Comp790-166: Computational Biology

Lecture 16

April 1, 2021

Announcements

- We will assign homework 2 next week.
- It will probably be related to multi-something integration/alignment.... :D
- Any comments on projects? How are they going?

Today

- Representing nodes with multiple relational definitions
- MASHUP + Protein Function Prediction
- Graph Alignmnet
- REGAL matrix factorization method for graph alignment.

Integrating Heterogeneous Information Sources

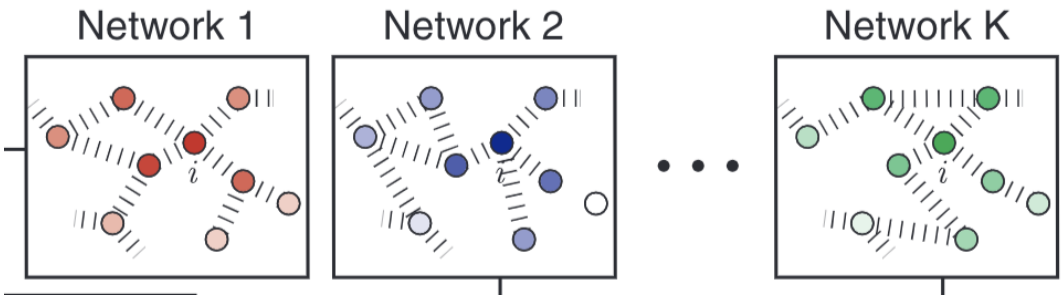

Figure: from Cho *et al.* Cell Systems. Each graph is representing a different relational definition between features.

Considering proteins, there are multiple methods for predicting whether these proteins interact .

- Physical binding
- gene expression
- co-localization
- experimentally determined
- text mined, etc.

We Seek a Unified Representations of these Nodes

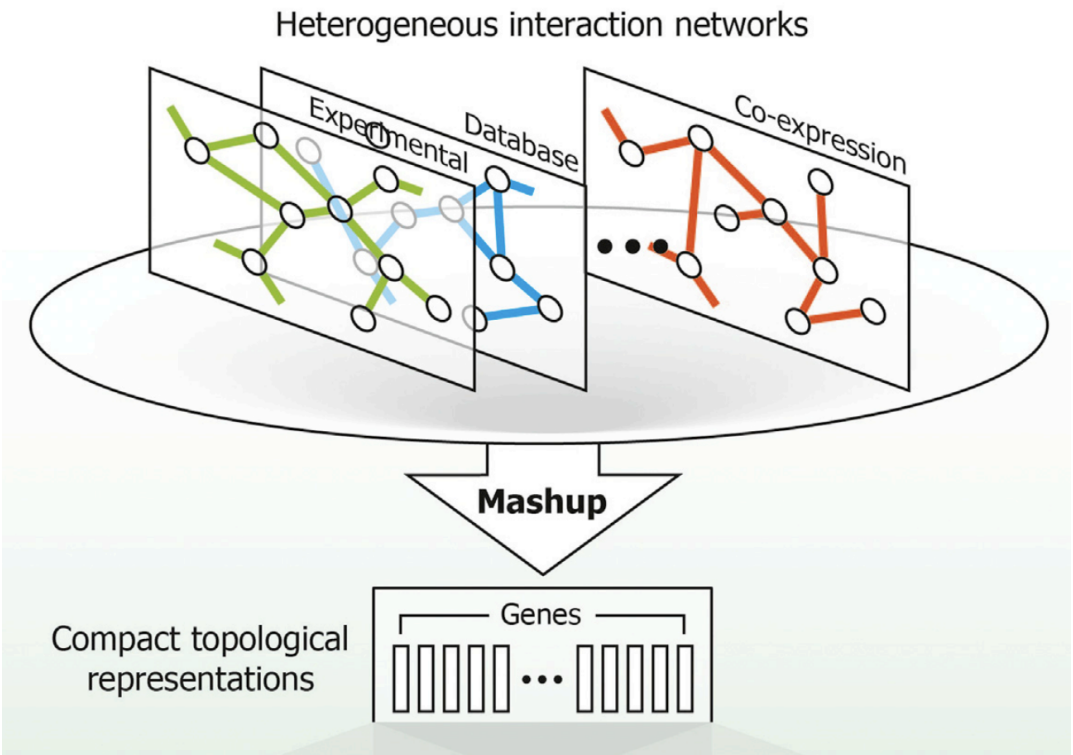

Figure: from Cho *et al.* Cell Systems. 2016.

Example from STRING

Using the STRING database, you can extract PPIs according to multiple relational definitions.

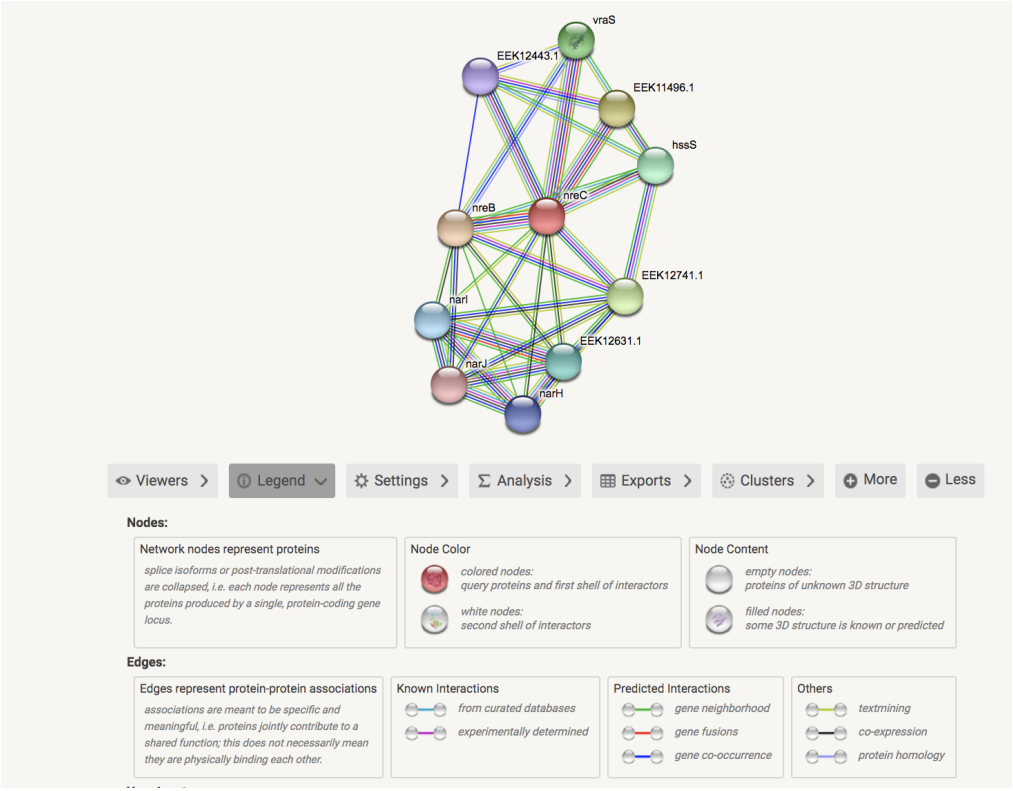

Figure: `https://string-db.org/`

Welcome Mashup

Given multiple relational definitions (e.g. multiple graphs) between a common set of nodes (features), define a consensus d-dimensional embedding vector, according to each graph.

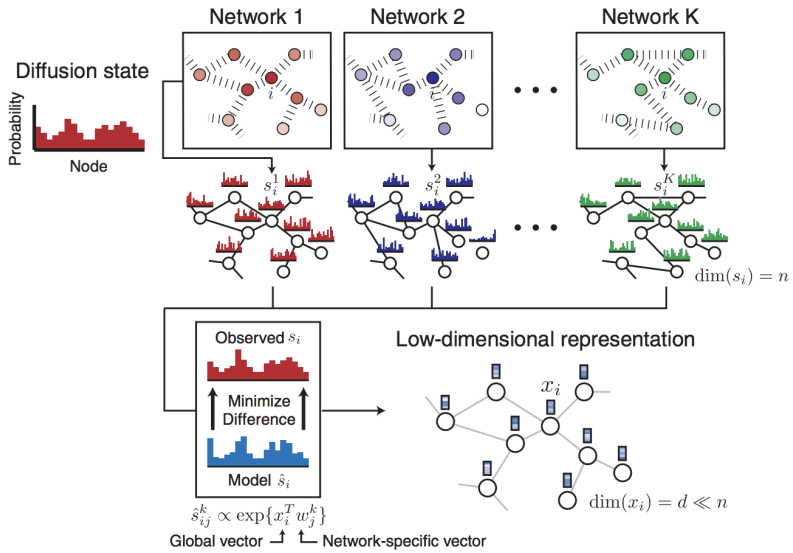

Figure: from Cho *et al.* Cell Systems. Each graph is representing a different relational definition between features.

Random Walk with Restart

- RWR is a way to account for both local and global ‘walk‘ information in the graph by giving your walker the chance to restart

But first, let's re-define the transition probability that a walker goes from node j to node i as,

$$B_{ij} = \frac{A_{ij}}{\sum_{i'} A_{i'j}}$$

RWR Formally Written

Given the transition matrix, B, the RWR from a node i is defined as,

$$s_i^{t+1} = (1 - p_r)Bs_j^t + p_r e_i$$

- p_r is the probability of restart
- $e_i(i) = 1$ and $e_i(j) = 0$ for $j \neq i$
- s_i^t is the probability of a node being visited after t steps in the random walk, starting from node i

Clarifying What is Happening Here

$$s_i^{t+1} = (1 - p_r)Bs_j^t + p_r e_i$$

- The first term corresponds to following a random edge connected to the current node
- The second term corresponds to restarting from node i.
- At some point, this reaches a stationary distribution, s_i^{∞}, or fixed point
- s_{ij} represents the probability of RWR starting at node i and ending at node j in equilibrium.

Quantifying Topological Overlap Between a Node Pair

Each node is given two vector representations, $\mathbf{w}_i, \mathbf{x}_i \in \mathbb{R}^d$

- Let $\mathbf{w}_i$ refer to the context feature of a node (e.g. per relational definition)
- Let $\mathbf{x}_i$ refer to the node feature of node i (e.g. overall)

Define a new similarity measure between nodes i and j as,

$$\hat{s_{ij}} = \frac{\exp\{x_i^T w_j\}}{\sum_{j'} \exp\{x_i^T w_{j'}\}}$$

Unpacking

$$\hat{s_{ij}} = \frac{\exp\{x_i^T w_j\}}{\sum_{j'} \exp\{x_i^T w_{j'}\}}$$

- If $\mathbf{x}_i$ and w_j are close in direction and hence have a large inner produce, then node j should be frequently visited in the random walk starting from node i.

Recap of what is happening

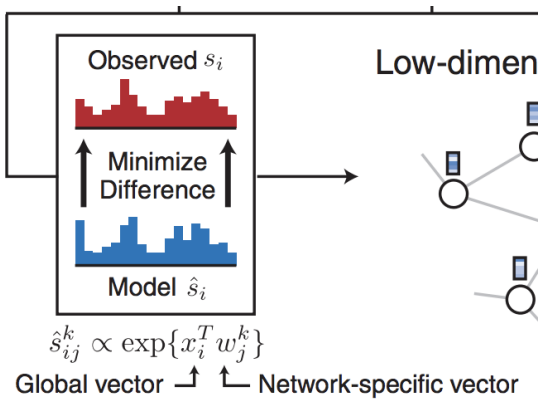

Figure: from Fig. 1. Given observed diffusion states from RWR, we should be able to find a global vector (x) and view-specific vector (y), such that a function of x and y gives a good diffusion state approximation.

Writing Out an Objective Functions for Embeddings

To find optimal d-dimensional representations for each node, x_i and w_j formulate an optimization problem that minimizes the notation between s and $\hat{s}$ as,

$$\underset{w,x}{\text{minimize}} C(s, \widehat{s}) = \frac{1}{n} \sum_{i=1}^{n} D_{KL}\left(s_i \| \widehat{s_i}\right)$$

Written out, given our definition of $\hat{s}$ gives the following (with $H(\cdot)$ denoting entropy,

$$C(s, \widehat{s}) = \frac{1}{n} \sum_{i=1}^{n} \left[-H\left(s_i\right) - \sum_{j=1}^{n} s_{ij} \left(x_i^T w_j - \log \left(\sum_{j'=1}^{n} \exp\left\{ x_i^T w_{j'} \right\} \right) \right) \right]$$

Integrating Heterogeneous Networks

You can do these RWRs on each individual network. At the same time, you can let x be fixed across all relational definitions.
Similar to what we have seen, yet adapted for modality k, we can write,

$$\widehat{s}_{ij}^{k} := \frac{\exp\left\{x_i^T w_j^k\right\}}{\sum_{j'} \exp\left\{x_i^T w_{j'}^k\right\}}$$

Writing the Objective Function Across All Modalities

Now, the objective function can be rewritten to take into account the recently-computed $\widehat{s}_{ij}^{k}s$, and sums over all modalities as,

$$\underset{w,x}{\text{minimize}} C(s,\widehat{s}) = \frac{1}{n}\sum_{k=1}^{k}\sum_{i=1}^{n} D_{KL}\left(s_i^k \| \hat{s}_i^k\right)$$

Implementation (the slow way)

To find the optimal ws and xs for each node, you could compute gradients, which turn out to be,

$$\nabla_{w_i^k} C(s, \widehat{s}) = \frac{1}{n} \sum_{j=1}^{n} \left(\widehat{s}_{ji}^k - s_{ji}^k \right) x_j$$

and

$$\nabla_{x_i} C(s, \widehat{s}) = \frac{1}{n} \sum_{k=1}^{K} \sum_{j=1}^{n} \left(\widehat{s}_{ij}^k - s_{ij}^k \right) w_j^k$$

An SVD Formulation: Setup

- Let S^k be the $N \times N$ diffusion state matrix for network k
- Also, let s_i^k be the ith column of this matrix, S^k

This matrices can be concatenated to form an $nK \times n$ matrix, S

Remember Truncated SVD?

The authors used truncated SVD as an alternative to estimating the w_i^ks and x_is as,

$$S = U\Sigma V$$

(Remember this implies that we will have some 0s on the diagonal of Σ)

- $\{w_i^k\} \rightarrow \Sigma^{1/2} U^T$
- $\{x_i\} \rightarrow \Sigma^{1/2} V$

Another Approach for $\{x_i\}$

Consider the top eigenvectors of R, where R is defined as,

$$R = \sum_{k=1}^{K} \left(\tilde{S}^k\right)^T \tilde{S}^k$$

Using the Learned $\mathbf{x}_i$s as Feature Vectors

- After Mashup each node, i has an embedding, $\mathbf{x}_i$.
- Each protein has a known function, which we can try to predict.

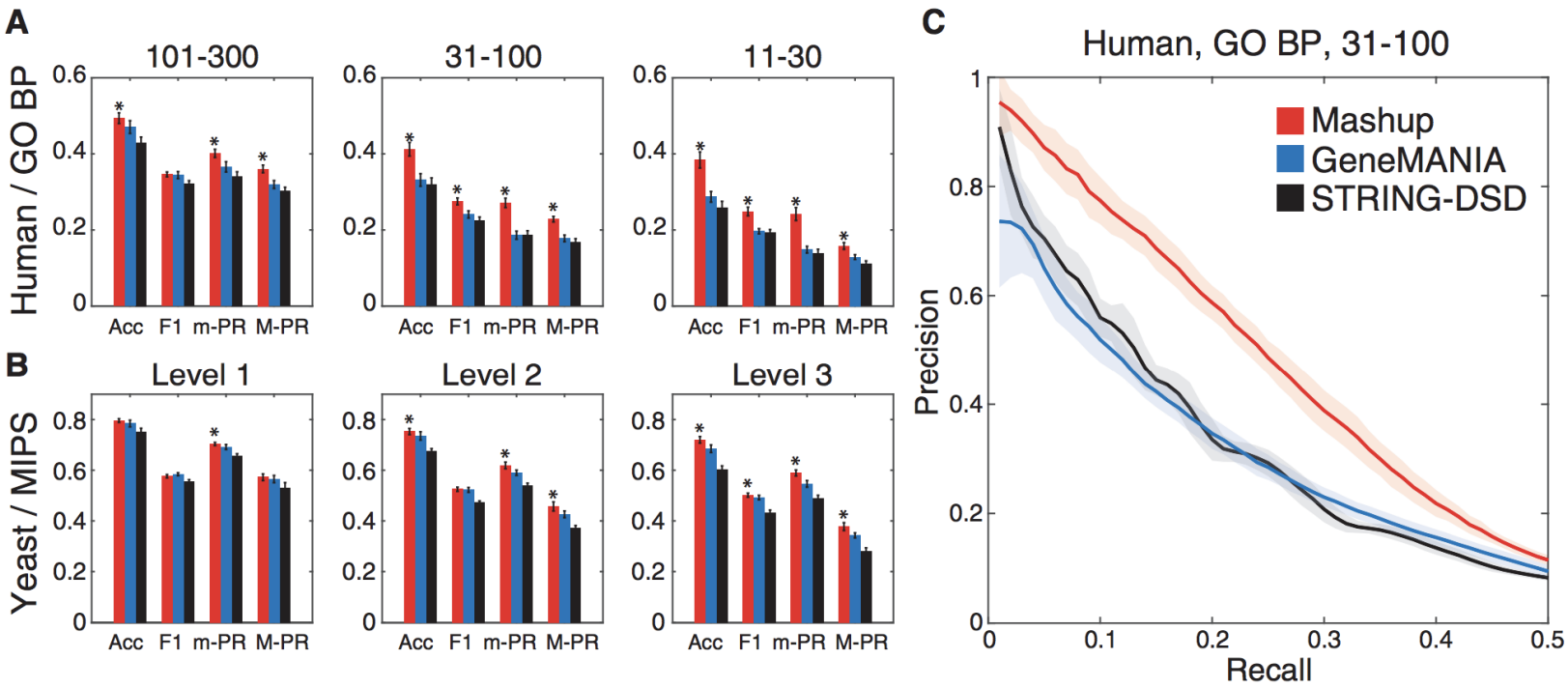

Figure: from Fig. 2. Performance is evaluated for multiple levels of annotation.

Similarly, Combining All Networks Leads to Better Protein Function Prediction

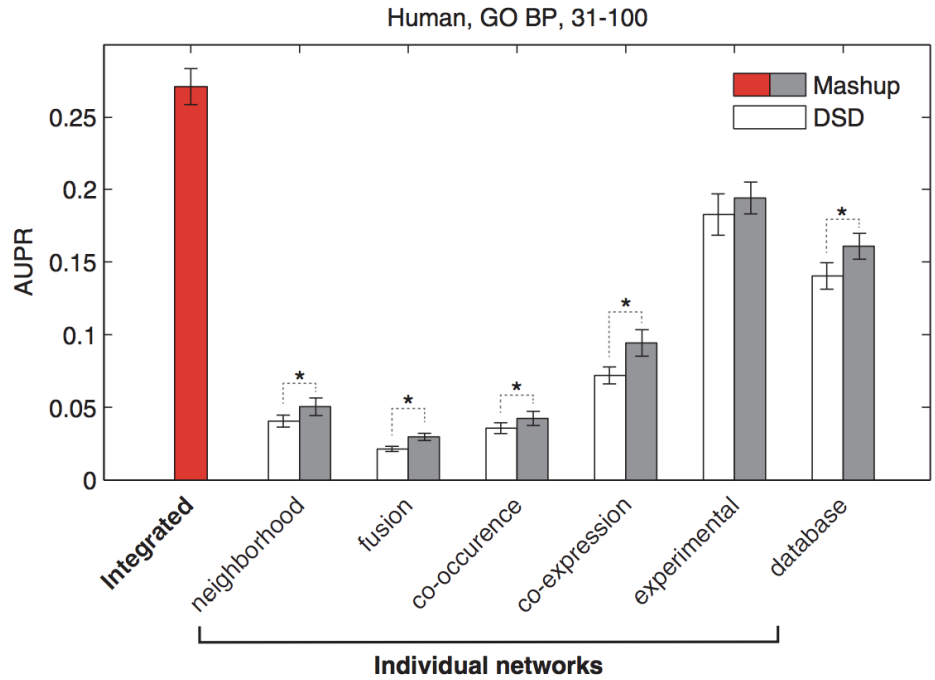

Figure: from Fig. 3. It's very reassuring to see that experimental is also a top performer!

Intuition about Parameters

The main parameters of interest is the restart probability, and the number of dimensions to keep.

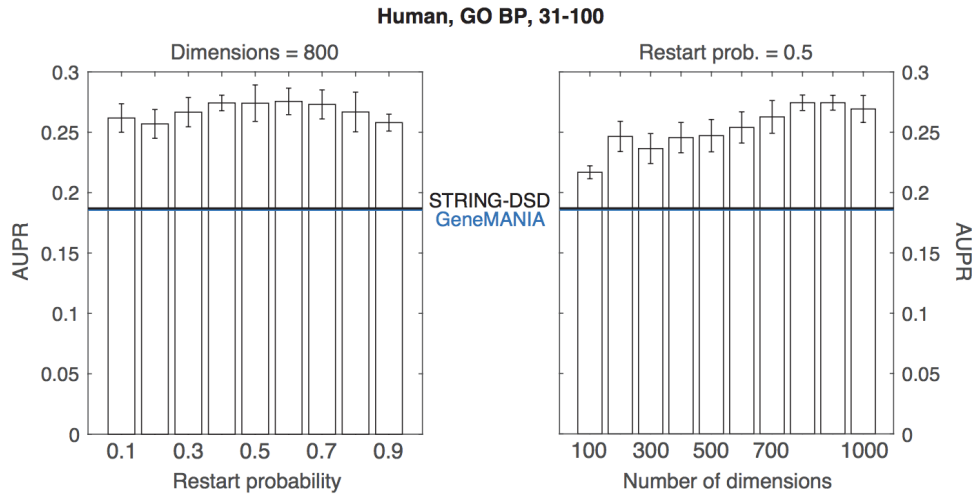

Figure: from Supp Fig. 4

*Btw, I recommend choosing your y-axis so that it is useful when making such a plot.

Mashup is More Robust to Noise in the Network

Here, noise was simulated by removing a subset of edges from the original graph.

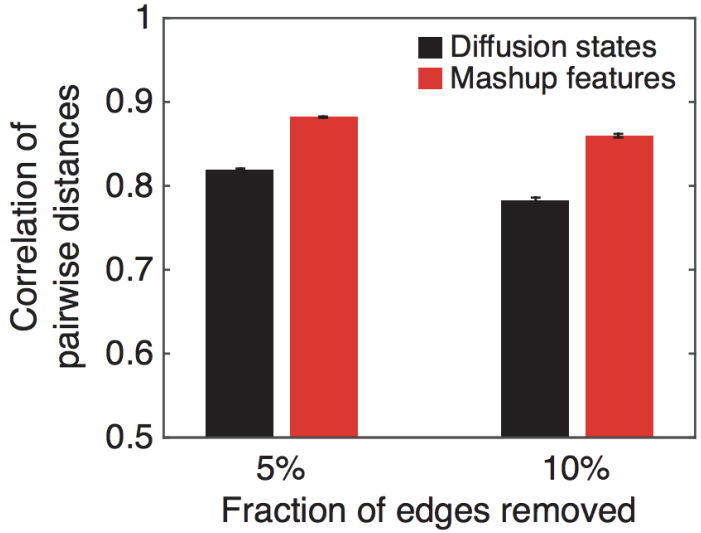

Figure: from Supp. Fig. 7. Edges were removed from the BioGrid physical interaction network. Similarities between nodes could be calculated based on diffusion state or mashup.

Since we are on the topic of multiple networks...

I'm so glad you asked. Let's talk about a related problem of graph alignment.

Problem

Here is a formal definition of the graph matching problem.

PROBLEM 1. *Given two graphs G_1 and G_2 with node-sets $\mathcal{V}_1$ and $\mathcal{V}_2$ and possibly node attributes $\mathcal{A}_1$ and $\mathcal{A}_2$ resp., devise an efficient* ***network alignment method*** *that aligns nodes by learning* ***directly comparable*** *node representations $\mathbf{Y}_1$ and $\mathbf{Y}_2$, from which a node mapping $\phi : \mathcal{V}_1 \rightarrow \mathcal{V}_2$ between the networks can be inferred.*

Figure: from from Heimann *et al.* CIKM 2018.

Overview of Regal Method

Regal → Representation Based Graph Alignment.

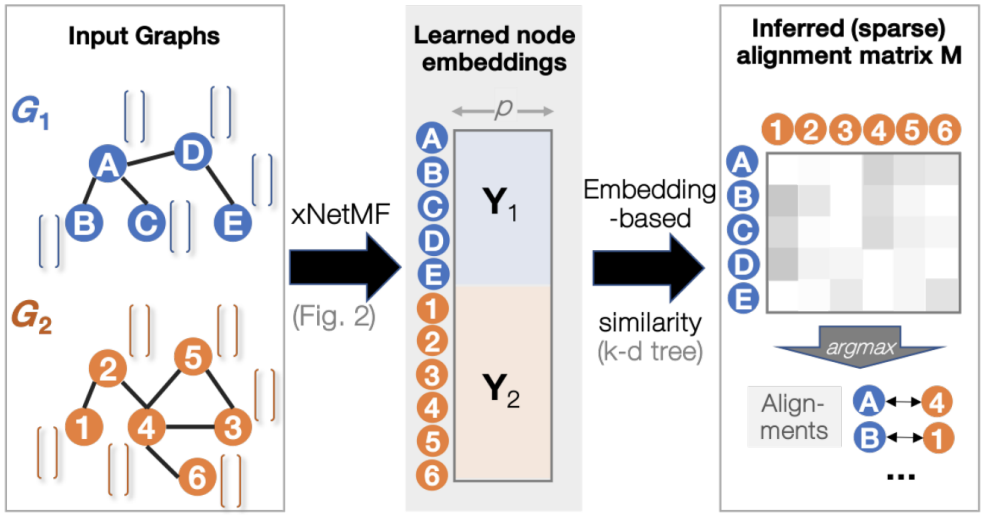

Figure: from from Heimann *et al.* CIKM 2018. Similar to Node2vec, the authors want to find a representation for each node

What does this mean for us? Aligning nodes across multiple biological networks, for example.

Typical Network Alignment Problem Formulation

Given two graphs, $\mathbf{A}_1$ and $\mathbf{A}_2$, find the permutation matrix, $\mathbf{P}$ that minimizes the following.

$$||\mathbf{P}\mathbf{A}_1\mathbf{P}^T - \mathbf{A}_2||_F^2$$

Many of the proposed solutions might start with a ‘seed‘ alignment to make the problem easier. We will see that thee REGAL solution does not require any seeds.

Defining Node Identity

- Previously (e.g. node2vec), we saw node representations were defined in terms of their neighbors through random walks.
- In this setting, of quantifying relationships between nodes between graphs, we cannot walk because we have two different graphs.
- The solution is to instead focus on nodes with similar structural roles (e.g. degree, degree of neighbors, etc).

Calculating Node Similarity Within and Between Graphs

Define distance between nodes u and v in terms of structure ($\mathbf{d}$), or attributes ($\mathbf{f}$) as,

$$\text{sim}(u, v) = \exp\left[-\gamma_s \cdot \|\mathbf{d}_u - \mathbf{d}_v\|_2^2 - \gamma_a \cdot \text{dist}\left(\mathbf{f}_u, \mathbf{f}_v\right)\right]$$

Each $\mathbf{d}_u$ is defined as,

$$\mathbf{d}_u = \sum_{k=1}^{K} \delta^{k-1} \mathbf{d}_u^k$$

Here, δ is a discount factor for greater hop distances.

Expensive Formulation

Given a factorization approach, write the similarity matrix, $\mathbf{S}$ as,

$$\mathbf{S} \approx \mathbf{Y}\mathbf{Z}^T$$

Here, $\mathbf{Y}$ gives the node-to-embedding matrix.
Intuitively, we want the following to be as close as possible to 0,

$$||\mathbf{S} - \mathbf{Y}\mathbf{Z}^T||$$

An Approximation with Landmarks

- The punchline is that $\mathbf{S}$ will be approximated with a low-rank matrix, $\tilde{\mathbf{S}}$, which is never explicitly computed!
- The solution is to choose $p \leq\leq n$ 'landmark' nodes, chosen across both graphs (G_1, G_2).
- Compute similarity between each node and each landmark. This produces an $n \times p$ matrix, $\mathbf{C}$.
- Further, the $p \times p$ landmark-to-landmark submatrix can also be extracted ($\mathbf{W}$).

The Low-Rank Matrix, $\tilde{\mathbf{S}}$

Finally, the low-rank matrix $\tilde{\mathbf{S}}$ is given as,

$$\tilde{\mathbf{S}} = \mathbf{C}\mathbf{W}^{-1}\mathbf{C}^T$$

Here $\mathbf{W}^{-1}$ is computed as the pseudoinverse of $\mathbf{W}$

- The problem is that because we need to consider similarity between all pairs of nodes, this is still an n^2 computation.

Approximation for the Embedding Matrix, $\mathbf{Y}$

THEOREM 3.1. *Given graphs $G_1(\mathcal{V}_1, \mathcal{E}_1)$ and $G_2(\mathcal{V}_2, \mathcal{E}_2)$ with $n \times n$ joint combined structural and attribute-based similarity matrix $\mathbf{S} \approx \mathbf{Y}\mathbf{Z}^T$, its node embedding matrix $\mathbf{Y}$ can be approximated as*

$$\tilde{\mathbf{Y}} = \mathbf{C}\mathbf{U}\Sigma^{1/2},$$

where $\mathbf{C}$ is the $n \times p$ matrix of similarities between the n nodes and p randomly chosen landmark nodes, and $\mathbf{W}^{\dagger} = \mathbf{U}\Sigma\mathbf{V}^{\top}$ is the full rank singular value decomposition of the pseudoinverse of the small $p \times p$ landmark-to-landmark similarity matrix $\mathbf{W}$.

So to Summarize the Entire xNetMF

Algorithm 2 xNetMF $(G_1, G_2, p, K, \gamma_s, \gamma_a)$

========== STEP 1. Node Identity Extraction ==========
for node u in $\mathcal{V}_1 \cup \mathcal{V}_2$ **do**
  for hop k up to K **do** ▷ counts of node degrees of k-hop neighbors of u
    $\mathbf{d}_u^k$ = CountDegreeDistributions($\mathcal{R}_u^k$) ▷ $1 \leq K \leq$ graph diameter
  end for
  $\mathbf{d}_u = \sum_{k=1}^{K} \delta^{k-1} \mathbf{d}_u^k$ ▷ discount factor $\delta \in (0, 1]$
end for
========== STEP 2. Efficient Similarity-based Representation ==========
========== STEP 2a. Reduced $n \times p$ Similarity Computation ==========
$\mathcal{L}$ = ChooseLandmarks(G_1, G_2,p) ▷ choose p nodes from G_1, G_2
for node u in $\mathcal{V}$ **do**
  for node v in $\mathcal{L}$ **do**
    $c_{uv} = e^{-\gamma_s \cdot ||\mathbf{d}_u - \mathbf{d}_v||_2^2 - \gamma_a \cdot \text{dist}(\mathbf{f}_u, \mathbf{f}_v)}$
  end for
end for ▷ Used in low-rank approx. of similarity graph (not constructed)
========== STEP 2b. From Similarity to Representation ==========
$\mathbf{W} = \mathbf{C}[\mathcal{L}, \mathcal{L}]$ ▷ Rows of $\mathbf{C}$ corresponding to landmark nodes
$[\mathbf{U}, \Sigma, \mathbf{V}] = \text{SVD}(\mathbf{W}^{\dagger})$
$\tilde{\mathbf{Y}} = \mathbf{C}\mathbf{U}\Sigma^{-\frac{1}{2}}$ ▷ **Embedding**: *implicit* factorization of similarity graph
$\tilde{\mathbf{Y}} = Normalize(\tilde{\mathbf{Y}})$ ▷ **Postprocessing**: make embeddings have magnitude 1
$\tilde{\mathbf{Y}}_1, \tilde{\mathbf{Y}}_2 = \text{Split}(\tilde{\mathbf{Y}})$ ▷ **Separate** representations for nodes in G_1, G_2
return $\tilde{\mathbf{Y}}_1, \tilde{\mathbf{Y}}_2$