# PROJECT REPORT FOR REU 2024: NETWORK REGRESSION...

QI KUANG, YI WEI, DAVID JIANG, AND HANBAEK LYU

Abstract. We present a novel approach to network regression by introducing the Supervised Network Dictionary Learning (SNDL) model. This model leverages k-path sampling and Supervised Matrix Factorization (SMF) to extract mesoscale structural patterns, achieving both high predictive accuracy and interpretability. By employing Block Coordinate Descent (BCD) algorithms, the SNDL model efficiently handles large-scale network data, making it a robust tool for both prediction and network affinity tasks. We validate the SNDL model with empirical experiments on datasets such as Facebook100 and BioGRID, demonstrating its applicability to both social and biological networks.

Contents

## 1. Introduction

Regression is a fundamental technique in machine learning, widely used to model and analyze the relationship between a dependent variable and one or more independent variables. By fitting a model to observed data, regression allows for the prediction of continuous outcomes, making it essential in a variety of applications, from financial forecasting to medical diagnosis [Hastie et al., 2009]. Beyond its predictive power, regression models, particularly linear regression, are highly valued for their interpretability. The coefficients of these models provide direct insights into the influence of each predictor, allowing practitioners to better understand the underlying patterns in the data. This interpretability is crucial for validating models, ensuring transparency, and supporting decision-making in fields where the rationale behind predictions is just as important as the predictions themselves [Murdoch et al., 2019].

However, extending regression models to discrete data, such as networks, presents significant challenges. Network data differs fundamentally from continuous data due to its complex structure and inherent dependencies. Unlike traditional regression models, which assume independence between observations, network data is defined by nodes connected through edges, introducing intricate dependencies that traditional models are not designed to handle [Newman, 2018]. Moreover, network data often exhibits high dimensionality and sparsity, with interactions between nodes that cannot be easily captured by linear models [Kolaczyk and Csárdi, 2014]. This complexity also hampers interpretability, as regression coefficients may lose their intuitive meaning in the context of networks [Goldenberg et al., 2009].

Motivated by these challenges, we address the following core question in this paper:

*Given a collection of networks $\{G_i\}_{i=1}^d$ with labels $\{y_i\}_{i=1}^d$, can we perform regression between the independent network data and the dependent label data? If so, how can we associate the obtained coefficients with predictors to achieve interpretability?*

### 1.1. Our Contributions.

In this paper, we propose an interpretable Supervised Network Dictionary Learning (SNDL) model for network regression. The model is designed to handle network data and label data, offering both high prediction accuracy and good interpretability. Our approach assumes that the mesoscale structure of the networks — the space of all $k$-node subgraphs — is low-dimensional, meaning that it can be efficiently captured and summarized. To achieve this, we use a $k$-path sampling algorithm [Lyu et al., 2023] to sample mesoscale structures, and employ Supervised Matrix Factorization (SMF) and the Block Coordinate Descent (BCD) algorithm [Lyu et al., 2024] to extract representative patterns (or *dictionary*) and their corresponding regression coefficients.

The key contributions of our work include:

- We propose an interpretable framework, SNDL, for network regression that achieves high prediction accuracy compared to baseline predictors. The model leverages $k$-path sampling and supervised matrix factorization techniques to explore the mesoscale structure of networks.
- We validate our model through empirical experiments on the Facebook100 dataset [Rossi and Ahmed, 2015] and BioGRID dataset [Rossi and Ahmed, 2015]. For the Facebook100 dataset, we conduct additional experiments with randomly generated networks as baseline networks and real networks as test cases.

## 2. Related Works

### 2.1. Network Regression.

Network regression extends traditional regression models to network-structured data, where though dependencies between nodes violate the independence assumption. Early work, such as by Krackhardt [Krackhardt, 1988], incorporated network attributes like centrality as covariates in regression models, acknowledging the role of network position on node outcomes. More advanced

models, like stochastic actor-oriented models (SAOMs), integrate both node-level attributes and network structure to capture dynamic relationships [Snijders et al., 2006].

Bayesian methods, such as those by Hoff [Hoff, 2008], address uncertainty in network structure by using hierarchical models, which have been effective in sparse networks. Non-parametric approaches, such as Li et al. [Li et al., 2019], allow for flexibility by capturing both global and local relationships in network data without strong parametric assumptions. These models are valuable for complex networks where traditional approaches fall short.

Network Generalized Linear Models (NGLMs) extend generalized linear models to network data, incorporating terms that account for node connectivity [Park and Newman, 2004]. These models have been applied to problems like disease spread in epidemiology and behavior analysis in social networks.

However, the interpretability of the models above is still not totally clear like regression to continuous data. And they don't provide insights of the networks' mesoscale structure. Furthermore, most of them are not computationally efficient given network data with large order.

### 2.2. **Metrics for Similarity between Networks.**

Accurate measurement of network similarity is essential for applications ranging from anomaly detection to transfer learning. This section will focus on various methodologies for assessing network similarity, highlighting specific comparison methods from recent studies.

Tantardini et al. (2019) explore quantitative methods that extend beyond traditional unweighted and undirected network comparisons to include directed and/or weighted networks. Their work emphasizes methods capable of handling richer information typically found in complex real-world networks, such as the European Air Transportation Network and the FAO Trade Network. Notably, they discuss DeltaCon, a method that utilizes node correspondences to assess network similarities through a root-node adjacency matrix, and NetSimile, which does not require known node-correspondence and uses statistical summaries of node features to determine similarity[Tantardini et al., 2019]. Grannis (2017) addresses the statistical challenges in network data, focusing on the biases introduced by non-independence of observations. The paper emphasizes the use of model-based approaches to counteract these sampling issues, though specific statistical models are not detailed, the discussion centers on the need for robust statistical methods that account for network data dependencies [Grannis, 2018]. Soundarajan et al. (2014) provide an empirical evaluation of twenty network similarity methods, categorizing them based on their operational complexity and requirement for node correspondence. They highlight two specific methods: Random Walk with Restarts (RWR), which measures similarity based on a restart probability at each node during random walks, and NetSimile, which employs a canonical correlation analysis on node-level features to determine network similarity. Their findings suggest that even complex methods can often be closely approximated by simpler methods like those comparing basic network features such as density or degree distributions [Soundarajan et al., 2014]. Wills and Meyer (2020) introduce practical metrics for graph comparison with a focus on practitioner-oriented applications. They discuss vector-based similarity methods, such as spectral distances that compare networks based on the eigenvalues of their Laplacian matrices, and matching-based approaches like the Graphlet Correlation Distance, which compares the frequencies of small subgraphs (graphlets) between networks. These methods are illustrated through their implementation in the NetComp software, which is designed to facilitate the application of various graph comparison metrics [Wills and Meyer, 2020].

These studies collectively enhance our understanding of network similarity assessment, providing a range of methods suited to different types of network data and analysis needs. The ongoing development and evaluation of these methods reflect the dynamic nature of network analysis research and its applications across diverse fields.

### 2.3. **Network Alignment for Biological Networks.**

Studying and comparing the structural representations in the form of complex networks has been a very fruitful and fascinating research area[Emmert-Streib et al., 2016]. To date, many techniques have been developed to compare real-world graph patterns structurally; concrete examples thereof are methods applied in linguistics[Dehmer and Emmert-Streib, 2011], web mining[Dehmer, 2007], chemoinformatics[Willett, 1987], and computational biology[Emmert-Streib and Dehmer, 2011].

Biological network alignment has transitioned significantly from focusing on simple pathway-based alignments to incorporating complex, integrative approaches that utilize a wide range of biological data. Early efforts, such as those introduced by PathBLAST, primarily aligned pathways based on sequence similarity and orthologous relationships [Kelley et al., 2003]. These foundational methods evolved through innovations like NetworkBLAST, which integrated interaction data for more sophisticated network mapping [Sharan et al., 2005]. Subsequent developments introduced global alignment strategies, exemplified by IsoRank and its successors, which applied PageRank-like algorithms to consider both protein sequence and topology simultaneously, leading to more robust many-to-many mappings [Singh et al., 2008, Liao et al., 2009].

Graphlet-based methods, starting with GRAAL and its extensions, marked a further evolution by using local topological features to inform alignments, enhancing accuracy through algorithms like the Hungarian method and incorporating multiple similarity metrics [Kuchaiev et al., 2010]. The latest advancements in the field have been driven by data-driven approaches such as TARA and TARA++, which integrate topological, sequence, and functional information. TARA++, in particular, has demonstrated superior performance over traditional methods by employing machine learning techniques to directly learn from network and protein data, thereby improving precision and functional prediction capabilities [Gu and Milenković, 2021].

Overall, the progression from simple pathway comparisons to sophisticated, data-driven frameworks in network alignment reflects significant advances in our ability to decipher functional conservation across species. Modern NA methods like TARA++ set new benchmarks for accuracy and efficiency, pushing the boundaries of comparative biology and offering enhanced insights into the evolutionary and functional relationships inherent in biological networks [Gu and Milenković, 2021]. These integrative approaches, leveraging diverse datasets and computational innovations, continue to transform the landscape of network alignment, highlighting the dynamic interplay between computational techniques and biological insights.

### 2.4. **Supervised Matrix Factorization.**

Supervised Matrix Factorization (SMF) is a variant of matrix factorization that incorporates supervision in the learning process by leveraging both feature and label information. It extends traditional matrix factorization, which decomposes a data matrix into low-rank latent factors, by utilizing the target labels to guide the factorization. SMF is widely used in machine learning tasks such as collaborative filtering, document modeling, and image analysis, where not only the data but also some form of supervision (e.g., class labels or ratings) is available to improve predictive performance [Mairal et al., 2008, Lee and Seung, 2001]. Unlike traditional unsupervised matrix factorization, SMF aims to optimize a supervised loss function that captures both the reconstruction error and the predictive power of the learned factors [Kim et al., 2016, Ritchie et al., 2020].

Lee et al. [2023a] introduced a novel method to reformulate SMF problems as low-rank matrix estimation by applying a "double-lifting" technique in the parameter space. This reformulation allowed them to show that low-rank projected gradient descent (LPGD) could find a global optimum at an exponential rate when the problem is well-conditioned. However, their approach faces limitations, particularly when constraints like nonnegativity need to be imposed on the factor matrices. Singular value decomposition (SVD), which is commonly used in such matrix factorization tasks, does not guarantee optimal nonnegative matrix decompositions, which are often required in practical applications like image processing and collaborative filtering [Cai et al., 2010].

To overcome this limitation, recent research has shifted toward analyzing the local (constrained) landscape of SMF, focusing on the robustness of local optima under L2-regularization. Block

coordinate descent (BCD) algorithms have been extensively applied to SMF problems due to their ability to optimize convex objectives with respect to each block of variables while keeping the others fixed. This iterative strategy improves solutions step by step [Wright, 2015, Mairal et al., 2008, Austin et al., 2018, Leuschner et al., 2019]. Nevertheless, theoretical guarantees for these methods are limited, with most studies ensuring only asymptotic convergence to stationary points [Grippo and Sciandrone, 2000, Xu and Yin, 2013], leaving a gap in understanding convergence rates and global optimality.

Lee et al. [2024a] makes significant contributions for this problem by performing an in-depth analysis of the local landscape of SMF, including a detailed calculation of the block structure of the Hessian matrix, and identifies the minimum L2-regularization required to ensure local strong convexity. The authors propose a BCD algorithm with global convergence guarantees and demonstrate that it achieves $\epsilon$-stationary points within a known iteration complexity, enhancing earlier works on constrained matrix factorization [Burer and Monteiro, 2003]. Additionally, the authors introduce a neural network-based implementation of the BCD algorithm, optimized for GPU acceleration, and validate their theoretical results with numerical experiments. And our SNDL model utilizes this algorithm and solve a well-modeled SMF problem to do network regression.

## 3. Preliminaries

We use $\mathbb{R}^p$ to represent the ambient space for data, equipped with standard inner product $\langle\cdot,\cdot\rangle$, inducing the Euclidean norm $\|\cdot\|$. We refer to the set $\{0,1,\ldots,\kappa\}$ as the space of class labels, containing $\kappa+1$ classes. For a convex subset $\Theta$ in an Euclidean space, we denote $\Pi_\Theta$ the projection operator onto $\Theta$.

For a matrix $\mathbf{A}=(a_{ij})\in\mathbb{R}^{m\times n}$, the expressions $\mathbf{A}[i,:]$ and $\mathbf{A}[:,j]$ refer to the $i$th row and the $j$th column of $\mathbf{A}$ for each $1\le i\le m$ and $1\le j\le n$, respectively. For each integer $n\ge 1$, $\mathbf{I}_n$ denotes the $n\times n$ identity matrix. We denote its Frobenius, (2-), and supremum norm by

$$\|\mathbf{A}\|_F^2:=\sum_{i,j}a_{ij}^2,\quad \|\mathbf{A}\|_2:=\sup_{\mathbf{x}\in\mathbb{R}^n,\|\mathbf{x}\|=1}\|\mathbf{A}\mathbf{x}\|,\quad \|\mathbf{A}\|_\infty:=\max_{i,j}|a_{ij}|,$$

respectively. For square symmetric matrices $\mathbf{A},\mathbf{B}\in\mathbb{R}^{n\times n}$, $\mathbf{A}\preceq\mathbf{B}$ indicates that $\mathbf{v}^T\mathbf{A}\mathbf{v}\le\mathbf{v}^T\mathbf{B}\mathbf{v}$ holds for all unit vectors $\mathbf{v}\in\mathbb{R}^n$. If $0<\alpha^-<\alpha^+$, then we write $\mathbf{A}\simeq\alpha^{\pm}\mathbf{B}$ to denote $\alpha^-\mathbf{B}\preceq\mathbf{A}\preceq\alpha^+\mathbf{B}$. The horizontal concatenation of two matrices $\mathbf{A}$ and $\mathbf{B}$ is denoted by $[\mathbf{A},\mathbf{B}]$ when their dimensions match.

A *network* is a $G=(V,A)$ with node set $V$ and weighted adjacency matrix $A:V^2\to\mathbb{R}_{\ge 0}$. We say $G$ is a *graph* if $A$ is symmetric and binary. Two nodes $u,v$ are *adjacent* if $A[u,v]>0$ and in this case $u$ is a *neighbor* of $v$ and vice versa. When $u,v\in G$ are adjacent, an edge in $G$ between $u,v\in G$ is a pair $(u,v)$. We say the network $G$ is *connected* if for every pair of nodes, there is a sequence of adjacent nodes leading from one to the other. A *$k$-walk* in $G$ is a sequence of $k$ (allowing repetition) adjacent nodes $\mathbf{x}:[k]\to V, A[\mathbf{x}(i),\mathbf{x}(i+1)]>0$ for $i=1,\ldots,k-1$ (denoting $[k]:=\{1,2,\ldots,k\}$). A *$k$-walk* is a *$k$-path* if all nodes are distinct. Another network $H=(V',A')$ is a *subnetwork* of $G=(V,A)$ if $V'\subseteq V$ and $A'[u,v]\le A[u,v]$ for all $u,v\in V'$. Such $H$ is a *$k$-node sub-network* if $|V'|=k$. Given a $k$-path $\mathbf{x}$ in $G$, denote by $\mathbf{A}_\mathbf{x}$ the $k\times k$ matrix where $\mathbf{A}_\mathbf{x}[i,j]=A[\mathbf{x}(i),\mathbf{x}(j)]$ for $1\le i,j\le k$.

We say a network has $d_1$-dimensional node feature if for every node $i\in V$, there is a vector $\boldsymbol{z}_i\in\mathbb{R}^{d_1}$ attached to $i$. Then for a network $G$ with $n_1$ nodes and $d_1$-dimensional node feature, there is a node feature matrix $\mathbf{Z}\in\mathbb{R}^{n_1\times d_1}$ for network $G$. Similarly, we say that a network has $d_2$-dimensional edge feature if for every edge $e\in G$, there is a vector $\boldsymbol{l}_e\in\mathbb{R}^{d_2}$ attached to $e$. Then for a network $G$ with $n_2$ nodes and $d_2$-dimensional edge feature, there is an edge feature matrix $\mathbf{L}\in\mathbb{R}^{n_2\times d_2}$ for network $G$.

## 4. Statement of the problem

**Problem 4.1.** *Given $k \in \mathbb{N}$, a collection of baseline networks $\{G_i\}_{i=1}^{d}$ (potentially with node/edge feature), in which every network has a label $y_i \in \{1, \ldots, \kappa\}$, and a test network $G$, encode $G$ as a single probability distribution vector $\mathbf{p} = (p_1, \ldots, p_\kappa)$, where $p_i$ means the predictive probability that a random $k$-node subgraph in $G$ is from a baseline network with label $i$.*

**Remark.** Here if $\kappa = d$, which means every network has a distinct label, the problem is to find affinity score between the test network $G$ and every network in the baseline networks.

## 5. Methods

### 5.1. SMF and BCD Algorithm.

In this section, we give a mathematical formulation of the Supervised Matrix Factorization (SMF) problem, and the explanation of BCD algorithm in [Lee et al., 2024b] to solve SMF.

Given a set of $n$ samples $(y_i, \mathbf{x}_i)$ for $i = 1, 2, \ldots, n$ where $y_i \in \{0, 1, \ldots, \kappa\}$ represents the observed label, and $\mathbf{x}_i \in \mathbb{R}^p$ denotes high-dimensional feature. Our task is that given any feature $\mathbf{x}$, we want to predict its corresponding label $y$.

In SMF, firstly we utilize a suitable $r \ll p$ to compress the sampled dataset's features into a matrix $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_r] \in \mathbb{R}^{p \times r}$, which we call it the *dictionary* with respect to the sampled feature matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{p \times n}$, using a suitable code matrix $\mathbf{H} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n] \in \mathbb{R}^{r \times n}$. This can be compactly expressed as an optimization problem:

$$\min_{\mathbf{W} \in \mathcal{C}_1, \mathbf{H} \in \mathcal{C}_2} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2$$

Here $\mathcal{C}_j$ for $j = 1, 2$ represent convex constraint sets of $\mathbf{W}$ and $\mathbf{H}$.

Then we add components of supervised learning into the matrix factorization problem above. Here we present our probabilistic modeling assumption. Given $\mathbf{W} \in \mathbb{R}^{p \times r}, \boldsymbol{h}_i \in \mathbb{R}^r$ and $\boldsymbol{\beta} \in \mathbb{R}^{\kappa \times r}$, suppose $y_i$ is a realization of a random variable of sample $i$ whose conditional distribution is defined as:

$$\begin{aligned} \mathbb{P}(y_i = 0 | \mathbf{x}_i) &= \frac{1}{1 + \sum_{c=1}^{\kappa} \exp(\boldsymbol{a}_{i,c})} \\ \mathbb{P}(y_i = l | \mathbf{x}_i) &= \frac{\exp(\boldsymbol{a}_{i,l})}{1 + \sum_{c=1}^{\kappa} \exp(\boldsymbol{a}_{i,c})} \qquad \forall\, l = 1, \ldots, \kappa \end{aligned} \tag{1}$$

where $\boldsymbol{a}_i \in \mathbb{R}^\kappa$ is the activation for $y_i$. The activation is defined in two ways corresponding to feature-based model (SMF-H) and filter-based model(SMF-W):

$$\boldsymbol{a}_i = \begin{cases} \boldsymbol{\beta}^\top \mathbf{h}_i & \text{for SMF-H} \\ \boldsymbol{\beta}^\top \mathbf{W}^\top \mathbf{x}_i & \text{for SMF-W} \end{cases} \tag{2}$$

Here, $\boldsymbol{\beta}$ is the logistic regression coefficient associated with input features $\mathbf{h}_i$ or $\mathbf{W}^\top \mathbf{x}_i$, respectively. In equation (2), the code $\mathbf{h}_i$ or the 'filtered feature' $\mathbf{W}^\top \mathbf{x}_i$ is the low-dimensional representation of $\mathbf{x}_i$. Notable differences between SMF-H and SMF-W arise when predicting the unknown label of a test point [Lee et al., 2023b].

Let $\mathbf{Z} := (\mathbf{W}, \mathbf{H}, \boldsymbol{\beta})$ be our block parameters of interest. In order to estimate $\mathbf{Z}$ from observed data $(\mathbf{x}_i, y_i)$ for $i = 1, \ldots, n$, we combine two parts above and consider the following multi-objective non-convex constrained optimization problem:

$$\min_{\mathbf{W} \in \mathcal{C}_1, \mathbf{H} \in \mathcal{C}_2, \boldsymbol{\beta} \in \mathcal{C}_3} f(\mathbf{Z}) := \xi \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2 + \sum_{i=1}^{n} \ell(y_i, \boldsymbol{a}_i) \tag{3}$$

**Algorithm 1** BCD algorithm for SMF-W

**Input:** $\mathbf{X} \in \mathbb{R}^{p\times n}$ (Data); $\mathbf{Y}_{\text{label}} \in \{0,\dots,\kappa\}^{1\times n}$ (Label)
**Constraints:** Convex subsets $\mathcal{C}_1 \subseteq \mathbb{R}^{p\times r}$, $\mathcal{C}_2 \subseteq \mathbb{R}^{r\times n}$, $\mathcal{C}_3 \subseteq \mathbb{R}^{r\times \kappa}$
**Parameters:** $\xi \geq 0$ (Tuning parameter); $T \in \mathbb{N}$ (number of iterations); $(\eta_{k;i})_{k\geq 1, 1\leq i\leq 4}$ (step-sizes)
Initialize $\mathbf{W} \in \mathcal{C}_1$, $\mathbf{H} \in \mathcal{C}_2$, $\boldsymbol{\beta} \in \mathcal{C}_3$
**for** $k = 1, 2, \dots, T$ **do**
  **(Update W)**
  Update activation $a_1, \dots, a_n$ and $\mathbf{K}$
  $\nabla_{\mathbf{W}} f(\mathbf{Z}) \leftarrow \mathbf{X}\mathbf{K}^\top \boldsymbol{\beta}^\top + 2\xi(\mathbf{W}\mathbf{H} - \mathbf{X})\mathbf{H}^\top$
  Choose $\eta_{k,1}^{-1} > L_1 := \alpha^+ \|\boldsymbol{\beta}\|_2^2 \cdot \|\mathbf{X}\|_2^2 + 2\xi\|\mathbf{H}\|_2^2$
  $\mathbf{W} \leftarrow \Pi_{\mathcal{C}_1}(\mathbf{W} - \eta_{k;1}\nabla_{\mathbf{W}} f(\mathbf{Z}))$
  **(Update H)**
  $\nabla_{\mathbf{H}} f(\mathbf{Z}) \leftarrow 2\xi\mathbf{W}^\top(\mathbf{W}\mathbf{H} - \mathbf{X})$
  Choose $\eta_{k,2}^{-1} > L_2 := 2\xi\|\mathbf{W}\|_2^2$
  $\mathbf{H} \leftarrow \Pi_{\mathcal{C}_2}(\mathbf{H} - \eta_{k;2}\nabla_{\mathbf{H}} f(\mathbf{Z}))$
  **(Update $\beta$)**
  Update activation $a_1, \dots, a_n$ and $\mathbf{K}$
  $\nabla_{\boldsymbol{\beta}} f(\mathbf{Z}) \leftarrow \mathbf{W}^\top\mathbf{X}\mathbf{K}^\top$
  Choose $\eta_{k,3}^{-1} > L_3 := \alpha^+ \|\mathbf{W}\|_2^2 \cdot \|\mathbf{X}\|_2^2$
  $\boldsymbol{\beta} \leftarrow \Pi_{\mathcal{C}_3}(\boldsymbol{\beta} - \eta_{k;3}\nabla_{\boldsymbol{\beta}} f(\mathbf{Z}))$
**Output:** $\mathbf{Z} = (\mathbf{W}, \mathbf{H}, \boldsymbol{\beta})$

where

$$\ell(y_i, \boldsymbol{a}_i) := \log(1 + \sum_{c=1}^{\kappa} \exp(\boldsymbol{a}_{i,c})) - \sum_{c=1}^{\kappa} \mathbf{1}_{\{y=c\}} \boldsymbol{a}_{i,c}$$

Here $\mathcal{C}_j$ for $j = 1, 2, 3$ represent convex constraint sets of each block parameter, $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbb{R}^{p\times n}$, $\boldsymbol{a}_i$ is as in (2), and the last term in (3) is the classification loss defined as the negative log-likelihood. Note that the four block parameters are *individually* assumed to be constrained in (3). A tuning parameter $\xi$ controls the trade-off between the dual objectives of classification and matrix factorization. The stated problem is inherently non-convex, involving three block parameters that may come with additional constraints such as bounded norm. This formulation encompasses several classical models as special cases. Specifically, when $\xi \gg 1$, it transforms into the classical matrix factorization with constraints [Lee and Seung, 2000][Lee and Seung, 1999].

To solve (3), we use the Block Coordinate Descent(BCD) algorithm in [Lee et al., 2024b]. We outline this algorithm for SMF-W in Algorithm 1 and for SMF-H in Algorithm 2. Our algorithm, outlined in Algorithm 1 and Algorithm 2, iteratively uses Block Coordinate Descent (BCD) on the three blocks with an adaptively chosen stepsize. Here we denote $\mathbf{K} := [\nabla_{\boldsymbol{a}}\ell(y_1, \boldsymbol{a}_1), \dots, \nabla_{\boldsymbol{a}}\ell(y_n, \boldsymbol{a}_n)] \in \mathbb{R}^{\kappa\times n}$ where

$$\left(\nabla_{\boldsymbol{a}}\ell(y_i, \boldsymbol{a}_i)\right)_j = \frac{\exp(\boldsymbol{a}_{i,j})}{1 + \sum_{c=1}^{\kappa}\exp(\boldsymbol{a}_{i,c})} - \mathbf{1}\{y_i = j\} \quad \forall\, i \in \{1, 2, \dots, n\},\ j \in \{1, \dots, \kappa\}$$

And let observed information $\ddot{\mathbf{H}}(y, \boldsymbol{a}) := \nabla_{\boldsymbol{a}}\nabla_{\boldsymbol{a}^\top}\ell(y, \boldsymbol{a})$ for $y$ and $\boldsymbol{a}$. Then, let $M > 0$ be the constant for the bounded activation, then $\alpha^-$ and $\alpha^+$ are defined as follows:

$$\alpha^- := \inf_{\|\mathbf{a}\|\leq M} \min_{1\leq s\leq n} \lambda_{\min}(\ddot{\mathbf{H}}(y_s, a))$$

$$\alpha^+ := \sup_{\|\mathbf{a}\|\leq M} \max_{1\leq s\leq n} \lambda_{\max}(\ddot{\mathbf{H}}(y_s, a))$$

**Algorithm 2** BCD algorithm for SMF-H

**Input:** $\mathbf{X} \in \mathbb{R}^{p\times n}$ (Data);$\mathbf{Y}_{\text{label}} \in \{0,\dots,\kappa\}^{1\times n}$ (Label);
**Constraints:** Convex subsets $\mathcal{C}_1 \subseteq \mathbb{R}^{p\times r}$, $\mathcal{C}_2 \subseteq \mathbb{R}^{r\times n}$, $\mathcal{C}_3 \subseteq \mathbb{R}^{r\times \kappa}$
**Parameters:** $\xi \geq 0$ (Tuning parameter); $T \in \mathbb{N}$ (number of iterations); $(\eta_{k;i})_{k\geq 1, 1\leq i\leq 4}$ (step-sizes)
Initialize $\mathbf{W} \in \mathcal{C}_1$, $\mathbf{H} \in \mathcal{C}_2$, $\boldsymbol{\beta} \in \mathcal{C}_3$
**for** $k = 1, 2, \dots, T$ **do**
  **(Update W)**
  $\nabla_{\mathbf{W}} f(\mathbf{Z}) \leftarrow 2\xi(\mathbf{WH} - \mathbf{X})\mathbf{H}^\top$
  Choose $\eta_{k,1}^{-1} > L_1 := 2\xi\|\mathbf{H}\|_2^2$
  $\mathbf{W} \leftarrow \Pi_{\mathcal{C}_1}(\mathbf{W} - \eta_{k;1}\nabla_{\mathbf{W}} f(\mathbf{Z}))$
  **(Update H)**
  Update activation $a_1, \dots, a_n$, and $\mathbf{K}$
  $\nabla_{\mathbf{H}} f(\mathbf{Z}) \leftarrow \boldsymbol{\beta}\mathbf{K} + 2\xi\mathbf{W}^\top(\mathbf{WH} - \mathbf{X})$
  Choose $\eta_{k,2}^{-1} > L_2 := \alpha^+\|\boldsymbol{\beta}\|_2^2 + 2\xi\|\mathbf{W}\|_2^2$
  $\mathbf{H} \leftarrow \Pi_{\mathcal{C}_2}(\mathbf{H} - \eta_{k;2}\nabla_{\mathbf{H}} f(\mathbf{Z}))$
  **(Update $\beta$)**
  Update activation $a_1, \dots, a_n$, and $\mathbf{K}$
  $\nabla_{\beta} f(\mathbf{Z}) \leftarrow \mathbf{XK}^\top$
  Choose $\eta_{k,3}^{-1} > L_3 := \alpha^+\|\mathbf{H}\|_2^2$
  $\mathbf{H} \leftarrow \Pi_{\mathcal{C}_3}(\mathbf{H} - \eta_{k;3}\nabla_{\beta} f(\mathbf{Z}))$
**Output:** $\mathbf{Z} = (\mathbf{W}, \mathbf{H}, \boldsymbol{\beta})$

For theoretical guarantee of BCD, we need the following assumptions:

**Assumption 5.1.** *(Constraint sets)* The constraint sets $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ in (3) are closed, convex, and compact.

**Assumption 5.2.** *(Bounded activation)* The activation $\boldsymbol{a} \in \mathbb{R}^\kappa$ defined in (2) assumes bounded norm, i.e., $\|\boldsymbol{a}\| \leq M$ for some constant $M \in (0, \infty)$.

And for each $\epsilon \geq 0$, we define $\theta^\star \in \Theta$ to be an $\epsilon$-stationary point of $f$ over $\Theta$ if

$$\text{Gap}(\theta^\star) := \sup_{\theta\in\Theta, \|\theta-\theta^\star\|\leq 1} \langle -\nabla f(\theta^\star), \theta - \theta^\star\rangle \leq \epsilon. \quad (14)$$

Then in [Lee et al., 2024b], we get convergence guarantee of BCD in the following theorem:

**Theorem 5.1.** *Suppose Assumptions 5.1 and 5.2 hold. Let* $\Theta = \mathcal{C}_1 \times \mathcal{C}_2 \times \mathcal{C}_3$, $\mathbf{Z}_t = (\mathbf{W}_t, \mathbf{H}_t, \boldsymbol{\beta}_t)$, $t \geq 1$ *denote the sequence of estimated parameters from Algorithm 1 or 2. Then for every initial estimate* $\mathbf{Z}_0$ *and choice of parameters* $\xi$*, the followings hold:*

*(i)* $\min_{1\leq t\leq T} Gap(\mathbf{Z}_t) = \mathcal{O}(T^{-1/2}\log T)$.
*(ii) For each* $\epsilon > 0$*, an* $\epsilon$*-stationary point is achieved within iteration* $\mathcal{O}(\epsilon^{-1}\log\epsilon^{-1})$.
*(iii) Further assume that the step sizes* $\eta_{k,i}$ *are uniformly upper bounded. Then* $\mathbf{Z}_t$ *converges to the set of stationary points of* $f$ *over* $\Theta$.

### 5.2. Supervised Network Dictionary Learning.

In this section, we propose our Supervised Network Dictionary Learning model (SNDL) for network regression problem based on filter-based Supervised Matrix Factorization (SMF-W) model.

#### 5.2.1. *Learning the latent motifs.*

In this subsection, we give a mathematical explanation about how to learn latent motifs and corresponding coefficients in our SNDL model. Given a collection of networks $\{G_i\}_{i=1}^d$ with labels $\{y_i\}_{i=1}^d \subseteq \{0,\dots,\kappa\}$ (potentially with node/edge feature), our task is to compress the mesoscale

information of the collection of networks (mesoscale subgraphs) and be able to utilize the compressed information of networks to predict the labels of networks. The general process is:

(1) For each network $G_i$ in the collection, we sample suitable number $n_i$ of subgraphs of it and get corresponding adjacency matrices. Let $n = \sum_{i=1}^{n} n_i$. Then we vectorize every adjacency matrix and horizontally concatenate them to get feature matrix $\mathbf{X} \in \mathbb{R}^{k^2 \times n}$. And for every sample, we get its corresponding label from the network where the sample is generated. Then we combine them to get the label vector $\mathbf{Y} \in \mathbb{R}^n$.
(2) Use BCD for SMF-W Algorithm 1 with input $\mathbf{X}$ and $\mathbf{Y}$ to get the output $\mathbf{Z} = (\mathbf{W}, \mathbf{H}, \boldsymbol{\beta})$, where $\mathbf{W}$ is the dictionary of $\{G_i\}_{i=1}^d$, a.k.a. *latent motifs* of $\{G_i\}_{i=1}^d$.

In detail, we choose suitable $k$ (number of nodes for subgraphs), $r$ (number of subgraphs in the dictionary), and $n_i$ for $i = 1, \ldots, d$ (number of samples from $G_i$). For sampling method, we use the Pivot Chain sampling algorithm with k-path embedding in [Lyu et al., 2023]. There are two main advantages of the algorithm:

(1) The sampled subgraphs are connected, which is convenient for identifying patterns of networks.
(2) The algorithm will output a k-path embedding for the sampled subgraph defining a unique order of nodes in the sampled subgraph, which is useful for constructing the unique adjacency matrix and SNDL for networks with node/edge feature.

And we have the convergence guarantee and convergence rate of Pivot Chain sampling method below. See [Lyu et al., 2023] for more details.

**Theorem 5.2** (Convergence of pivot chain)**.** *Let $\mathcal{G} = ([n], A, \alpha)$ be an irreducible network with $A(i, j) > 0$ for some $j \in [n]$ for each $i \in [n]$. Let $F = ([k], A_F)$ be a rooted tree motif. Then pivot chain $F \to \mathcal{G}$ is irreducible with unique stationary distribution $\pi_{F\to\mathcal{G}}$.*

**Theorem 5.3** (Mixing time of pivot chain)**.** *Let $F = ([k], E_F)$ be a directed rooted tree and $\mathcal{G} = ([n], A, \alpha)$ be an irreducible network. Further assume that for each $i \in [n], A(i, j) > 0$ for some $j \in [n]$. Let $P$ denote the transition kernel of the random walk on $\mathcal{G}$ defined at (21). Then the mixing time $t_{mix}(\epsilon)$ of the pivot chain $(\mathbf{x}_t)_{t\geq 0}$ of homomorphisms $F \to \mathcal{G}$ satisfies the following.*

*(i) Let $t_{mix}^{(1)}(\epsilon)$ be the mixing time of the pivot with kernel $P$. Then*

$$t_{mix}(\epsilon) = t_{mix}^{(1)}(\epsilon).$$

*(ii) Let $\lambda_\star$ be the eigenvalue of $P$ with largest modulus that is less than 1. Then*

$$\frac{\lambda_\star \log(1/2\epsilon)}{1 - \lambda_\star} \leq t_{mix}(\epsilon) \leq \frac{\max_{x\in[n]} \log(1/\alpha(x)\epsilon)}{1 - \lambda_\star}.$$

*(iii) Suppose $n \geq 13$, $A$ is the adjacency matrix of some simple graph, and $\alpha(i) \propto \deg(i)$ for each $i \in [n]$. Then*

$$t_{mix}(\epsilon) \leq \log_2(\epsilon^{-1}) \left( \frac{4}{27} n^3 + \frac{4}{3} n^2 + \frac{2}{9} n - \frac{296}{27} \right).$$

After sampling subgraphs from $\{G_i\}_i$, we obtain ajacency matrices $\{A_{i,j}\}_{j=1}^{n_i} \subseteq \mathbb{R}^{k\times k}$ of subgraphs for each $G_i$, $i = 1, \ldots, d$. Suppose for any $i, j$, $A_{i,j} = [A_{(i,j),1}, \ldots, A_{(i,j),k}]^\top$, where $A_{(i,j),c} = A[c, :]$ for $c = 1, \ldots, k$. We define vectorized adjacency matrices $\boldsymbol{a}_{i,j}^{\text{adj}} = [A_{(i,j),1}^\top, \ldots, A_{(i,j),k}^\top]^\top \in \mathbb{R}^{k^2}$ for $i = 1, \ldots, d$; $j = 1, \ldots, n_i$.

If $\{G_i\}_i$ have $d_1$-dimensional node feature, for $j$-th sampled subgraph from $G_i$, $i = 1, \ldots, d$, $j = 1, \ldots, n_i$, we obtain the corresponding node feature matrix $\mathbf{X}_{i,j}^{\text{node}} \in \mathbb{R}^{k\times d_1}$ with node order from the sampled k-path. Then we vectorize every $\mathbf{X}_{i,j}^{\text{node}}$ in the same way above to get the corresponding vectorized node feature vector $\mathbf{x}_{i,j}^{\text{node}} \in \mathbb{R}^{kd_1}$. Similarly, if $\{G_i\}_i$ have $d_2$-dimensional edge feature, for j-th sampled subgraph from $G_i$, $i = 1, \ldots, d$, $j = 1, \ldots, n_i$, we obtain the corresponding edge feature

tensor $\mathbf{X}_{i,j}^{\text{edge}} \in \mathbb{R}^{k\times k\times d_2}$ with edge order same as $A_{i,j}$. Then we vectorize every $\mathbf{X}_{i,j}^{\text{edge}} \in \mathbb{R}^{k\times k\times d_2}$ in the same way above to get the corresponding vectorized edge feature vector $\mathbf{x}_{i,j}^{\text{edge}} \in \mathbb{R}^{k^2 d_2}$.

Then for any $i = 1, \ldots, d$, $j = 1, \ldots, n_i$, we concatenate the vectorized adjacency matrices $\boldsymbol{a}_{i,j}$, node feature vector $\mathbf{x}_{i,j}^{\text{node}}$ (if $\{G_i\}_i$ have), and edge feature vector $\mathbf{x}_{i,j}^{\text{edge}}$ (if $\{G_i\}_i$ have) to get the feature vector $\mathbf{x}_{i,j} = [\boldsymbol{a}_{i,j}^{\text{adj}\top}, \mathbf{x}_{i,j}^{\text{node}\top}, \mathbf{x}_{i,j}^{\text{edge}\top}]^\top$ for the $j$-th sample of network $G_i$. Eventually, we get the feature matrix $\mathbf{X} = [\mathbf{x}_{1,1}, \ldots, \mathbf{x}_{1,n_1}, \ldots, \mathbf{x}_{d,1}, \ldots, \mathbf{x}_{d,n_d}]$ for $\{G_i\}_i$. Furthermore, we get the label vector $\mathbf{Y} = [\underbrace{y_1, \ldots, y_1}_{n_1 \text{ terms}}, \ldots, \underbrace{y_j, \ldots, y_j}_{n_j \text{ terms}}, \ldots, \underbrace{y_d, \ldots, y_d}_{n_d \text{ terms}}]$.

Having obtained the feature matrix $\mathbf{X}$ and the label vector $\mathbf{Y}$, we utilize Algorithm 1 with input $\mathbf{X}$, $\mathbf{Y}$, and hyperparameter $r$ to get the output $\mathbf{Z} = (\mathbf{W}, \mathbf{H}, \boldsymbol{\beta})$, where $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_r] \in \mathbb{R}^{r\times(k^2+kd_1+k^2d_2)}$ is the dictionary, i.e, latent motifs of $\{G_i\}_i$ with respect to subgraph size $k$. And $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_r]^\top \in \mathbb{R}^{r\times\kappa}$ is the multinomial logistic regression coefficients with input features $\mathbf{W}^\top\mathbf{X}$.

**Remark.** In our SNDL model, we implicitly assume that the feature space of all $k$-subgraphs from $\{G_i\}_i$ is low-dimensional. In other words, there exists $r \ll n$ and $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_r] \in \mathbb{R}^{r\times(k^2+kd_1+k^2d_2)}$ such that for every feature vector $\mathbf{x} \in \mathbb{R}^{k^2+kd_1+k^2d_2}$ from some $G_j$, $j \in \{1, \ldots, d\}$, $\mathbf{x} \in lin\{\mathbf{w}_1, \ldots, \mathbf{w}_r\}$, where $lin(\cdot)$ means the linear space spanned by the vectors in the parenthesis.

5.2.2. *Network Affinity Prediction.*

In this subsection, we explain the methodology of SNDL to solve Problem 4.1 continuing section 5.1 and subsection 5.2.1. The assumptions and initial data are the same as subsection 5.2.1.

After dealing with the given data using the SNDL model described in subsection 5.2.1, we obtain the output $\mathbf{Z} = (\mathbf{W}, \mathbf{H}, \boldsymbol{\beta})$. Now given a test network $G$, we choose a suitable $n$ (number of samples from $G$) and construct a feature matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{(k^2+kd_1+k^2d_2)\times n}$ in the same way described in subsection 5.2.1. Then by our probabilistic modeling assumption (1) and SMF-W model (2), for $i = 1, \ldots, n$, we get activations $\boldsymbol{a}_i = \boldsymbol{\beta}^\top\mathbf{W}^\top\mathbf{x}_i$, and a probability distribution $\boldsymbol{p}_i = [\frac{1}{1+\sum_{c=1}^{\kappa}\exp(\boldsymbol{a}_{i,c})}, \frac{\boldsymbol{a}_{i,1}}{1+\sum_{c=1}^{\kappa}\exp(\boldsymbol{a}_{i,c})}, \ldots, \frac{\boldsymbol{a}_{i,\kappa}}{1+\sum_{c=1}^{\kappa}\exp(\boldsymbol{a}_{i,c})}]$, where the $j$-th entry of $\boldsymbol{p}_i$ is the predictive probability that the $i$-th k-node subgraph sample in $G$ is from a baseline network with label $j$. Then we average all $\boldsymbol{p}_i$, i.e., we define $\boldsymbol{p} = \frac{\sum_{i=1}^{n}\boldsymbol{p}_i}{n}$, which is our solution of SNDL to Problem 4.1.

**Remark.** Suppose $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_r]$, $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_r]^\top$. Notice that for each $\mathbf{w}_i \in \mathbb{R}^{k^2+kd_1+k^2d_2}$, $i = 1, \ldots, r$, (a representative subgraph in the dictionary), there is a corresponding $\boldsymbol{\beta}_i \in \mathbb{R}^{\kappa}$ serving as the impact factor of this representative subgraph for the predictive probability distribution, which is useful to interpret what label the representative subgraph has and what mesoscale pattern the test network $G$ has to be characterized with some specific label.

## 6. Random network models

Before we continue talking about our experiments and applications, it is important to talk about these random graphs and how we set our parameters for them. Given graph $1 = g_1$, which is the real world graph that we are looking at, we want to tune graph $2 = g_2$, the random graph, such that they are roughly equivalent in density and equivalent in number of nodes.

6.1. **Erdös-Rényi.** Erdös-Rényi (ER) consists of 2 parameters: number of nodes ($n$), and edge probability ($p$). In order to generate an ER graph we will place $n$ nodes all unconnected. Then between each pair of nodes we will draw an edge at probability $p$.

We will let $n = g_1(n)$, and then in order to approximate the density, we just need to set $p$ to be equal to $g_1(\rho)$.

This results in graphs in which edges are assumed to be independent and appear at equal probability. This is a stark contrast from real world graphs where edges influence other edges, and edges do not appear at equal probability. For this reason we should expect ER graphs to be very polarizing when comparing to social networks.

6.2. **Barabási–Albert.** Real world networks are often scale-free which means that the degree distribution of the nodes follow a power law. A downfall of ER is that they do not produce scale-free networks. In order to do so we can use preferential attachment, which Barabási–Albert (BA) takes advantage of.

BA consists of 2 parameters as well: number of nodes ($n$), and number of edges to attach from a new node to existing nodes ($m$). In order to generate a BA graph we will place $m_0$ where $m_0 > m$ nodes all unconnected. Then we will add a new node that is also unconnected. Then we will draw $m$ links to our existing nodes $i$ by probability:

$$p_i = \frac{k_i}{\sum_j k_j}$$

where $k_i$ is the degree of node $i$. This can be thought of as the probability being proportional to the node's degree.

We will let $n$ be equivalent to $g_1(n)$. Then in order to match density, we want to know the number of edges that we will have. Let $g_1(\rho)$ be the density of $g_1$. This means that $g_1(\rho) = \frac{n}{e}$, since each time we add a node we are adding $m$ new edges. This means we have $m$ many edges added $n - m_0$ times. Let's let $m_0 = m + 1$, this means we are adding $m(n - m + 1)$ edges total. Meaning that we want the following to be true:

$$\begin{aligned} e &= m(n - m + 1) \\ e &\approx m(n - m) \end{aligned}$$

We note that we can make such an approximation since we are working with $n$ with a magnitude of $10^4 >$. Using the density of an undirected simple graph we get that:

$$\begin{aligned} g_1(\rho) &\approx \frac{2e}{n(n-1)} \\ g_1(\rho) &\approx \frac{2m(n-m)}{n(n-1)} \\ g_1(\rho)(n(n-1)) &\approx 2m(n-m) \\ \frac{g_1(\rho)(n(n-1))}{2n} + \frac{m^2}{n} &\approx m \\ \frac{g_1(\rho)(n-1)}{2} &\approx m \end{aligned}$$

We can again make an assumption that $\frac{g_1(\rho)(n-1)}{2} \gg \frac{m^2}{n}$. Through this calculation we get our two parameters that approximate the density of the graph that we are trying to approximate using BA.

This method of graph generation results in a structure where earlier placed nodes have a much higher degree, which can be thought of as akin to social networks. Earlier users are likely to have more connections due to longer time with potential to make connections. This means we can expect BA to better predict social networks.

6.3. **Watts–Strogatz.** The Watts-Strogatz model (WS) was designed to also counter some of the downfalls of ER when looking at real world networks. WS solves some of the issues that BA attempts to solve but with an easier method.

WS consists of 3 parameters: number of nodes ($n$), rewiring probability ($p$), and neighbors that each node is joined to ($k$) . In order to generate a WS graph we start by constructing a graph with $n$ nodes each connected to $k$ neighbors, $k/2$ on each side. That is if the nodes are labels $0, \ldots, n-1$ , there is an edge $(i, j)$ if and only if

$$0 < |i - j| \pmod{\left(n - 1 - \frac{K}{2}\right)} \leq \frac{K}{2}$$

Then for every node $i = 0, \ldots, n-1$we take every edge connecting $i$ to its $k/2$ rightmost neighbors, that is every edge $(i, j)$ such that

$$0 < (j - i) \pmod{n} \leq \frac{k}{2}$$

And we rewire it with probability $p$. Where it is rewired by replacing $(i, j)$ with $(i, \ell)$ where $\ell$ is chosen uniformly at random from all possible nodes while avoiding self-loops $\ell \neq i$ and link duplication.

We again will let $n$ be equivalent to $g_1(n)$. We will arbitrarily let $p = 0.5$ since this parameter is responsible for rewiring the graph and does not impact the density of the graph. The number of edges in a WS graph can be approximate by:

$$e = \frac{n * k}{2}$$

Since each node is connected to $k$ neighbors on one side of the ring, and then we must divide by 2 to avoid double counting. Then using the same edge density formula from earlier we have that:

$$\begin{aligned} g_1(\rho) &\approx \frac{2e}{n(n-1)} \\ g_1(\rho) &\approx \frac{n * k}{n(n-1)} \\ g_1(\rho) &\approx \frac{k}{(n-1)} \\ (n-1)g_1(\rho) &\approx k \end{aligned}$$

Through this calculation we get our parameters that approximate the density of the graph that we are trying to approximate through WS.

Since we are starting with a lattice like structure, we expect a locally clustered network. The rewiring is responsible to reduce the average path-length of the graph. More importantly it again produces a scale-free network similar to BA, however, computationally WS is simpler.

6.4. **Configuration Model.** The last model we will use is the configuration model (CM). CM is arguably the best in terms of approximately real world networks. This is because we are using actual information of $g_1$ in order to produce a CM graph.

The only parameter that a CM graph needs is a list of the degrees of each node. The way the graph is by representing the degrees of the vertices as stubs. Then we choose two stubs uniformly at random and connecting them to form an edge. Then we choose another pair from the remaining $2m - 2$ stubs where $m$ is half the sum of our degree list. This is guaranteed to always be

an integer since a graph must have an even sum of degrees. We will continue this process until we run out of stubs, and as a result we will get a network with a pre-defined degree sequence.

The only parameter we need is the degree list which can be found using python.

## 7. Experiments

### 7.1. **Facebook100.**

We study the Facebook social network of friendships at 100 American colleges and universities at a single moment of time in September 2005[Rossi and Ahmed, 2015]. The network consist of 100 independent networks, where every network corresponds to one university. Friendships are recorded only between people from the same university. Besides the information about friendships, network also contains limited demographic information. The Network is unweighted and undirected.

7.1.1. *Binary Network Regression.*

As stated in 5.2, given baseline networks $G_1$ and $G_2$ and test network $G$, we would like to encode $G$ as a single probability distribution vector $\mathbf{p} = (p_1, p_2)$, where $p_i$ means the predictive probability that a random $k$-node subgraph in $G$ is from the baseline network $G_i$ for $i = 1, 2$.

We developed a function that for each ordered network pair $(G_1, G_2)$ in the `ntwk_list` ($\frac{n(n-1)}{2}$ number of pairs for an $n$-length list), it will give the numeric predictive probability of each network in the list that it is from the baseline network $G_2$. For example, in the cell $(4, 1)$ in Figure **??**, it means that there is 0.79 probability that `MIT8` is from the network `UCLA26` and $0.21(= 1 - 0.79)$ probability that it is from the network `Caltech36`.

From 1, we can oberserve that `Caltech36` and `UCLA26` can be very well distinguished from each other, showing that the structures of them are quite different, which is corresponding to the fact. However, it seems that it is more difficult to distinguish `Harvard1, MIT8` and `UCLA26` from each other, indicating that there structures may be more similar and need further understanding.

7.1.2. *Random Graph Testing.* The main allure of using random graphs is this allows us to have a lot of control over our experiment. When looking at two separate real world graphs there are various factors that might cause the two graphs to either be similar or vastly different. When looking at these well documented and studied random graphs we will be able to arrive to conclusions much easier about our models.

One of the main goals we have is to look at how our accuracy changes as we vary both $k$ and $n$. We can do this by repeatedly using `SMF_BCD` on various values of $k$ and $n$ and seeing how well they can differentiate between our social graphs and our random graphs. We begin by observing how our accuracy changes based on the hyperparameters and random graph we use. We will do this by generating a heatmap of `n_components` and $k$.

There are some differences between the Facebook networks that are apparent and should be mentioned before we talk about observations when comparing with random graphs.

| | **Caltech** | **UCLA** | **MIT** | **Harvard** |
|---|---|---|---|---|
| **Nodes** | 769 | 20.5k | 6.4k | 15.1k |
| **Edges** | 16.7k | 747.7k | 251.3k | 824.6k |
| **Density** | 0.056 | 0.003 | 0.012 | 0.007 |

As we see based on our values, Caltech is by far the smallest graph out of all the ones that we are looking at. Additionally, Caltech is the densest by a decent margin as well. UCLA is the polar opposite where is has the most noes and lowest density. This means that when looking at the data, it is beneficial to look at comparing UCLA with Caltech since they can be thought of as polar opposites.

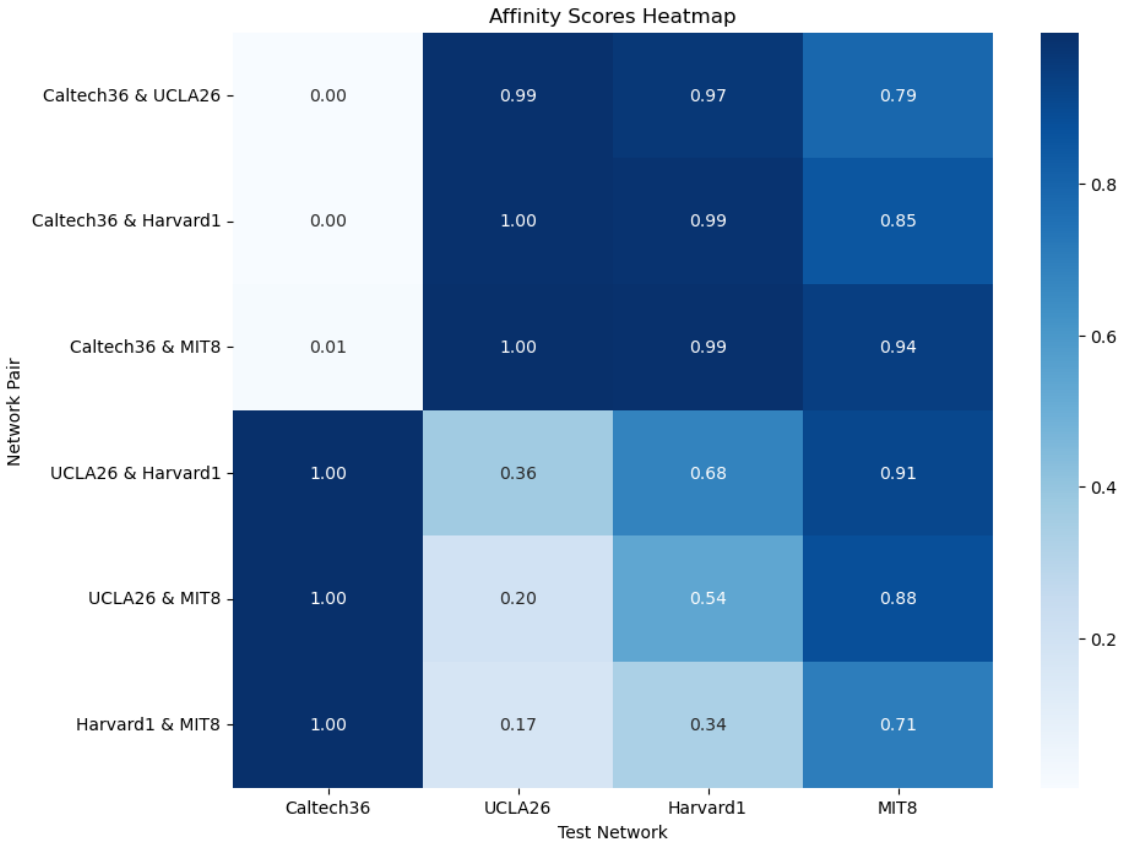



FIGURE 1. Binary Network Regression - Facebook 100 ($k = 50$)

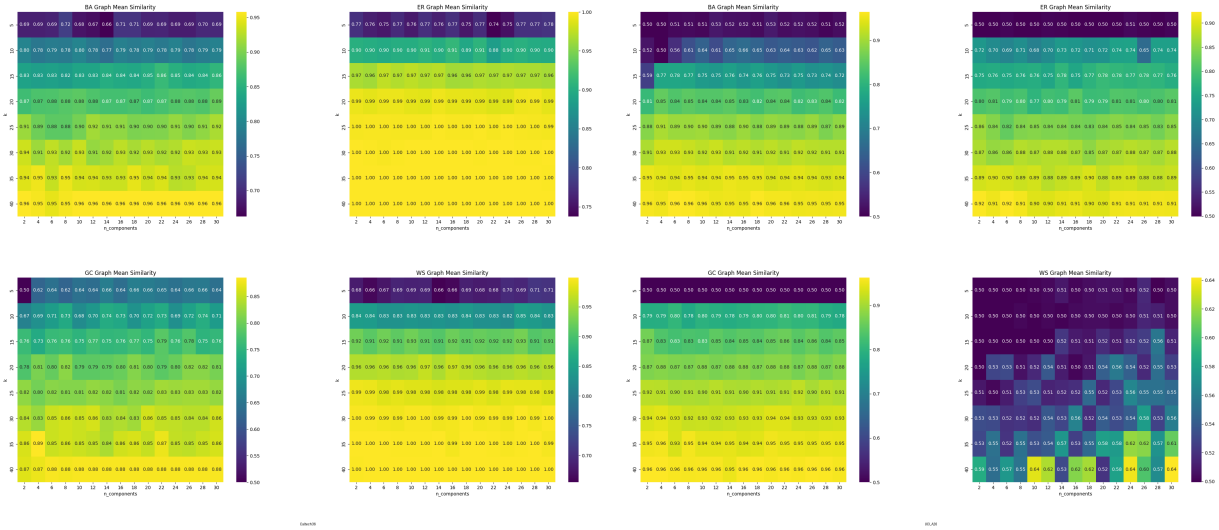
For Caltech, at $k = 40$ it appears that all of our synthetic models reach relatively high accuracy. The random graph that does the worse is CM, which is what we expected given that CM most closely aligns a graphs properties while retaining randomness within the graph.

Looking at UCLA's graph we notice that WS falls significantly behind in terms of accuracy. Additionally, we notice that for our trials WS also has the greatest standard deviation meaning that even amongst our trials our regression wasn't too sure about WS. We also notice that the second derivative of WS is positive, this would suggest that perhaps at $k = 50$ we could see much higher accuracy between UCLA and WS.

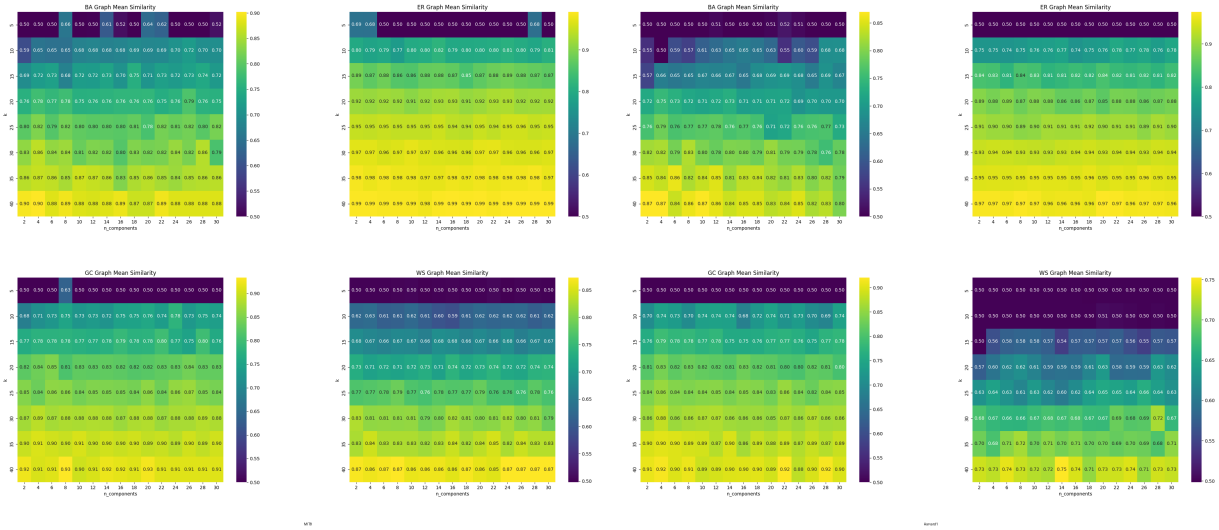
Looking at Harvard we notice again that WS is the lowest accuracy and ends at around 70%. But comparatively to UCLA, we see that the second derivative of WS is much lower than UCLA.

There are some sparse examples in which while we vary $n$ we notice an increase in accuracy, it seems in general that while we vary $n$ the accuracy stays relatively close. The physical interpretation of why `n_components`doesn't impact the accuracy that much would be that there are only a few subgraph patterns between our social networks and our random networks that we need to learn.

Some of these trends can also be explained once we plot the latent motifs that `SMF_BCD` is learning. The graphs we will take a look at will be UCLA26 with ER and WS. This is because looking at UCLA with WS we see that despite increasing $k$ our accuracy doesn't necessarily increase. What we would expect from this fact would be that their subgraph structure looks very similar. In the following graphs we have the subgraph that has the lowest and highest regression coefficient. In both cases we see that the lowest coefficient subgraph is a path with some sparse edges connecting the




(A) Caltech36 (B) UCLA26

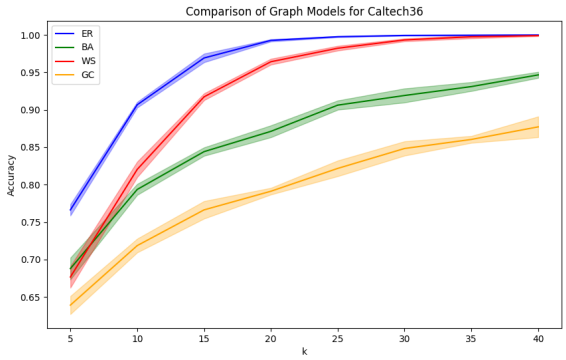



(C) MIT8 (D) Harvard1

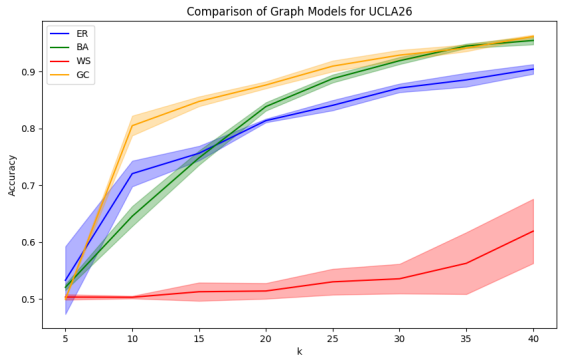
path while the higher valued coefficient is a path with much more pronounced edges interconnecting the path nodes. Although there is a slight difference in WS for UCLA, we notice how much more of a difference ER has when comparing to WS and this most likely contributes to why the model does so poorly when predicting between WS compared to ER for UCLA.

Since it appears that `n_components` makes little to no difference on our accuracy, we can take a brief look at $\xi$. We see that the altering of $\xi$ leads to much more variation in the accuracy. There is not necessarily a distinct pattern to say with certainty how $\xi$ acts when varying it. It would be preferred to repeat this experiment through several trials and see the average accuracy as it is likely the variance of each trial is very high.

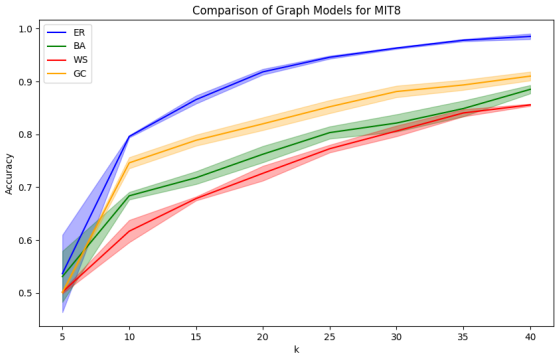
7.1.3. *Multiclass Network Regression.*



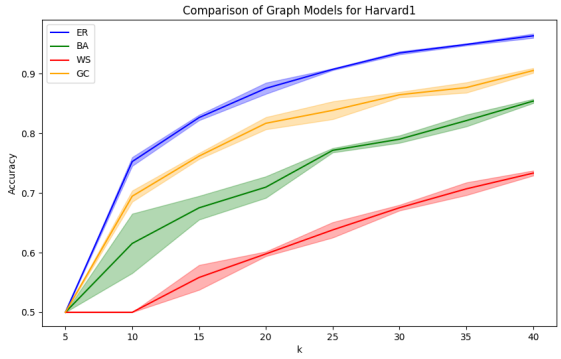

(A) Mean-Graph-Caltech36




(B) Mean-Graph-UCLA26




(C) Mean-Graph-MIT8



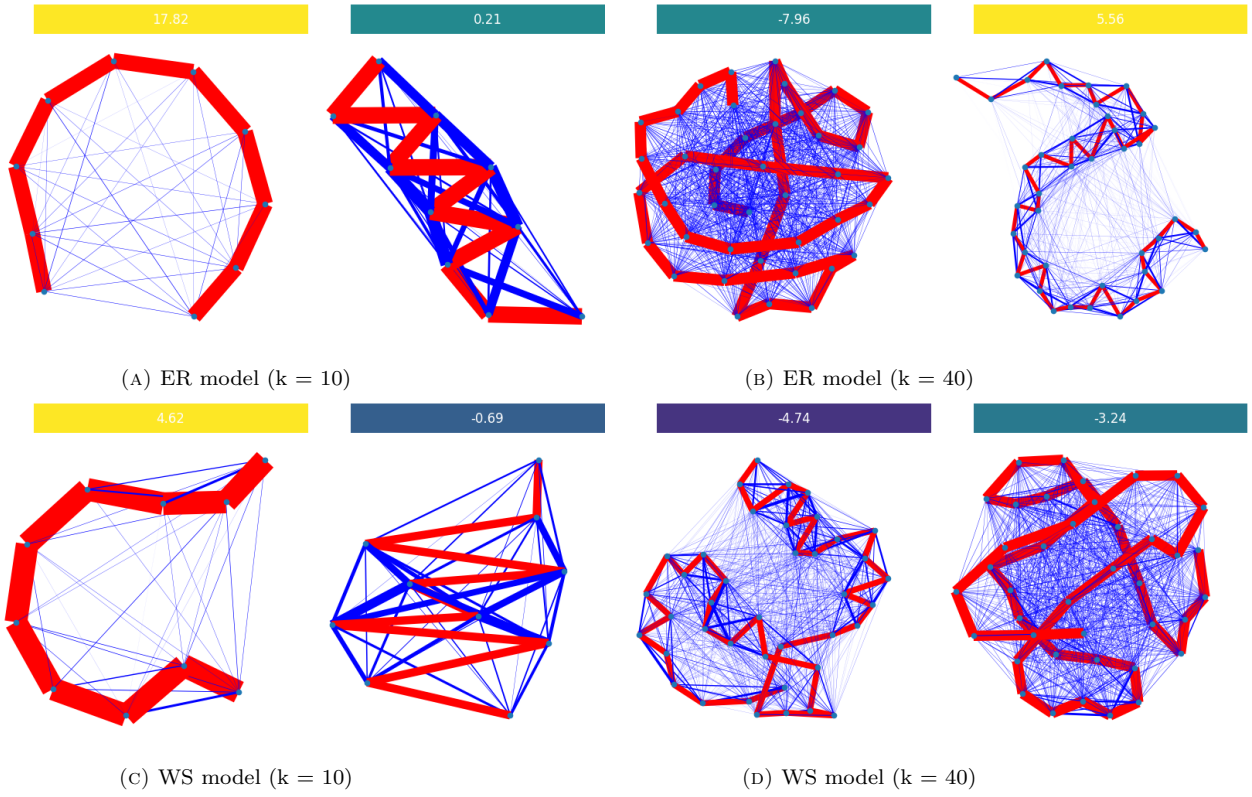

(D) Mean-Graph-Harvard1

In light of the previous result in 7.1.1, we took a step further to the multiclass-case regression. Similarly, given baseline networks $G_1, \ldots, G_d$ and test network $G$, we would like to encode $G$ as a single probability distribution vector $\mathbf{p} = (p_1, \ldots, p_d)$, where $p_i$ means the predictive probability that a random $k$-node subgraph in $G$ is from the baseline network $G_i$ for $i = 1, \ldots, d$.

To visualize the result through proper manners, we here developed a function to plot a triangle plots to show the prediction probabilities. Other than the 7 parameters used for the binary case, we introduced one more parameter, `baseline_i` here.

For `baseline_i`$= i$, the function will use the three networks: `ntwk_list[`$i$`]`, `ntwk_list[`$i+1$`]` and `ntwk_list[`$i+2$`]` as baseline networks and calculate the prediction probability (a 3-dimensional array) for each network in the input list, based on the $k$-node sampled graphs. We provided three standard points $(1,0,0), (0,1,0)$ and $(0,0,1)$ on the plot, comparing to the three baseline networks points inside the triangle (in the same corresponding colors) to demonstrate how well is the algorithm distinguishing the other two baseline networks to itself. We introduced the area ratio between the small triangle formed by the three regressed baseline network plots and the big triangle formed by the three standard plots to quantify this metric.

For example, we experimented with 10 college Facebook networks retrived from [Rossi and Ahmed, 2015], including `Caltech36, UCLA26, MIT8, Wisconsin87, Harvard1, Berkeley13, Columbia2, Michigan23, Princeton12` and `Stanford3`. From 6a, we noticed that with $k = 50$, the algorithm can well distinguish the three baseline networks, with an area ratio of 0.67 and the structure of `Caltech36` network is very diffrent from other 9 college Networks, which are mostly lying in between `MIT8` and `UCLA26`.




(A) ER model (k = 10)

(B) ER model (k = 40)

(C) WS model (k = 10)

(D) WS model (k = 40)

FIGURE 4. Comparing ER and WS models with k = 10 and k = 40

This result inspired us to use networks other than `Caltech36` as baseline to further examine the structure similarity 6b. We found that even with the same $k$, it is harder to distinguish `UCLA26`, `MIT8` and `Harvard1`. We also found that `Wisconsin87` and `Berkeley13` are more like `UCLA26`, as well as `Princeton12` is more like `MIT8`, while `Columbia2` and `Stanford3` are approximately in the middle of the plot, which altogether reveals more information about the network characteristics of **Facebook100**.

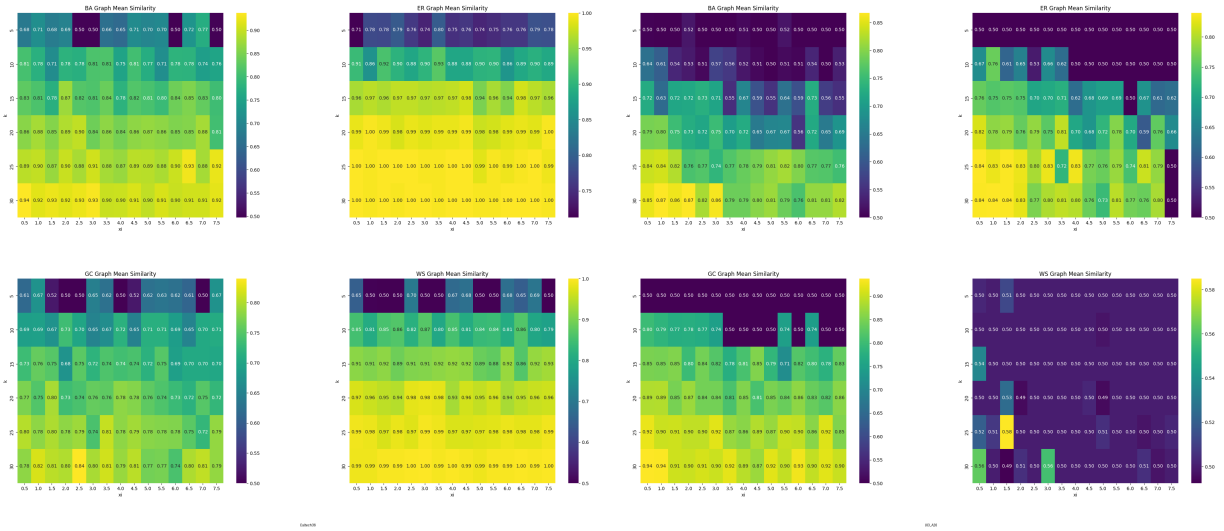
### 7.2. **Biological PPI Network.**

The Protein-Protein Interaction (PPI) networks retrieved from BioGRID for various organisms such as human, fruitfly, yeast, mouse, and worm provide critical insights into biological processes and disease mechanisms[Rossi and Ahmed, 2015]. These networks are invaluable for comparative studies, highlighting conserved interactions and unique pathways across species. Utilized in genetic, developmental, and medical research, these PPI networks from BioGRID help elucidate the molecular foundations of life, aiding researchers in understanding the complexity of cellular functions and evolutionary biology.

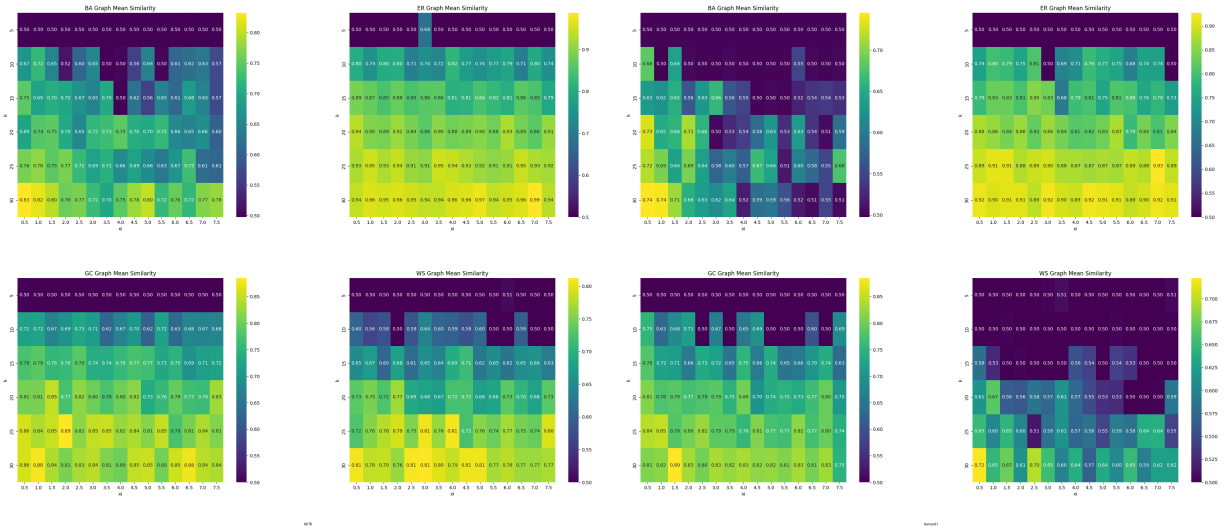
We want to conduct experiments with these PPI networks to better understand the biological network similarities among species and infer the unknown functions and properties based on the well-studied proteins.

#### 7.2.1. *Binary Network Regression.*

Similarly, we used function introduced in 7.1.1 to first understand the potential similarities among the PPI networks of 5 different species, including yeast, fruitfly, worm, human and mouse.




(A) Caltech36 (B) UCLA26

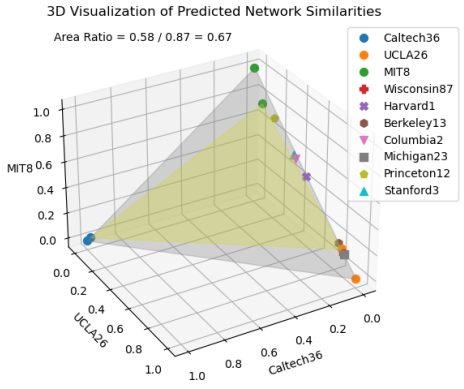



(C) MIT8 (D) Harvard1

FIGURE 5. Comparison of $\xi$ for different datasets (Caltech36, UCLA26, MIT8, and Harvard1)

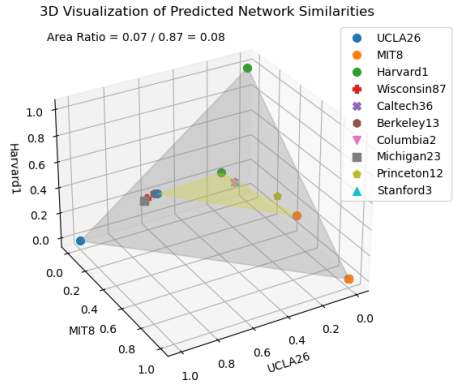
From the binary heatmap we find that yeast can be well distinguished from fruitfly, worm and human, but it is hard for yeast and mouse, suggeseting the potential similarity of their latent motifs. It also can be easily distinguished between fruitfly and mouse. For the pairs of fruitfly & worm and fruitfly & human, the probability predicted for mouse are both surprisingly 1.0, indicating that the mesoscale structure of mouse PPI networks is much more alike worm and human, rather than fruitfly. Compared to worm, yeast's PPI patterns are more silimar to human's, which aligns with the result of some biological studies.

7.2.2. *Multiclass Network Regression.*




| Network | Coordinates |
|---|---|
| Caltech36 | (0.98, 0.00, 0.02) |
| UCLA26 | (0.00, 0.86, 0.14) |
| MIT8 | (0.04, 0.14, 0.82) |
| Wisconsin87 | (0.00, 0.88, 0.11) |
| Harvard1 | (0.00, 0.52, 0.48) |
| Berkeley13 | (0.00, 0.83, 0.17) |
| Columbia2 | (0.01, 0.43, 0.56) |
| Michigan23 | (0.00, 0.89, 0.11) |
| Princeton12 | (0.02, 0.23, 0.75) |
| Stanford3 | (0.01, 0.42, 0.58) |

(A) Multiclass Network Regression ($k = 50$) with `Caltech36` as baseline




| Network | Coordinates |
|---|---|
| UCLA26 | (0.60, 0.11, 0.29) |
| MIT8 | (0.08, 0.64, 0.28) |
| Harvard1 | (0.30, 0.25, 0.44) |
| Wisconsin87 | (0.65, 0.09, 0.26) |
| Caltech36 | (0.00, 1.00, 0.00) |
| Berkeley13 | (0.61, 0.10, 0.29) |
| Columbia2 | (0.27, 0.33, 0.40) |
| Michigan23 | (0.67, 0.09, 0.25) |
| Princeton12 | (0.13, 0.51, 0.36) |
| Stanford3 | (0.26, 0.33, 0.41) |

(B) Multiclass Network Regression ($k = 50$) without `Caltech36` as baseline

FIGURE 6. Some Experiments of Multiclass Prediction - Facebook100

TABLE 1. Overview of PPI Networks from BioGRID

| **Organism** | **Model Use** | **Key Focus in PPI Network** |
|---|---|---|
| Human (*Homo sapiens*) | Medical and genetic research | Complex cellular processes, disease pathways |
| Fruitfly (*Drosophila melanogaster*) | Genetic and developmental studies | Development, behavior |
| Plant (*Arabidopsis thaliana*) | Plant biology | Development, metabolism, environmental responses |
| Yeast (*Saccharomyces cerevisiae*) | Cellular biochemistry | Fundamental cellular processes, metabolism |
| Mouse (*Mus musculus*) | Biomedical models | Genetic models of human diseases |
| Worm (*Caenorhabditis elegans*) | Developmental and neural biology | Development, aging, neural circuitry |

We moved one step further in the biological networks experiments as well. In the experiments, we found that due to the randomness of the sampling and the selection of $k$'s, there are times when we are not able to sample enough $k-$paths and trapped into the sampling process. In light of this, we introduced parameters `average` and `times`, which enables the function to conduct the supervised network dictionary learning process `times` tiems and calculate the average prediction probability when `average = True` and `skip_folded_hom = False` (sampling $k-$walks), also it gives the standard deviation of the coordinates to show the variability across iterations (as shown in Figure 8b).

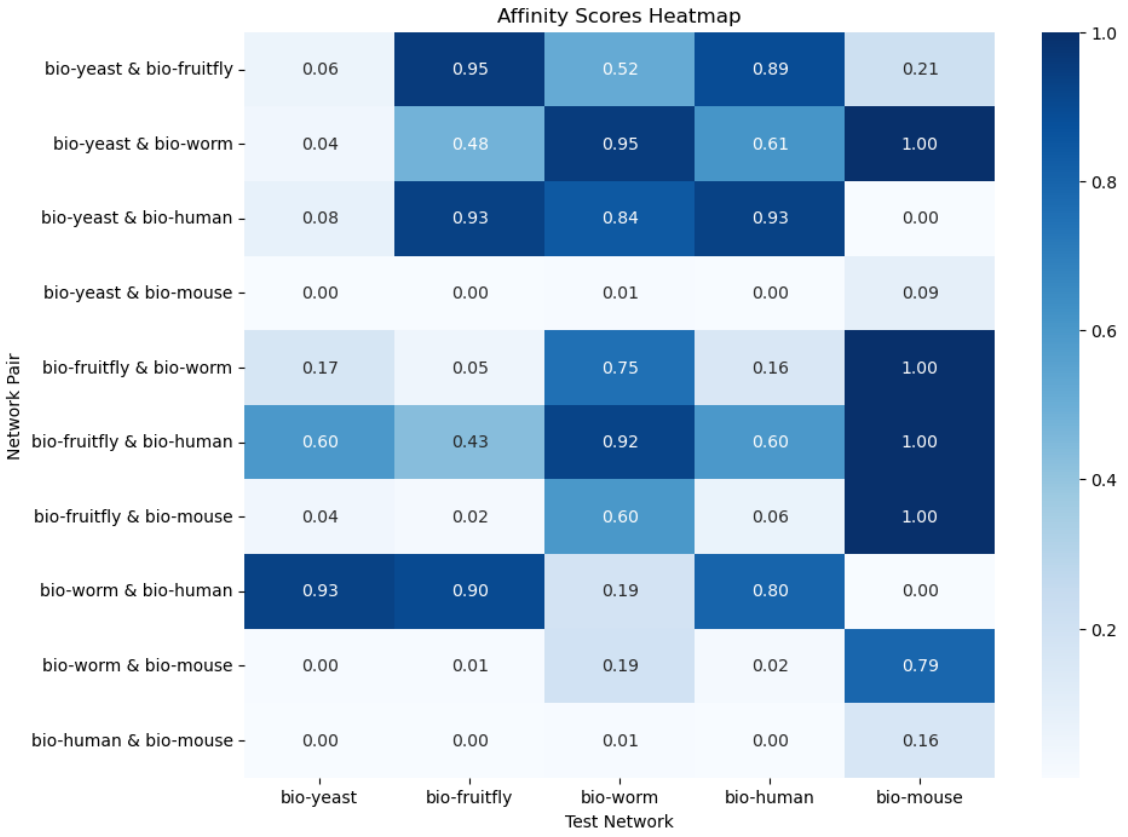
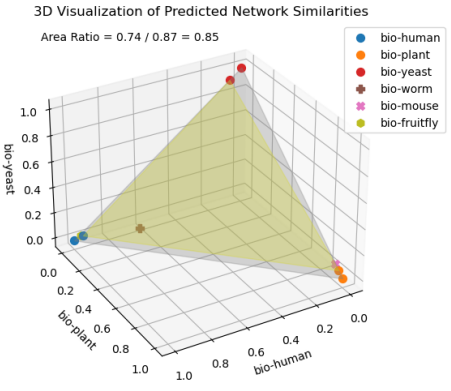



FIGURE 7. Binary Network Regression - PPI (k = 50)

From Figure 8a, where the baseline networks are `bio-human, bio-plant` and `bio-yeast`, we notice that first these three networks can be well distinguished from each other, with an area ration high 0f 0.85. Also, we find that mouse has a probability 0.93 to be predicted as plant and fruitfly has a probability of 0.96 to be predicted as human, which is interesting and need further exmaination with current biology studies. Worm is less likely to be predicted as yeast ad plant but more likely to be predicted as human.

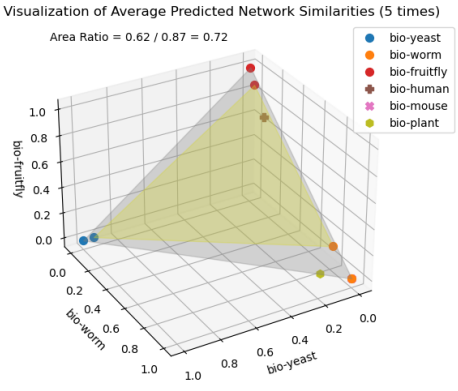
We display the average result of the 5 iterations of the experiment with baseline `bio-yeast, bio-worm` and `bio-fruitfly` in Figure 8b. Similarily, the baseline networks can be well distinguished. It suggested that compared to other two networks, human PPI network has more similar structure with fruitflt, which is aligning with our observations in 8a. Mouse is predicted as worm with a high probability 1.0 and a relatively low standard deviation (0, 0, 0). Plant is closet to worm, while the standard deviation is a little bit larger compared to other results on the plot.

These observations inspires us to do further reproducible and verifiable experiments to reveal the potential inference using network regression.




| Network | Coordinates |
|---|---|
| bio-human | (0.96, 0.01, 0.03) |
| bio-plant | (0.00, 0.96, 0.04) |
| bio-yeast | (0.07, 0.00, 0.93) |
| bio-worm | (0.71, 0.18, 0.11) |
| bio-mouse | (0.00, 0.93, 0.07) |
| bio-fruitfly | (0.96, 0.01, 0.03) |

(A) Multiclass Network Regression Baseline 1




| Network | Coordinates | Std Dev |
|---|---|---|
| bio-yeast | (0.95, 0.02, 0.02) | (0.00, 0.00, 0.00) |
| bio-worm | (0.01, 0.84, 0.15) | (0.00, 0.01, 0.01) |
| bio-fruitfly | (0.02, 0.07, 0.91) | (0.00, 0.01, 0.01) |
| bio-human | (0.04, 0.21, 0.75) | (0.01, 0.02, 0.02) |
| bio-mouse | (0.00, 1.00, 0.00) | (0.00, 0.00, 0.00) |
| bio-plant | (0.11, 0.88, 0.00) | (0.13, 0.12, 0.00) |

(B) Multiclass Network Regression Baseline 2

FIGURE 8. Some Experiments of Multiclass Prediction - Biological Networks

## 8. Conclusion and Future Works

In this paper, we propose our interpretable Supervised Network Dictionary Learning (SNDL) model for network regression, which is robust and efficient for prediction tasks and network affinity prediction tasks. We validate our results through several numerical experiments using datasets from both social and biological networks.

Future research directions include investigating the theoretical guarantees of the SNDL model for prediction tasks, identifying appropriate methods or principles for selecting hyperparameters such as $r$, $n$, and $k$, and conducting additional experiments to demonstrate that SNDL remains computationally efficient for large-scale network data compared to other baseline methods. Additionally, synthetic networks with hard-coded properties could be generated to enhance clustering performance by distinguishing between different categories. This approach would be especially useful in handling challenging cases like the all-inclusive synthetic networks. Further exploration could also involve incorporating node features into the dictionary learning and regression processes, which may enhance the predictive capabilities of the SNDL algorithm, particularly for biological networks where node attributes, such as protein function, are essential for classification. Moreover, the methods developed in this project could be extended to brain networks, focusing on distinguishing different states, such as active versus resting, or benign versus malignant, by adapting the SNDL algorithm to address the unique properties of brain networks.

## 9. Reconstruction error bounds

The classical network reconstruction bound: $G = (V, A)$, $G_{\text{recons}} = (V, \hat{A})$

$$\frac{\|A - \hat{A}\|_{1,\pi}}{\|A \vee \hat{A}\|_{1,\pi}} = d_{JD}(G, G_{\text{recons}}) \leq \frac{1}{k}\mathbb{E}_{\mathbf{x}}[\|A_{\mathbf{x}} - \hat{A}_{\mathbf{x};\mathbf{W}}\|_1]. \tag{9.1}$$

We need to assume that our dictionary $\mathbf{W}$ is effective to minimize the expected subgraph approximation error above.

Can we have a similar reconstruction error bound for predictive probabilities? Let $\hat{\mathbf{p}}(G) = (\hat{p}_1(G), \ldots, \hat{p}_\kappa(G))$, where $p_i(G)$ =predictive probability that $G$ belongs to class $i$. $\beta \in \mathbb{R}^{\kappa \times r}$. $\beta_i = i$th row of $\beta$. Then

$$\hat{p}_i(G) = \mathbb{E}_{\mathbf{x}}[\hat{p}_i(A_{\mathbf{x}})] \tag{9.2}$$

$$= \mathbb{E}_{\mathbf{x}}\left[\frac{\exp(\beta_i^T \mathbf{W}^T \mathrm{Vec}(A_{\mathbf{x}}))}{1 + \sum_{c=1}^{\kappa} \exp(\beta_c^T \mathbf{W}^T \mathrm{Vec}(A_{\mathbf{x}}))}\right] \tag{9.3}$$

Suppose $G$ has class 1. Then

$$\hat{\mathbf{p}}(G) \approx (0, 1, 0, \ldots, 0). \tag{9.4}$$

$$|\mathbf{1}_{(i=1)} - \hat{p}_i(G)| = \text{small} \tag{9.5}$$

$$\mathbf{1}_{(i=c(G))} - \mathbb{E}_{\mathbf{x}}[\hat{p}_i(A_{\mathbf{x}})] = \mathbb{E}_{\mathbf{x}}[\mathbf{1}_{(i=c(G))}] - \mathbb{E}_{\mathbf{x}}[\hat{p}_i(A_{\mathbf{x}})] \tag{9.6}$$

$$= \mathbb{E}_{\mathbf{x}}[\mathbf{1}_{(i=c(G))} - \hat{p}_i(A_{\mathbf{x}})] \tag{9.7}$$

$$\mathbb{E}_{\mathbf{x}}[\text{log-likelihood function?}] \tag{9.8}$$

9.1. **Notations.**

- **Networks:**
  - $G = (V, A)$: Original network with node set $V$ and adjacency matrix $A \in \mathbb{R}^{n \times n}$.
  - $G' = (V, A')$: Another network with the same node set $V$ and adjacency matrix $A' \in \mathbb{R}^{n \times n}$.
- **Predictive Probabilities:**
  - For each class $i \in \{0, 1, 2, \ldots, \kappa\}$, the predictive probability $p_i(G)$ is defined as:

$$p_i(G) = \mathbb{E}_{\mathbf{x}}[\hat{p}_i(A_{\mathbf{x}})],$$

    where:
    - $\mathbf{x}$: A subgraph sampled from $G$.
    - $A_{\mathbf{x}}$: Adjacency matrix of subgraph $\mathbf{x}$.
    - $\hat{p}_i(A_{\mathbf{x}})$: Predictive probability for class $i$ based on subgraph $\mathbf{x}$.
- **Activation Function:**

$$a_i(A_{\mathbf{x}}) = \beta_i^\top \mathbf{W}^\top \mathrm{Vec}(A_{\mathbf{x}}),$$

  where:
  - $\beta_i \in \mathbb{R}^r$: Coefficient vector for class $i$.
  - $\mathbf{W} \in \mathbb{R}^{d \times r}$: Dictionary matrix.
  - $\mathrm{Vec}(A_{\mathbf{x}}) \in \mathbb{R}^d$: Vectorized form of $A_{\mathbf{x}}$.
- **Predictive Probability for Subgraph x:**

$$\hat{p}_i(A_{\mathbf{x}}) = \frac{\exp(a_i(A_{\mathbf{x}}))}{\sum_{c=0}^{\kappa} \exp(a_c(A_{\mathbf{x}}))}.$$

- **Jaccard Reconstruction Error:**

$$d_{JD}(G, G_{\text{recons}}) = \frac{\|A - \hat{A}\|_1}{\|A \vee \hat{A}\|_1}.$$

9.2. **Assumptions.**

(1) **Boundedness of Model Parameters:**

$$\|\beta_i\|_2 \leq C_\beta, \quad \|\mathbf{W}\|_2 \leq C_W, \quad \forall i.$$

(2) **Bounded Activations:**

$$|a_i(A_{\mathbf{x}})| \leq M_a, \quad \forall i, \forall \mathbf{x}.$$

(3) **Uniform Sampling of Subgraphs:** Subgraphs $\mathbf{x}$ are sampled uniformly at random from all subgraphs of size $k$.

(4) **Bounded Norm of $A \vee \hat{A}$:**

$$\|A \vee \hat{A}\|_1 \leq M, \quad \text{for some constant } M.$$

9.3. **Theorems.** The *network denoising and reconstruction* (NDR) algorithm in Lyu et al. [2024] works as follows. We seek to build a weighted network $G_{\text{recons}}$ using the node set $V$ and a weighted adjacency matrix $A_{\text{recons}} : V^2 \to \mathbf{r}$. This network best approximates the observed network $G$, whose subgraphs are generated by the latent motifs in $W$. Namely, for each nodes $x, y \in V$, we define

$$A_{\text{recons}}(x, y) = \mathbb{E}_{\boldsymbol{x}} \left[ \hat{A}_{\boldsymbol{x}}(i_{\boldsymbol{x}}, j_{\boldsymbol{x}}) \,|\, x, y \in \boldsymbol{x} \right], \tag{9.9}$$

where $\boldsymbol{x}$ is uniformly random $k$-paths in $G$, $\hat{A}_{\boldsymbol{x}}$ is the best linear approximation of the adjacency matrix $A_{\boldsymbol{x}}$ of the $k$-node induced subgraph on $\boldsymbol{x}$ in terms of the latent motifs $\mathcal{L}_1, \ldots, \mathcal{L}_r$, $\{x, y \in \boldsymbol{x}\}$ denotes the event that $\boldsymbol{x}$ uses both nodes $x$ and $y$, and $i_{\boldsymbol{x}}$ and $j_{\boldsymbol{x}}$ are the unique integers in $[k]$ such that $\boldsymbol{x}(i_{\boldsymbol{x}}) = x$ and $\boldsymbol{x}(j_{\boldsymbol{x}}) = y$, respectively. In Lyu et al. [2024], it was established that latent motifs that are effective in reconstructing most subgraph patterns are also effective in reconstructing the whole graph:

$$\frac{\|A - A_{\text{recons}}\|_1}{\|A \vee A_{\text{recons}}\|_1} \leq \frac{1}{2k} \mathbb{E}_{\boldsymbol{x}}[\|A_{\boldsymbol{x}} - \hat{A}_{\boldsymbol{x}}\|_1], \tag{9.10}$$

where $a \vee b := \max(a, b)$.

**1.** Two baseline graphs $G, G'$

**2.** A test graph $H \approx G$.

**3.** Learn $(\mathbf{W}, \beta)$ s.t. $G \approx \hat{G}_{\mathbf{W}}$, $G' \approx \hat{G}'_{\mathbf{W}}$, $\mathbf{p}_{\mathbf{W},\beta}(H) \approx (1, 0)$, where

$$\mathbf{p}_{\mathbf{W},\beta}(H) = \left( \mathbb{E}_{\boldsymbol{x}}[\hat{p}_0(A^H_{\boldsymbol{x}})], \mathbb{E}_{\boldsymbol{x}}[\hat{p}_1(A^H_{\boldsymbol{x}})] \right) \tag{9.11}$$

$$d(G, \hat{G}_{\mathbf{W}}) + d(G', \hat{G}'_{\mathbf{W}}) + d((1, 0), \mathbf{p}_{\mathbf{W},\beta}(H)) \leq \text{SMF objective}(\mathbf{W}, \beta) \tag{9.12}$$

Training:

$$d_{KL}((1, 0), \mathbf{p}_{\mathbf{W},\beta}(G)) + d_{KL}((0, 1), \mathbf{p}_{\mathbf{W},\beta}(G')) \leq C + \text{classification part of SMF objective}(\mathbf{W}, \beta) \tag{9.13}$$

$$d(G, \hat{G}_{\mathbf{W}}) + d(G', \hat{G}_{\mathbf{W}}) \leq \text{recons. part of SMF objective}(\mathbf{W}, \beta) \tag{9.14}$$

[**YW:**Connect the loss with KL divergence, and do the Jensen.]

(9.15)
$$\frac{1}{2n}\sum_{i=1}^{2n}\ell(y_i,\mathbf{a}_i)=\frac{1}{n}\sum_{i=1}^{n}\ell(0,\mathbf{a}_i)+\frac{1}{n}\sum_{i=n+1}^{2n}\ell(1,\mathbf{a}_i)$$

(9.16)
$$=C+\frac{1}{n}\sum_{i=1}^{n}D_{KL}((1,0),1-\hat{\mathbf{p}}(A_{\boldsymbol{x}_i}))+\frac{1}{n}\sum_{i=n+1}^{2n}D_{KL}((0,1),\hat{\mathbf{p}}(A'_{\boldsymbol{x}_i}))$$

(9.17)
$$\geq C+D_{KL}((1,0),\underbrace{\frac{1}{n}\sum_{i=1}^{n}1-\hat{\mathbf{p}}(A_{\boldsymbol{x}})}_{=\hat{p}_0(G)})+D_{KL}((0,1),\underbrace{\frac{1}{n}\sum_{i>n}\hat{\mathbf{p}}(A'_{\boldsymbol{x}})}_{=\hat{p}_1(G')})\quad\text{(Jensen, convexity of KL)}$$

Same computation in the population version:

(9.18)
$$\mathbb{E}_{\boldsymbol{x}}[\ell(y,\mathbf{a})]\geq C+\sum_{c=0}^{1}D_{KL}(\delta_c,\mathbb{E}_{\boldsymbol{x}}[\hat{\mathbf{p}}_c(A_{\boldsymbol{x}})\mathbf{1}(y=c)])$$

Given $(\boldsymbol{x}_i,y_i)$,

(9.19)
$$\ell(\beta)=\frac{1}{n}\sum_{i=1}^{n}\log\mathbb{P}_\beta(\hat{y}_i=y_i\,|\,\boldsymbol{x}_i)$$

(9.20)
$$=\frac{1}{n}\sum_{i=1}^{n}\underbrace{y_i\log p_i+(1-y_i)\log p_i}_{=\text{CrossEntropy}(y,x,\beta)}$$

where $p_i=\mathbb{P}(\hat{y}=1\,|\,\boldsymbol{x}_i)$. The above relation follows from noting that

(9.21)
$$\mathbb{P}(\hat{y}=y_i\,|\,\boldsymbol{x}_i)=p_i^{y_i}(1-p_i)^{1-y_i}.$$

To verify this claim, $y_i=1$. Then

(9.22)
$$\log\mathbb{P}_\beta(\hat{y}_i=y_i\,|\,\boldsymbol{x}_i)=\log\frac{\exp(\langle\beta,\boldsymbol{x}\rangle)}{1+\exp(\langle\beta,\boldsymbol{x}\rangle)}$$

(9.23)
$$=$$

Define $q_{i,0}=0$; For $c=1,...,\kappa$

(9.24)
$$q_{i,c}=\begin{cases}1\text{ if }c=y_i,\\0\text{ o.w..}\end{cases}$$

Since

(9.25)
$$\ell(y_i,\boldsymbol{a}_i)=-(\boldsymbol{a}_iy_i-\log(1+\sum_{k=1}^{\kappa}\exp\boldsymbol{a}_{i,k})).$$

then we have

$$\begin{aligned} D_{KL}(q\|p) &= \sum_{c=0}^{\kappa} \underbrace{q_{i,c}}_{=\mathbf{1}(y_i=c)} \log(\frac{q_{i,c}}{p_{i,c}}) \\ &= \sum_{c=0}^{\kappa} \mathbf{1}(y_i=c)\log \mathbf{1}(y_i=c) - \mathbf{1}(y_i=c)\log p_i(\beta) \\ &= Const. - \sum_{c=0}^{\kappa} \mathbf{1}(y_i=c)\log p_i(\beta) \\ &= Const. + \text{NegativeLogLikelihood} \end{aligned}$$

$$d_{KL}((1,0), \mathbf{p}_{\mathbf{W},\beta}(G)) \approx |1-\hat{p}_0(G)| + |0-\hat{p}_1(G)| \tag{9.26}$$

**Theorem 9.1.** *For any two networks $G$ and $G'$ with adjacency matrices $A$ and $A'$, the predictive probabilities $p_i(G)$ are Lipschitz continuous with respect to $A$, i.e., there exists a constant $L$ such that:*

$$|p_i(G) - p_i(G')| \leq L\|A - A'\|_F,$$

*where $\|\cdot\|_F$ denotes the Frobenius norm.*

$$|\hat{p}_i(A_{\mathbf{x}}) - \hat{p}_i(A'_{\mathbf{x}})| \leq L_1\|A_{\mathbf{x}} - A'_{\mathbf{x}}\|_1 \tag{9.27}$$

$$\begin{aligned} |p_i(G) - p_i(G')| &= |\mathbb{E}[\hat{p}_i(A_{\mathbf{x}})] - \mathbb{E}[\hat{p}_i(A'_{\mathbf{x}})]| && (9.28)\\ &\leq \mathbb{E}[|\hat{p}_i(A_{\mathbf{x}}) - \hat{p}_i(A'_{\mathbf{x}})|] && (9.29)\\ &\leq L_1\mathbb{E}\left[\|A_{\mathbf{x}} - A'_{\mathbf{x}}\|_1\right] && (9.30) \end{aligned}$$

$$\begin{aligned} |p_i(G) - p_i(G_{\text{recons}})| &= |\mathbb{E}[\hat{p}_i(A_{\mathbf{x}})] - \mathbb{E}[\hat{p}_i(A'_{\mathbf{x}})]| && (9.31)\\ &\leq \mathbb{E}[|\hat{p}_i(A_{\mathbf{x}}) - \hat{p}_i(A'_{\mathbf{x}})|] && (9.32)\\ &\leq L_1\mathbb{E}\left[\|A_{\mathbf{x}} - A'_{\mathbf{x}}\|_1\right] && (9.33) \end{aligned}$$

$$F(\mathbf{W},\beta) = \text{SMF objective that we minimize to find optimal } W \text{ and } \beta \tag{9.34}$$

$$= \text{measure of accuracy in the subgraph level} \tag{9.35}$$

$$\text{(some measure of global accuracy of our problem)} \leq \frac{1}{k}F(\mathbf{W},\beta)?? \tag{9.36}$$

*Given baseline graphs $G_1$, $G_2$:*

$$\text{Global network recons error} = d(G_1, G_{1;recons})^2 + d(G_1, G_{2;recons})^2 \tag{9.37}$$

*What is the global level of regression model? Formulate a regression model for the true labels following logistic regression.*

*Compare with*

$$\frac{\|A - \hat{A}\|_{1,\pi}}{\|A \vee \hat{A}\|_{1,\pi}} = d_{JD}(G, G_{\text{recons}}) \leq \frac{1}{k}\mathbb{E}_{\mathbf{x}}[\|A_{\mathbf{x}} - \hat{A}_{\mathbf{x};\mathbf{W}}\|_1]. \tag{9.38}$$

*Proof.* Consider corresponding subgraphs $\mathbf{x}$ from $G$ and $\mathbf{x}$ from $G'$ (over the same nodes).

$$\|A_{\mathbf{x}} - A'_{\mathbf{x}}\|_F \leq \|A - A'\|_F,$$

since $A_{\mathbf{x}}$ and $A'_{\mathbf{x}}$ are submatrices of $A$ and $A'$, respectively.

For each class $i$:

$$\begin{aligned}
|a_i(A_{\mathbf{x}}) - a_i(A'_{\mathbf{x}})| &= \left|\beta_i^\top \mathbf{W}^\top (\mathrm{Vec}(A_{\mathbf{x}}) - \mathrm{Vec}(A'_{\mathbf{x}}))\right| \\
&\leq \|\beta_i\|_2 \|\mathbf{W}^\top\|_2 \, \|\mathrm{Vec}(A_{\mathbf{x}}) - \mathrm{Vec}(A'_{\mathbf{x}})\|_2 \\
&\leq C_\beta C_W \|\mathrm{Vec}(A_{\mathbf{x}}) - \mathrm{Vec}(A'_{\mathbf{x}})\|_2 \qquad \text{[YW:tight bound in the last step?]} \\
&\leq C_\beta C_W \|A_{\mathbf{x}} - A'_{\mathbf{x}}\|_F \\
&\leq C_\beta C_W \|A - A'\|_F.
\end{aligned}$$

Suppose the softmax function $\sigma : \mathbb{R}^{\kappa+1} \to [0,1]^{\kappa+1}$ maps $\mathbf{a}$ to $\hat{\mathbf{p}}(A_{\mathbf{x}})$.

Let $\mathbf{a}(A_{\mathbf{x}}) = [a_0(A_{\mathbf{x}}), a_1(A_{\mathbf{x}}), \ldots, a_\kappa(A_{\mathbf{x}})]^\top$. Over the domain where $|a_i(A_{\mathbf{x}})| \leq M_a$, the softmax function is Lipschitz continuous with constant $L_s$ depending on $\kappa + 1$.

Therefore:

$$\|\hat{\mathbf{p}}(A_{\mathbf{x}}) - \hat{\mathbf{p}}(A'_{\mathbf{x}})\|_1 \leq L_s \|\mathbf{a}(A_{\mathbf{x}}) - \mathbf{a}(A'_{\mathbf{x}})\|_2.$$

Using the bound on activations:

$$\begin{aligned}
\|\mathbf{a}(A_{\mathbf{x}}) - \mathbf{a}(A'_{\mathbf{x}})\|_2 &\leq \sqrt{\kappa+1} \max_i |a_i(A_{\mathbf{x}}) - a_i(A'_{\mathbf{x}})| \\
&\leq \sqrt{\kappa+1} \cdot C_\beta C_W \|A - A'\|_F.
\end{aligned}$$

[YW:Keep the last step to subgraph not the full graph $\|A - A'\|_F$]

Therefore:

$$\|\hat{\mathbf{p}}(A_{\mathbf{x}}) - \hat{\mathbf{p}}(A'_{\mathbf{x}})\|_1 \leq L_s \sqrt{\kappa+1}\, C_\beta C_W \|A - A'\|_F.$$

[YW:Normalize the vector by difference between softmax distributions.]

Since:

$$p_i(G) = \mathbb{E}_{\mathbf{x}}[\hat{p}_i(A_{\mathbf{x}})], \quad p_i(G') = \mathbb{E}_{\mathbf{x}}[\hat{p}_i(A'_{\mathbf{x}})],$$

we have:

$$|p_i(G) - p_i(G')| = |\mathbb{E}_{\mathbf{x}}[\hat{p}_i(A_{\mathbf{x}}) - \hat{p}_i(A'_{\mathbf{x}})]| \leq \mathbb{E}_{\mathbf{x}}\left[|\hat{p}_i(A_{\mathbf{x}}) - \hat{p}_i(A'_{\mathbf{x}})|\right].$$

Using the bound from Step 1:

$$|\hat{p}_i(A_{\mathbf{x}}) - \hat{p}_i(A'_{\mathbf{x}})| \leq \|\hat{\mathbf{p}}(A_{\mathbf{x}}) - \hat{\mathbf{p}}(A'_{\mathbf{x}})\|_1 \leq L_s \sqrt{\kappa+1}\, C_\beta C_W \|A - A'\|_F.$$

Therefore:

$$|p_i(G) - p_i(G')| \leq \mathbb{E}_{\mathbf{x}}\left[L_s \sqrt{\kappa+1}\, C_\beta C_W \|A - A'\|_F\right] = L_s \sqrt{\kappa+1}\, C_\beta C_W \|A - A'\|_F.$$

We have shown that:

$$|p_i(G) - p_i(G')| \leq L \|A - A'\|_F,$$

where:

$$L = L_s \sqrt{\kappa+1}\, C_\beta C_W.$$

□

**Theorem 9.2.** *Under the above assumptions, the error in the estimated predictive probabilities for each class $i \in \{0, 1, \ldots, \kappa\}$ satisfies:*

$$|p_i(G) - \hat{p}_i(G)| \leq \frac{\sqrt{n} L M}{k} \mathbb{E}_{\mathbf{x}}[\|A_{\mathbf{x}} - \hat{A}_{\mathbf{x};\mathbf{W}}\|_1].$$

*Proof.* From the Lipschitz continuity assumption:

$$|p_i(G) - p_i(G_{\text{recons}})| \leq L\|A - \hat{A}\|_F.$$

Since $\|A - \hat{A}\|_F \leq \sqrt{n}\|A - \hat{A}\|_1$, we have:

$$|p_i(G) - p_i(G_{\text{recons}})| \leq \sqrt{n}L\|A - \hat{A}\|_1. \tag{9.39}$$

From the Jaccard reconstruction error:

$$d_{JD}(G, G_{\text{recons}}) = \frac{\|A - \hat{A}\|_1}{\|A \vee \hat{A}\|_1} \leq \frac{1}{k}\mathbb{E}_{\mathbf{x}}[\|A_{\mathbf{x}} - \hat{A}_{\mathbf{x};\mathbf{W}}\|_1].$$

Therefore:

$$\|A - \hat{A}\|_1 \leq M \cdot d_{JD}(G, G_{\text{recons}}) \leq \frac{M}{k}\mathbb{E}_{\mathbf{x}}[\|A_{\mathbf{x}} - \hat{A}_{\mathbf{x};\mathbf{W}}\|_1].$$

Substituting the bound from (9.39):

$$|p_i(G) - \hat{p}_i(G)| \leq \sqrt{n}L \cdot \frac{M}{k}\mathbb{E}_{\mathbf{x}}[\|A_{\mathbf{x}} - \hat{A}_{\mathbf{x};\mathbf{W}}\|_1] = \frac{\sqrt{n}LM}{k}\mathbb{E}_{\mathbf{x}}[\|A_{\mathbf{x}} - \hat{A}_{\mathbf{x};\mathbf{W}}\|_1].$$

In conclusion, we obtain:

$$|p_i(G) - \hat{p}_i(G)| \leq \frac{\sqrt{n}LM}{k}\mathbb{E}_{\mathbf{x}}[\|A_{\mathbf{x}} - \hat{A}_{\mathbf{x};\mathbf{W}}\|_1].$$

□

## References


W. Austin, D. Anderson, and J. Ghosh. Fully supervised non-negative matrix factorization for feature extraction. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 5772–5775, 2018.

Samuel Burer and Renato DC Monteiro. Nonlinear programming and low-rank semidefinite programming with applications to combinatorial optimization. In *SIAM Journal on Optimization*, volume 14, pages 434–470, 2003.

Deng Cai, Xiaofei He, and Jiawei Han. Graph regularized non-negative matrix factorization for data representation. *IEEE transactions on pattern analysis and machine intelligence*, 33(8): 1548–1560, 2010.

Matthias Dehmer. *Strukturelle analyse web-basierter dokumente*. Springer-Verlag, 2007.

Matthias Dehmer and Frank Emmert-Streib. Mining graph patterns in web-based systems: A conceptual view. *Genres on the web: Computational models and empirical studies*, pages 237–253, 2011.

Frank Emmert-Streib and Matthias Dehmer. Networks for systems biology: conceptual connection of data and function. *IET systems biology*, 5(3):185–207, 2011.

Frank Emmert-Streib, Matthias Dehmer, and Yongtang Shi. Fifty years of graph matching, network alignment and network comparison. *Information Sciences*, 346-347:180–197, 2016. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2016.01.074. URL https://www.sciencedirect.com/science/article/pii/S002002551630010X.

Anna Goldenberg, Alice X Zheng, Stephen E Fienberg, and Edoardo M Airoldi. A survey of statistical network models, 2009. URL https://arxiv.org/abs/0912.5410.

Rick Grannis. *Sampling Effects in Social Network Analysis*, page 2281–2290. Springer New York, 2018. ISBN 9781493971312. doi: 10.1007/978-1-4939-7131-2_37. URL http://dx.doi.org/10.1007/978-1-4939-7131-2_37.

L. Grippo and M. Sciandrone. On the convergence of the block nonlinear gauss–seidel method under convex constraints. *Operations research letters*, 26(3):127–136, 2000.

Shawn Gu and Tijana Milenković. Data-driven biological network alignment that uses topological, sequence, and functional information. *BMC Bioinformatics*, 22(1), January 2021. ISSN 1471-2105. doi: 10.1186/s12859-021-03971-6. URL http://dx.doi.org/10.1186/s12859-021-03971-6.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

Peter D Hoff. Modeling homophily and stochastic equivalence in symmetric relational data. In *Advances in Neural Information Processing Systems*, pages 657–664, 2008.

Brian P. Kelley, Roded Sharan, Richard M. Karp, Taylor Sittler, David E. Root, Brent R. Stockwell, and Trey Ideker. Conserved pathways within bacteria and yeast as revealed by global protein network alignment. *Proceedings of the National Academy of Sciences*, 100(20):11394–11399, September 2003. ISSN 1091-6490. doi: 10.1073/pnas.1534710100. URL http://dx.doi.org/10.1073/pnas.1534710100.

Yoon Kim, Emily Denton, Luong Hoang, and Alexander M. Rush. Convolutional matrix factorization for document modeling. *Proceedings of the 33rd International Conference on Machine Learning*, pages 699–707, 2016.

Eric D Kolaczyk and Gábor Csárdi. *Statistical analysis of network data with R*, volume 65. Springer, 2014.

David Krackhardt. Predicting with networks: Nonparametric multiple regression analysis of dyadic data. *Social Networks*, 10(4):359–381, 1988.

Oleksii Kuchaiev, Tijana Milenković, Vesna Memišević, Wayne Hayes, and Nataša Pržulj. Topological network alignment uncovers biological function and phylogeny. *Journal of The Royal Society Interface*, 7(50):1341–1354, March 2010. ISSN 1742-5662. doi: 10.1098/rsif.2010.0063. URL http://dx.doi.org/10.1098/rsif.2010.0063.

Daniel Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13:556–562, 2000.

Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788, 1999.

Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13:556–562, 2001.

J. Lee, H. Lyu, and W. Yao. Exponentially convergent algorithms for supervised matrix factorization. *Neural Information Processing Systems*, 2023a.

J. Lee, H. Lyu, and W. Yao. Supervised matrix factorization: Local landscape analysis and applications. In *Proceedings of the 41st International Conference on Machine Learning*, 2024a.

Joowon Lee, Hanbaek Lyu, and Weixin Yao. Exponentially convergent algorithms for supervised matrix factorization. *Neural Information Processing Systems*, 2023b.

Joowon Lee, Hanbaek Lyu, and Weixin Yao. Supervised matrix factorization: Local landscape analysis and applications. In *Forty-first International Conference on Machine Learning*, 2024b. URL https://openreview.net/forum?id=YlJy1FcM9E.

J. Leuschner, M. Schmidt, P. Fernsel, et al. Supervised non-negative matrix factorization methods for maldi imaging applications. *Bioinformatics*, 35(11):1940–1947, 2019.

Yingjie Li, Elizaveta Levina, and Ji Zhu. Generalized linear models with network-linked responses. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 81(2):453–476, 2019.

Chung-Shou Liao, Kanghao Lu, Michael Baym, Rohit Singh, and Bonnie Berger. Isorankn: spectral methods for global alignment of multiple protein networks. *Bioinformatics*, 25(12):i253–i258, May 2009. ISSN 1367-4803. doi: 10.1093/bioinformatics/btp203. URL http://dx.doi.org/10.1093/bioinformatics/btp203.

Hanbaek Lyu, Facundo Memoli, and David Sivakoff. Sampling random graph homomorphisms and applications to network data analysis. *Journal of Machine Learning Research*, 24:1–79, 2023.

Hanbaek Lyu, Yacoub H Kureh, Joshua Vendrow, and Mason A Porter. Learning low-rank latent mesoscale structures in networks. *Nature Communications*, 15(1):224, 2024.

J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. Bach. Supervised dictionary learning. In *Advances in Neural Information Processing Systems*, volume 21, pages 1033–1040, 2008.

W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, October 2019. ISSN 1091-6490. doi: 10.1073/pnas.1900654116. URL http://dx.doi.org/10.1073/pnas.1900654116.

Mark E. J. Newman. *Networks*. Oxford University Press, Oxford, UK, second edition, 2018.

Juyong Park and Mark EJ Newman. Statistical mechanics of networks. *Physical Review E*, 70(6): 066117, 2004.

A. Ritchie, L. Balzano, D. Kessler, et al. Supervised pca: A multiobjective approach. *arXiv preprint arXiv:2011.05309*, 2020.

Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015. URL https://networkrepository.com.

Roded Sharan, Silpa Suthram, Ryan M. Kelley, Tanja Kuhn, Scott McCuine, Peter Uetz, Taylor Sittler, Richard M. Karp, and Trey Ideker. Conserved patterns of protein interaction in multiple species. *Proceedings of the National Academy of Sciences*, 102(6):1974–1979, February 2005. ISSN 1091-6490. doi: 10.1073/pnas.0409522102. URL http://dx.doi.org/10.1073/pnas.0409522102.

Rohit Singh, Jinbo Xu, and Bonnie Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences*, 105(35):12763–12768, September 2008. ISSN 1091-6490. doi: 10.1073/pnas.0806627105. URL http://dx.doi.org/10.1073/pnas.0806627105.

Tom AB Snijders, Gerhard GG van de Bunt, and Christian EG Steglich. Introduction to stochastic actor-based models for network dynamics. *Social Networks*, 32(1):44–60, 2006.

Sucheta Soundarajan, Tina Eliassi-Rad, and Brian Gallagher. A guide to selecting a network similarity method. In *SDM*, 2014. URL https://api.semanticscholar.org/CorpusID:10509701.

Mattia Tantardini, Francesca Ieva, Lucia Tajoli, and Carlo Piccardi. Comparing methods for comparing networks. *Scientific Reports*, 9(1), November 2019. ISSN 2045-2322. doi: 10.1038/s41598-019-53708-y. URL http://dx.doi.org/10.1038/s41598-019-53708-y.

John 1953 Willett. *Similarity and clustering in chemical information systems*. John Wiley & Sons, Inc., 1987.

Peter Wills and François G. Meyer. Metrics for graph comparison: A practitioner's guide. *PLOS ONE*, 15(2):e0228728, February 2020. ISSN 1932-6203. doi: 10.1371/journal.pone.0228728. URL http://dx.doi.org/10.1371/journal.pone.0228728.

S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

Y. Xu and W. Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on Imaging Sciences*, 6(3):1758–1789, 2013.

Qi Kuang, Department of Mathematics, University of Wisconsin – Madison, Madison, WI 53706
*Email address*: qkuang5@wisc.edu

Yi Wei, Department of Mathematics, University of Wisconsin – Madison, Madison, WI 53706
*Email address*: ywei224@wisc.edu

David Jiang, Department of Mathematics, University of Wisconsin – Madison, Madison, WI 53706
*Email address*: djiang38@wisc.edu

Hanbaek Lyu, Department of Mathematics, University of Wisconsin – Madison, Madison, WI 53706
*Email address*: hlyu@math.wisc.edu