sparsity and the absolute size of the non-zero values in the precision matrix are varied to create different circumstances under which the models are tested. Sparsity refers to the fraction of zero entries in the matrix. A sparse matrix has a larger fraction of zero entries, meaning most variables are conditionally independent of one another, whereas a dense matrix has a smaller fraction of zero entries and therefore more conditionally dependent variables. The absolute value of the non-zero entries can represent the strength of these conditional dependent relations. Low values represent weak relations and higher values indicate strong relations.

The combinations used in this study are sparse-low (few, weak relations) and dense-high (many, strong relations). The setting of a sparse-low precision matrix corresponds to a situation where most variables are conditionally independent of each other, except for a few weak relations. This data condition will test how well the models perform at identifying subtle dependencies amongst much noise and their ability to discern meaningful relations in sparse, high-dimensional datasets. On the other hand, the dense-high setting corresponds to a situation with many, relatively strong conditional dependencies among variables. This data condition studies the models' ability to identify conditional dependencies in high-dimensional datasets where they receive a lot of strong signals.

The correlation matrix is created from the precision matrix once it has been set up, and the binary datasets are subsequently generated using it. The used sparse-low and dense-high precision matrices can be found in Tables 3 and 4 in Appendix B, respectively. Each of the 100 datasets consists of 10 variables and 50 observations, such that they are relatively high-dimensional. At the same time, the relatively small number of variables and observations ensures the computational feasibility of running and analysing all models with different parameter settings.

5 Results

This section presents the results of the analysis following the methodology described in previous sections. First, the performance of the models under ideal conditions is described, then under sparse-low and dense-high conditions, and finally, a sensitivity analysis of the bagging parameter π_{cut} is performed. All models and analyses are implemented in R (v4.3.3).

5.1 Basic DGP

Figure 2 shows the SHD values for the models under the basic DGP, as described in Section 4.1. For each model, the left box represents the results with the AND-rule and the right box with the OR-rule. The thick black lines represent the median SHD value of each model across 100 simulations, and the red line marks the median value across all models and datasets. The bagged variants of the Lasso and AL are denoted as Bolasso and Boal, respectively.

As explained in Section 3.4, SHD is defined as the number of edge transformations needed to get the true edge set from the estimated set. Therefore, an SDH value of 0 represents a perfect estimation, and higher values indicate more discrepancies between the estimated and true edge sets.

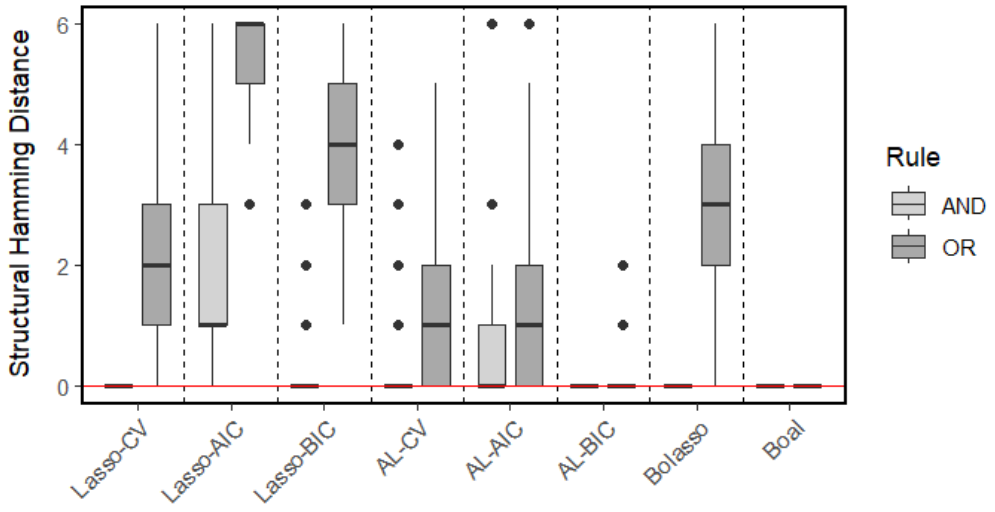

Figure 2: Box plot of the SHD for all models in the basic DGP. The left, light-grey boxes represent the AND-rule and the right, dark-grey boxes represent the OR-rule. The thick black lines are the median values of the individual models, and the red line is the overall median across all models and simulation runs. The first six models are the Lasso and AL models using the different λ optimisation criteria, and the last two are the bagging models with $B = 200$ and optimised π_{cut}.

From Figure 2, it is evident that most models have perfect or near-perfect estimations in this basic setting, especially when the AND-rule is applied. The AL is generally more accurate and consistent than Lasso, both with and without bagging. The Boal models outperform most other models with superior accuracy and consistency, particularly when comparing the models under the OR-rule. AL's better performance over Lasso can be attributed to its adaptive weighting, which provides more robust variable selection. This adaptive weighting also helps AL models maintain better performance under the OR-rule compared to Lasso, which tends to select too many edges due to the one-sided requirement, leading to false positives. In contrast, AL's

adaptive weights filter out weak signals, reducing false positives.

Both Lasso and AL benefit from bagging in terms of consistency, likely due to their more stable variable selection. Among the non-bagging models, the best performance is observed with CV and BIC for optimising λ in Lasso and AL, respectively. The difference in optimal selection criteria between Lasso and AL may be because of their different penalty schemes: AL benefits from a criterion focusing on model complexity (BIC), while Lasso benefits from a criterion minimising prediction errors (CV).

Lasso-AIC is the only model that does not achieve a median SHD of 0 with the AND-rule, indicating it selects too many edges, increasing the SHD and reducing specificity. This could be due to AIC's smaller penalty on model complexity compared to BIC, leading AIC to prefer more parameters and, consequently, more edges. The same trend is observed in AL models, where AIC is less consistent and slightly less accurate under the OR-rule than BIC. For these simple datasets with few variables and dependencies, BIC's stronger penalty yields better performance, though in more complex settings, this penalty could produce a too conservative model.

All bagging models except the Bolasso-OR have perfect estimations in all simulation runs. Table 1 below shows the average tuned π_{cut} values of these models. It can be seen that Boal tends to have lower optimal thresholds than Bolasso, indicating that Boal is better at picking up relevant signals in this basic setting and can provide perfect estimations even with less strict values of π_{cut}. On the other hand, Bolasso-OR requires quite a strict threshold for optimal performance but still shows relatively inaccurate estimations. This model selects too many edges under the one-sided definition of the OR-rule, meaning it often mistakenly picks up irrelevant signals under this setting even with strict threshold values.

Model	**Average π_{cut}**
Bolasso-AND	0.741
Bolasso-OR	0.960
Boal-AND	0.432
Boal-OR	0.629

Table 1: Average tuned π_{cut} values for the basic DGP. The models are the bagging models using the AND and OR- rules.

Across all models, the AND-rule outperforms the OR-rule in terms of accuracy and consistency, as the OR-rule tends to select too many edges, leading to more false positives and a higher SHD. A box plot of the obtained Youden indices can be found in Figure 9 in Appendix B and supports the results discussed above.

5.2 Sparse-Low Conditions

In this subsection, the results found after applying the models to datasets generated from a sparse and low-value precision matrix are presented and discussed.

Figure 3 below presents the SHD values of the models for these sparse-low datasets. Again, the thick black lines are the median SHD values of the individual models, and the red line is the overall median. The light-grey boxes represent the models using the AND-rule, and the dark-grey boxes are the ones using the OR-rule.

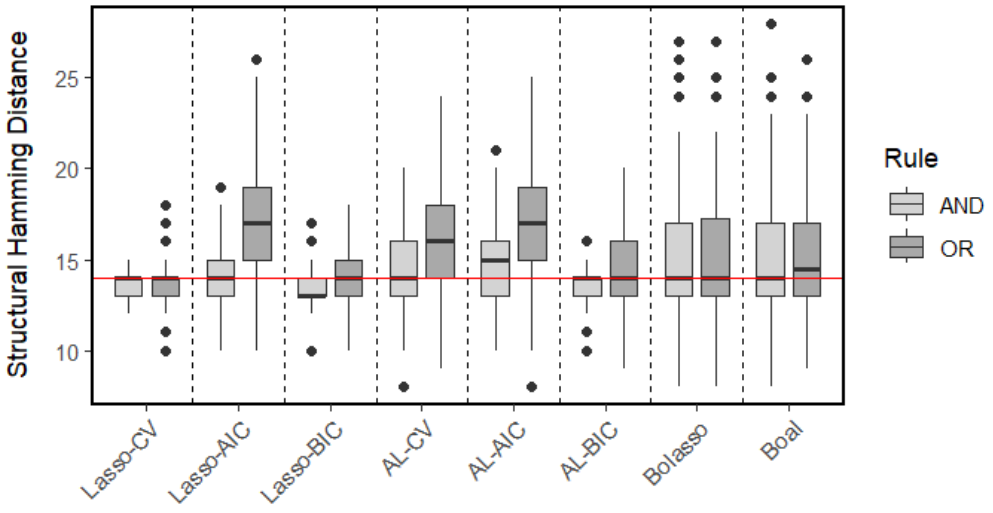

Figure 3: Box plot of the SHD for all models in the sparse-low conditions. The left, light-grey boxes represent the AND-rule and the right, dark-grey boxes represent the OR-rule. The thick black lines are the median values of the individual models, and the red line is the overall median across all models and simulation runs. The first six models are the Lasso and AL models using the different λ optimisation criteria, and the last two are the bagging models with $B = 200$ and optimised π_{cut}.

It can be seen that the bagging models exhibit less consistent performance compared to non-bagging models and generally have slightly less accurate SHD values. This inconsistency could be due to the bagging parameter π_{cut} being optimised based on the Youden index, potentially negatively affecting the SHD scores. Among the non-bagging models, those using AIC as the λ selection criterion have the highest SHD values, especially with the OR-rule. This is likely due to AIC's smaller penalty on model complexity, resulting in the selection of too many edges and more false positives. BIC, with its stricter penalty, performs best for non-bagging models under sparse-low conditions when considering the SHD as a measure.

However, only looking at the SHD might not accurately reflect model performance, especially in sparse matrices where there are few positives and models predicting only negatives can still achieve relatively low SHD values. Thus, the Youden index is also considered to assess how well the models identify true positives. Figure 4 below contains the box plot of the Youden indices. As mentioned in Section 3.4, an index of 1 indicates perfect estimations, whereas with an index of 0, model estimations are no better than random.

In terms of the Youden index, bagging models generally outperform non-bagging models, showing better accuracy and consistency than when using SHD as a measure. Some of the non-bagging models have quite a large spread in Youden indices, with even negative values, indicating worse than random performance. Unlike the basic DGP, there is little difference in performance between the Lasso and AL, except for a somewhat larger spread in values for the AL. The high-dimensional and sparse nature of the data may hinder AL's ability to accurately estimate the weights, making its weighted penalty not much more effective than Lasso's equal penalty in these conditions.

The non-bagging models generally exhibit low sensitivity and high specificity as they select too few edges. The more conservative models (using BIC or AND-rule) maintain lower SHD and high specificity values but miss many true positives, reducing sensitivity. Less conservative

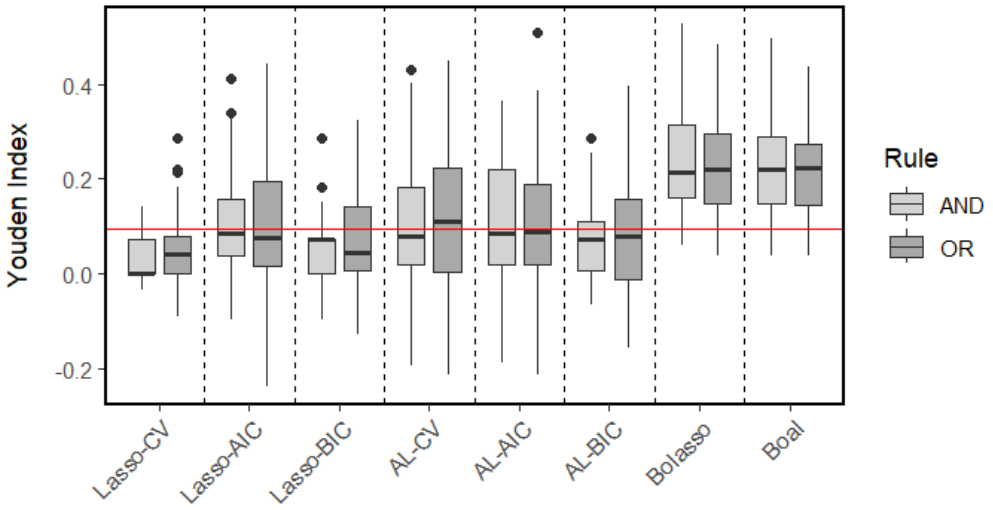

Figure 4: Box plot of the Youden index for all models in the sparse-low conditions. The left, light-grey boxes represent the AND-rule and the right, dark-grey boxes represent the OR-rule. The thick black lines are the median values of the individual models, and the red line is the overall median across all models and simulation runs. The first six models are the Lasso and AL models using the different λ optimisation criteria, and the last two are the bagging models with $B = 200$ and optimised π_{cut}.

models (using AIC or OR-rule) tend to have slightly higher Youden indices due to more true positives but also achieve higher SHD values and introduce more false positives.

The bagging models provide a better balance between the true positive and true negative rates, which might make them the most fitting in this setting. However, if the aim is to minimise false positives, they might not be as appropriate due to their low level of accuracy and consistency when considering the SHD. In this case, the more conservative non-bagging models might be preferred despite their lower sensitivity. The choice of model and optimisation criterion depends on this trade-off between sensitivity, specificity, and consistency, where a choice can be made based on the application's aim and the loss of false positives and negatives.

5.3 Dense-High Conditions

In this subsection, the results found after applying the models to datasets generated from a dense and high-value precision matrix are presented and discussed. Figures 5 and 6 below show the box plots of the SHD values and Youden indices for all models, respectively.

In this setting, bagging models again show less consistent SHD values but now also generally have lower values than non-bagging models, indicating better performance. The difference between Lasso and AL is more pronounced, especially when applying bagging. The non-bagging models employing AIC also have relatively low SHD values due to their less conservative nature and selection of more edges, benefiting them in this dense setting.

From Figure 6, bagging models outperform non-bagging models in terms of Youden indices, showing more accurate performance and more consistency than with SHD as a measure. The lack of accuracy of the AL compared to Lasso could be because of the dense setting, where many positives and high non-zero values cause the AL to over-penalise significant variables, missing relevant relations. Lasso, without bias from weights, distributes these strong signals evenly,

leading to fewer exclusions and higher Youden indices.

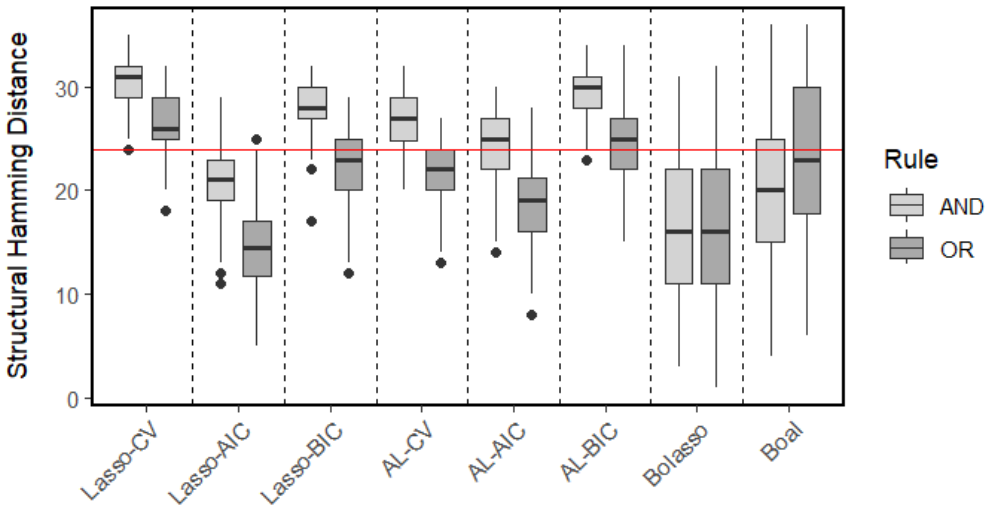

Figure 5: Box plot of the SHD for all models in the dense-high conditions. The left, light-grey boxes represent the AND-rule and the right, dark-grey boxes represent the OR-rule. The thick black lines are the median values of the individual models, and the red line is the overall median across all models and simulation runs. The first six models are the Lasso and AL models using the different λ optimisation criteria, and the last two are the bagging models with $B = 200$ and optimised π_{cut}.

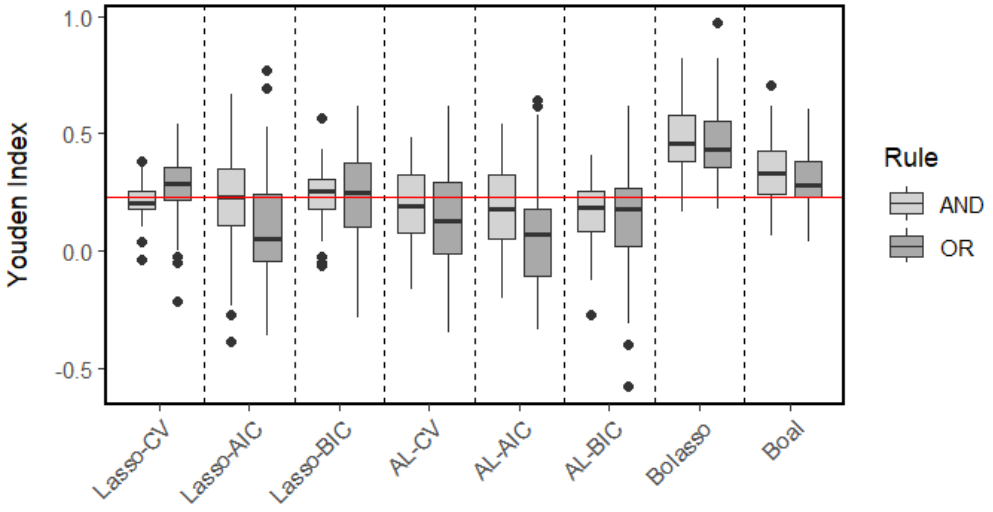

Figure 6: Box plot of the Youden index for all models in the dense-high conditions. The left, light-grey boxes represent the AND-rule and the right, dark-grey boxes represent the OR-rule. The thick black lines are the median values of the individual models, and the red line is the overall median across all models and simulation runs. The first six models are the Lasso and AL models using the different λ optimisation criteria, and the last two are the bagging models with $B = 200$ and optimised π_{cut}.

Considering the non-bagging models, in the previous setting, less conservative non-bagging models gained in Youden index due to the increase in sensitivity being greater than the loss in specificity. This does not always seem to be the case in this dense-high setting, where the use of OR-rule on top of the already less conservative AIC causes a decrease in Youden index. The SHD values for these models remain relatively low due to many edges being selected, but now the gain in sensitivity is smaller than the loss in specificity as too many false positives are

introduced. This difference in trade-off between the sparse and dense settings is because in the sparse setting, the gain in sensitivity is relatively larger, as finding the few true positives there are will boost the model's performance. However, in the dense setting where there are already quite a few true positives, the marginal gain in sensitivity is smaller.

In the dense-high setting, bagging models outperform non-bagging models in both SHD and Youden indices, making them an appropriate choice of model. This suggests that bagging is particularly beneficial in this setting. The choice between Lasso and AL is also more straightforward, with Lasso showing better performance in these data conditions.

5.4 Sensitivity Analysis of π_{cut}

In the previous subsections, the threshold π_{cut} was optimised based on the Youden index. However, this procedure does not give much insight into how different π_{cut} values affect model performance. To investigate this effect, a sensitivity analysis is performed under both sparse-low and dense-high conditions. The π_{cut} values range from 0 to 1 in increments of 0.05, and the Youden index is used to compare performance across values and conditions. SHD is less suitable for this due to it being more dependent on the underlying dependency structure. Figure 7 shows the average Youden indices of the bagging models across 100 simulations at different threshold values. The left graph displays the results under sparse-low conditions and the right graph under dense-high conditions.

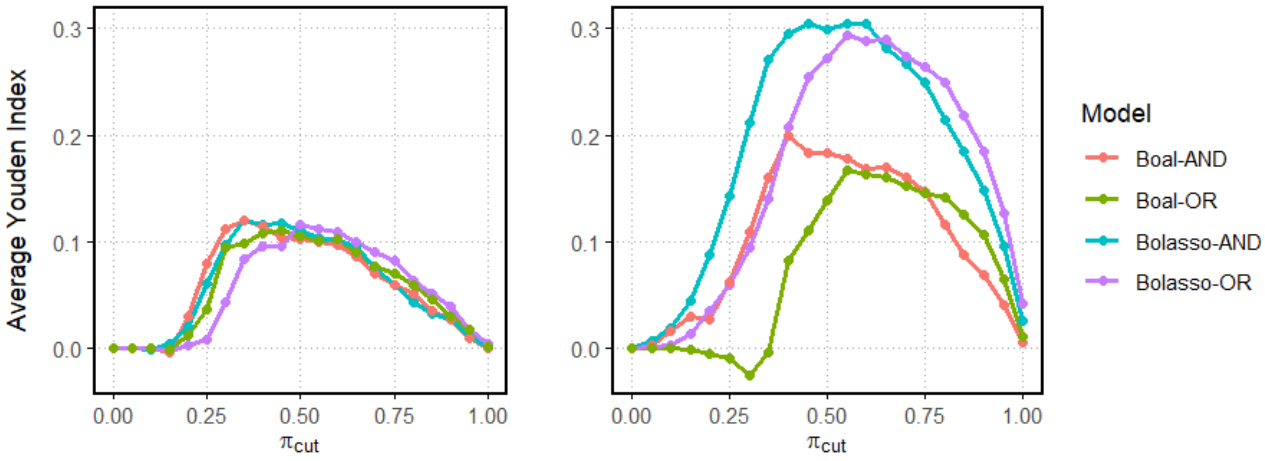

Figure 7: Average Youden indices of all bagging models and π_{cut}'s. The left graph shows the results under sparse-low conditions and the right one under dense-high conditions. The different bagging models and rules are represented by different colours.

Starting from a π_{cut} value of 0, all edges are selected, leading to a sensitivity of 1, a specificity of 0, and a Youden index of 0. As the threshold value increases, estimations become more conservative, with increasing specificity and a slight drop in sensitivity until an optimal trade-off is reached at certain π_{cut} values. Beyond these peaks, models become more conservative, leading to higher specificity but even lower sensitivity, decreasing the Youden index. At $\pi_{cut} = 1.00$, nearly all estimates are negative, and the Youden index is back close to 0.

The bagging models perform better in dense-high conditions across almost all values of π_{cut}, especially Bolasso. In sparse-low conditions, the models struggle to detect the weak, sparse

relations, making them more suited for situations where relations among variables are strong and dense. Another difference is that average optimal π_{cut} values appear lower for sparse-low conditions, peaking before or around 0.50, while for dense-high conditions, peaks are around or slightly above 0.50. This could suggest that weak conditional relations in sparse precision matrices require a lower threshold to still accurately detect these relations. Notably, at their optimal cut-off value, all models achieve higher specificity than sensitivity in most simulation runs, indicating better performance at estimating true negatives compared to true positives.

It also shows that the Bolasso models are more accurate than Boal, specifically in dense-high conditions with moderate values of π_{cut}. For lower threshold values, the AND-rule is preferred, whereas for higher, more strict thresholds, the OR-rule outperforms. In other words, a loose threshold is better accompanied by a strict edge set definition, and vice versa.

Though some models and threshold values attain better performance in terms of the average Youden index, the choice of model and parameter setting still depends on the context. Lower values of π_{cut} have a higher sensitivity but introduce more false positives, while higher values provide better specificity but lead to more false negatives. Model selection depends on these trade-offs and the relative loss of false positives versus false negatives.

Figure 8 shows the Youden indices for the bagging models using a π_{cut} of 0.90 and their optimal values based on the highest average Youden indices as shown in Figure 7. Table 5 in Appendix B presents these optimal values of π_{cut} that are now set beforehand rather than tuning the threshold as done in previous sections. The threshold value of 0.90 was chosen based on the works of Bach (2008) and Strobl et al. (2012), where this value is proposed and used as a soft threshold.

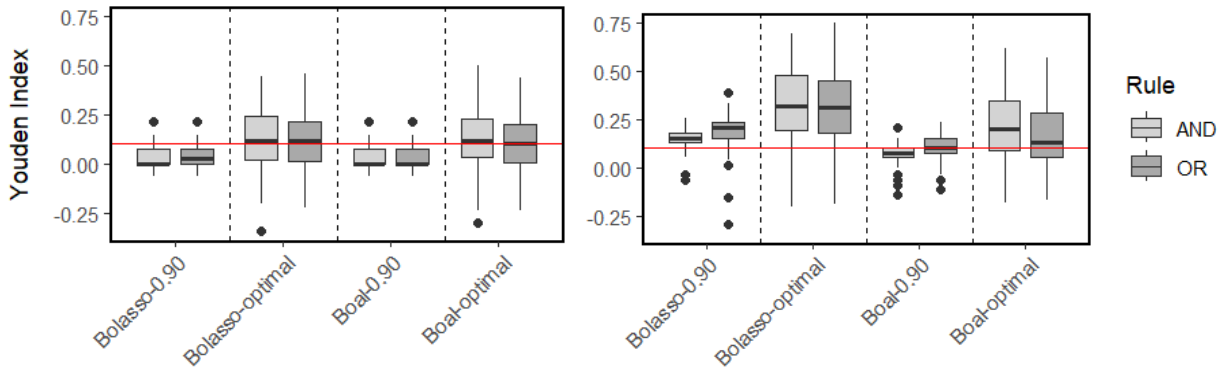

Figure 8: Box plot of the Youden index for the bagging models using 0.90 and optimal thresholds. The left graph shows the results under sparse-low conditions and the right one under dense-high conditions. The left, light-grey boxes represent the AND-rule, and the right, dark-grey boxes represent the OR-rule. The thick black lines are the median values of the individual models, and the red line is the overall median across all models and simulation runs.

This figure reaffirms that the models perform better in dense-high conditions and that Bolasso generally outperforms Boal. However, the optimal thresholds, although providing better average Youden indices, result in less consistent performance, including more negative scores. This variability may be due to lower values being more susceptible to noise and irrelevant signals, leading to mixed performance across simulations. Thus, the choice of cut-off value not only depends on balancing sensitivity and specificity but also on the desired consistency.

6 Conclusion

This study explores the use of ℓ_1-regularised regression techniques in identifying the conditional dependence structure in high-dimensional binary data. By focusing on the performance of different models, including Lasso, Adaptive Lasso, and their bagged variants, under various data conditions and parameter settings, a comprehensive analysis is presented on how these models can be applied. The results give insight into the effectiveness and trade-offs associated with the models, particularly in different data conditions.

Under ideal conditions, the models demonstrate high accuracy, with AL outperforming Lasso and bagging stabilising variable selection. In the sparse-low setting, bagging models show the best balance between sensitivity and specificity. However, if minimising false positives is of importance, more conservative non-bagging models would be more appropriate despite lower sensitivity. There is a trade-off between sensitivity, specificity, and consistency, which is not as present in dense-high conditions. In this setting, the bagging models are more of an obvious choice, with Lasso outperforming AL.

The sensitivity analysis of π_{cut} for the bagging models shows the effect of the threshold value on performance and highlights the trade-off between sensitivity and specificity. Lower thresholds improve sensitivity by identifying more true positives but at the cost of more false positives, whereas higher thresholds provide better specificity but miss many true edges, especially in the sparse-low setting. It was also concluded that the bagging models might be better suited for dense-high conditions and are better at identifying true negatives than true positives.

The results highlight the importance of model selection and parameter tuning according to the context in which they will be applied. Especially in more complex data structures, it was seen that the choice of model and settings is dependent on the application. Based on the losses associated with false negatives and positives, an appropriate choice can be made regarding the model's score on sensitivity versus specificity. Another consideration is the need for consistent performance, in which higher threshold values for the bagging models might be preferred.

Future studies could explore the use of ℓ_1-regularised techniques other than the ones presented in this paper. There have been many adaptations of the Lasso that could potentially work well in detecting the underlying dependence structures. Additionally, for the AL, the effect of different values and optimisation schemes for the parameter γ could be investigated. Another direction of research would be to try to improve performance by combining the strengths of the different models presented here and finding a better trade-off between sensitivity, specificity, and consistency.

This study provides a comparative analysis of various models and their ability to identify conditional dependent relations across different data conditions. The findings highlight the necessity of a context-specific choice of model and parameter settings and provide useful insights for both practical applications and further research.

References

Bach, F. R. (2008). Bolasso: model consistent lasso estimation through the bootstrap. In *Proceedings of the 25th international conference on machine learning* (pp. 33–40).

Breiman, L. (1996a). Bagging predictors. *Machine learning*, *24*, 123–140.

Breiman, L. (1996b). Heuristics of instability and stabilization in model selection. *The annals of statistics*, *24*(6), 2350–2383.

Davison, A. C. & Hinkley, D. V. (1997). *Bootstrap methods and their application* (No. 1). Cambridge university press.

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

Fan, J. & Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, *96*(456), 1348–1360.

Friedman, J., Hastie, T. & Tibshirani, R. (2010a). *Applications of the lasso and grouped lasso to the estimation of sparse graphical models* (Tech. Rep.). Technical report, Stanford University.

Friedman, J., Hastie, T. & Tibshirani, R. (2010b). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, *33*(1), 1.

Gao, X., Pu, D. Q., Wu, Y. & Xu, H. (2009). Tuning parameter selection for penalized likelihood estimation of inverse covariance matrix. *arXiv preprint arXiv:0909.0934*.

Krämer, N., Schäfer, J. & Boulesteix, A.-L. (2009). Regularized estimation of large-scale gene association networks using graphical gaussian models. *BMC bioinformatics*, *10*, 1–24.

Lauritzen, S. L. (1996). *Graphical models* (Vol. 17). Clarendon Press.

Meinshausen, N. & Bühlmann, P. (2006). High-dimensional graphs and variable selection with the lasso.

Ravikumar, P., Wainwright, M. J. & Lafferty, J. D. (2010). High-dimensional ising model selection using 1-regularized logistic regression.

Strobl, R., Grill, E. & Mansmann, U. (2012). Graphical modeling of binary data using the lasso: a simulation study. *BMC medical research methodology*, *12*, 1–13.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *58*(1), 267–288.

Tsai, C.-L. & Camargo Jr, C. A. (2009). Methodological considerations, such as directed acyclic graphs, for studying "acute on chronic" disease epidemiology: chronic obstructive pulmonary disease example. *Journal of clinical epidemiology*, *62*(9), 982–990.

Tsamardinos, I., Brown, L. E. & Aliferis, C. F. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, *65*, 31–78.

Wong, F., Carter, C. K. & Kohn, R. (2003). Efficient estimation of covariance selection models. *Biometrika*, *90*(4), 809–830.

Youden, W. J. (1950). Index for rating diagnostic tests. *Cancer*, *3*(1), 32–35.

Yuan, M. & Lin, Y. (2007). Model selection and estimation in the gaussian graphical model. *Biometrika*, *94*(1), 19–35.

Zhou, S., van de Geer, S. & Bühlmann, P. (2009). Adaptive lasso for high dimensional regression and gaussian graphical modeling. *arXiv preprint arXiv:0903.2515*.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American statistical*

association, *101*(476), 1418–1429.

A Programming code

This section gives an overview of the steps and functions implemented in R to obtain the results. First, the algorithm for the non-bagging models:

1. Define functions to calculate the performance measures, AIC, and BIC. These functions are detailed in Algorithms 1, 2, and 3, respectively.

2. Set parameters including the number of variables, level of sparsity, range of precision matrix values, number of observations of each dataset, number of datasets, and marginal probabilities of binary variables.

3. Create the precision matrix based on the parameters from the previous step. This matrix is used to set up the true edge set and the correlation matrix for data generation.

4. For the number of datasets, do the following:

    4.1 Generate the binary variables based on certain parameters from Step 2 and the correlation matrix from Step 3.

    4.2 For every variable, do the following:

        4.2.1 Run an ℓ_1-regularised regression (Lasso or AL) using all others as predictors and one of the three λ optimisation criteria: CV, AIC, or BIC. This is done using the `glmnet()` and `cv.glmnet()` functions from the R-package `glmnet`.

        4.2.2 Identify and store the non-zero coefficients, as these represent dependent relations.

    4.3 Create the edge set based on these non-zero coefficients and one of the edge set definitions (AND/OR) as defined in Section 3.1.

    4.4 Calculate the performance measures using the function defined in Step 1, the true edge set from Step 3, and the estimated edge set from Step 4.3.

5. Compute and display the (average) results.

The algorithm for the bagging models:

1. Define functions to calculate the performance measures, AIC, and BIC. These functions are detailed in Algorithms 1, 2, and 3, respectively.

2. Set parameters the including number of variables, level of sparsity, range of precision matrix values, number of observations of each dataset, number of datasets, marginal probabilities of binary variables, number of bootstrap samples, and threshold value candidates.

3. Create the precision matrix based on the parameters from the previous step. This matrix is used to set up the true edge set and the correlation matrix for data generation.

4. For the number of datasets, do the following:

    4.1 Generate the binary variables based on certain parameters from Step 2 and the correlation matrix from Step 3.

4.2 Define an ℓ_1-regularised regression function (Lasso or AL) for bagging where all other variables are used as predictors and λ is optimised using one of the three criteria: CV, AIC, or BIC. This is done using the `glmnet()` and `cv.glmnet()` functions from the R-package `glmnet`.

4.3 Generate the bootstrap samples and apply the function from Step 4.2. This is done using the `boot()` function from the R-package `boot`.

4.4 Calculate the neighbourhood percentages representing how often nodes are in each other's neighbourhood across all bootstrap samples.

5. Define and run a function for the optimisation of the threshold value where, for every dataset, the optimal value is found by iteratively going through the set threshold value candidates from Step 2 and selecting those that achieve the highest Youden index based on the given edge set definition (AND/OR). This function is detailed in Algorithm 4.

6. Compute and display the final (average) results using the optimised threshold values from Step 5.

Below are some of the functions used in the steps described above. First is the function to calculate the performance measures described in Section 3.4.

Algorithm 1: Calculate performance measures

Input: True edge set matrix, *true_edge_set*; Estimated edge set matrix, *edge_set*

Output: SHD, Sensitivity, Specificity, Youden index

`/* Extract upper triangles to avoid counting double */`

$true\_upper \leftarrow true\_edge\_set[\text{upper.tri}(true\_edge\_set)]$;

$estimated\_upper \leftarrow edge\_set[\text{upper.tri}(edge\_set)]$;

`/* Compute SHD */`

$SHD \leftarrow \text{sum}(true\_upper \neq estimated\_upper)$;

`/* Calculate true/false positives/negatives */`

$TP \leftarrow \text{sum}(true\_upper \wedge estimated\_upper)$;

$FP \leftarrow \text{sum}(\neg true\_upper \wedge estimated\_upper)$;

$TN \leftarrow \text{sum}(\neg true\_upper \wedge \neg estimated\_upper)$;

$FN \leftarrow \text{sum}(true\_upper \wedge \neg estimated\_upper)$;

`/* Compute sensitivity, specificity, and the Youden index */`

Sensitivity $\leftarrow \frac{TP}{TP+FN}$;

Specificity $\leftarrow \frac{TN}{TN+FP}$;

$J \leftarrow \text{Sensitivity} + \text{Specificity} - 1$;

return SHD, Sensitivity, Specificity, and J;

Next are the two functions to calculate the AIC and BIC scores from an ℓ_1-regularised regression fit.

Algorithm 2: Calculate AIC (Akaike Information Criterion)

Input: ℓ_1-regularised model fit, fit

Output: AIC value

`/* Extract negative log likelihood, number of estimated parameters, and number of observations */`

$tLL \leftarrow -\text{deviance}(fit)$;

$k \leftarrow fit.df$;

$n \leftarrow fit.nobs$;

`/* Compute AIC */`

$AIC \leftarrow -tLL + 2 \times k$;

return AIC;

Algorithm 3: Calculate BIC (Bayesian Information Criterion)

Input: ℓ_1-regularised model fit, fit

Output: BIC value

`/* Extract negative log likelihood, number of estimated parameters, and number of observations */`

$tLL \leftarrow -\text{deviance}(fit)$;

$k \leftarrow fit.df$;

$n \leftarrow fit.nobs$;

`/* Compute BIC */`

$BIC \leftarrow \log(n) \times k - tLL$;

return BIC;

Lastly, the function used to optimise the bagging parameter π_{cut} based on the Youden index.

Algorithm 4: Optimise π_{cut} based on the Youden index

Input: Rule for edge set definition ("AND" or "OR"), *rule*; true edge set, *true_edge_set*; list of results containing neighborhood percentages, *results*; set of π_{cut} value candidates, *pi_cuts*

Output: List of optimal π_{cut} values and corresponding best performance, $(optimal\_pi\_cut, best\_performance)$

`/* Initialise vectors to store output */`

$optimal\_pi\_cut \leftarrow \text{numeric}(n\_datasets)$;

$best\_performance \leftarrow \text{numeric}(n\_datasets)$;

for $i \leftarrow 1$ **to** $n\_datasets$ **do**

`/* Extract neighbourhood percentage and number of variables, initialize the best metric */`

$neighbourhood\_percentage \leftarrow results[[i]]\$neighbourhood\_percentage$;

$n \leftarrow \text{ncol}(neighbourhood\_percentage)$;

$best\_metric \leftarrow -\infty$;

for $pi\_cut \in pi\_cuts$ **do**

$edge\_set \leftarrow \text{matrix}(0, n, n)$;

for $j \leftarrow 1$ **to** n **do**

`/* Identify neighbours and edge set */`

$neighbours \leftarrow \text{which}(neighbourhood\_percentage[j,] \geq pi\_cut)$;

$edge\_set[j, neighbours] \leftarrow 1$;

end

`/* Create final edge set based on given rule */`

if *rule is "OR"* **then**

$final\_edge\_set \leftarrow edge\_set + \text{t}(edge\_set)$;

$final\_edge\_set[final\_edge\_set > 1] \leftarrow 1$;

else

$final\_edge\_set \leftarrow edge\_set \times \text{t}(edge\_set)$;

end

`/* Calculate performance measures using final edge set and function as given in Algorithm 1 */`

$measures \leftarrow calculate\_performance\_measures(true\_edge\_set, final\_edge\_set)$;

if *Youden index greater than best_metric* **then**

$best\_metric \leftarrow measures\$J$;

$optimal\_pi\_cut[i] \leftarrow pi\_cut$;

end

end

$best\_performance[i] \leftarrow best\_metric$;

end

return $optimal\_pi\_cut, best\_performance$

B Tables and figures

	X1	X2	X3	X4	X5
X1	1	0.5	0	0	0.5
X2	0.5	1	0	0	0
X3	0	0	1	0.5	0
X4	0	0	0.5	1	0
X5	0.5	0	0	0	1

Table 2: The correlation matrix for the ideal data conditions. Based on this matrix the datasets for the simple DGP are generated.

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
X1	1.4800	-0.0460	0.3117	0.0994	-0.1131	0.0000	0.0000	-0.0982	0.0000	0.0000
X2	-0.0460	1.4800	0.0000	0.0000	-0.1112	0.0000	-0.1005	0.0000	0.1403	0.0000
X3	0.3117	0.0000	1.4800	0.0000	0.0000	0.0000	-0.0666	0.0000	0.0000	0.0000
X4	0.0994	0.0000	0.0000	1.4800	0.0000	0.0000	0.1108	-0.0124	0.0000	0.0000
X5	-0.1131	-0.1112	0.0000	0.0000	1.4800	0.0000	-0.0806	0.0000	0.0000	0.0000
X6	0.0000	0.0000	0.0000	0.0000	0.0000	1.4800	0.0000	0.0000	0.0000	-0.1201
X7	0.0000	-0.1005	-0.0666	0.1108	-0.0806	0.0000	1.4800	0.0000	0.4038	0.0000
X8	-0.0982	0.0000	0.0000	-0.0124	0.0000	0.0000	0.0000	1.4800	0.0000	0.0000
X9	0.0000	0.1403	0.0000	0.0000	0.0000	0.0000	0.4038	0.0000	1.4800	0.0000
X10	0.0000	0.0000	0.0000	0.0000	0.0000	-0.1201	0.0000	0.0000	0.0000	1.4800

Table 3: The sparse-low precision matrix. This matrix defines the true dependency structure the models aim to uncover where the non-zero entries represent conditionally dependent relations. Based on this matrix the datasets for the sparse-low conditions are generated.

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
X1	20.8320	-1.5000	1.2272	3.0000	-0.2368	5.7877	3.0000	-4.6627	-0.2171	0.9838
X2	-1.5000	20.8320	1.5000	0.0000	-1.5000	0.0000	0.0000	-10.6895	3.2534	1.5000
X3	1.2272	1.5000	20.8320	0.4156	-4.7261	-5.7167	0.5945	4.0143	-4.3453	1.5000
X4	3.0000	0.0000	0.4156	20.8320	7.4763	0.0000	-1.0464	-3.2730	-1.5000	-1.5000
X5	-0.2368	-1.5000	-4.7261	7.4763	20.8320	-2.8078	-4.1795	0.0000	-1.5000	1.9016
X6	5.7877	0.0000	-5.7167	0.0000	-2.8078	20.8320	-2.3719	4.0639	1.5000	-0.0006
X7	3.0000	0.0000	0.5945	-1.0464	-4.1795	-2.3719	20.8320	0.0000	5.0478	10.5935
X8	-4.6627	-10.6895	4.0143	-3.2730	0.0000	4.0639	0.0000	20.8320	1.5000	2.3315
X9	-0.2171	3.2534	-4.3453	-1.5000	-1.5000	1.5000	5.0478	1.5000	20.8320	1.3720
X10	0.9838	1.5000	1.5000	-1.5000	1.9016	-0.0006	10.5935	2.3315	1.3720	20.8320

Table 4: The dense-high precision matrix. This matrix defines the true dependency structure the models aim to uncover where the non-zero entries represent conditionally dependent relations. Based on this matrix the datasets for the dense-high conditions are generated.

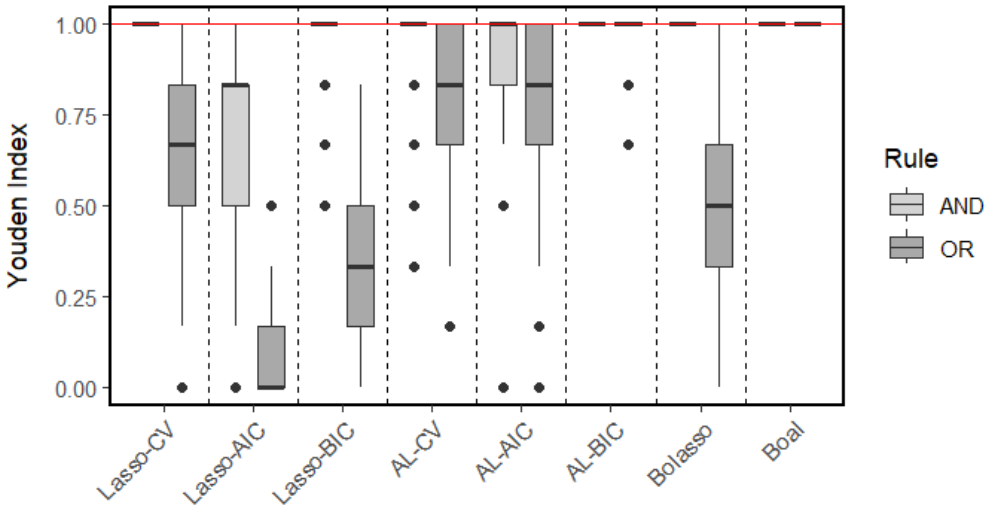

Figure 9: Box plot of the Youden index for all models in the basic DGP. The left, light-grey boxes represent the AND-rule and the right, dark-grey boxes the OR-rule. The thick black lines are the median values of the individual models and the red line is the overall median across all models and simulation runs. The first six models are the Lasso and AL models using the different λ optimisation criteria, and the last two are the bagging models with $B = 200$ and optimised π_{cut}.

sparse-low	**model**	**optimal π_{cut}**
	Bolasso-AND	0.35
	Bolasso-OR	0.50
	Boal-AND	0.35
	Boal-OR	0.45
dense-high	**model**	**optimal π_{cut}**
	Bolasso-AND	0.45
	Bolasso-OR	0.55
	Boal-AND	0.40
	Boal-OR	0.55

Table 5: Optimal π_{cut} values for all bagging models and data conditions. These optimal values are based on the highest average Youden index across simulation runs for π_{cut} values ranging from 0 to 1 with increments of 0.05.