README

1. Team members

Thomas P Wytock
e-mail: t-wytock@northwestern.edu
Northwestern University

István Kovács
e-mail: istvan.kovacs@northwestern.edu
Northwestern University

Contributions: TPW wrote the code and developed the prediction algorithm. IK conceived the algorithm and supervised the development. Both authors wrote and edited the submission materials.

2. Code description

predict_ppis.py

This file takes an input network as input in edge list in tab-delimited format. It returns the statistics requested by the challenge. Please note that the file `L3.cpp` must be compiled before running the python script.
To run the program, simply type `python predict_ppis.py -s 0.95 -K 3 -C 0 -a 3`
Hyperparameters:

- $M = 0.95\lambda_1$, a constant representing the noise magnitude in the spectrum, λ_1 is the largest eigenvalue of graph matrix. It functions as a soft cutoff for parts of the adjacency matrix spectrum below threshold.
- $K = 3$, a scaling constant to adjust the n^{th} root model.
- $C = 0$, an offset to adjust the cube root model.
- $a = 3$, an exponent to adjust the n^{th} root model.

L3.cpp

Source code for the $L3$ method [1]. To compile this code, simply type `g++ -O3 L3.cpp -o L3.out`.

binheap_2.h

Header file implementing binary heap for the $L3$ method.

3. Computing environment

Operating system	Linux
Processor Name	Intel(R) Xeon(R) Gold 6230R
Processor Speed	2.1 GHz
Number of Processors	52
Memory	1 TB

4. Packages used

pandas	1.1.3
numpy	1.19.2
networkx	2.5
scikit-learn	0.23.25

5. Additional dataset

We used information that mapped each interaction to each assay to generate the external validation predictions. This information is contained in the file `HURI_DATA.txt`.

6. **Method description**

6.1. **MATHEMATICS FOR THE THREE METHODS.** We use an unpublished method, currently in development in the Kovacs group. Our starting point is an edge prediction model of the form

$$\hat{s} = \mathbf{R}\mathbf{A}\mathbf{R}^{\intercal} \tag{1}$$

where A is the $n \times n$ adjacency matrix of the observed protein-protein interaction network, R is an $n \times n$ similarity matrix to be determined, and n is the number of proteins in the screen. Motivated by a generalized Wiener filter, we assume that

$$\mathbf{R} = \mathbf{S}(\mathbf{S}+\mathbf{N})^{-1}, \tag{2}$$

where S and N are $n \times n$ "signal" and "noise" matrices respectively.

Recognizing the success of L3 models in PPI prediction, we take $\mathbf{S} = |\mathbf{A}|$ and focus on finding an appropriate noise model $\mathbf{N}$. We consider two noise models, the first is

$$\mathbf{N} = K\mathrm{diag}\left(d_i^{1/a}\right) + C, \tag{3}$$

where d_i is the degree of the i^{th} node, and K is a proportionality constant that can be specified by the user or optimized using cross-validation. This model is referred to as the "n^{th}-root" noise model. We have found that $a = 3$, $K = 3$, and $C = 0$ work reasonably well.

Alternatively, we could consider a N matrix that is approximated by N', which is simultaneously diagonalizable with $|A|$ with associated eigenvalues μ_i. Then we can rewrite Eq. (1) as:

$$\hat{s} = \mathbf{P}\frac{|\lambda_i|^2\lambda_i}{(|\lambda_i| + \mu_i)^2}\mathbf{P}^{\intercal}. \tag{4}$$

We find that most of the predictive benefit of this model can be achieved by setting $\mu_i = M$, which can be optimized or specified by the user. We have found that $M = 0.95$ works well.

6.2. **MERGING THE PREDICTIONS.**

6.2.1. ***Internal merging.*** Each of the three methods generates its own list of scores, which are ranked with the highest score being 1 and the lowest score being 0. These ranks are averaged across the three methods for each pair of proteins yielding a single set of predictions. This averaged list of predictions is evaluated against the test set in terms of the required statistics.

6.2.2. ***External merging.*** For the external validation, predictions are made for all three methods for each assay. Within each assay, the predictions are merged and averaged as described in the preceding subsection. Since the assays cover largely disjoint sets of genes, the absence of a prediction of a particular pair in one assay can result from that assay's failure to cover the pair in question, or it may reflect a genuine absence of interaction. Meanwhile, pairs that are highly ranked in all 3 assays should take precedence over pairs appearing in only a single assay.

To merge the predictions of the three assays, we take the top 10,000 entries 3 lists of pairs and concatenate them in to a matrix p_{ij}. To average the ranks for each of the i pairs, we must first assign values to the missing entries. If a pair is missing from the top 10,000 entries, but it is queried in the j^{th} assay, it is assigned a rank of 0. On the other hand, if the i^{th} pair is missing due to a lack of coverage in assay j, then $p_{ij} = \min(0.25, \min'_j p_{ij'})$. The value 0.25 is chosen so as to prioritize pairs that appear in multiple assays as these pairs are likely more robust than the top predictions in each assay. The $\min$ function is used to that the absence of a pair in assay j cannot increase the rank of the pair.

7. Time complexity analysis

The n^{th} root noise model requires solving for the eigenvalues and eigenvectors of $\mathbf{A}$ and the inversion of $\mathbf{S}+\mathbf{N}$. Since both of these are matrices of size $n \times n$, the time complexity is $O(n^3)$. The simultaneously diagonalizable noise model requires only the eigenvalues and eigenvectors of $\mathbf{A}$. The time complexity is still $O(n^3)$, but it runs faster because parameter optimizations do not require additional matrix inversions.

References Cited

[1] Kovács, I.A., Luck, K., Spirohn, K. et al. Network-based prediction of protein interactions. *Nat Commun* **10**, 1240 (2019). `https://doi.org/10.1038/s41467-019-09177-y`