1 Supplementary material

1.1 Justification of degree separation for nodewise regression

Let’s assume that there are \( r \) hub and \( p - r \) non-hub genes in the network. Let’s denote the degree of hub and non-hub genes by \( \bar{d}_i, i = 1, \ldots, r \) and \( \tilde{d}_j, j = r + 1, \ldots, p \), respectively. Assume that \( \bar{d}_i > n, i = 1, \ldots, r \) and \( \tilde{d}_j < n, j = r + 1, \ldots, p \) such that for hub genes the Lasso estimation involves more coefficients than samples to estimate, while for non-hub genes it involves less coefficients than the samples. In case of hub genes, it leads to an underestimation problem, meaning that some coefficients cannot be uniquely estimated. In contrast, for non-hubs, the coefficients can be uniquely estimated using large penalty parameters. If we assume that each hub node is connected to most non-hub nodes, some edges connected to hubs can be recovered by the edges estimated from non-hub nodes. Therefore, the nodewise regression is able to partially infer the edges connected to hub nodes where the number of false positive edges is controlled for large penalty parameters. Simulations with nodewise regression demonstrate that the degree of hub and non-hubs can be separated for the intermediate values of the penalty parameter (Supplementary Figure S1).

1.2 Convergence of degree estimates of DW-Lasso

One main feature of DW-Lasso is the convergence of degree (weight) estimates that is demonstrated in Supplementary Figures S2 and S3. For a carefully chosen

![Figure 1: Heatmap of weights computed for different values of penalty parameter using nodewise regression. Initially for small penalty values, all genes have similar weights. For intermediate penalty values, there is a separation of weights between hub and non-hub genes (indicates separation of degrees between hubs and non-hubs). For large penalty values, the network is very sparse and hence all genes have the similar weights.](image-url)
Figure 2: The convergence of the weight $w^{(k)}$ for DW-Lasso method demonstrated for $p = 50$, $n = 40$, $\lambda_1 = 0.5$. (A) The trajectories of weights (red lines) over different iterations that converge to the steady state. Blue lines indicate the true weights computed from the true network. Initially, all weights are same and equal to one. (B) The prediction accuracy of the DW-Lasso increases with the number of iterations.

$\lambda_1$ as described in Supplementary Section 1.3, most of the time, the DW-Lasso weight estimates converge close to the weights determined by the true graph (Supplementary Figure S2A and Supplementary Figure S3). The performance of DW-Lasso improves with the number of iterations and achieves the best estimates in terms of AUROC at steady state (Supplementary Figure S2B). For prediction accuracy, the convergence of the estimates is highly dependent on the choice of the penalty parameter $\lambda_1$. In particular, if $\lambda_1$ is chosen outside of the high performance region (see Section 3.4), the weight estimates of DW-Lasso converge to different steady states with low prediction accuracy.

### 1.3 Choosing penalty parameters

In this section, we discuss how to select the penalty parameter $\lambda_1$ for the degree estimation step. Therefore, we propose to employ a stability selection [8] to choose the penalty parameters from the data. According to [8], we denote a set of estimated graphs as $\hat{G}^\lambda = \{(i, j) \in E; \hat{\Theta}_{ij} \neq 0\}$ obtained for every value $\lambda \in \mathbb{R}_+$. Denote a set of samples as $I_n = \{1, \ldots, n\}$. Let $I$ be a random subsample of $I_n$ of size $n/2$ drawn without replacement. If a random subsample $I$ is drawn $J$ times, for every set $D \subseteq E$, one can define the probability of being in the selected set $\hat{G}^\lambda$

$$\hat{\Pi}_D^\lambda = J^{-1} \sum_{j=1}^{J} \mathbb{1}\{D \subseteq \hat{G}^\lambda(P)\}$$

(1)

For a set of regularization parameters $\Lambda$ and a cutoff $\pi_{thr}$, the set of stable edges is defined as

$$\hat{G}^{\text{stable}} = \{D : \max_{\lambda \in \Lambda} \hat{\Pi}_D^\lambda \geq \pi_{thr}\}, \quad \pi_{thr} \in (0, 1)$$

(2)

where the edges with high selection probabilities are selected, while the edges with low selection probabilities are excluded. As justified in [8], for $\pi_{thr} \in$
Figure 3: The convergence of the degree weighted Lasso (DW-Lasso). Blue circles indicate the true weights; red circles indicate the predicted weights. The plots indicate that weights converge closely to true weights at later iterations.

(0.5, 1), it is possible to control the number of falsely selected edges \( W \) given the expected number of selected edges \( q_A \), the total number of edges \( \bar{q} \) and the threshold parameter \( \pi_{thr} \)

\[
E(W) \leq \frac{q_A^2}{\bar{q}(2\pi_{thr} - 1)}
\]

under the assumption that the DW-Lasso performs better than the random guess (exchangeability condition, for more information see [8]). For a user-defined \( \pi_{thr} \) and given number of maximum false positives \( E(W) \), we compute the expected number of edges \( q_A \) in the graph. Using the computed value of \( q_A \), we select the penalty parameter \( \lambda_1 \).

1.4 Graph generation procedure

To generate a hub graph, we generate a sparse symmetric adjacency matrix \( A \in \mathbb{R}^{p \times p} \) with the edge probability \( p_1 \). We next randomly select a set of hub nodes of size \( h \) which is a predefined parameter. Finally, we generate a set of neighboring nodes around hubs with probability \( p_2 \). The parameters \( p_1 \) and \( p_2 \) allow us to generate various hub graphs with different sparsity levels. In our case, we set \( p_1 = 0.01 \) and \( p_2 = 0.95 \).

We also evaluate the performance of our method on scale-free graphs [1], [3]. We use a linear preferential attachment approach to generate an adjacency matrix \( A \in \mathbb{R}^{p \times p} \) for a scale-free graph with degree exponent \( \gamma \). The approach
Given the adjacency matrix $A \in \mathbb{R}^{p \times p}$, we uniformly generate a new weighted adjacency matrix

$$B_{ij} \overset{i.i.d.}{\sim} \begin{cases} \text{Unif}([-0.75, -0.25] \cup [0.25, 0.75]), & \text{if } A_{ij} \neq 0 \\ 0, & \text{if } A_{ij} = 0 \end{cases}$$

Finally, we convert $B$ into a symmetric matrix $\tilde{B} = (B + B^T)/2$. We transform the matrix $\tilde{B}$ to the concentration matrix using the expression

$$\Theta = B + (0.1 + |\lambda_{\min}(B)|)I,$$

where $\lambda_{\min}(.)$ is the smallest eigenvalue of the matrix.

Given the concentration matrix $\Theta$, we generate samples that follow a multivariate Gaussian distribution with mean $\mathbf{0}$ and covariance matrix $\Sigma = \Theta^{-1}$. In
Figure 5: Strength of association between gene sets from GO ontology and networks inferred with DW-Lasso, nodewise regression (MB-Lasso) and Glasso from Kidney Clear Carcinoma (Knet score). The comparison of strength of association of gene sets for (left) DW-Lasso and Glasso networks, and DW-Lasso and nodewise regression networks (middle). (Right) Quantification of genesets that significantly cluster on inferred networks.

Figure 6: Strength of association between gene sets from GO ontology and networks inferred with DW-Lasso with different penalty parameters from Kidney Clear Carcinoma (compactness score).

the final step, the data is standardized to mean 0 and standard deviation $\sigma = 1$.

1.5 Quantifying the functional content of a graph

To quantify the association between publically available gene sets and the inferred networks, we compute two metrics: Knet score implemented in SANTA package [2] that takes into account the global topology of the network, and compactness score that quantifies the average distance between the genes in the network [6]. Knet function is a modified form of Ripley’s K-function [5] and is defined as

$$K_{net}(s) = \frac{2}{(kn)^2} \sum_{i} k_i \sum_{j} (k_j - \bar{k}) I(d(i,j) \leq s) \quad (5)$$

where, $k_i$ is the phenotype observed at gene $i$, $\bar{k} = \frac{1}{n} \sum_{i=1}^{n} k_i$, $d(i,j)$ is the shortest distance between two hits in the network, and $I(d(i,j) \leq s)$ is the indicator function which equals 1, if the distance $d(i,j)$ between hits $i$ and $j$ is...
smaller or equal to \( s \), and 0 otherwise. For a given gene set \( M \), the compactness score is defined as

\[
CS(M) = \frac{2 \sum_{i,j \in M; i < j} d(M_i, M_j)}{|M|(|M| - 1)}
\]

where \( d(M_i, M_j) \) is the shortest distance between any two pairs of genes in gene set \( M \), and \( |M| \) is the cardinality of gene set \( M \).

References


