

# Node wise Lasso - LIN

LIN

07/12/2019

## 2.2 Simulation

For the target of comparing the sample performance (such as False Positives, False Negatives) between Node-wise Lasso approach and Graphical Lasso approach, simulation data is generated by the following settings:

Consider the  $p$ -dimensional multivariate Gaussian distributed random variable

$$\mathbf{X} = (X_1, X_2, \dots, X_p)^T \sim \mathcal{N}(0, \mathbf{\Sigma})$$

, where  $\mathbf{\Sigma} = \mathbf{\Theta}^{-1}$ .

In detail, we let  $\mathbf{\Theta} = \mathbf{B} + \delta \mathbf{I}_p \in \mathbb{R}^{p \times p}$  where  $\mathbf{I}_p$  stands for the identity matrix and symmetric matrix  $\mathbf{B}$  has its each off-diagonal entry generated respectively and independently, which will equal to 0 with probability 0.9 or equal to 0.5 with probability 0.1. We calculated and chose  $\delta > 0$  to guarantee  $\mathbf{\Theta}$  is positive definite. We standardized the  $\mathbf{\Theta}$  to achieve correlation matrix instead of covariance matrix.

Of particular interest is the identification of entries that equals to 0 in the  $\mathbf{\Theta}$ , the sparsity pattern in  $\mathbf{\Theta}$  links to the true edge set  $E$  in the graphical model  $G = (V, E)$ , where  $V = 1, \dots, p$  is the set of nodes and  $E$  is the set of edges  $\in V \times V$  between nodes.

Finally, we can specify two arguments in the `simulation(n, p)` function, with `n` as the number of dataset generated, and with `p` as the number of dimension or, say, the number of nodes generated in  $\mathbf{X} = (X_1, X_2, \dots, X_p)^T$ . As the return of the function, a list consisting of:

1. A  $n \times p$  data set.
2.  $\mathbf{\Theta}$
3. Standardized  $\mathbf{\Theta}$

For instance, an  $100 \times 50$  dataset is generated by the following codes.

```
data_list <- simulation(50,100)
simulation_data <- data_list$data
simulation_StandardTheta <- data_list$standardtheta
```

To have a stable and reliable result, we are conducting 50 iterations and having mean value of the results from all the method introduced and discussed in the following chapters. ## 2.3 Node-wise Lasso approach

### 2.3.1 Introduction

One of the approaches that has been deployed to estimate the edge set  $E$  is based on the linear regression. Under the assumption of  $\mathbf{X}$  being multivariate Gaussian distributed random variable, the method of node-wise Lasso is estimate each random variables  $X_i$  in  $\mathbf{X} = (X_1, X_2, \dots, X_p)^T$  by applying linear regression on the remaining variables. More specifically, for each node  $i \in V$ , we regress  $X_i$  on the remaining variables  $X_j = \{X_j : j \in V, j \neq i\}$  to achieve this form

$$X_i = \sum_{1 \leq j \leq p, j \neq i} \beta_{ij} X_j + \epsilon_{ij}$$

Due to the high dimensionality of our setting, we could implement the Lasso method while regressing all variables to achieve sparse solution.

For response variable  $y_i$  and predictors  $x_{ij}$ , the lasso coefficient  $\hat{\beta}_\lambda$  is chosen to minimise the equation

$$\sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j|$$

where  $\lambda \geq 0$  is the tuning parameter. As the  $l_1$  norm of penalty has the effect of forcing some of the coefficients to be exactly zero where a large enough  $\lambda$  is chosen, Lasso can perform variable selection and build a sparse model containing a subset of all  $p$  predictors.

**Section 2.3.3** will introduce and compare different methods to choose optimal tuning parameter  $\lambda$  and the performance each of them achieves. After choosing the tuning parameter  $\lambda$ , we could attain the Lasso estimation for each coefficient  $\beta_{ij}$ , which is  $\hat{\beta}_{ij}$ . We consider the following rules, named **node-wise Lasso 1** and **node-wise Lasso 2** to estimate the Edge Set  $E$ :

1. **node-wise Lasso 1**: if  $\hat{\beta}_{ij} \neq 0$  and  $\hat{\beta}_{ji} \neq 0$  are satisfied, then we say nodes  $X_i$  and  $X_j$  are estimated to be connected.

$$\hat{E}_1 = \left\{ (i, j) : \hat{\beta}_{ij}, \hat{\beta}_{ji} \neq 0, 1 \leq i, j \leq p, i \neq j \right\}$$

2. **node-wise Lasso 2**: if either  $\hat{\beta}_{ij} = 0$  or  $\hat{\beta}_{ji} = 0$ , we say nodes  $X_i$  and  $X_j$  are estimated to be connected.

$$\hat{E}_2 = \left\{ (i, j) : \exists \hat{\beta}_{ij}, \hat{\beta}_{ji} \neq 0, 1 \leq i, j \leq p, i \neq j \right\}$$

For each rule and a certain choice of the tuning parameter  $\lambda$ , by comparing the true edge set  $E$  and the estimated edge set  $\hat{E}$ , the elements in the Confusion Matrix can be calculated and based on that, the True Positive Rate ( $TPR_\lambda$ ) and False Positive Rate ( $FPR_\lambda$ ) can be attained consequently. As a result, we are able to plot the ROC curve over a fine grid of values of  $\lambda$ . Furthermore, the Area Under the ROC ( $AUROC$ ) for each method will be plotted to compare the overall performance on recovering the true edge set  $E$ . These will be introduced in the **Chapter 2.3.2**.

### 2.3.2 ROC Curve

After conducting node-wise lasso with different tuning parameter  $\lambda$  and different choices of method ( $\hat{E}_1$  and  $\hat{E}_2$ ), one way to compare their performance is to plot the Receiver Operating Characteristic Curve (ROC Curve), created by plotting the true positive rate ( $TPR_\lambda$ ) against the false positive rate ( $FPR_\lambda$ ) at various choices of  $\lambda$ . ROC analysis provides us methods to select potentially optimal models and to compare the performance. Further interpretations such as Area Under Curve ( $AUC$ ) can be calculated for more analysis.

For each specific  $\lambda$ , Lasso estimation for each coefficient  $\beta_{ij}$  can be achieved, here in the function `edge_table(data_set, lambda_choice)`. By treating the simulated data and lambda choice, the function will return a  $p \times p$  matrix where each entry will be either TRUE or FALSE which respectively suggests that, for example, at  $(i, j)$ -entry, TRUE means  $\hat{\beta}_{ij} \neq 0$  and FALSE means  $\hat{\beta}_{ij} = 0$ . Corresponds to the rules **node-wise Lasso 1**, we say nodes  $i$  and  $j$  are estimated to be connected as long as Both  $(i, j)$ -entry and  $(j, i)$ -entry are TRUE. Equivalently, **node-wise Lasso 2** only requires either  $(i, j)$ -entry or  $(j, i)$ -entry are TRUE for connecting nodes  $i$  and  $j$ .

Since the sparsity pattern in the  $\Theta$  identify the true edge set  $E$

$$E = \{(i, j) : \Theta_{ij} \neq 0, 1 \leq i, j \leq p, i \neq j\}$$

, function `true_edge(theta)` visualise these and return a  $p \times p$  matrix similar to the one returned by the function `edge_table(data_set, lambda_choice)`.

Finally, by making the results from `edge_table` and `true_edge` as arguments function `confusion_matrix(estimate_edge, trueedge, estimate_way)` will perform calculation of confumatrix and return relevant result including  $TPR_\lambda$  and  $FPR_\lambda$ . Setting `estimate_way = "both"` will allow function perform the calculation through **node-wise Lasso 1**, while `estimate_way = "either"` makes it go through **node-wise Lasso 2**.

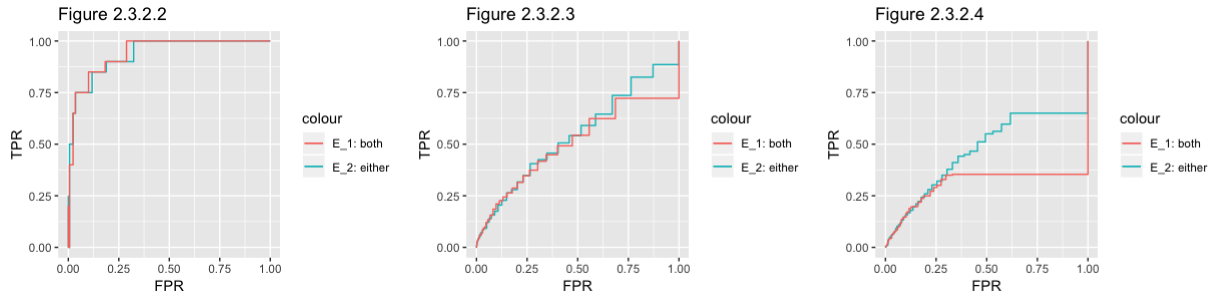
As a result, each  $\lambda$  will leads to a specific pair of  $TPR_\lambda$  and  $FPR_\lambda$ , function `ROC_curve` automates this process by computing through all the lambda and return all the coordinates ( $TPR_\lambda, FPR_\lambda$ ).

### ROC Curve of different methods with same $\frac{p}{n}$ ratio.

In this section, the ROC Curves of methods  $\hat{E}_1$  and  $\hat{E}_2$  will be compared in parallel with  $p$  and  $n$  being controlled.

First of all, for  $p = 20$  and  $n = 1000$ , which leads to the ratio  $\frac{p}{n} = 0.02$ , we have the following ROC curve (Figure 2.3.2.2). In this low  $\frac{p}{n}$  ratio settings, a large enough volumn of training data will help to achieve a good optimal performance on both methods and  $\hat{E}_2$  has slightly higher  $AUC$  than the  $\hat{E}_1$ . More specifically,  $\hat{E}_1$  has  $AUC = 0.99199$  while  $AUC = 0.99140$  for  $\hat{E}_2$ . (Under 50-times iterations setting).

Secondly, for  $p = 100$  and  $n = 100$ , which leads to the ratio  $\frac{p}{n} = 1$ , we have the following ROC curve (Figure 2.3.2.3). In this case, we have a low number of observations and also a relatively high  $\frac{p}{n}$  ratio, both methods achieved a lower performance compared to the last case. In detail,  $\hat{E}_1$  has  $AUC = 0.67574$  while  $AUC = 0.74338$  for  $\hat{E}_2$ . (Under 50-times iterations setting). Moreover, for  $p = 100$  and  $n = 200$ , which corresponds to the ratio  $\frac{p}{n} = 2$ , ROC curve (Figure 2.3.2.4) is plotted.  $\hat{E}_1$  has  $AUC = 0.43343$  while  $AUC = 0.59768$  for  $\hat{E}_2$ .



In conclusion, by conducting more simulations on different  $\frac{p}{n}$  settings, Table 2.3.2.5 is generated to show all the outcomes. **Obviously, by increasing the  $\frac{p}{n}$  ratio from 0.02 to 2, Both  $\hat{E}_1$  and  $\hat{E}_2$  had their performance decreased over the ratio.** When  $\frac{p}{n} = 0.02$ , both methods achieved impressive performance which is larger than  $AUC = 0.99$ . In the  $n = p$  situation, Both methods perform worse than the last case, while  $\hat{E}_1$ 's  $AUC$  decreases 31.88% and  $AUC$  decreases 25.02% for  $\hat{E}_2$  from  $\frac{p}{n} = 0.02$  case. In the end, if  $\frac{p}{n} = 2$ , we can see that  $AUC(\hat{E}_1)$  dropped below 0.5 for the first time, while  $AUC(\hat{E}_2)$  still higher than 0.5.

Vertically, **under each  $\frac{p}{n}$  setting,  $\hat{E}_2$  will usually achieve better result than the  $\hat{E}_1$ ,** although both of the methods perform very well in  $\frac{p}{n} = 0.02$ , in which big enough data allows lasso and linear regression to inference the hidden structure of the edge set. Similarly, both methods perform not well in the setting  $\frac{p}{n} = 2$ , in which “lack of data” leads to this outcome.