2.1 Introduction

2.1.1 The Node-wise Lasso Approach

Node-wise regression which is also called neighbourhood selection created by Nicolai Mein shausen and Peter Buhlmann in 2006. The main idea of node-wise regression is using a projection of every column of the design matrix on all the remaining columns to get every detailed column of the precision matrix [4]. This method is useful for sparse high-dimension data and similar to the variable selection for Gaussian linear models. Therefore, the assumption of this method is that X is normally distributed, and each X is independently and identically distributed. [1].

For more details about the node-wise regression. This method builds the approximate inverse $\hat{\Theta}$, which is the approximate inverse of the variance of X, is given by the lasso on the design matrix X for the nodewise regression and can also be calculated under the Karush–Kuhn–Tucker (KKT) conditions which are first derivative tests (sometimes called first-order necessary conditions) for a solution in non-linear programming to be optimal [6] and aims to get the optimal parameters in this method. Second, it use the lasso p times where we regress X_j (the j-th column of X) on the remaining columns X_{-j} [3].

The predicted edge set is then derived according to the following two rules, named node-wise lasso 1 (NWL1) and node-wise lasso 2 (NWL2), respectively, defined as follows:

$$\hat{E}_{1,\lambda} := \{(j,k) \ : \ \text{both } \hat{\beta}_{jk,\lambda} \text{ and } \hat{\beta}_{kj,\lambda} \text{ are non-zero }, 1 \leq j,k \leq p, j \neq k\}$$

$$\hat{E}_{2,\lambda} := \{(j,k) \ : \ \text{either } \hat{\beta}_{jk,\lambda} \text{ or } \hat{\beta}_{kj,\lambda} \text{ are non-zero }, 1 \leq j,k \leq p, j \neq k\}$$

2.1.2 The Graphical Lasso Approach

Friedman, Hastie, and Tibshirani [2] initially introduced the graphical lasso approach to estimate the graphical model as an alternative to the node-wise lasso regression approach offered by Meinshausen and Bühlmann [1]. Using the ℓ_1 penalty on the matrix elements, the graphical lasso approach is built for the estimation of the sparse inverse covariance matrix.

We can demonstrate that $c_{j,k} = 0$ if and only if $\Theta_{j,k} = 0$, where $\Theta_{j,k}$ is the (j,k)-th element of the inverse covariance matrix. This is predicated on the idea that X has a multivariate normal distribution with a covariance matrix. Therefore, different from the node-wise lasso approach, the edge set under graphical lasso approach can be alternatively shown as:

$$\hat{E}_{G,\lambda} = \{(j,k) : \hat{\Theta}_{jk} \neq 0\}$$

So how can we get an estimator $\hat{\Theta}$?

To begin with, the gaussian log-likelihood graphical models can be expressed as:

$$\log\det\Theta - \text{trace}(\hat{\Sigma}\Theta)$$

Then, the graphical Lasso approach maximizes log-likelihood with a penalty to estimate $\hat{\Theta}$:

$$\hat{\Theta} = \arg\max_{\Theta} \left(\log\det\Theta - \text{trace}(\hat{\Sigma}\Theta) - \lambda \sum_{j \neq k} |\Theta_{jk}| \right)$$

2.2 Data Generation

Firstly, we generate n random samples (fix $n = 500$ at first):

$$X_1, ..., X_n \overset{i.i.d.}{\sim} N(0, \Sigma) \quad \text{where } \Sigma = \Theta^{-1}$$

Secondly, we generate the inverse covariance matrix (fix $n = 500$, $p = 50$ at first):

$$\Theta = B + rI_p \in \mathbb{R}^{p \times p}$$

where I_p is an identity matrix, $r > 0$ is some constant such that the smallest eigenvalue of Θ is at least 1, and the matrix B is generated:

$$B_{ij} \sim 0.5 \times \text{Bernoulli}(p = 0.1) \quad (i \neq j)$$

Under those conditions above, the sparsity pattern in θ corresponds to the true edge set E. Observe that the estimation will be affected by the sample size n, the number of nodes p and the sparsity pattern which is represented by the probability in the Bernoulli distribution. As a result, we fix $n = 500$, $p = 50$, and Bernoulli($p = 0.1$) at first for the following few sections, test different parameters in the simulation section and present some results in the appendix.

2.3 TPR and FPR with a choice of $\lambda = 0.08$

At first, using the $\lambda = 0.08$ to estimate the true positive rate (TPR) and false positive rate (FPR) under these three methods.

	TPR	FPR	Error rate
Node-wise 1	84.03%	7.50%	8.32%
Node-wise 2	87.39%	10.04%	10.29%
Graphical	87.39%	11.21%	11.35%

Table 5: The summary of TPR, FPR and error rate for three methods

True positive rate (TPR) is also called sensitivity to reflect the ability of screening tests to detect. Under the λ, the TPR of three methods are above 80 percent which shows that all three methods are robust. The error rate is the misclassification rate, so we can see how three methods behave under the $\lambda = 0.08$. A higher error rate means the model behaves worse.

In general, the most accurate method is node-wise lasso 1 (NWL1) and the least accurate method is graphical lasso with the λ.

2.4 ROC Curves and AUC

In order to get the ROC curves for three methods, the project initially produces a grid of 100 different λs from 0.01 to 1. The area below the ROC (AUC) summarises the performance of each method, and the closer the value of AUC is to 1, the better the overall performance of this method.

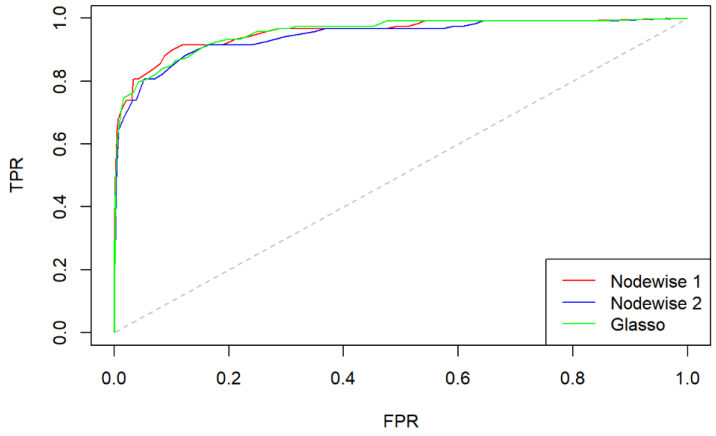

Figure 2: The ROC curves of the three methods

	AUC
Node-wise 1	0.9551
Node-wise 2	0.9548
Graphical	0.9543

Table 6: The summary of AUC for the three methods

After checking Table 6 and Figure 2, this project shows that all of these three methods have good performance with a high slope coefficient (FPR between 0-0.1). What's more, it seems that graphical lasso method is not as accurate as two node-wise methods, because AUC of graphical lasso is less than two node-wise methods.

For n=2000 and p=50, the AUC of all three methods have higher values than those of n=500 in these methods. However, the value of AUC in graphical lasso is higher than that in node-wise one method, and the value of AUC in node-wise 2 is the highest among these three methods.The specific figure 8 can be seen in the Appendix

2.5 Selection of Optimal Tuning Parameter λ

The project uses four methods to choose the optimal λ. They are cross-validation, AIC, BIC and M1 which minimizes the misclassification error rate to find the best λ. After testing different ranges of λ, it shows that the performance is best under the grid of λ between 0.1 to 10.

2.5.1 Methods used to choose optimal λ

In the M1 method, the misclassification error rate is defined as the performance metric. Each λ makes a prediction for the edge set and from which calculates the misclassification error. This process is repeated 100 times with the different values of λ from the grid that was set before. After getting the whole 100 misclassification rates, the λ within one standard error from the λ with the lowest error rates is chosen to be the optional λ.

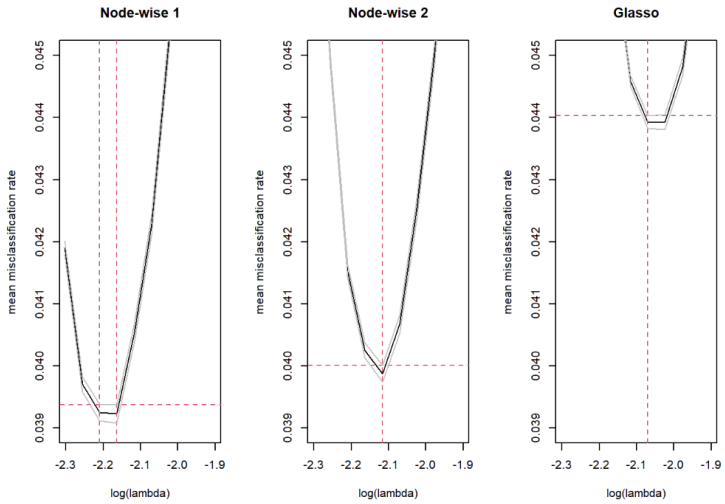

Figure 3: Mean misclassification rate against $\log(\lambda)$ for these three methods

Cross-validation is an important method to evaluate the performance of a model. It is mainly used to select the best model in the current problem scenario among multiple models (with different λ in this project) (model selection), and the project utilizes k-fold cross-validation to choose the best λ determining β's which could be changed into 0. Then applying λ with a minimum error rate in these lasso regressions. From the perspective of prediction, AIC selects a good model to predict, while from the perspective of fitting, BIC selects a model that best fits the existing data and from the interpretation of Bayesian factors, BIC is the model with the largest marginal likelihood.

2.5.2 CV Error, AIC and BIC of Node-wise Lasso

$$\text{CVerror} = \text{MSE} = \frac{1}{p}\sum_{j=1}^{p} \|X_j - \frac{1}{n}\sum_{1\le j\le p, j\neq l} \hat{\beta}_{jl} X_l\|_2^2$$

$$\text{AIC} = 2k - 2\ln(\hat{L})$$

$$\text{BIC} = k\ln(n) - 2\ln(\hat{L})$$

where k is the number of non-zero parameters of $\hat{\beta}$ and $\hat{L}$ is the log-likelihood of the objective function.

2.5.3 CV Error, AIC and BIC of Graphical Lasso

$$\text{CVerror} = \ell(\Theta) = \frac{n}{2}\left(\text{trace}(\hat{\Sigma}\hat{\Theta}) - \log|\det\hat{\Theta}|\right)$$

where $\bar{X} = \sum_{i=1}^{n} X_i$ is the sample mean and $\hat{\Sigma} = n^{-1}\sum_{i=1}^{n}(X_i - \bar{x})(X_i - \bar{X})^\top$ is the sample covariance matrix.

$$\text{AIC} = \ell(\Theta) + \sum_{1\le i<j\le p} \mathbf{1}\{\hat{\Theta}_{ij} \neq 0\}$$

$$\text{BIC} = \ell(\Theta) + \frac{\log(n)}{2}\sum_{1\le i<j\le p} \mathbf{1}\{\hat{\Theta}_{ij} \neq 0\}$$

2.5.4 Support recovery

This project chooses the TPR, FPR and error rates to be the metrics for measuring support recovery. The result of the support recovery under different methods and λ are present in the following Table 7.

	Node-wise 1				Node-wise 2				Graphical			
	M1	CV	AIC	BIC	M1	CV	AIC	BIC	M1	CV	AIC	BIC
λ	0.1097	0.1	0.1	0.1	0.1205	0.1	0.1	0.1	0.1262	0.1	0.1150	0.2783
TPR	0.9725	0.7395	0.7395	0.7395	0.9725	0.7899	0.7899	0.7899	0.9633	0.8067	0.7311	0.2783
FPR	0.0134	0.0271	0.0271	0.0271	0.0134	0.0361	0.0361	0.0361	0.0278	0.0470	0.0154	0
Error rate	0.0147	0.0497	0.0497	0.0497	0.0147	0.5306	0.5306	0.5306	0.0286	0.0612	0.04	0.0963

Table 7: Supply recovery and optimal λ

$$\text{ErrorRate} = \frac{FN + FP}{N + P}$$

M1 which uses the misclassification error rate repeated 100 times to find the optimal tuning parameter λ has the lowest error rate among these three different kinds of lassos. In node-wise 1 whose λ is 0.1097 has the lowest error rate 0.0147. In the graphical lasso, CV performs better in TPR while worse than AIC in error rate.

For $n = 2000$ and $p = 50$, M1 also has the lowest error rate among these three different kinds of lassos. In node-wise 1 has the lowest error rate 0.0008163 which is lower than that of $n = 500$. However, in the graphical lasso, BIC performs best in error rate besides M1. In general, the error rate of each method is lower when the number of n becomes bigger (more specific table 8 see Appendix).

2.5.5 Replicating and Testing

For M1, under the original parameters, the entire procedure is repeated 100 times from generating data to calculating AUC and Minimum Misclassification Rate over a grid of λ values. In addition, the mean and standard error of every measurement are calculated.

Based on the results demonstrated in figure 4, graphical lasso surprisingly produces the highest AUC with the lowest standard error. Node-wise1 has the lowest AUC and the highest standard error. The performance in misclassification rate does not follow the same pattern, in which graphical lasso has the highest misclassification rate and the lowest standard error, while node-wise1 and node-wise2 have almost the same misclassification rate and standard error.

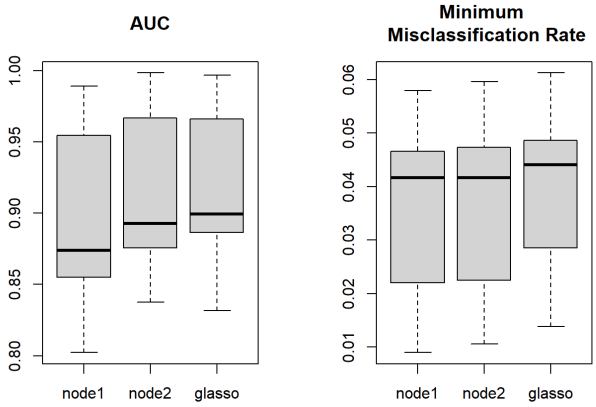

Figure 4: Boxplot of AUC and Minimum Misclassification Rate from M1 for each lasso method

For the remaining three methods(CV, AIC, BIC), to evaluate the performance of a chosen ideal λ, multiple sets of data samples are simulated with the same parameters ($n = 500$, $p = 50$, and Bernoulli($p = 0.1$)).

The result displayed in figure 5 is generated from 100 different simulations. For the three lasso methods, the mean of node-wise 1 is the smallest, while the graphical lasso is the largest in the three selection methods. It is consistent with the results in the above sample. In terms of the methods to select the optimal tuning parameter λ, we think CV performs the best because the figure 5 shows that all three lasso methods have a relatively small and balanced mean and standard error in the CV method.

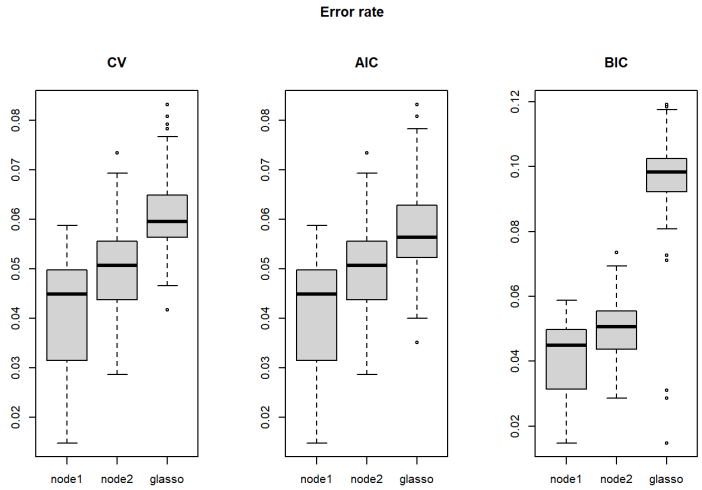

Figure 5: Boxplots of error rate from CV, AIC and BIC selection methods in 100 simulations

However, when we change from $n = 500$ to $n = 2000$, it comes to a different conclusion that node-wise 1 performs the best AUC and minimum misclassification rate, while graphical lasso performs the worst. The specific figure 10 can be seen in the Appendix. However, the error rate of graphical lasso method is the lowest among the three lasso methods in each method of choosing the optimal λ when $n = 2000$. The specific figure 11 can be seen in the Appendix.

2.6 Simulation & Numerical Results

2.6.1 Simulation Settings

To explore various simulation parameters, the sample size (n) is adjusted while the number of variables (p) and Bernoulli(Prob) remain constant. Then after, Bernoulli(Prob) which represents the sparsity structure is adjusted while the other two parameters remain constant.

2.6.2 Simulation under Different Values of n for a Fixed Value of p

In figure 6, the AUC values are displayed against a range of n values from 0 to 1000. Overall, all three lasso methods perform very similarly: (1) At first, from $n = 0$ to approx.$n = 50$, as n increases, the AUC also increases; (2) Then, between approx.$n = 50$ and 100, the value of AUC fluctuates dramatically; (3) After that, as n increases, n and AUC are positively correlated; (4) Finally, as n approaches 400, the AUC approaches 1, and as n increases, the AUC keeps fluctuating in a small range around 1.

However, there are some slight changes when we fix different values of p. These changes are mainly reflected in the fluctuations part which is (2) above. As figure 6 shows, when we fix $p = 20$, the fluctuation range of AUC is between 0.7 and 0.9, but when we adjust $p = 50$ and $p = 100$, it is between 0.5 and 0.7.

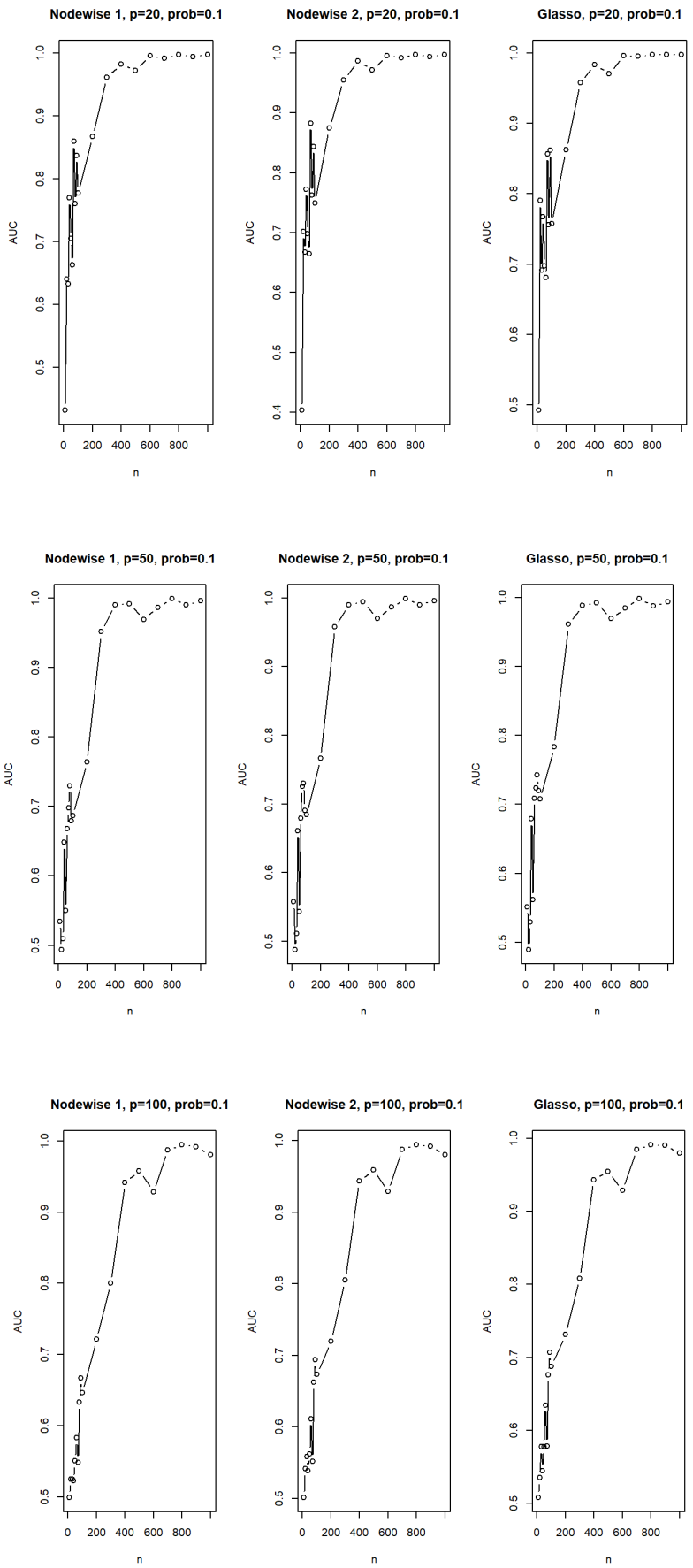

Figure 6: AUC against n when $p = 20, 50, 100$ and Bernoulli ($prob = 0.1$)

2.6.3 Simulation under Different Sparsity Patterns for $p = 50, n = 500$

In figure 7, the AUC values are displayed against a range of sparsity patterns which is Bernoulli(Prob) from 0.0 to 1.0.

Figure 7 illustrates how AUC is steady and high for low Prob but rapidly decreases once Prob exceeds a particular threshold. This behaviour is a result of the sparsity structure and unbalanced classification problems. For small Prob, Θ is sparse and there are more positive outcomes in the vector representing edge set E. This results in a situation where TPR closes to 1 and FPR closes to 0. Thus, AUC is quite near to 1.0. Nonetheless, when the prob exceeds 0.2, Θ becomes less sparse, resulting in a substantial decrease in AUC.

Comparing three lasso methods, it is possible to deduce that graphical lasso performs worse than both node-wise approaches when Θ is not sparse, for graphical lasso always has the smallest AUC value when the Prob is the same.

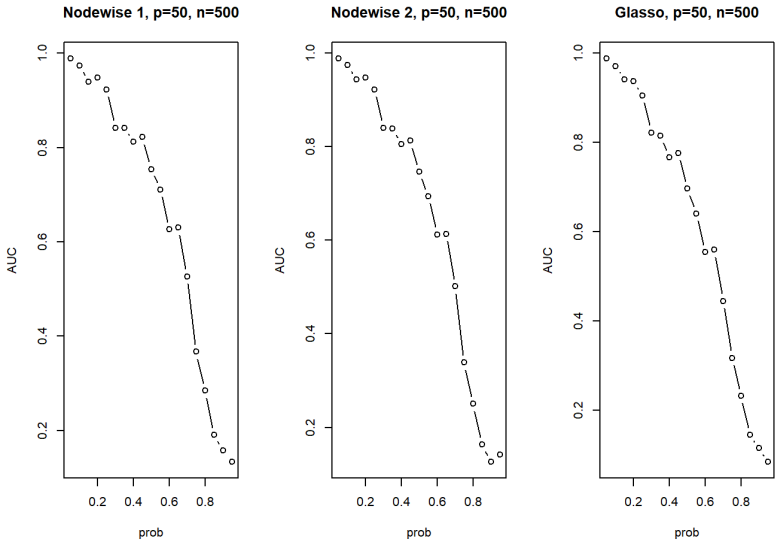

Figure 7: AUC against *prob* when $p = 50$ and $n = 500$

2.7 Summary

This report introduces the graphical lasso and two node-wise lasso methods for estimating a graphical model and the conditional dependency structure. The estimation performance of a method is evaluated by FPR, FNR, Error Rate, ROC and AUC.

In terms of choosing the optimal tuning parameter λ, this report uses cross-validation, AIC, BIC and M1 which minimizes the misclassification error rate. Based on the selected optimal λ, it turns out that M1 has the lowest error rate among above selection approaches in all three lasso methods. In addition, the graphical lasso method is slightly less accurate than two node-wise lasso methods in M1, CV and BIC selection approaches. However, in AIC selection approach, the graphical lasso method performs the best.

Furthermore, this report replicates the above procedure 100 times to test our findings and conclusions. It shows that most of the conclusions hold true, except for graphical lasso method which performs the worst in the AIC selection approach.

Finally, we explore different choices of parameters in the simulations. When we simulate different values of n for a fixed value of p, all three lasso methods perform very similarly. However, there are some slight changes in terms of the fluctuation range when we fix different values of p. When we simulate different sparsity patterns, graphical lasso performs worse than both node-wise methods when Θ is not sparse.

A Appendix for Task 2

The following are figures and tables after we tune the parameters from $n = 500$, $p = 50$, Bernoulli(Prob $= 0.1$) to $n = 2000$, $p = 50$, Bernoulli(Prob $= 0.1$).

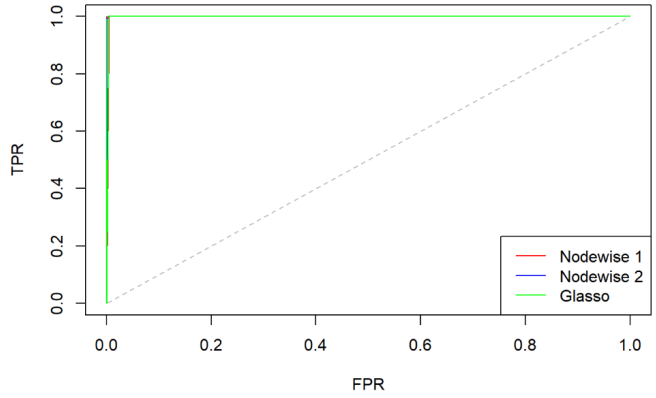

Figure 8: ROC curves of the three methods for $n = 2000$

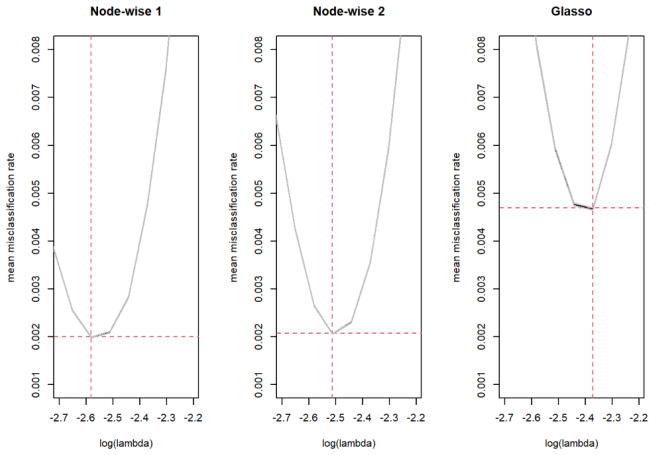

Figure 9: Mean misclassification rate against $\log(\lambda)$ in M1 for $n = 2000$

	Node-wise 1				Node-wise 2				Graphical			
	M1	CV	AIC	BIC	M1	CV	AIC	BIC	M1	CV	AIC	BIC
λ	0.0756	0.1	0.1	0.1	0.0811	0.1	0.1	0.1	0.0933	0.0266	0.0433	0.0614
TPR	0.9912	1	1	1	0.9912	1	1	1	0.9561	1	1	1
FPR	0	0.591	0.591	0.591	0	0.6624	0.6624	0.6624	0	0.2715	0.0977	0.0262
Error rate	0.0008	0.5331	0.5331	0.5331	0.0008	0.5975	0.5975	0.5975	0.0041	0.2449	0.0882	0.0237

Table 8: Supply recovery and optimal λ for $n = 2000$

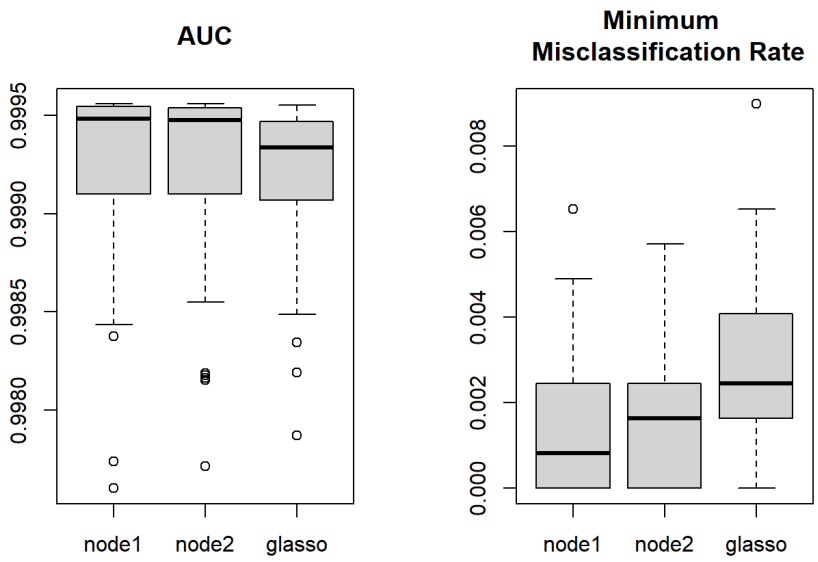

Figure 10: Boxplot of AUC and Minimum Misclassification Rate from M1, for $n = 2000$

As we can see in the figure 11, after we change $n = 2000$, the differences between CV, AIV and BIC selection methods are really small, for the results are almost the same.

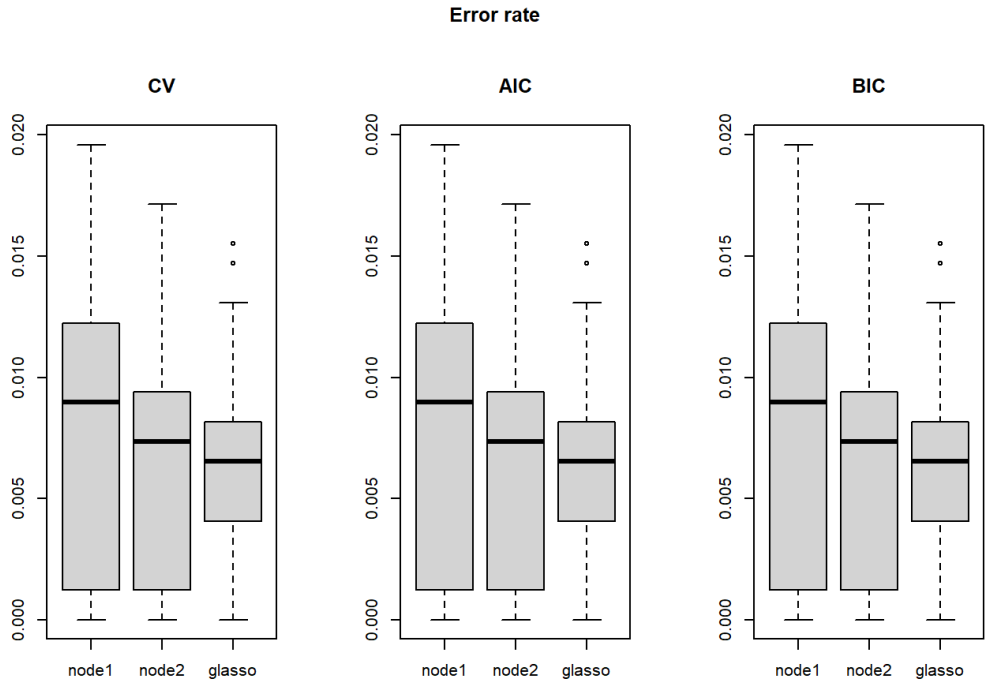

Figure 11: Boxplots of error rate from CV, AIC and BIC in 100 simulations, for $n = 2000$

References

[1] Nicolai Meinshausen and Peter Bühlmann. "High-dimensional graphs and variable selection with the lasso". In: *The annals of statistics* 34.3 (2006), pp. 1436–1462.

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

[3] Sara Van de Geer et al. "On asymptotically optimal confidence regions and tests for high-dimensional models". In: *The Annals of Statistics* 42.3 (2014), pp. 1166–1202.

[4] Jana Janková. "Asymptotic inference in sparse high-dimensional models". PhD thesis. ETH Zurich, 2017.

[5] Jannie Borst et al. "CD4 + T cell help in cancer immunology and immunotherapy". In: *Nature Reviews Immunology* 18 (July 2018), p. 1. DOI: 10.1038/s41577-018-0044-0.

[6] Laurent Callot et al. "A nodewise regression approach to estimating large portfolios". In: *Journal of Business & Economic Statistics* 39.2 (2021), pp. 520–531.

[7] Chenqu Suo et al. "Mapping the developing human immune system across organs". In: (Jan. 2022). DOI: 10.1101/2022.01.17.476665.