Confidence estimation for network propagation-based biological studies

Yang Lu[1], Winston Chen[3], and William Stafford Noble[1,2]

[1]Department of Genome Sciences, University of Washington
[2]Paul G. Allen School of Computer Science and Engineering, University of Washington
[3]Department of Electrical and Computer Engineering, University of Washington

November 30, 2021

1 Introduction

Biological networks are powerful abstractions of real biological phenomena. A biological network model consists of *nodes* and *edges* that connect the nodes. In molecular biology, network nodes might represent, for example, genes or proteins, with edges representing pairwise relationships such as gene regulatory relationships, protein-protein interactions, or protein sequence similarities.

Various computational methods aim to draw inferences from biological networks by using *network propagation* [1]. These methods amplify biological signals associated with nodes of a network based on the assumption that the signal at one node is influenced by the signals of its direct network neighbors, as well as the neighbors of its neighbors, and so on. A network propagation algorithm propagates biological signals to adjacent nodes in an iterative manner until some convergence conditions are satisfied. In the end, the final score of each node is the summation of signals from that node's neighbors, their neighbors, and so on, subject to a diffusion term that attenuates signal from distal nodes in the network. The general form of network propagation is summarized in Section 2.1. Despite the widespread use of network propagation methods, an outstanding challenge is to assign confidence estimates to the resulting scores.

Here we focus on one particular application of network propagation, namely, detecting homologies between distantly related protein sequences [2]. In particular, we focus on a network propagation-based method, Rankprop [2] (Section 2.2). Given a Rankprop-induced ranking of proteins, we would like to assign each target protein a q-value, defined as the minimal false discovery rate (FDR) threshold at which the target's associated score is deemed significant. To estimate such q-values, we need a suitable null model as well as an efficient means of estimating FDR relative to that model.

In this work, we describe a method to provide confidence estimates for the predictions made via network propagation with the help of the recently proposed knockoffs framework [3]. The key idea is to embed the nodes of the network into a latent space, then generate knockoffs in the embedding space by solving an optimization problem. Specifically, we search for a knockoff embedding that fulfills two criteria: (1) the original embeddings and the knockoff embeddings are indistinguishable in their ability to reconstruct the original network, and (2) the networks reconstructed by using the original embeddings and the knockoff embeddings are very different from one another. Any network propagation algorithm can be applied to the original network and the knockoff network to obtain FDR estimates.

2 Background

2.1 Network propagation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a biological network, where $\mathcal{V} = \{v_1, \cdots, v_n\}$ denotes the set of n node nodes and $\mathcal{E}$ represents the edge set. Each edge $e_{ij} \in \mathcal{E}$ connects nodes v_i and v_j. Depending on the network type, the

edges can be either directed or undirected. Without loss of generality, we assume that edges are directed, to which the undirected case can be trivially adapted. Denote $W \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ as the normalized weight matrix, where $w_{ij} \geq 0$ represents the weight for edge e_{ij}, encoding the confidence of the interaction between v_i and v_j.

The starting point for the propagation is a vector $p^0 \in \mathbb{R}^{|\mathcal{V}|}$ of node activations, where p_i^0 is the score assigned to node v_i. At each time point k, the propagation process is essentially a random walk with restart:

$$p^k = \alpha p^0 + (1-\alpha) W p^{k-1} \tag{1}$$

where the parameter α describes the trade-off between the initial node activations and the network smoothing.

2.2 The Rankprop algorithm

In this paper, we focus on a particular network propagation algorithm, Rankprop algorithm [2], which detects homologies between distantly related protein sequences. The Rankprop algorithm takes as input a protein similarity network, where proteins are nodes and edges are weighted by scores from a pairwise sequence alignment algorithm such as PSI-BLAST [4]. Given a designated query protein in the network, Rankprop uses network propagation to return a ranking of proteins in the database ("targets") with respect to the query. Empirically, the ranking of targets relative to a given query after network propagation is superior to a ranking based on the original pairwise sequence similarity scores, in the sense that true homologs of the query (identified using 3D structural information) are more highly enriched at the top of the ranked list [2]. The details of the Rankprop algorithm are shown in Algorithm 1.

Algorithm 1 The Rankprop algorithm

Input: The protein similarity weight matrix $W \in \mathbb{R}^{n\times n}$, the initial values $p^0 \in \mathbb{R}^n$ indicating the similarity between the query protein and any other proteins in the network, the index q indicating the query protein, and the number of iterations m.

1: Normalize columns of W and p^0 so that each one sums to 1
2: Initialize $p^1 = \mathbf{0} \in \mathbb{R}^n$
3: Set $p_q^1 = 1$
4: **for** $i = 2, \ldots, m$ **do**
5: $\quad p^i = \alpha' W p^{i-1} + p^0$
6: $\quad$ Set $p_q^i = 1$ ▷ set the q-th entry of p^i to 1
7: **end for**

Output: p^m

In this work, we use a slightly modified version of the Rankprop algorithm. The formulation in Algorithm 1 has no closed-form solution because the network propagation is not properly normalized and therefore fails to converge. However, to combine Rankprop with the knockoff (Section 2.3), we need to guarantee convergence. We therefore implemented a variant of Rankprop which has a closed-form solution. This "convergent Rankprop" variant incorporates two changes relative to the original Rankprop algorithm: we removed step 4, which sets the query protein's Rankprop value to 1 after each iteration, and we require the coefficients of the two propagation terms to sum to 1 (Algorithm 2).

Algorithm 2 The convergent Rankprop aglorithm

Input: The protein similarity weight matrix: $W \in \mathbb{R}^{n\times n}$, the initial values $p^0 \in \mathbb{R}^n$, and the number of iterations m

1: Normalize W and p^0 by column
2: **for** $i = 1, \cdots, m$ **do**
3: $\quad p^i = (1-\alpha) \cdot W \cdot p^{i-1} + \alpha \cdot p^0, \quad$ where $\alpha = \frac{1}{1+\alpha'}$
4: **end for**

Output: p^m

The convergent Rankprop algorithm achieves the stationary distribution of the Rankprop vector $p \in \mathbb{R}^n$:

$$p = (1-\alpha) \cdot W \cdot p + \alpha \cdot p^0 \tag{2}$$

The vector $p \in \mathbb{R}^n$ can be obtained by solving the following linear equation:

$$\left(W - \frac{1}{(1-\alpha)} I\right) \mathrm{p} = -\frac{\alpha}{(1-\alpha)} p^0 \tag{3}$$

Hence, the problem now is reduced to finding the solution of a linear system with the design matrix $A = \left[W - \frac{1}{(1-\alpha)} I\right] \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$.

2.3 The knockoff filter

The model-X knockoffs framework [3] achieves false discovery rate (FDR) control in a feature selection setting. Assume that there exists a feature subset $\mathcal{S}_0 \subset \{1, \ldots, p\}$ such that, conditional on features in $\mathcal{S}_0$, the response Y_i is independent of features in the complement $\mathcal{S}_0^c$. Then for a subset of features $\widehat{S} \subset \{1, \ldots, p\}$ selected by some feature selection procedure, the FDR is defined as

$$\mathrm{FDR} = \mathbb{E}[\mathrm{FDP}] \text{ with } \mathrm{FDP} = \frac{|\widehat{S} \cap \mathcal{S}_0^c|}{|\widehat{S}|},$$

where $|\cdot|$ stands for the cardinality of a set.

The knockoff filter achieves FDR control in two steps: construction of knockoff features followed by filtering using knockoff statistics. For the first step, we define the knockoff features as follows:

Definition 1 ([3])**.** Model-X knockoff features for the family of random features $\mathbf{x} = (X_1, \ldots, X_p)^T$ are a new family of random features $\tilde{\mathbf{x}} = (\tilde{X}_1, \ldots, \tilde{X}_p)^T$ that satisfy two properties:

1. $(\mathbf{x}, \tilde{\mathbf{x}})_{\mathrm{swap}(\mathcal{S})} \overset{d}{=} (\mathbf{x}, \tilde{\mathbf{x}})$ for any subset $\mathcal{S} \subset \{1, \ldots, p\}$, where swap$(\mathcal{S})$ means swapping X_j and $\tilde{X}_j$ for each $j \in \mathcal{S}$ and $\overset{d}{=}$ denotes equal in distribution, and
2. $\tilde{\mathbf{x}} \perp\!\!\!\perp Y | \mathbf{x}$, i.e., $\tilde{\mathbf{x}}$ is independent of response Y given feature $\mathbf{x}$.

According to Definition 1, the construction of knockoffs is independent of the response Y. If we can construct a set $\tilde{x}$ of model-X knockoff features, then by comparing the original features with these control features, FDR can be controlled at target level q. See [3] for theoretical guarantees of FDR control with knockoff filters.

With the constructed knockoff features $\tilde{\mathbf{x}}$, we quantify importance of each feature by computing the knockoff statistics $W_j = g_j(Z_j, \tilde{Z}_j)$ for $1 \le j \le p$, where Z_j and $\tilde{Z}_j$ represent feature importance measures for the jth feature X_j and its knockoff counterpart $\tilde{X}_j$, respectively, and $g_j(\cdot, \cdot)$ is an antisymmetric function satisfying $g_j(Z_j, \tilde{Z}_j) = -g_j(\tilde{Z}_j, Z_j)$. Note that the feature importance measures as well as the knockoff statistics depend on the specific algorithm used to fit the model. For example, in linear regression models one can choose Z_j and $\tilde{Z}_j$ as the Lasso regression coefficients of X_j and $\tilde{X}_j$, respectively, and a valid knockoff statistic could be $W_j = |Z_j| - |\tilde{Z}_j|$.

In principle, the knockoff statistics W_j should satisfy a coin-flip property such that swapping an arbitrary pair of X_j and its knockoff counterpart $\widetilde{X}_j$ only changes the sign of W_j but keeps the signs of other W_k $(k \neq j)$ unchanged [3]. A desirable property for knockoff statistics W_j's is that important features are expected to have large positive values, whereas unimportant ones should have small magnitudes symmetric around 0.

Given the knockoff statistics as feature importance measures, we sort $|W_j|$'s in decreasing order and select features whose W_j's exceed some threshold T. In particular, two choices of threshold are suggested [3, 5]

$$T = \min\left\{t \in \mathcal{W}, \frac{|\{j : W_j \le -t\}|}{|\{j : W_j \ge t\}|} \le q\right\}, \quad T_+ = \min\left\{t \in \mathcal{W}, \frac{1 + |\{j : W_j \le -t\}|}{1 \vee |\{j : W_j \ge t\}|} \le q\right\}, \tag{4}$$

where $\mathcal{W} = \{|W_j| : 1 \leq j \leq p\} \setminus \{0\}$ is the set of unique nonzero values obtained by $|W_j|$'s, and $q \in (0, 1)$ is the desired FDR level specified by the user. Candes *et al.* [3] proved that under mild conditions on the W_j's, the knockoff filter with T controls a slightly modified version of FDR, and the knockoff$_+$ filter with T_+ controls the exact FDR.

3 Approach

In this section, we integrate the idea of applying knockoff filters to the Rankprop setting so as to achieve FDR control relative to the ranking of nodes by their importance scores. Specifically, the approach involves first constructing a knockoff network (Section 3.1) which is strictly subject to both the knockoff definition (Definition 1) and the requirements of the network propagation algorithm (*i.e.,* edge nonnegativity and column stochasticity). From this knockoff network, we obtain knockoff Rankprop scores (Section 3.2), and we use these scores to estimate FDR.

3.1 Constructing the knockoff for the protein similarity network

We aim to couple the convergent Rankprop algorithm with the knockoff procedure to obtain confidence estimates. Specifically, following the knockoff procedure in Section 2.3, we want to construct a knockoff design matrix $\widetilde{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ to mimic the correlation structure of A. That is,

$$[A\ \widetilde{A}]^T[A\ \widetilde{A}] = \begin{bmatrix} A^TA & A^T\widetilde{A} \\ \widetilde{A}^TA & \widetilde{A}^T\widetilde{A} \end{bmatrix} = \begin{bmatrix} A^TA & A^TA - \text{diag}(\mathbf{s}) \\ A^TA - \text{diag}(\mathbf{s}) & A^TA \end{bmatrix} \tag{5}$$

where $\mathbf{s}$ is a $|\mathcal{V}|$-dimensional nonnegative vector. Hence, corresponding non-diagonal entries of A^TA and $A^T\widetilde{A}$ are equal. To ensure that the knockoff procedure has good statistical power, the entries of $\mathbf{s}$ should be as large as possible so that A and $\widetilde{A}$ are not too similar to each other.

To find a knockoff matrix that satisfies Equation 5, we construct a knockoff adjacency matrix $\widetilde{W} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, so that the corresponding knockoff design matrix $\widetilde{A} = \left[\widetilde{W} - \frac{1}{(1-\alpha)}I\right]$ mimics the correlation structure of A. We construct the unitary matrix $U \in \mathbb{R}^{|\mathcal{V}|\times d}$, the original embedding matrix $P \in \mathbb{R}^{|\mathcal{V}|\times d}$, and the knockoff embedding matrix $\widetilde{P} \in \mathbb{R}^{|\mathcal{V}|\times d}$ in the following manner:

$$\begin{aligned} I &= U^TU \\ A &= UP^T \\ \widetilde{A} &= U\widetilde{P}^T \end{aligned} \tag{6}$$

where the embedding dimensionality $d \ll |\mathcal{V}|$ is a user-specified parameter.

By combining the swappable condition in Definition 1 and Equation 6, the knockoff adjacency matrix should satisfy

$$\begin{aligned} \left[A^TA\right]_{ij} &= \mathbf{p}_i\mathbf{p}_j^T \\ \left[A^TA\right]_{ij} &= \widetilde{\mathbf{p}}_i\widetilde{\mathbf{p}}_j^T \\ \left[A^TA - \text{diag}(\mathbf{s})\right]_{ij} &= \mathbf{p}_i\widetilde{\mathbf{p}}_j^T \\ \left[A^TA - \text{diag}(\mathbf{s})\right]_{ij} &= \widetilde{\mathbf{p}}_i\mathbf{p}_j^T \end{aligned} \tag{7}$$

Thus, given an input network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the network knockoff generator constructs a perturbed network $\tilde{\mathcal{G}} = (\tilde{\mathcal{V}}, \tilde{\mathcal{E}})$, where the node set $\tilde{\mathcal{V}}$ is identical to $\mathcal{V}$ but the edge set $\tilde{\mathcal{E}}$ is totally different from $\mathcal{E}$. Analogous to the input network $\mathcal{G}$ having an embedding matrix $U \in \mathbb{R}^{|\mathcal{V}|\times d}$, $\tilde{\mathcal{G}}$ has its own embedding matrix $\tilde{U} \in \mathbb{R}^{|\mathcal{V}|\times d}$ where the i-th row $\tilde{\mathbf{u}}_i \in \mathbb{R}^d$ represents the embedding vector for node v_i.

For computational efficiency, we use singular value decomposition (SVD) to find a desirable U and P, and we narrow down the target to optimize $\widetilde{P}$ only. In doing so, SVD factorizes the matrix A into $\bar{U}\Sigma\bar{V}^T$, where $\bar{U}$ and $\bar{V}$ are orthonormal matrices and Σ is a diagonal matrix consisting of an ordered list of singular values. We can approximate the original matrix A with A_d so that $A \approx A_d = \bar{U}_d\Sigma_d\bar{V}_d^T$, where Σ_d is the

matrix composed of the top d singular values, and $\bar{U}_d$ and $\bar{V}_d$ are the first d columns of $\bar{U}$ and $\bar{V}$, respectively (which are the first d eigenvectors of AA^T and A^TA respectively). Therefore, we can simply let $U = \bar{U}_d$ and $P = \bar{V}_d\Sigma_d^T$.

To learn the second-order embeddings indicated in Equation 7, a straightforward approach is to minimize the following objective:

$$\begin{aligned} O(\widetilde{P}) = &\sum_{i,j} \left\| \left[A^TA\right]_{ij} - \widetilde{\mathbf{p}}_i\widetilde{\mathbf{p}}_j^T \right\|^2 + \sum_{i,j} \left\| \left[A^TA - \mathrm{diag}(\mathbf{s})\right]_{ij} - \mathbf{p}_i\widetilde{\mathbf{p}}_j^T \right\|^2 \\ &+ \sum_{i,j} \left\| \left[A^TA - \mathrm{diag}(\mathbf{s})\right]_{ij} - \widetilde{\mathbf{p}}_i\mathbf{p}_j^T \right\|^2 \end{aligned} \tag{8}$$

That is to say, the swappable condition ensures that the original embedding of $v_i \in \mathcal{V}$ and the knockoff embedding of $\tilde{v}_i \in \tilde{\mathcal{V}}$ are indistinguishable in reconstructing the original network $\mathcal{G}$. Note in particular that the construction of the knockoff network $\tilde{\mathcal{G}}$ relies solely on the original network $\mathcal{G}$ and is independent of the labels $\mathbf{Y}$.

Meanwhile, we want the entries of the knockoff design matrix $\widetilde{W} = U\widetilde{P}^T + \frac{1}{(1-\alpha)}I$ to be nonnegative, requiring the knockoff adjacency matrix to be nonnegative. Hence, we also minimize the following nonnegativity objective:

$$O_1(\widetilde{P}) = -\sum_{i,j} \min\left(0, \left[U\widetilde{P}^T + \frac{1}{(1-\alpha)}I\right]_{ij}\right) \tag{9}$$

Additionally, because the column stochasticity property of weight matrix W is important for the random walk to converge, we want the knockoff design matrix $\widetilde{W} = U\widetilde{P}^T + \frac{1}{(1-\alpha)}I$ to be column stochastic too. Therefore, we also minimize the following column stochasticity objective:

$$\begin{aligned} O_2(\widetilde{P}) &= \left\| \mathbf{1} \cdot \widetilde{W} - \mathbf{1} \right\|^2 \\ &= \left\| \mathbf{1} \cdot \left(U\widetilde{P}^T + \frac{1}{(1-\alpha)} \right) - \mathbf{1} \right\|^2 \end{aligned} \tag{10}$$

where $\mathbf{1}$ is a row vector with the dimensionality $|\mathcal{V}|$.

Finally, the combined objective function we optimize is defined as

$$O(\widetilde{P}) = O(\widetilde{P}) + \lambda_1 O_1(\widetilde{P}) + \lambda_2 O_2(\widetilde{P}) \tag{11}$$

where the parameters $\lambda_1 > 0$ and $\lambda_2 > 0$ balance the regularization of nonnegativity and column stochasticity, respectively.

Once we obtain the unitary matrix U and two second-order embedding matrices P and $\widetilde{P}$, we can reconstruct the knockoff adjacency matrix $\widetilde{W} = U\widetilde{P}^T$. The network $\tilde{\mathcal{G}}$ is said to be a knockoff copy of $\mathcal{G}$ if its weight matrix is $\widetilde{W}$. Because of Equation 5, $\tilde{\mathcal{G}}$ is a valid knockoff satisfying swappable condition [5].

3.2 Calculating the knockoff Rankprop values and estimating FDR

Having constructed the knockoff network $\tilde{\mathcal{G}}$, we next calculate corresponding knockoff Rankprop scores $\tilde{p}^k \in \mathbb{R}^{|\mathcal{V}|}$ by using the Rankprop algorithm shown in Algorithm 2. The knockoff statistic is defined as $\hat{p}^k = p^k - \tilde{p}^k$, and the standard procedure described in Section 2.3 can be used to estimate the q-value.

4 Methods

4.1 Defining and validating a baseline method

We compare the performance of the knockoff approach to a simple but computationally expensive empirical null model. The baseline method generates q-values in four steps (Algorithm 3). First, we generate from the given network a large collection of perturbed networks, each of which preserves the in-degree distribution

Algorithm 3 Baseline method for calculating empirical p-values and q-values

Input: The protein similarity weight matrix: $W \in \mathbb{R}^{n \times n}$, the initial values $p^0 \in \mathbb{R}^n$, the number of iterations m, and the number of perturbation K.

calculate the Rankprop vector $p \in \mathbb{R}^n$ (Algorithm 2) by using W, p^0, and m
for $k = 1, \ldots, K$ **do**
    generate a perturbed network $W_k \in \mathbb{R}^{n \times n}$ by in-degree-preserved edge rewiring
    calculate the the Rankprop vector $p_k \in \mathbb{R}^n$ (Algorithm 2) by using W_k, p^0, and m
end for
let $\mathcal{P} = \{p_1 \cup p_2 \cup \cdots \cup p_k\}$ ▷ Build the null distribution by using all perturbed network-derived Rankprop vectors
for $i = 1, \ldots, n$ **do** ▷ calculate the empirical p-value for each protein
    calculate $\text{pval}_i = \frac{1+|\{x \in \mathcal{P} | x > p_i\}|}{|\mathcal{P}|}$
end for

Output: $(\text{pval}_1, \ldots, \text{pval}_n)$, BenjaminiHochberg$(\text{pval}_1, \ldots, \text{pval}_n)$

of the original network, analogous to [6]. Specifically, we define an edge rewiring operation that selects two directed edges at random and switches their start nodes. This rewiring procedure is repeated for a pre-defined number of iterations (in our case 1,000,000 times) to produce a single perturbed network. We repeat this procedure 5000 times, yielding a collection of 5000 perturbed networks. In the second step, we run the Rankprop algorithm on each of the perturbed networks to obtain a corresponding Rankprop vector. Third, for each entry of the original Rankprop vector (i.e., the Rankprop score for each protein), we calculate its empirical p-value based on the null distribution built by using all perturbed network-derived Rankprop vectors. Fourth, we estimate the q-values for the calculated set of p-values using the Benjamini-Hochberg procedure.

5 Results

5.1 The original and convergent Rankprop algorithms perform similarly

We compared the empirical performance of the original Rankprop algorithm and the convergent variant by using the benchmarks laid out in the original Rankprop paper. In this benchmark, performance is measured using the ROC_{50} measure (i.e., the area under the receiver operating characteristic curve, up to the 50th false positive). This comparison shows that the convergent Rankprop performs nearly as well as the original Rankprop algorithm (Figure 1).

5.2 Downsampling the network does not degrade performance very much

For the sake of computational efficiency, we first test our method on subsampled protein similarity network with only 1000 proteins. As shown in Figure 2, the performance of Rankprop algorithm using the subsampled protein similarity network is nearly as good as the performance using the much larger network (Figure 2).

5.3 The baseline method yields valid p-values

As a sanity check, we test whether the empirical p-values produced by the baseline method are uniformly distributed. Specifically, we use Algorithm 3 to calculate the empirical p-values given a perturbed similarity network (by still preserving the column stochasticity) as the input. The results confirm that the baseline p-values are uniformly distributed (Figure 3A). We also plot the relationship between the Rankprop score and the corresponding p-value (Figure 3B), which shows the expected monotonic trend.

5.4 Knockoff Rankprop q-values are problematic

We next calculate the Rankprop knockoff q-values generated by our proposed method. Though the empirical p-values and knockoff q-values are not directly comparable, we should at least expect that these two types of

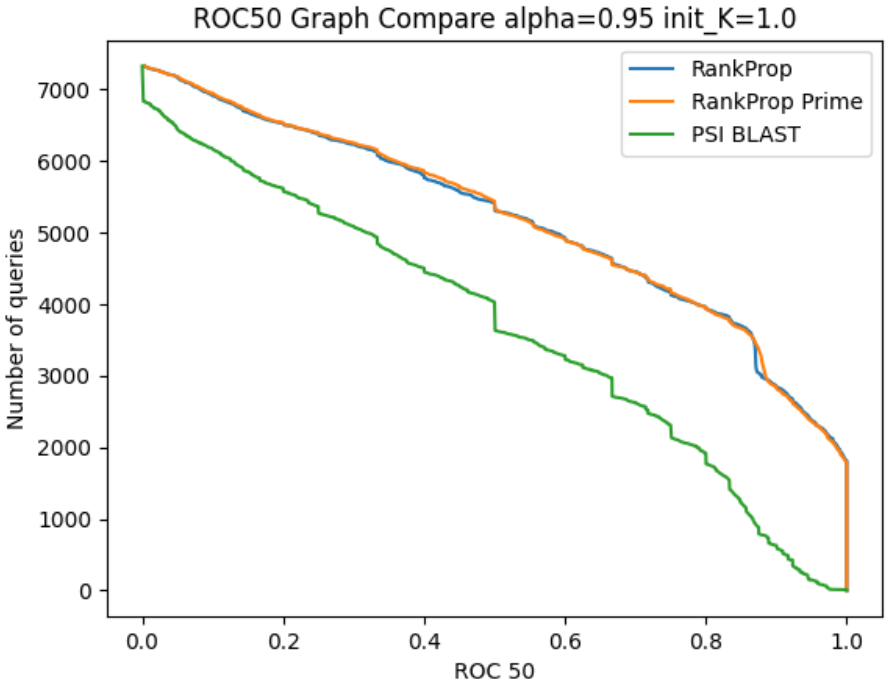


Figure 1: ROC_{50} graph of 7329 search queries with PSI-BLAAT, Rankprop, and normalized Rankprop. Labels are derived from the SCOP database, following [2].

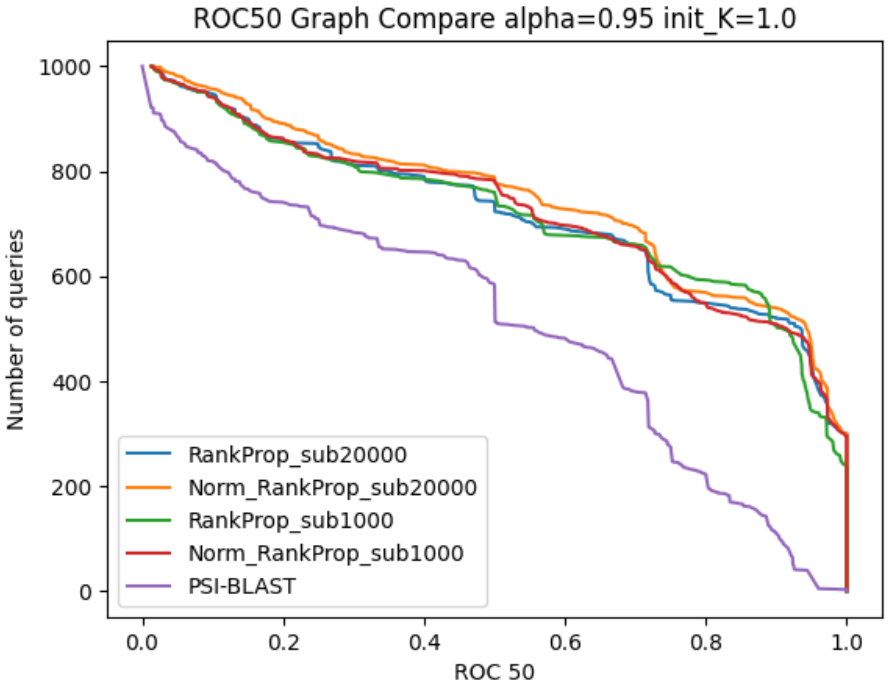
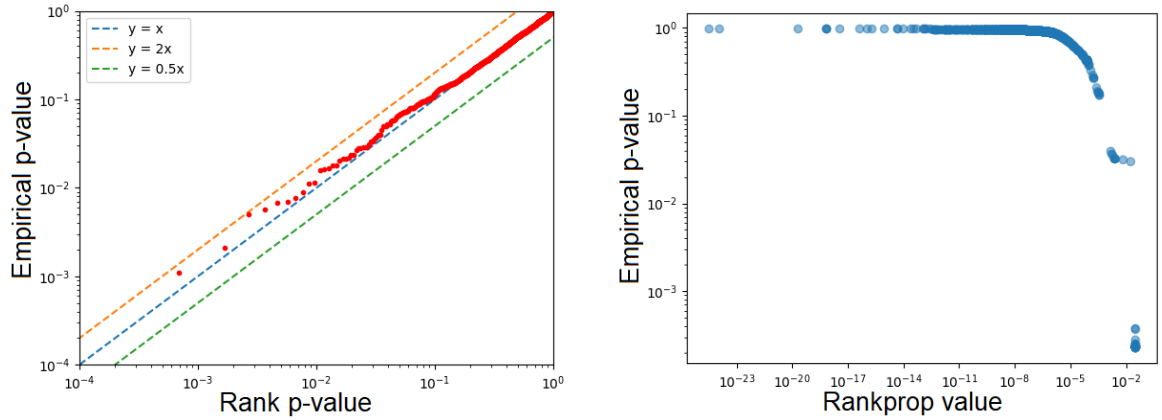


Figure 2: ROC_{50} graph of 7329 search queries with Rankprop using the protein similarity network of size 1000 and 20,000.



A B

Figure 3: (A) The figure plots the empirical Rankprop p-values produced by the baseline method, using a perturbed similarity network, given a particular query protein. The p-values are sorted and plotted (x-axis) relative to their relative rank in the sorted list (y-axis). (B) The relationship between the empirical p-values and the Rankprop values, given a particular query protein. As we expected, two types of values should be inversely correlated in the sense that smaller empirical p-values should correspond to larger Rankprop values.

values are correlated in the sense that smaller empirical p-values should also correspond to smaller knockoff q-values. Empirically, we do not observe such a trend (Figure **??**).

We further investigate the relationship between the Rankprop values and the Rankprop empirical p-values. Though the empirical p-values and the Rankprop values are not directly comparable, we expect these two types of values to be inversely correlated. Again, we do not observe the expected trend in practice (Figure 4).

Our hypothesis is that the non-uniformity of the observed knockoff p-values arises because of the sum-to-one constraint of the Rankprop vector. For the node activation vector of the original network, the Rankprop values exhibit few very large values and many very small values. By contrast, the Rankprop values obtained by using a perturbed network are more evenly distributed. Given the procedure for calculating empirical p-values (Algorithm 3), the majority of small Rankprop values will have relatively large empirical p-values.

A possible solution to overcome this problem is to break the sum-to-one constraint by dropping one node (*i.e.,* removing a specified row and column of W) in the original network. The rationale behind this

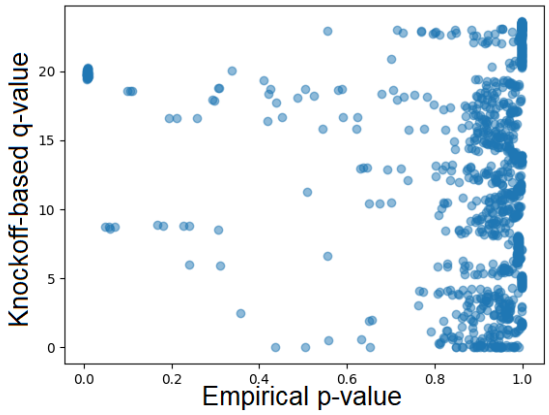


Figure 4: The relationship between the empirical p-values and knockoff q-values, given a particular query protein.

procedure is that if the sum-to-one constraint no longer exists in the original network, then there shouldn't be the same constraint in the knockoff network too. However, selecting such a node is challenging.

References

[1] Lenore Cowen, Trey Ideker, Benjamin J Raphael, and Roded Sharan. Network propagation: a universal amplifier of genetic associations. *Nature Reviews Genetics*, 18(9):551, 2017.

[2] Jason Weston, Andre Elisseeff, Dengyong Zhou, Christina S Leslie, and William Stafford Noble. Protein ranking: from local to global structure in the protein similarity network. *Proceedings of the National Academy of Sciences*, 101(17):6559–6563, 2004.

[3] Emmanuel J Candès, Yingying Fan, Lucas Janson, and Jinchi Lv. Panning for gold: Model-X knockoffs for high-dimensional controlled variable selection. *Journal of the Royal Statistical Society Series B*, 2018. to appear.

[4] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[5] Rina Foygel Barber and Emmanuel J Candès. Controlling the false discovery rate via knockoffs. *The Annals of Statistics*, 43(5):2055–2085, 2015.

[6] Molly Megraw, Sayan Mukherjee, and Uwe Ohler. Sustained-input switches for transcription factors and microrna s are central building blocks of eukaryotic gene circuits. *Genome Biology*, 14(8):R85, 2013.