The probability of edge existence due to node degree: a baseline for network-based predictions

Authors

- **Michael Zietz**
  ⓘ 0000-0003-0539-630X · ⓖ zietzm · 🐦 ZietzMichael
  Department of Physics & Astronomy, University of Pennsylvania, Philadelphia, Pennsylvania, United States of America; Department of Systems Pharmacology and Translational Therapeutics, University of Pennsylvania, Philadelphia, Pennsylvania, United States of America · Funded by ['Roy and Diana Vagelos Scholars Program in the Molecular Life Sciences', 'the Gordon and Betty Moore Foundation (GBMF4552)']

- **Daniel S. Himmelstein**
  ⓘ 0000-0002-3012-7446 · ⓖ dhimmel · 🐦 dhimmel
  Department of Systems Pharmacology and Translational Therapeutics, University of Pennsylvania, Philadelphia, Pennsylvania, United States of America · Funded by Pfizer Worldwide Research, Development, and Medical; the Gordon and Betty Moore Foundation (GBMF4552)

- **Christopher Williams**
  · ⓖ chrsunwil
  Department of Systems Pharmacology and Translational Therapeutics, University of Pennsylvania, Philadelphia, Pennsylvania, United States of America

- **Michael W. Nagle**
  ⓘ 0000-0002-4677-7582 · ⓖ naglem · 🐦 MikeNagle84
  Internal Medicine Research Unit, Pfizer Worldwide Research, Development, and Medical

- **Casey S. Greene**
  ⓘ 0000-0001-8713-9213 · ⓖ cgreene · 🐦 greenescientist
  Department of Systems Pharmacology and Translational Therapeutics, University of Pennsylvania, Philadelphia, Pennsylvania, United States of America · Funded by ['Pfizer Worldwide Research, Development, and Medical', 'the Gordon and Betty Moore Foundation (GBMF4552)', 'the National Institutes of Health (R01 HG010067)']

Abstract

Networks of biomedical data rarely consist of all true relationships. Instead, networks contain spurious relationships while omitting actual relationships. How a network deviates from the real set of relationships is often biased according to node degree, resulting from processes such as inspection bias and experimental methods. While degree is subject to potentially substantial biases, link prediction methods can be strongly affected by degree. In the present work, we introduce a network permutation framework to quantify the effect of node degree on network-based methods and prediction tasks. We introduce the "edge prior" to quantify the probability that two nodes are connected based only on their degree. After demonstrating that this prior feature shows excellent discrimination and calibration performance for 20 different biomedical networks (16 bipartite, 3 undirected, 1 directed), we conclude that our prior feature represents a suitable baseline for network link prediction tasks, as performance exceeding the baseline is attributable to factors other than degree alone. Additionally, we propose methods to incorporate network permutation and the edge prior into other predictive methods. Our results highlight the importance of degree for link prediction and provide a way to account for its effects when degree bias may be present. We have released a full implementation of our network permutation method and the edge prior as an open-source Python package on GitHub.

Introduction

Node degree

Figure 1: Degree figure

Edge prediction

Feature-degree correlation

Methods

Network permutation

Network permutation is a way to produce new networks by randomizing the connections of an existing network. Specialized permutation strategies can be devised that permute networks while retaining chosen features. Comparing between permuted and unpermuted networks gives insight to the effects of the retained network features. For example, an edge prediction method that has superior reconstruction performance on a network compared to its permutations likely relies on information that is eliminated in permutation. Conversely, identical predictive performance on true and permuted networks indicates that a method relies on information that is preserved during permutation. Network permutation is a flexible framework for analyzing other methods, because it generates networks with identical formats to the original network. We propose using network permutation to isolate degree and determine its effects in different contexts. Degree-preserving network permutation obscures true connections and higher-order connectivity information while retaining node degree, and thereby, the network's degree sequence. Thanks to the flexibility of permutation, our framework can quantify the effect of degree on any network edge prediction method.

XSwap algorithm

Hanhijärvi, et al. presented XSwap [1], an algorithm for the randomization ("permutation") of unweighted networks (Figure 3A). The algorithm picks two existing edges at random, and if the edges constitute a valid swap, exchanges the targets between the edges (Figure 2).
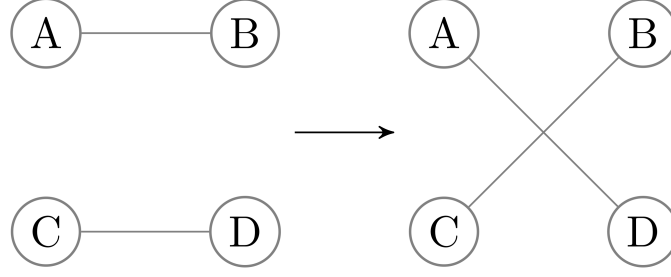

Figure 2: Graphical representation of the XSwap algorithm applied to two edges. The algorithm preserves both the source- and target-degree for all nodes.

To allow greater flexibility, we modified the algorithm by adding two parameters, "`allow_self_loops`", and "`allow_antiparallel`" that allow a greater variety of network types to be permuted (Figure 3B). Specifically, two chosen edges constitute a valid swap if they preserve degree for all four involved nodes and do not violate the above condition options. The motivation for these generalizations is to make the permutation method applicable both to directed and undirected graphs, as well as to networks with different types of nodes, variously called multipartite, heterogeneous, or multimodal networks. We provide documentation for parameter choices depending on the type of network being permuted in the GitHub repository (https://github.com/hetio/xswap). The original algorithm and our proposed modification are given in Figure 3.

A

Input: Undirected graph G, distribution ρ, and number of steps T
Output: Edge-swapped graph G_s
for i = 1,...,T **do**
   Select two edges (i,j), $(k,l) \in E(G_s)$
   if $(i,l) \notin E(G_s)$ and $(k,j) \notin E(G_s)$ **then**
      $E(\widehat{G_s}) \leftarrow E(G_s) \setminus \{(i,j),(k,l)\} \cup \{(i,l),(k,j)\}$
      $G_s \leftarrow \widehat{G_s}$ with probability $\min(\rho(\widehat{G_s})/\rho(G_s),1)$
   end if
end for

B

Input:
   Undirected graph G,
   number of steps T,
   booleans `allow_antiparallel` and `allow_loops`
Output: Edge-swapped graph G_T
Initialize $G_0 \leftarrow G$
for i = 1,...,T **do**
   Select two edges (i,j), $(k,l) \in E(G_{i-1})$
   condition 1 $(i,l) \notin E(G_{i-1})$ **and** $(k,j) \notin E(G_{i-1})$
   condition 2 `allow_antiparallel` or $((l,i) \notin E(G_{i-1})$ and $(k,j) \notin E(G_{i-1}))$
   condition 3 `allow_loops` or $(i \neq l$ and $k \neq j)$
   if all conditions met **then**
      $E(G_i) \leftarrow (E(G_{i-1}) \setminus \{(i,j),(k,l)\}) \cup \{(i,l),(k,j)\}$
   else
      $G_i \leftarrow G_{i-1}$
   end if
end for

Figure 3: **A.** XSwap algorithm due to Hanhijärvi, et al. [1]. **B.** Proposed modification to XSwap algorithm

Edge prior

We introduce the "edge prior" to quantify the probability that two nodes are connected based only on their degree. The edge prior can be estimated using the maximum likelihood estimate for the binomial distribution success probability–the fraction of permuted networks in which a given edge exists. Based only on permuted networks, the edge prior does not contain any information about the true edges in the (unpermuted) network. The edge prior is a numerical feature that can be computed for each node pair in a network, and we compared its ability to predict edges in three tasks, discussed below.

Edge prior approximation

We also considered the possibility that the probability of an edge existing across permuted networks could be written as a closed form equation involving the node pair's degree. We were unable to find a closed-form solution giving the edge prior without assuming independent node pairs, which we believe is incorrect for XSwap. Nonetheless, we discovered a good approximation to the edge prior for networks with many nodes and relatively low edge density.

Let m be the total number of edge in the network, and $d(u_i)$, $d(v_j)$ be the source and target degrees of a node pair, respectively. A good approximation of the edge prior is given by the following:

$$P_{i,j} = \frac{d(u_i)d(v_j)}{\sqrt{(d(u_i)d(v_j))^2 + (m - d(u_i) - d(v_j) + 1)^2}}$$

Further discussion of this approximate edge prior and an derivation are available in [the supplement](#).

Prediction tasks

Degree-grouping

Implementation and source code

We implemented the modified XSwap algorithm as a Python package, with the actual edge swap mechanism implemented in C++ for greater speed. In addition to functions that permute networks (represented as edge lists), the package contains utilities for computing the edge prior, converting a network between adjacency matrix and edge list formats, and for assigning unique identifiers to nodes. The Python package is available on the Python Packaging Index under the name "xswap". The full source code for our method of degree-preserving network permutation has also been made freely available (https://github.com/hetio/xswap), as has the code for the analysis, figure generation (https://github.com/greenelab/xswap-analysis), and manuscript (https://github.com/greenelab/xswap-manuscript).

Results

Figure 4: Discrimination figure

Figure 4: Discrimination figure

Figure 5: Calibration figure

Figure 5: Calibration figure

Discussion

Conclusion

References

1. **Randomization Techniques for Graphs**
Sami Hanhijärvi, Gemma C. Garriga, Kai Puolamäki
Proceedings of the 2009 SIAM International Conference on Data Mining (2009-04-30)
https://doi.org/f3mn58
DOI: 10.1137/1.9781611972795.67

Supplemental information

Approximate edge prior

To approximate the edge prior, we began by making two simplifications. First, we assumed independence between node pairs. This assumption does not actually hold for the XSwap algorithm, though it is a reasonable simplification for large, sparse networks. Second, we assumed that the XSwap process is stationary. This assumption also does not actually hold, but it was made because it significantly simplifies the problem. A single node pair has two possible states, "edge" and "no edge". These states are not transient, and they are not periodic so long as more than one possible swap exists in the network. In almost all cases, then, our simplified model of the algorithm gives the state of a node pair as an ergodic process, independent of other node pairs.

Let $A_{i,j}$ represent the existence of edge (i, j) For a given node pair, (i, j), then, let $q_{i,j}$ represent the transition probability from the "no edge" state to the "edge" state in one successful iteration of the XSwap algorithm. Let $r_{i,j}$ represent the probability of the opposite transition ("edge" to "no edge") in one successful iteration. With "no edge" represented as $[1, 0]^T$ and "edge" represented as $[0, 1]^T$, the transition matrix, P, is given by the following:

$$P^T = \begin{bmatrix} 1 - q & r \\ q & 1 - r \end{bmatrix}$$

The stationary distribution of this system should correspond to the distribution when the number of swaps goes to infinity. It can be found by computing the eigenvectors of the system, as we know that the stationary distribution vector, $\mathbf{v}$ satisfies $P^T \mathbf{v} = \mathbf{v}$. The normalized eigenvector $\mathbf{v}$ is given by

$$\mathbf{v} = \frac{1}{r/q + 1} \begin{bmatrix} r/q \\ 1 \end{bmatrix}$$

The asymptotic edge probability is therefore

$$\frac{1}{r/q + 1}.$$

Since node pairs are being treated as independent, the probability of an edge being created in one successful iteration, given that the edge does not currently exist, is the ratio of the number of edge choices involving nodes i and j to the total number of possible swaps, S. Let $d(u_i)$ represent the degree of source node i and $d(v_j)$ represent the degree of target node j.

$$q_{i,j} = \frac{d(u_i)d(v_j)}{S}$$

Similarly, the probability of an edge being eliminated in one iteration is the ratio of the number of edge choices involving (i, j) and any other valid edge to the total number of possible swaps. Let m be the total number of edges in the network.

$$r_{i,j} = \frac{m - d(u_i) - d(v_j) + 1}{S}$$

The approximate edge prior is, therefore,

$$\frac{d(u_i)d(v_j)}{m - d(u_i) - d(v_j) + 1 + d(u_i)d(v_j)}.$$

Unfortunately, we found that the above edge prior approximation is a poor approximation in many cases. We found that the following modified form (introduced in Methods) affords a superior approximation:

$$P_{i,j} = \frac{d(u_i)d(v_j)}{\sqrt{(d(u_i)d(v_j))^2 + (m - d(u_i) - d(v_j) + 1)^2}}$$

Because the modified form of the approximation offers a much superior fit to the data, we chose to include only the modified version in the Python package released, and we used only the modified form throughout our analysis.