Covariance Selection using Node-wise and Graphical Lasso

1 Introduction

Given a set of p random variables, $X = (X_1, \dots, X_p)^T$. Two variables X_j and X_l, $j, l \in \{1, \dots, p\}$, are said to be *conditionally dependent* if

$$c_{jl} = Cov(X_j, X_l | X_k : 1 \leq k \leq p, k \neq l, j) \neq 0$$

A common and convenient (Edwards, 2000; Koller and Friedman, 2009) description of the conditional dependence structure of X is through a graph $G = (V, E)$, where $V = \{1, \dots, p\}$, a set of vertices where each vertex corresponds to one variable and the edge set $E = \{(j, l) : 1 \leq j < l \leq p \;\&\; c_{jl} \neq 0\}$. Thus, an edge (j, l) is only in the graph G if j and l are dependent when conditoned on all the other random variables in X.

The problem of recovering the edge set E of G is very useful and extensively studied in statistics (Edwards, 2000). The construction of the graph can be used to perform dimensionality reduction and causal inference. The graphical model developed is also useful in several fields, especially in biology. Probabilistic graphical models have been used to great effect in genetic counseling, prediction and linkage analysis (Koller and Friedman, 2009). The wide-ranging applicability of the probabilistic graphical models makes their construction very important.

In this report, a special case of the problem is focussed upon, where X is a multivariate normally distributed variable with mean μ with a positive-definite covariance matrix Σ, that is $X \sim \mathcal{N}(\mu, \Sigma)$. In this case the estimation of the precision or concentration matrix $\Theta = \Sigma^{-1}$ is vital, as it can be shown that $c_{jl} = 0$ if and only if $\Theta_{jl} = 0$ (Dempster, 1972), where Θ_{jl} is the j, lth entry of Θ. In this report, Θ is assumed to be sparse, that is it contains many entries which are zero. Several approaches have been developed to tackle this case. The most naive approach to recover E is to go through every possible edge set, and see which one fits the best. Such an exhaustive search is practically infeasible except in problems where p is very small (Meinshausen and Bühlmann, 2006), as the total possible edge sets are $2^{\frac{p*(p-1)}{2}}$. This led to the usage of forward-selection and backward-deletion (Dempster, 1972), but these also suffer from high computational complexity. Hence, more efficient methods were developed, which exploit the normality of the variables and the properties of the inverse covariance matrix.

Meinshausen and Bühlmann (2006) use the lasso developed by Tibshirani (1996) on each node to construct a neighbourhood, then use the estimated neighbourhoods to generate an edge set and use the lasso coefficients to estimate conditional dependencies. This method is called the node-wise lasso, and has succesfully been used for different purposes, like for estimating large portfolios (Callot et al., 2019). However, this method requires a two step approach to estimating the entries of the precision matrix. Buhl (1993) introduces conditions under which maximum likelihood estimator for the covariance matrix exists and can be used to estimate the precision matrix for Gaussian graphical models. Using the maximum likelihood estimator, along with a penalized approach on off diagonal entries similar to the lasso, Yuan and Lin (2007); Dahl et al. (2008); Banerjee et al. (2008); Friedman et al. (2007) developed methods to estimate the edge set and the precision matrix simultaneously. The relatively computationally fast method developed by Friedman et al. (2007) is called the graphical lasso, and uses blockwise coordinate descent to solve the convex optimization problem of finding the penalized maximum likelihood. Many variations of this algorithm have been made to make it faster in general and for specific problems. More recently, Kovács et al. (2021) developed an elastic penalty for solving the general problem of retrieving a Gaussian graphical model from data, and have provided statistical packages to do so.

The aim of this report is to compare the efficiency of the node-wise lasso developed by Meinshausen and Bühlmann (2006) and the graphical lasso by Friedman et al. (2007) in retrieving the edge sets in various simulated settings. Further, the strength of the two approaches when the assumption of multivariate normality no longer holds is tested.

Section 2 expands upon the simulation settings and the variables used to determine it. Section 3 expands upon the node-wise lasso, explaining the general idea, optimal tuning parameter selection and hypothesizing expected results. Section 4 details the graphical lasso, and discusses how the optimal tuning parameter is selected for this method. In Section 5, the two approaches are compared under varying sample sizes, dimensionalities and sparsity structures along with testing the assumption of normality. Finally, the report is concluded in Section 6.

2 Creation of Simulated Datasets

In order to test the two methods, simulated data sets are required to apply them upon. As the methods are defined for multivariate normally distributed variables, a mean μ and a covariance matrix Σ are required. As the conditional dependence can directly be linked to the precision matrix, in this report precision matrices are simulated. This is done as follows.

Let $n > 0$ denote the sample size of the simulated data, $p > 0$ the number of number of normal variables (parameters, features), $s_t \in [0, 1]$ the sparsity threshold and $\delta > 0$ the variance determining coefficient. Using these four parameters, a multivariate normal sample of size n of p variables is created, with conditional dependencies whose magnitudes are controlled by δ, and the number of conditionally dependent variables is controlled by s_t.

The procedure to make this sample is as follows. $\frac{p*(p-1)}{2}$ numbers are drawn from a uniform[0,1] distribution, each corresponding to a chance that there is a conditional dependence between two variables. If a number drawn is greater than s_t, then a value of 0.5 is attached to the corresponding entries of the inverse covariance matrix. In this way, increasing s_t increases the sparsity in Θ, and can be considered the probability that two variables are conditionally independent of each other. Then, the diagonal entries of the simulated Θ are set to δ, and then all the entries of the matrix are divided by delta, with delta large enough to ensure positive definiteness. (Taking $\delta > 2$ ensures positive definiteness of the inverse covariance matrix in this case Meinshausen and Bühlmann (2006) as the absolute value of the partial correlation between neighbours becomes less than 0.25, however the implementation used generally uses a smaller delta and explicitly tests for positive definiteness of Θ). Θ is then inverted and $\Sigma = \Theta^{-1}$ is used as the covariance matrix of the random variables. Then n observations of p multivariate normal random variables with mean 0 and covariance Σ are generated. The mean can be set to zero without loss of generality, as the variables are scaled prior to applying the node-wise lasso, and the sample covariance matrix, which is relevant to the graphical lasso, also looks at demeaned deviations.

3 Node-Wise Lasso

This technique was developed by Meinshausen and Bühlmann (2006), and is essentially an approximation to relatively quickly find the edge set $\hat{E}$. The idea is to estimate the neighbourhood ne_a for each vertex $a \in V$, by using the lasso (Tibshirani, 1996). The neighbourhood of a vertex a is defined to be the minimum subset of vertices in V such that given these, a is conditionally independent of all the other vertices. To estimate the neighbourhood, p lasso regressions of the following type are used to find coefficient estimates for each variable. Here, β_{ab} refers to the entry of vector β_a which corresponds to the random variable X_b.

$$\beta_a^\lambda = \underset{\beta_a : \beta_{aa}=0}{\arg\min}\, n^{-1}\|X_a - X\beta_a\|_2^2 + \lambda\|\beta_a\|_1$$

Using the lasso ensures that some of the coefficient estimates are exactly zero. Once the β_a^λ is obtained for each $a \in V$, first the neighbourhood of a is estimated as follows:

$$\hat{ne_a}^\lambda = \{v \in V|\ \beta_{av}^\lambda \neq 0\}$$

The size of the neighbourhood can be controlled by the tuning parameter λ. High values of λ correspond to smaller neighbourhoods. In general, λ is allowed to vary when constructing neighbourhoods for different variables. Once the neighbourhoods are constructed for every variable, edge sets can be constructed using the following two rules:

$$\hat{E}_{and}^\lambda = \{(j,l) : j \in \hat{ne_l}^\lambda \textbf{ and } l \in \hat{ne_j}^\lambda\}$$
$$\hat{E}_{or}^\lambda = \{(j,l) : j \in \hat{ne_l}^\lambda \textbf{ or } l \in \hat{ne_j}^\lambda\}$$

Meinshausen and Bühlmann (2006) show that these two rules converge to the same edge set asymptotically. They also describe a set of assumptions, which can be satisfied using the correct parameters in the simulation setting in Section 2 (and in fact are a bit more general, allowing for the number of parameters to vary with the sample size). Under these conditions, with a certain value for the tuning parameter lambda, the neighbourhood estimates are consistent for the true neighbourhood, and the both the estimated edge sets converge in probability to the true edge set as the sample size grows to infinity. This can be seen in Table 1, which details the area under the receiver operating characteristics (AUROC) for different sample sizes and sparsity patterns for data containing 100 variables. The higher s_t is, the more sparse the precision matrix. Note that the receiver operating characteristic (ROC) curves such as those in Fig. 1 were produced by repeatedly performing the node-wise lasso at different levels of λ and plotting the true positive rate (number of edges correctly predicted/total edges) against the false positive rate (number of edges falsely predicted/number of pairs of conditionally independent variables).

Sample Size	s_t	AUROC (and)	AUROC (or)
50	0.9	0.2158	0.5177
100	0.9	0.7117	0.8323
500	0.9	0.9540	0.9661
500	0.6	0.7572	0.7500
500	0.1	0.0307	0.0335

Table 1: AUROC for node-wise lasso under varying sample size, $p = 100$.

The patterns in Table 1 are consistent with Meinshausen and Bühlmann (2006), as when the sample size increases, the model becomes more accurate at recovering the edge set. However, as the sparsity of the graph decreases, the node-wise lasso fails to provide reasonable predictions of the true edge set. Hence, the assumption of sparsity is crucial to the functioning of the node-wise lasso. This will be reiterated in Section 5. Also, under sparse conditions, the or rule tends to give better results. However, as the sample size grows, the results from the two constructions tend to be similar. Moreover, the and rule controls false positives better by construction. Hence, both the rules situationally have merit. Of course, when faced with real data, one would have to choose the λ, and that can change the results of the method significantly. However, it is not clear yet as to how to choose the λ in finite samples.

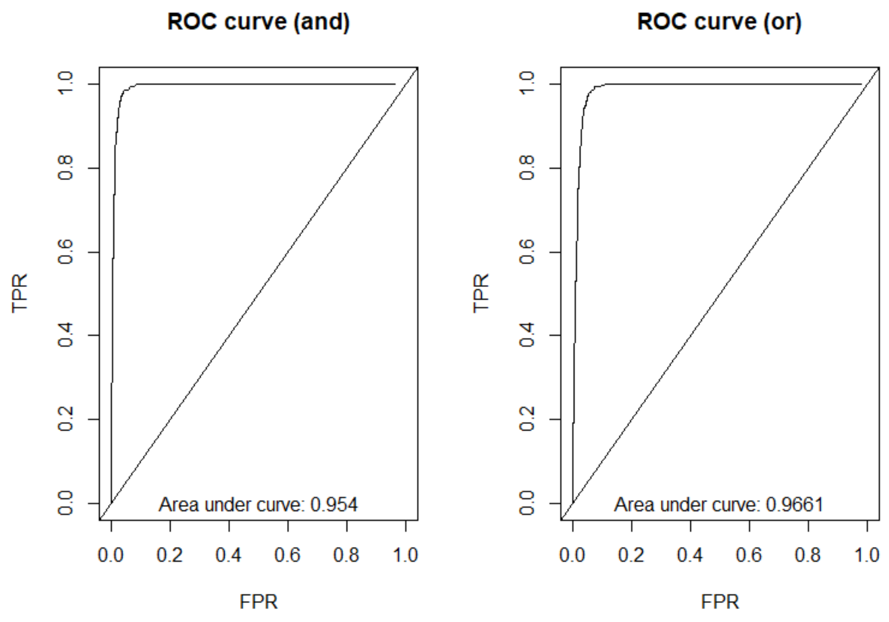

Figure 1: Node-wise Lasso ROC curve for n=500, p = 100, $s_t = 0.9$.

3.1 Selecting λ

Typically, in lasso estimates, cross-validation is used to select the optimal tuning parameter. One can split the data into a training and a test set, or use a K-fold cross validation for each of the p regressions in the node-wise lasso, and then pick the λ which minimises the test mean squared error or the estimated cross-validation mean squared error from K-fold cross validation. However, in the case of K-fold cross validation, this requires fitting $p * K * n_\lambda$ models, where n_λ is the number of lambdas to cross validate with. This can be computationally quite expensive. Moreover, Meinshausen and Bühlmann (2006) show that the estimated λ from cross-validation ensures that asymptotically the wrong neighbourhoods are chosen for each vertex.

Instead for most of the simulation, the Meinshausen-Bühlmann penalty (MBpen) is used. Using this penalty controls the probability of spuriously connecting distinct connectivity components of the real graph (Meinshausen and Bühlmann, 2006). The connectivity component for a given vertex a in a graph $G = (V, E)$ is the subset of V which are connected to a using the edges in E. The MBpen is given by:

$$\lambda(\alpha, p, n, a) = \frac{2 * \hat{\sigma}_a}{\sqrt{n}} * \left(1 - \Phi^{-1}\left(\frac{\alpha}{2 * p^2}\right)\right)$$

where Φ is the cumulative distribution function of a $\mathcal{N}(0,1)$ variable, $\hat{\sigma}_a^2$ is the variance of X_a, n is the sample size, and p is the number of variables. α controls the probability of connecting two distinct connectivity components wrongly. Unless explicitly stated, whenever MBpen is used in this report, alpha is set to 5%. In the simulations performed in this report, MBpen tends to work better and is much faster than cross-validation. Table 2 reports the mean misclassifaction error rate and mean true positive rate (TPR) of selecting lambda by 5-fold cross validation versus using MBpen repeating each procedure 50 times. This is done over varying sample sizes. The and rule is used to construct edge sets. Similar results are found by the or rule, with higher true positive rates and higher error rates, as should be expected.

Sample Size	TPR (5-fold CV)	TPR (MBpen)	Error (5-fold CV)	Error (MBpen)
50	13.2%	38.31%	9.44%	11.1%
100	37.7%	60.6%	7.86%	9.34 %
500	91.9%	96.8%	2.59%	5.52%

Table 2: K-fold cross validation λ selection versus MBpen, $p = 100$.

Table 2 shows that although the overall misclassification error rate is smaller when λ is chosen by cross-validation, the true positive rate is always lower for the 5-fold cross validation, especially when the sample size is closer to the number of parameters. However, as n grows, the true positive rate for both methods goes to 1. Hence, both methods are viable when estimating sparse precision matrices given enough data. MBpen is significantly computationally faster however, as each penalized regression is done for only once and for only one value of λ. It also tends to predict more true positives, which is vital here.

Note the added problem of choosing α and K. Both of these are generally chosen heuristically. Also, more methods exist to choose the regularization parameter here, for instance Callot et al. (2019) use a λ which minimizes the generalized information criterion of Fan and Tang (2013) for each penalized regression.

4 Graphical Lasso

The graphical lasso (glasso) algorithm was developed by Friedman et al. (2007), and focusses on using blockwise coordinate descent to maximise the following penalized log-likelihood:

$$\max_{\Theta > 0} \log det\Theta - tr(S\Theta) - \lambda \|\Theta\|_1$$

where Θ is the positive definite precision matrix, and S is the estimated covariance matrix of X. The third term is an l_1 penalty similar to that of the lasso, just adapted to the maximisation case rather than the minimisation case. Doing this involves a trade off between maximising the log-likelihood and ensuring that the precision matrix is sparse. Friedman et al. (2007) focus on the dual of the maximisation problem described above, using an iterative approach which uses the lasso in each iteration to estimate the true covariance matrix, based on the empirical covariance matrix. Banerjee et al. (2008) claim that this is very similar to Meinshausen and Bühlmann

(2006), and in fact the node-wise lasso is an approximation to the graphical lasso. Hence, one should expect the graphical lasso to provide similar results, but also be computationally slower than the node-wise lasso.

Sample Size	s_t	AUROC
50	0.9	0.5741
100	0.9	0.8296
500	0.9	0.9317
500	0.6	0.6958
500	0.1	0.06326

Table 3: AUROC for graphical lasso under varying sample size, $p = 100$

Table 3 gives the AUROC for the graphical lasso under various sample sizes and sparsity conditions, tested on the same samples as those in Table 1. Again, the AUROC increases with sample size and increases with sparsity. Note that the node-wise lasso (or) has slightly higher AUROCs, however, it cannot be concluded that the node-wise lasso is a better technique as the values are quite close, and hence appropriate parameter selection for each sample size is the key to comparing the approaches. Fig. 2 shows the ROC curve for the graphical lasso in a sparse setting with a sample size of 500 and 100 variables.

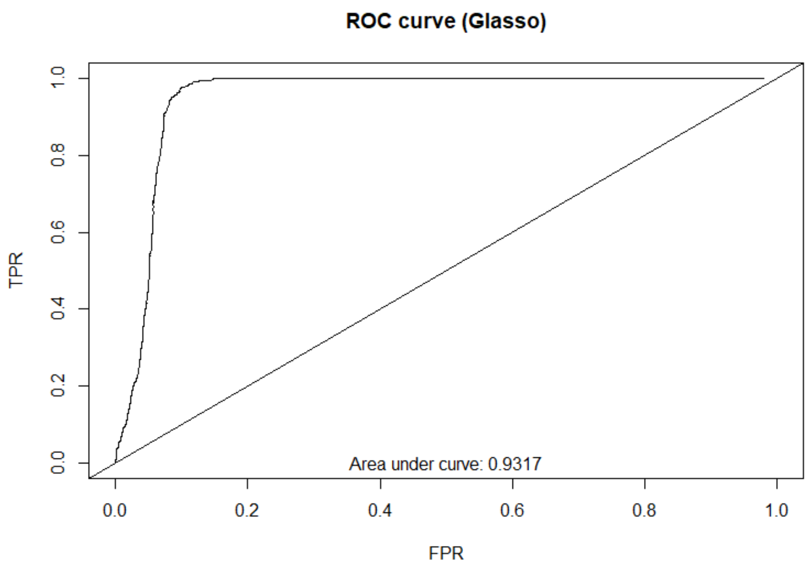

Figure 2: Graphical Lasso ROC curve for n=500, p = 100, $s_t = 0.9$

4.1 Selecting λ

The penalization parameter λ controls the sparsity of the resulting estimate of the precision matrix. The higher the λ, the more sparse the resulting precision matrix, as shown in Fig. 3.

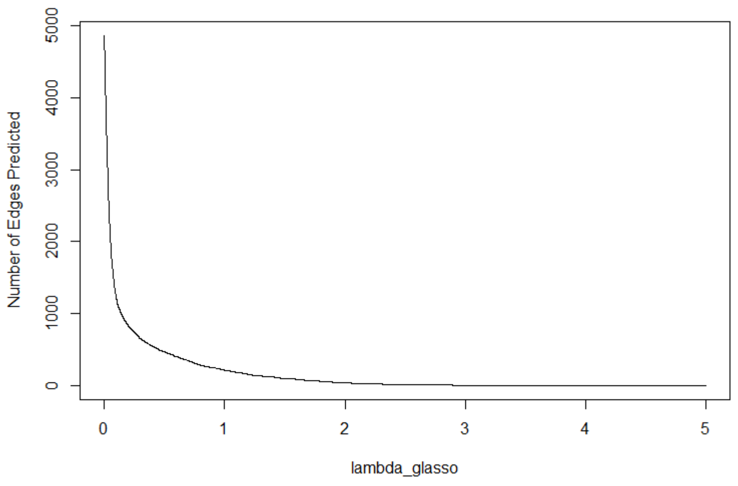

Figure 3: Change in number of edges predicted with λ

There are various ways to select λ. First, Banerjee et al. (2008) provide a finite sample value which is based upon the MBpen introduced in Section 3.1, which also controls the asymptotic probability of falsely connecting two distinct connectivity components of the graph. However, Liu et al. (2010) find this aim of the penalty to be unnatural for many applications of the graphical lasso. They introduce a stability criterion which tends to have higher true (and false) positive rates than the adapted MBpen criterion, with the aim of sparsity as well as inclusiveness.

Finally, a cross validation approach can be used to select the regularization parameter (Friedman et al., 2007; Galloway, 2018). In the simple case, suppose the data was split into a training and test set, the precision matrix is calculated using the training data for various values of λ. Using the testing data, a sample covariance matrix is generated. The precision matrix is used to calculate the log-likelihood of the covariance matrix. The λ is chosen so as to maximise the log-likelihood, or equivalently minimize the negative log-likelihood, using the following formula (or a scaled variant):

$$loglikelihood(S_{test}|\Theta^{\lambda}_{train}) = \log det\Theta^{\lambda}_{train} - tr(S_{test}\Theta^{\lambda}_{train})$$

Other criterion such as the Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC) can also be used depending upon the objective of the research. In this report, 5-fold cross validation and the log-likelihood maximisation is used to choose the regularization parameter λ. The sample performance of this method is detailed in the next section.

5 Sample Performance

5.1 Base Cases

The initial focus is on 3 cases:

- **Case 1:** $n = 500$, $p = 100$, $s_t = 0.9$
- **Case 2:** $n = 100$, $p = 100$, $s_t = 0.9$

- **Case 3:** $n = 500$, $p = 100$, $s_t = 0.6$

In each case, the node-wise lasso with MBpen and the graphical lasso with 5-fold cross validation are repeated 50 times, and the mean and standard errors of the true positive rate (TPR), false positive rate (FPR), true negative rate (TNR), false negative rate (FNR), proportion of dependencies correctly classified (TC%) and proportion of dependencies incorrectly classifed (error %) are reported. Note that although graphs have a % sign in them next to the TC and Error, they are not actually in percentages, but in proportions.

Case 1 corresponds to a scenario where everything is relatively ideal. The sample size exceeds the number of variables, and the precision matrix is relatively sparse. In this case, it should be expected that both methods manage to consistently recover the edge set. Indeed, as Fig. 4 shows, both methods have an average true positive rate very close to 1, however, the mean error rate is significantly higher for the graphical lasso. Also, the false positives are higher for the graphical lasso, which implies that it is over estimating the edge set. The standard errors are also much higher for the graphical lasso. This could potentially be a result of simply a wrong choice of the penalty parameter. It is possible that better tuning of the penalty parameter, or using a different approach to select the λ decreases the error. Note that also a list of only 30 penalty parameters is provided to cross-validate with, and increasing this number could also make the results better or less variable for the graphical lasso.

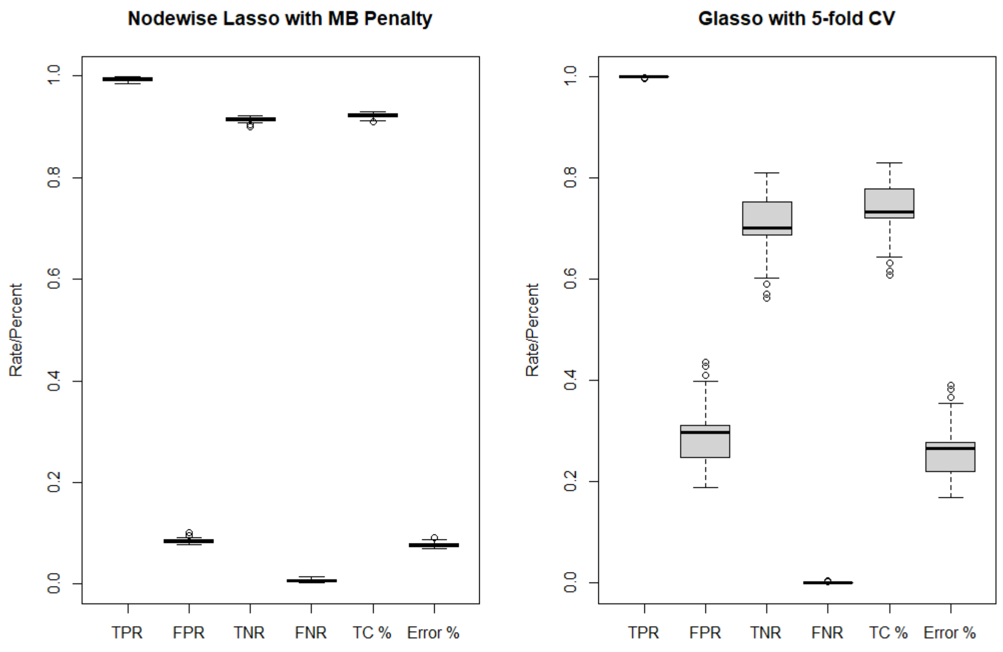

Figure 4: Node-wise lasso (MBpen) versus Glasso (5-fold cross validation). P = Positive, N = Negative, TC = Total Correct, $n = 500$, $p = 100$, $s_t = 0.9$

Case 2 corresponds to a scenario where the application of these methods is vital - when the number of variables is very close to the size of the data. As seen in Fig. 6, the graphical lasso with 5-fold cross validation ends up outperforming the node-wise lasso significantly in retrieving the true edge set, although its estimates are quite variable. The node-wise lasso still has a lower error rate, however this is simply a result of the fact that the precision matrix is sparse and the node-wise lasso is predicting more edges as non-existent. The findings here suggest that the graphical lasso may outperform the node-wise lasso when the sample size is close to the number of parameters, but further investigation is required to prove this.

Case 3 occurs when the precision matrix is dense - typically, both methods do not perform well in recovering the edge set in this case. As Fig. 7 shows, the node-wise lasso biases towards predicting less positives, while the graphical lasso biases towards prediciting too many positives. The node-wise lasso has high false negatives, while the graphical lasso has high false positives. Both methods have error rates of over 30% in this case. The suggestion, in line with Meinshausen and Bühlmann (2006) and Friedman et al. (2007), is that both methods are better at predicting sparse graphs to dense graphs. This result is explored further in Section 5.4.

5.2 Varying Sample Size

Given a fixed number of variables p, the experimentation so far suggests that an increase in the sample size should lead to an increase in correctly predicted edges for both the methods. Fig. 5 is obtained by looking at the error and true positive rate over n going from 50 to 500, fixing p at 50 and s_t at 0.9. As the sample size increases, the true positive rate increases for the node-wise lasso, converging to 1, while its error rate remains relatively constant. The true positive rate also converges to 1 for the graphical lasso, however its error rate is higher and more variable. In this case, it looks like once all the edges have been identified, a higher sample size deceases the error rate of the graphical lasso.

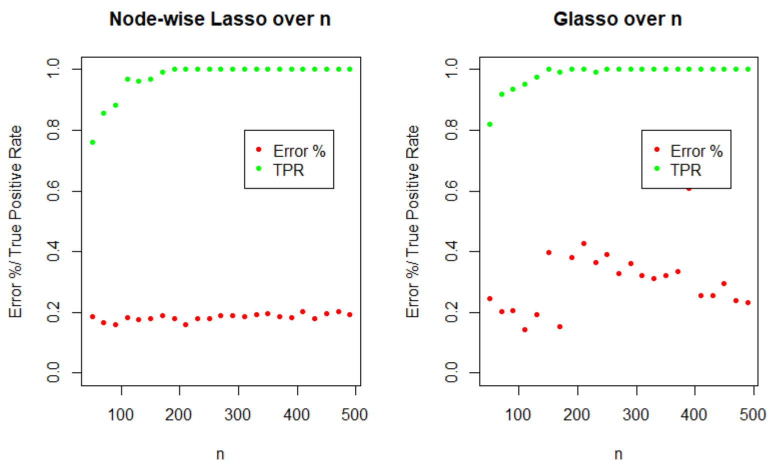

Figure 5: Evolution of TPR and Error % over n. $p = 50$, $s_t = 0.9$.

The same procedure is repeated in a denser setting, setting s_t at 0.6. It can be seen in Fig. 8 that the true positive rate still converges to 1 for both methods, just slightly slower. The error rate consistently goes down for the node-wise lasso, while the it seems to be relatively stable for the graphical lasso. Still the convergence to the true edge-set is faster for the graphical lasso. Hence, depending on what is required from the data and the purpose of research, both methods can be used to estimate a somewhat denser matrix.

5.3 Varying number of variables

For a given n, it seems relatively straightforward that the increasing p should decrease the rate at which the edge set is retrieved. Fig. 9 shows the variation in the true postive rate and error for $n = 500$ and p varying from 10 to 230, in a sparse setting. As expected, increasing p tends to result in a decrease in the true positive rate. However, for the node-wise lasso the error rate keeps going down - probably as a result of there being more negatives in a higher dimensional setting. As seen before, the graphical lasso manages to recover the true edge set even when the number of data points are small relative to the dimensionality of the problem, however, it has a slightly higher error rate. Of course, a difference in penalty parameter choices could change these results significantly.

5.4 Varying the sparsity structure

In the experimentation so far, decreasing the sparsity has resulted in much poorer estimates. Of course, the two methods are designed with sparsity in mind, and seek to be used only in cases where the precision matrix should be sparse. Nevertheless, in this report we look at the impact of sparsity when the sample size is much bigger than the number of predictors. This is the case where a lot of information is available in the data, and previous results suggest that decent estimates may still be obtained by the two methods under these conditions. Hence, fixing n at a 1000, and p at 50, we look at the impact of sparsity of the precision matrix on the true positive and error rates. Fig. 10 shows this variation for the node-wise lasso with MBpen and the node-wise lasso with cross-validation. In this large data, low dimensionality setting, the cross-validation approach outperforms MBpen significantly. It recovers the true edge set at all sparsity levels, and has a lower error rate when the precision matrix is most sparse. Fig. 11 shows the performance of the cross-validated glasso. The true edge set is always recovered, however the error rate is quite high and hence there are also a lot of false positives, which is a trend that is seen with this method of choosing the optimal tuning parameter.

5.5 Deviation from Normality

A crucial assumption here is that X is multivariately normal. The validity of the methods is proved under this assumption (Meinshausen and Bühlmann, 2006; Friedman et al., 2007). But what happens when X deviates from normality? Similar to Meinshausen and Bühlmann (2006), let Z be a sample from a $\mathcal{N}(0, \Sigma)$, sampled as in Section 2. Let Y be a $n \times p$ matrix of identically and independentally distributed uniform[-1,1] variables. Then, let $X = Z + Y$. Adding the independent uniform noise does not change the conditional dependence structure of the data. Both the methods are applied on Z and X, and the results are summarized in Table 4.

Method	TPR- Z	TPR - X	FPR - Z	FPR - X	Error - Z	Error - X
Node-wise Lasso	99.80%	98.80%	17.48%	22.61%	15.73%	20.44%
Graphical Lasso	100%	98.60%	13.07%	19.03%	11.75%	17.25%

Table 4: Performance of methods when assumption of multivariate normality fails to hold (X) versus when it does (Z), $n = 500$, $p = 50$.

Both the methods have significantly higher error rates (and in other simulation tests the maginitude of difference is even higher). However, both the methods still manage to capture the true edge sets. Hence, in the presence of some noise in the data, the conditional dependence structure should still be captured by the methods implemented, although some modifications to the penalty selection and the methods themselves could lead to even better results.

6 Conclusion

In this report the node-wise lasso and the graphical lasso were used and evaluated in various simulation settings to recover the edge set which represents the conditional dependence between variables. It is found that under the correct conditions, the methods are very strong at recovering the edge set as defined in the introduction. Some key trends are observed in this process. The first is that an increase in sample size improves results for both methods significantly. Secondly, the assumption of sparsity of the precision matrix is crucial to the functioning of the two methods, although under the correct conditions even denser matrices can be estimated. Third, the ratio of the number of variables to the sample size is important. The sample should be large relatively to the dimensionality of the problem. Fourth is that small deviations from normality in the observed data do not completely break down the two methods. Finally, the most important factor when using these methods is regularization parameter selection, as it completely determines the final result of the two methods.

Unfortunately, this is also where this report is limited. Only two methods of regularization selection are used for the node-wise lasso, and just one is used for the graphical lasso. Moreover, parameters like how many folds to cross-validate with were chosen heuristically. However, Section 3.1 and Section 4.1 describe more ways in which λ can be selected for the two methods. Future research can attempt to find conditions under which different methods of choosing the penalty paramter are optimal. Also, this report looks at the maximum dimensionality of $p = 230$. The validity of the results found here are not guaranteed under very high dimensional settings, which is where these methods are often interesting. Hence, there is a lot of potential for future work in assessing the strength of the two methods described here in doing covariance selection of high-dimensional, sparse graphs.

References

Banerjee, O., Ghaoui, L. E., and d'Aspremont, A. (2008). Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *Journal of Machine Learning Research*, 9:485–516.

Buhl, S. L. (1993). On the existence of maximum likelihood estimators for graphical gaussian models. *Scandanavian Journal Of Statistics*, 20(3):263–270.

Callot, L., Caner, M., Ulasan, E., and Özlem Önder, A. (2019). A nodewise regression approach to estimating large portfolios.

Dahl, J., Vandenberghe, L., and Roychowdhury, V. (2008). Covariance selection for non-chordal graphs via chordal embedding. *Optimization and Software*, 23(4):501–520.

Dempster, A. (1972). Covariance selection. *Biometrics*, 28(1):157–175.

Edwards, D. (2000). *Introduction to Graphical Modelling*. Springer, second edition.

Fan, Y. and Tang, C. Y. (2013). Tuning parameter selection in high dimensional penalized likelihood. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3):531–552.

Friedman, J., Hastie, T., and Tibshirani, R. (2007). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 0(0):1–10.

Galloway, M. (2018). Cvglasso. `https://github.com/MGallow/CVglasso`.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models - Principles and Techniques*. The MIT Press, second edition.

Kovács, S., Ruckstuhl, T., Obrist, H., and Bühlmann, P. (2021). Graphical elastic net and target matrices: Fast algorithms and software for sparse precision matrix estimation.

Liu, H., Roeder, K., and Wasserman, L. (2010). Stability approach to regularization selection (stars) for high dimensional graphical models.

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

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

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

Appendices

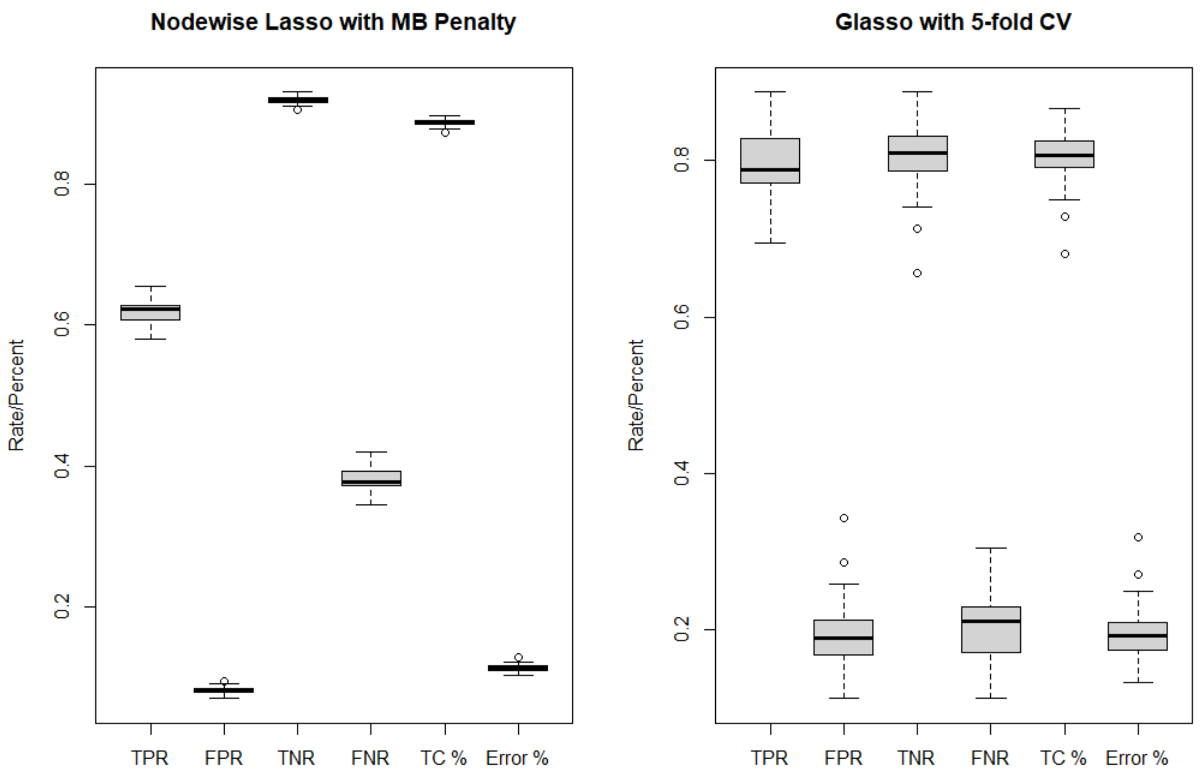

Figure 6: Node-wise lasso (MBpen) versus Glasso (5-fold cross validation). n = 100, p = 100, s_t = 0.9. Section 5.1.

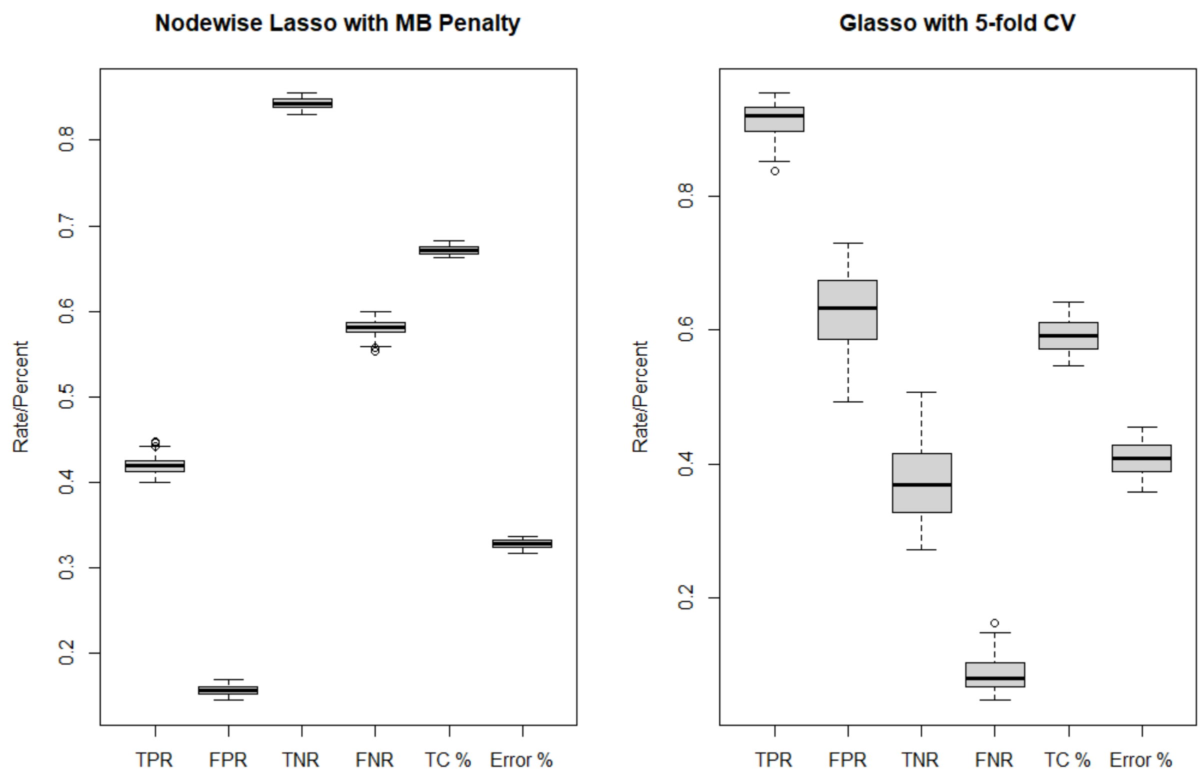

Figure 7: Node-wise lasso (MBpen) versus Glasso (5-fold cross validation). n = 100, p = 100, s_t = 0.6. Section 5.1.

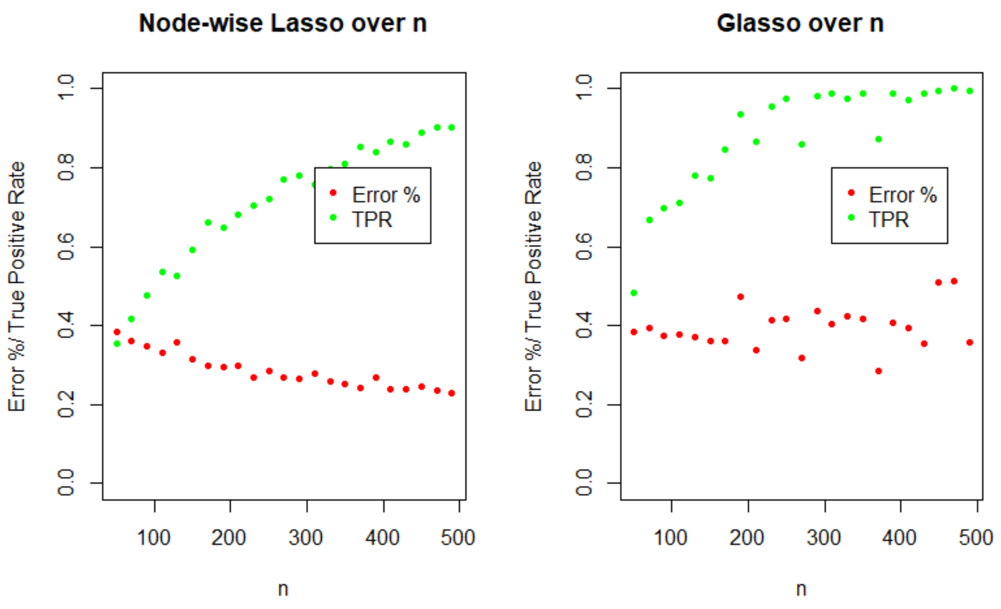

Figure 8: Evolution of TPR and Error % over n. $p = 50$, $s_t = 0.6$. Section 5.2.

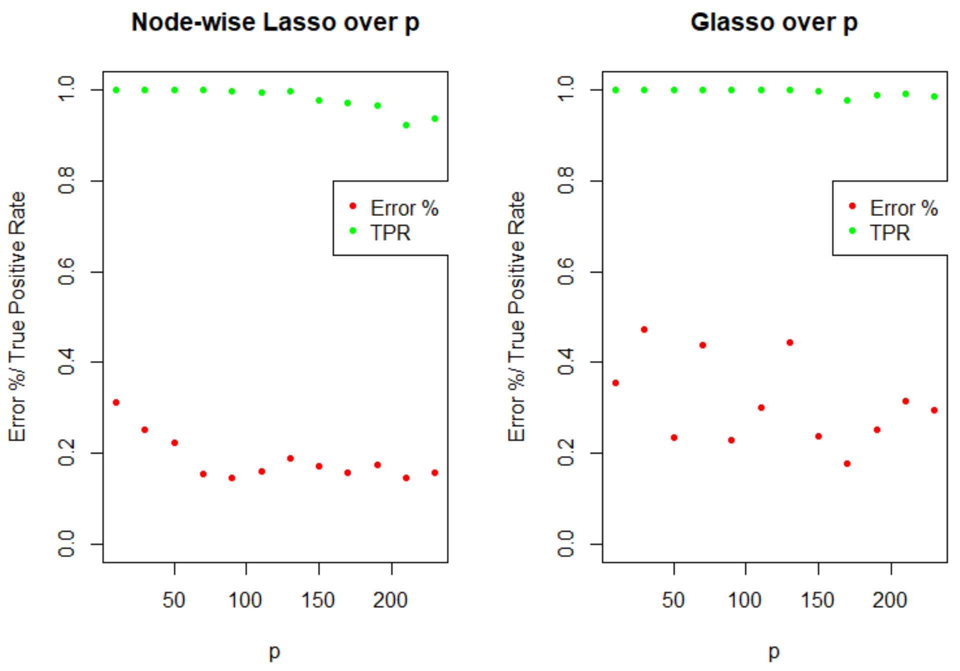

Figure 9: Evolution of TPR and Error % over p. $n = 500$, $s_t = 0.9$. Section 5.3.

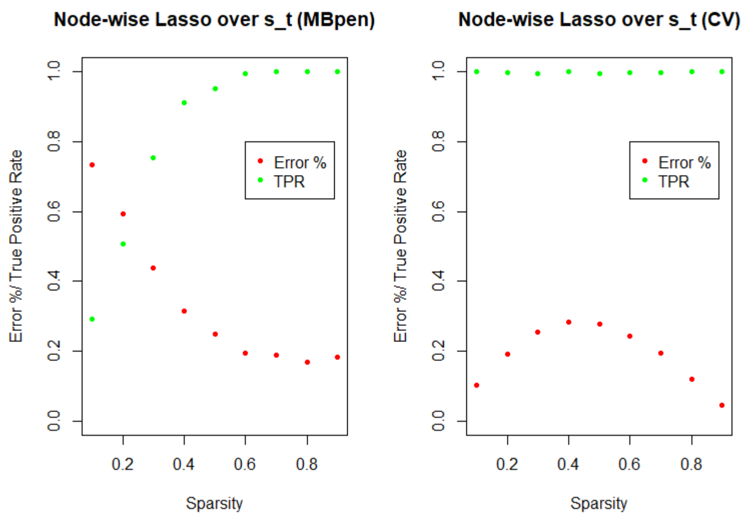

Figure 10: Evolution of TPR and Error % over s_t. $p = 50$, $n = 1000$. Section 5.4.

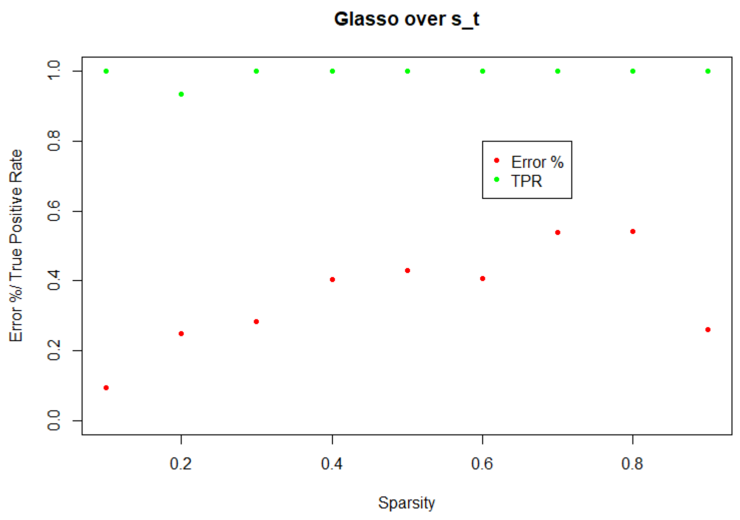

Figure 11: Evolution of TPR and Error % over s_t. $p = 50$, $n = 1000$. Section 5.4.