Structured discovery in graphs: Recommender systems and temporal graph analysis

Sheyda Peyman

1 Introduction

Graph-valued data arises in numerous diverse scientific fields ranging from sociology, epidemiology and genomics to neuroscience and economics. For example, sociologists have used graphs to examine the roles of user attributes (gender, class, year) at American colleges and universities through the study of Facebook friendship networks [47] and have studied segregation and homophily in social networks [34]; epidemiologists have recently modeled Human-nCov protein-protein interactions via graphs [41], and neuroscientists have used graphs to model neuronal connectomes [14].

In this work, a graph is an ordered pair $G = (V, E)$ where V is the set of vertices/nodes of the graph and E is the set of edges. A network (or a graph: for our purposes, these words are used interchangibly) can be thought of as a set of objects (V) and interactions between them (E). The objects can be anything from people, proteins, to neurons and the interactions can be anything from phone calls (emails sent) between individuals, being friends on social media, to synapses between neurons or protein-protein interactions. A *directed* graph is one in which the interactions are only "one-way" (i.e., $E \subset V \times V$) . For example, the synapses between a set of neurons can be modeled by a directed graph, since each synapse starts at one neuron and ends at another. An *undirected* graph, on the other hand, is a graph in which the interactions are "two-way" (i.e., $E \subset \binom{V}{2}$). Most graphs under consideration in this dissertation will be undirected graphs.

The structure of graphs, including latent features, relationships between the vertex and importance of each vertex are all highly important graph properties that are main aspects of graph analysis/inference. While it is common to imbue nodes and/or edges with implicitly observed numeric or qualitative features, in this work we will consider latent network features that must be estimated from the network topology. The main focus of this text is to find ways of extracting the latent structure in the presence of network anomalies. These anomalies occur in different scenarios: including cases when the graph is subject to an adversarial attack and the anomaly is inhibiting inference (Section 2-3), and in the scenario when detecting the anomaly is the key inference task (Section 4). The former case is explored in the context of vertex nomination information retrieval, where we consider both analytic methods for countering the adversarial noise (Section 2) and also the addition of a user-in-the-loop in the retrieval algorithm to counter potential adversarial noise (Section 3). In the latter case we use graph embedding methods to discover sequential anomalies in network time series. Below we describe each of these three projects in more detail.

2 Adversarial Contamination

In many cases of interest, inference of graph data is further complicated by the presence of adversarial data contamination designed to reduce algorithmic effectiveness [12, 28, 29]. Often, the effect of the adversary is to change the data distribution in ways that negatively affect statistical inference and algorithmic performance. In this chapter we study this phenomenon in the context of the *vertex nomination* (VN) inference task [9, 10, 16, 31, 33], a semi-supervised information retrieval task akin to personalized recommender systems [38] on graphs. Informally, the vertex nomination (VN) problem we pursue herein can be stated as follows: Given vertices of interest S_1 in a graph G_1 and a second graph G_2 with a portion of its vertices being also of interest, $S_2 \subset V(G_2)$, produce a rank list of the vertices of G_2 with the unknown vertices in S_2 ideally concentrating at the top of the rank list. What defines vertices as "interesting" is vague above, and this is intentional. Indeed, we allow the user broad leeway in defining what makes a vertex interesting, whether it

Algorithm 1: Block regularization pseudocode

Data: G_1, G_2

1. Compute the adjacency spectral embeddings $\hat{\mathbf{X}}_1 = \text{ASE}(G_1, d_1)$ and $\hat{\mathbf{X}}_2 = \text{ASE}(G_2, d_2)$, where G_2 is the contaminated graph. (see [40])
2. Separately cluster the rows of $\hat{\mathbf{X}}_1$ and $\hat{\mathbf{X}}_2$ via `MClust` (see [42]); Denote the cluster centers obtained from clustering $\hat{\mathbf{X}}_1$ (resp., $\hat{\mathbf{X}}_2$) via ξ_1 (resp., ξ_2);
3. Model G_1 and G_2 via Stochastic Blockmodel (SBM) random graphs [24]; estimate block probability matrices via $\hat{\mathbf{B}} = \xi_1 I_{p_1,q_1} \xi_1^T \in \mathbb{R}^{K_1 \times K_1}$ and $\hat{\mathbf{B}}^{(c)} = \xi_2 I_{p_2,q_2} \xi_2^T \in \mathbb{R}^{K_2 \times K_2}$. Theorem 1 shows the Frobenius norm consistency of these estimates in the SBM setting.
4. Find $P \in \text{argmin}_{Q \in \Pi_{K_1,K_2}} \|\hat{\mathbf{B}} - Q\hat{\mathbf{B}}^{(c)}Q^T\|_F$;
5. For P in the above argmin, denote $\xi_{2,t} := P\xi_2$. Let
$$\mathcal{I} = \{v \in V_2 \text{ s.t. } v's \text{ corresponding cluster center is selected by } P\};$$
redefine $\hat{\mathbf{X}}_2 = \text{ASE}(G_2[\mathcal{I}], d_1)$;
6. Solve $\mathbf{W} \in \text{argmin}_{O \in \mathcal{O}_{p_1,q_1}} \|\xi_1 O - P\xi_2\|_F$ and set $\hat{\mathbf{X}}_{1,a} = \hat{\mathbf{X}}_1 \mathbf{W}$;
7. Cluster the rows of $[\widehat{X}_{1,a}^T | \widehat{X}_2^T]^T$ via `MClust`, and rank the vertices according to the Mahalanobis distance to the vertices of interest in G_1.

be membership in a community of interest [16], vertices involved in illicit activity, or vertices corresponding to a particular user in a social network [36]. Note that a formal definition of the above is presented in [2, 31], we do not present the full formal definition here as it would introduce a needless notational complexity, and the informal definition suffices for our present purposes.

While the theoretical and practical impacts of adversarial noise and subsequent regularization have been widely studied in graph-valued data applications (see, for example, [5, 12, 15, 27, 51]), the development of these concepts in vertex nomination is relatively nascent and is an area of current research. In [2], the effect of adversarial data contamination was introduced and studied in the context of vertex nomination. The authors of [2] both develop a theoretical basis for understanding the action of an adversary in VN, and empirically demonstrate the expected cycle of performance degradation due to adversarial noise followed by data regularization (via the trimming method inspired by [13]) recovering much of the lost performance. The adversarial contamination model in [2] is formulated as a probabilistic mechanism acting on the network via edge/vertex deletion or addition. In [2] and the present work, the goal of the adversary is to move the distribution out of the consistency class of a given vertex nomination rule (see [31]), and thus diminish algorithmic performance.

This is done within the context of stochastic blockmodel graphs [24], and a noise model (initially proposed in [5]) is introduced in [2] in which the addition/deletion of edges and vertices in the network effectively introduces "noise" blocks into the original blockmodel distribution. A network regularization method is then introduced which seeks to trim the noise blocks via a network analogue of the classical trimmed-mean estimator [26, 43] for outlier contaminated data. While the trimming regularization of [2] is empirically demonstrated to be effective in both real and synthetic applications, it is both difficult to analyze theoretically and practically limited, as it is designed to combat a specific noise-type. Building upon this work, we propose an alternative trimming procedure (presented in Algorithm 1) that trims the graph in model space—as opposed to trimming the graph directly as in [2]. In addition to being more theoretically tractable (indeed, in [37], we prove that steps 1-4 of Algorithm 1 consistently recovers the signal blocks in the presence of the general adversarial block model presented in [2, 5]), our new approach yields superior performance as compared to the existing methodology (see Figure 1).

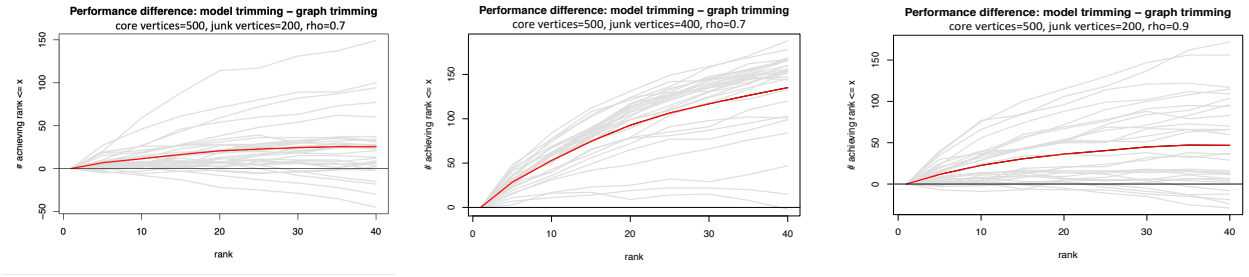

Figure 1: In each panel of the figure, we plot on the y-axis the number of vertices in G_1 (when considered as the vertex of interest) with their corresponding vertex of interest ranked in the top x in a block-contaminated SBM setting. Each panel represents the the difference in performance across the the model-based trimming method proposed herein and the graph-based trimming method of [2]. Each figure represents 30 Monte Carlo simulations (paired within each column), with performance in each simulation plotted in gray and the average over all MC plotted in red (top) or black (bottom two rows).

In Algorithm 1, we estimate the block probability matrices via $\hat{\mathbf{B}} = \xi_1 I_{p_1,q_1} \xi_1^T \in \mathbb{R}^{K_1 \times K_1}$ and $\hat{\mathbf{B}}^{(c)} = \xi_2 I_{p_2,q_2} \xi_2^T \in \mathbb{R}^{K_2 \times K_2}$. The following Theorem shows the Frobenius norm consistency of the analogous estimates obtained by the optimal Mean Squared Error-based clustering (in Step 2 of Algorithm 1) in the Stochastic Blockmodel (SBM) setting.

Theorem 1. *Let $\hat{\boldsymbol{\mu}}$ be the matrix of cluster centroids provided by the optimal Mean Squared Error-based clustering of $\hat{\mathbf{X}}$ in the uncontaminated network (with $\hat{\boldsymbol{\mu}}^{(c)}$ defined analogously for the contaminated SBM network as defined in [37]). Let $\mathbf{B}$ and $\mathbf{B}^{(c)}$ be the block probability matrices for the uncontaminated and contaminated networks, under suitable assumptions on the SBM model parameters (see Theorem 1 of [37] for specific details), we have that*

$$\|\underbrace{\hat{\boldsymbol{\mu}} I_{p_1,q_1} \hat{\boldsymbol{\mu}}^T}_{:=\hat{\mathbf{B}}} - \nu_n \mathbf{B}\|_F = O_{\mathbb{P}}\left(\frac{K_1 \log^{\alpha} n}{n}\right) \quad \|\underbrace{(\hat{\boldsymbol{\mu}}^{(c)}) I_{p_2,q_2} (\hat{\boldsymbol{\mu}}^{(c)})^T}_{:=\hat{\mathbf{B}}_c} - \nu_n \mathbf{B}^{(c)}\|_F = O_{\mathbb{P}}\left(\frac{K_2 \log^{\alpha} n}{n}\right)$$

We next use the estimates $\hat{\mathbf{B}}$ and $\hat{\mathbf{B}}^{(c)}$ to trim the contaminated vertices in G_2. We can accomplish this by solving the following graph matching type problem. For each $a, b \in \mathbb{Z} > 0$, define $\Pi_{a,b} = \{P \in \{0,1\}^{a \times b} : P\vec{1}_b = \vec{1}_a,\ \vec{1}_a^T P \leq \vec{1}_b\}$. Note that there exists a $P^* \in \Pi_{K,3K}$ such that $\|\mathbf{B} - P^* \mathbf{B}^{(c)} (P^*)^T\|_F = 0$. We then seek to show that with high probability, $P^* = \text{argmin}_{P \in \Pi_{K,3K}} \|\hat{\mathbf{B}} - P \hat{\mathbf{B}}^{(c)} P^T\|_F^2$. This would imply that P^* also recovers the correspondence between the original, non-contaminated blocks in $\hat{\mathbf{B}}$ and the uncontaminated portion of $\hat{\mathbf{B}}^{(c)}$, with the remaining blocks (which ideally capture the contamination) then trimmed by our procedure. This is formalized in the following result:

Theorem 2. *Let $\mathbf{B}$ and $\mathbf{B}^{(c)}$ be such that $\min_{P \in \Pi_{K,3K} \setminus \{P^*\}} \|\mathbf{B} - P \mathbf{B}^{(c)} P^T\|_F = \Omega(1)$. For K_1 fixed, we then have $\lim_n \mathbb{P}\left(P^* = argmin_{P \in \Pi_{K,3K}} \|\hat{\mathbf{B}} - P \hat{\mathbf{B}}^{(c)} P^T\|_F\right) = 1$.*

Although the above-described block regularization method is empirically (and theoretically) effective in the contaminated SBM setting above, in many real data settings the distribution of the noise is more nuanced or altogether unknown. Therefore, in [37] we also developed another regularization strategy to deal with broader noise models, based on a robust K-means clustering method, that can be used in the setting where the contamination is unstructured (or less structured) diffuse noise. Consider a K-block SBM model graph G_2 that we contaminate with unstructured noise (for our purposes we consider the noise to be uniformly distributed) such that, when viewing the SBM as a *Generalized Random Dot Product Graph* (GRDPG) of [40], the latent positions of our contaminated model can be written as $\mathbf{X} = [\mathbf{Y}^T | \mathbf{Z}^T]^T \in \mathbb{R}^{(n+m) \times d}$, where $\mathbf{Y} \in \mathbb{R}^{n \times d}$ are n latent positions from the uncontaminated base SBM graph and $\mathbf{Z} \in \mathbb{R}^{m \times d}$ are the i.i.d "white noise" latent positions. The contaminated graph G_2^c is then a GRDPG with latent positions $\mathbf{X}$. In order to regularize the diffuse noise out of G_2^c, we develop a robust K-means clustering algorithm for

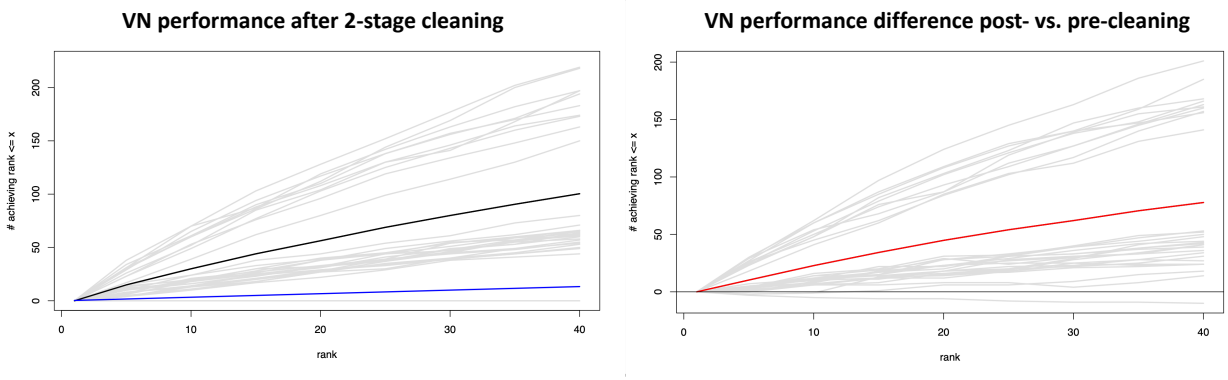

Figure 2: Vertex Nomination performance after the implementation of the two-stage regularization with $\lambda = 0.2$ in a diffuse and block contaminated SBM. We plot on the y-axis the number of vertices in G_1 (when considered as the vertex of interest) with their corresponding vertex of interest ranked in the top x. In the (L) panel, we plot the absolute performance of the 2-stage cleaning regularization procedure; in the (R) panel, we plot the difference in performance post-regularization versus without regularization (post-pre). Each grey line represents one of 30 Monte Carlo simulations, with the average performance over all MC plotted in black (L) or red (R). The blue line represents chance performance.

$\hat{\mathbf{X}} = \text{ASE}(G_2^c, d)$. Let $\mathcal{P}_2$ be the set of partitions of $[n+m]$ into two groups, and $\mathcal{C}_K$ the collection of sets of K distinct points in $\mathcal{X}_d$. We seek to solve the following optimization problem with tuning parameter $\lambda > 0$,

$$\min_{\Phi \in \mathcal{C}_K, \mathfrak{p} \in \mathcal{P}_2} \Big(\sum_{i \in \mathfrak{p}_1} \min_{\phi \in \Phi} \|\phi - \hat{X}_i\|_2 \Big) + \lambda |\{j : j \in \mathfrak{p}_2\}|. \tag{1}$$

The partition of the vertices provided by $\mathfrak{p} \in \mathcal{P}_2$ divides the vertex set of G_2^c into estimated signal vertices (i.e., those in $\mathfrak{p}_1$) and estimated noise vertices (those in $\mathfrak{p}_2$). The vertices in $\mathfrak{p}_1$ contribute the the error in Eq. (1) via the usual K-means term and are further clustered into K disjoint groups. Those vertices in $\mathfrak{p}_2$ are far from the cluster centers, and are not included in one of the K clusters. These unclustered points incur a constant cost (here λ) in Eq. (1); λ here is designed to penalize partitions that would cluster too few vertices in the noisy graph G_2 while also allowing for noise vertices to be excluded from the final clustering. The graph G_2^c is then regularized by considering the "cleaned" graph, $G_2^c[\mathfrak{p}_1]$, in which the unclustered (ideally noise) vertices have been trimmed. In addition to developing an algorithm for approximately solving Eq. 1 in [37], we also provide heuristics for choosing the penalization parameter λ. Under suitable assumptions, we also prove that the optimal solution to Eq. (1) will perfectly cluster the signal vertices and will only improperly assign a cluster label to relatively few of the noise vertices.

Heterogeneous and multimodal noise settings are very common when working with real data. The two above-described cleaning methods can be strung together seamlessly to potentially ameliorate multiple noise sources simultaneously. Consider the setting where G_2 is contaminated by both block-noise and diffuse noise. Combining the two regularization strategies outlined above yields a two-step approach in which we first clean out the unstructured noise using the robust K-means method and then use our block-noise regularization algorithm to trim the structured noise; see Figure 2 for a simulation example. The 2-stage trimming algorithm also is shown to be effective in [37] across a number of real data domains from neuroscience connectomics, social network analysis, and a political blog dataset.

3 Learning to rank with human-in-the-loop

The above vertex nomination task can be seen as a graph-based example of the more general task of ranking objects relative to a given query. This more general task has numerous important applications, such as identifying potential human traffickers [18], identifying users across social networks [36] or improving search engines [2, 7, 17]. Often, in order to rank the given objects relative to the query, a domain specific notion of similarity needs to be defined or computed, and this is not always an easy task [22, 23]. While in some

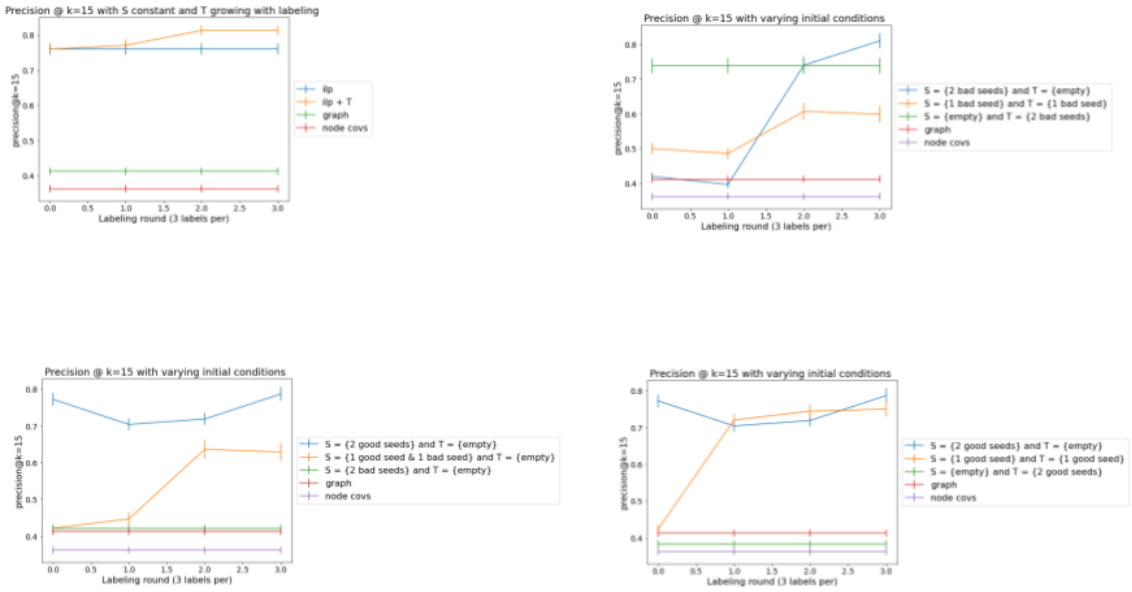

Figure 3: On the top left, we plot the performance of the ranking procedure with a fixed S set and a T set that grows with the iterations of the human-in-the-loop. Here S contains only one element (a vertex from S^*) and T is initially empty and grows with each iteration of the human-in-the-loop. The presence of the T set seems to increase algorithmic performance. We run 100 Monte Carlo iterations for each experiment.

settings a notion of similarity is provided a priori and the ranking is straightforward, in many settings the definition of similarity is poorly defined and needs to be learned using supervisory information or data [32, 39]. The latter usually requires a significant amount of data, and one possible solution for this problem is to leverage pre-existing models or representations in the similarity learning task, that have proven to be useful for similar downstream inference tasks. In settings where resources are scarce, this has proven to be successful, including transfer learning [35], domain adaptation [49], learning-to-rank [30] and continual learning [48].

In this work, we extend the integer linear programming (ILP) learning-to-rank framework proposed in [23], in which knowledge of some given dissimilarity measures, relative to a query, are leveraged to learn an approximately optimal linear combination of these dissimilarities for discovering new data points near to the query. This new dissimilarity measure is customized to the query under consideration though, and does not necessarily imply a generalization to other settings. Considering the ILP framework in the context of networks, informally the ILP proceeds as follows. Letting v^* be our query vertex, we posit an a priori unknown oracle dissimilarity Δ^*—oracle in that if $\Delta^*(v^*, v)$ is small (resp., large) then v is (resp., is not) a vertex of interest in the graph A set S of objects known to be similar (under Δ^*) to the query is assumed to be provided. Given a collection of perhaps disparate dissimilarities $\Delta_1, \cdots, \Delta_k$, the goal is to discover an "optimal" linear combination of the Δ_i's that well approximates Δ^*. This is achieved in the ILP framework of [23] by ranking the elements of S as highly as possible in the combined measure—here a linear combination of the $\Delta_1, \cdots, \Delta_k$.

We extend the above setting to one in which, in addition to the S set, we are given a set T of objects known to be dissimilar under Δ^* to the query under consideration. Our task now is to learn an approximately optimal linear combination of the dissimilarities provided which will simultaneously rank elements of S as high as possible in the ranked list and elements of T relatively lower in the same list. More specifically, let $\{v_1, ..., v_n\}$ be the vertices in our graph and without loss of generality, let $v_1 = v^*$. Let $\{\Delta_1, ..., \Delta_k\}$ be a collection of k distinct dissimilarity measures and assume we are given the dissimilarity between v_1 and

v_i, $i = 2, ..., n$ for each of these measures. Let $d_{ji} = \Delta_j(v_1, v_i)$ denote the dissimilarity between v_1 and v_i in the j-th dissimilarity, where $j = 1, ..., k$. Consider real valued decision variables/weights $\alpha_1, ..., \alpha_k$ that are constrained to be nonnegative. We impose the following normalization constraint, $\alpha_1 + ... + \alpha_k = 1$, given that the solution does not change when scaling the weights by the same positive factor. We define two integer variables x_v and y_v for each $v \in C = V \setminus \{S \cup T \cup v^*\}$ and impose the linear constraints $0 \leq x_v \leq 1$ and $0 \leq y_v \leq 1$ (as x_v and y_v are constrained to be integral, this effectively forces $x_v, y_v \in \{0, 1\}$ to maintain feasibility). These added slack variables capture the following behaviors: If $x_v = 1$, then v is ranked better than at least one element in S (under α) . If $y_v = 1$, then v is ranked worse than at least one element in T (under α). We then frame the ILP (an extension of that considered in [23]) as

$$\begin{aligned}
&\min\Big(\sum_{v \notin \{S \cup T\}} x_v + \sum_{v \notin \{S \cup T\}} y_v \Big) \qquad (2)\\
&\text{s.t. } x_v,\, y_v \in \{0, 1\}\\
&\quad \sum_i \alpha_i = 1\\
&\quad \text{for all } v_i \in S, v_\ell \in V \setminus (S \cup T), \quad \sum_{j=1}^{k} \alpha_j d_{ji} \leq \sum_{j=1}^{k} \alpha_j d_{j\ell} + M x_{v_\ell}\\
&\quad \text{for all } v_i \in T, v_\ell \in V \setminus (S \cup T), \quad \sum_{j=1}^{k} \alpha_j d_{j\ell} - M y_{v_\ell} \leq \sum_{j=1}^{k} \alpha_j d_{ji}
\end{aligned}$$

where $M := \max_{i,j} d_{ji}$. The inequality constraints are designed to choose weights (α_i's) that rank the all v with $x_v = 0$ (resp., $y_v = 0$) below all elements of S (resp., above all elements of T). If $x_v = 1$ or $y_v = 1$, the constraints are vacuously satisfied. This ILP formulation can be solved with any mixed integer program solver (e.g., Gurobi [1], SCIP [19]).

In addition to the above incorporation of T into the ILP of [23], we also introduce a human-in-the-loop (HitL) retraining step into the ILP ranking setting. At each iteration in the HitL framework, the user is provided with the top k elements of the ranked list, and returns information on whether each object is similar or dissimilar to the query at hand. Given this information, each of the top k elements in the list is then added to the S or T sets in the next iteration (depending on whether the object is judged similar / dissimilar to the query). This process is continued until certain stopping criteria are met. Both the addition of the T set and the introduction of the human-in-the-loop to the setting empirically (in both simulations and real data experiments) increase algorithmic performance and provide an improved ranking, especially in settings where there are mislabeled training vertices in the initial S and T sets. We support our claims with simulations and real data experiments (see Figure 3 for a simulation example).

We also consider and formulate an approximate ILP akin to Eq. 2 that achieves greater scalability by incorporating the affine equality constraint on the α_i's into the objective function; this is achieved here by adding an $\|\alpha\|_2^2$ term to the above minimization. Along the way, we discovered that the formulation of this newly-introduced extension is structurally similar to the classical Support Vector Machine (SVM) [4, 6, 25]. Therefore, in addition to the ILP, we consider a hybridized ILP-SVM approach to solve the problem at hand, which obtains better results while being more computationally efficient.

4 DCRDPG

In the previous two projects, the anomaly came in the form of adversarial noise which was regularized via trimming or the incorporation of a HitL. In this chapter, the anomaly is the signal that we wish to uncover, as this last project explores using graph embedding methods to discover sequential signal anomalies in network time series. Our motivation is the following biological setting. We are given temporal recordings of the dorsal pedal ganglia of Apylsia Californica slugs. In the experiment setting, the slug is given nerve stimuli at several different times, and the goal is to understand the behaviour of the neurons in the brain through their reaction to the sequential stimuli. We seek to uncover potential *learning* behaviour, where the firing rates of certain brain neurons do not return as quickly to resting state after the second or third stimuli, and

full activation of the motor program does not require as significant a change in firing rate. This may be implying that they have learned to expect/are primed for sequential stimuli, given previous stimuli.

One way to analyze the behaviour of the neurons is by looking at graph embeddings. In general terms, we can use these embeddings to cluster neurons into groups of similar activity patterns which in turn provides information about the way each neuron behaves when sequential stimuli are applied. We posit a tractable model to understand these time-series, namely the Degree Correlated Random Dot Product Model, abbreviated as the DCRDPG.

Definition 1 (d-dimensional Degree Corrected Random Dot Product Graph Model (DCRDPG))**.** *Let F be a distribution with support in a set $\mathcal{X} \subset \mathbb{R}^d$ where if $x, y \in \mathcal{X}$ then $\langle x, y\rangle \in (0, 1)$. Let the matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ have rows that are distributed i.i.d. according to F. A random graph $(G, \mathbf{X}) \sim DCRDPG(F, \nu, \vec{c})$ is an instantiation of a degree corrected random dot product graph with parameters F, $\vec{c}$ and sparsity factor ν, if, conditional on $\mathbf{X}$ and $\vec{c}$, edges in the graph are independently distributed with*

$$\mathbb{1}\{\{i, j\} \in E(G)\} \sim Bernoulli(\nu c_i c_j X_i^T X_j).$$

The upper triangular portion of the adjacency matrix A of G is then conditionally distributed via

$$P(A|\mathbf{X}, \vec{c}) = \prod_{i<j} (\nu c_i c_j X_i^T X_j)^{A_{ij}} (1 - \nu c_i c_j X_i^T X_j)^{1-A_{ij}}.$$

We use the DCRDPG to naturally encode a network time-series $(G_t)_{t=1}^m$ as follows. We sample a common latent position matrix $\mathbf{X}$ with rows i.i.d. according to F. For each $t \in [m]$, we then have

$$P(A_t|\mathbf{X}, \vec{c}_t) = \prod_{i<j} (\nu c_{t,i} c_{t,j} X_i^T X_j)^{A_{ij}} (1 - \nu c_{t,i} c_{t,j} X_i^T X_j)^{1-A_{ij}}.$$

so that we assume $\mathbf{X}$ and ν are invariant in t, but that the degree correction parameters $\vec{c}_t = (c_{t,1}, c_{t,2}, \ldots, c_{t,n})$ vary in t. In this regard, $\vec{c}^{(i)} := (c_{1,i}, c_{2,i}, ..., c_{m,i})$ is a one-dimensional time-series for each vertex v_i, where we uniformly set $c_{1,i} = 1$ for identifiability reasons. Our goal is to estimate and cluster the $\vec{c}^{(i)}$'s with the applied goal of better understanding the differences in temporal behavior across the different neurons in the motor program. Our estimation procedure begins by computing $\hat{Y}^{(t)} = ASE(G_t, d)$ (here ASE is the Adjacency Spectral Embedding of [3, 45]) and then estimating each $c_{t,i}$ via $\hat{c}_{t,i} = \|\hat{Y}_i^{(t)}\|/\|\hat{Y}_i^{(1)}\|$. In our work we show that $\hat{c}_{t,i}$ is indeed a consistent estimate of the parameter $c_{t,i}$, as seen below.

Theorem 3. *With notation as above, let m be fixed, and assume that $\vec{c}_t = \Theta(1)$ entry-wise for all t. Assume that (where the implicit dependence on n is dropped to each notation) there exists a constant c such that if $\nu n = \omega(\log^{4c} n)$, then for all i and t,*

$$|c_{t,i} - \widehat{c}_{t,i}| = O\left(\frac{\log^{\alpha} n}{n^{1/2}\nu^{1/2}}\right) \tag{3}$$

holds with probability at least $1 - 1/n^2$ for all n sufficiently large.

The data under consideration is data of neuronal activity of a sea slug brain. Specifically, we are provided with 33 min recordings of the dorsal pedal ganglia of Apylsia Californica slugs. These recordings involve 5 minutes of baseline data (which can be thought of as a "steady state" or resting state sample) and the subsequent application of four different nerve stimuli over the course of 33-mins. We cut the time (in seconds) into 36 intervals and for each of these intervals we create a matrix of the computed **sttcs** (Spike Time Tiling coefficients) for the neuron [11]. We then embed these matrices into a lower dimension to obtain the latent positions Y_i (where the dimension is chosen via an automated elbow detection in the singular value scree plot [8, 50]). We can then easily obtain the estimated parameters $\hat{c}_{t,i}$ for all i and t. We can hence deduce information about the neuron's activity by clustering the estimated time-series $\hat{\vec{c}}^{(i)}$. To this end, we can use Functional Data Clustering (FDC) to group neurons based on the 'similarity' of their trajectories over time. Here, each function, $\hat{\vec{c}}^{(i)}$, is considered a data point, and we compute a distance between pairs of functions

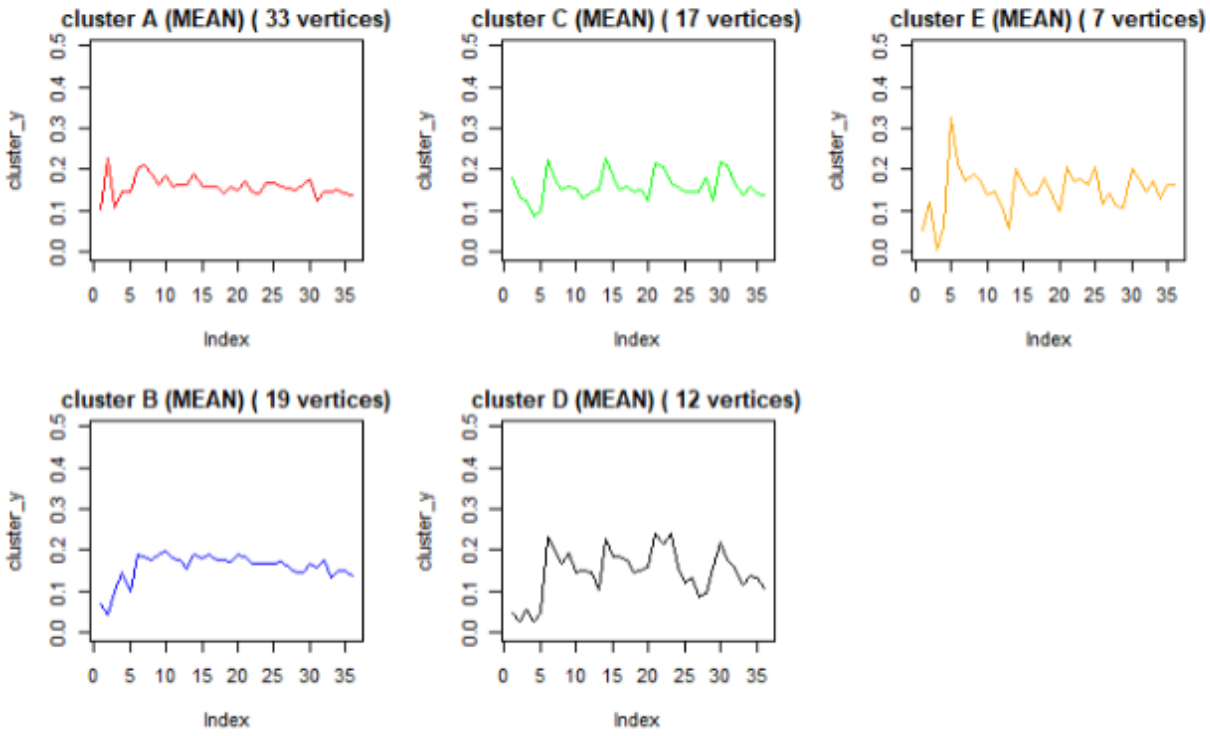

Figure 4: Mean trajectories of the FDC clustering of the Aplysia Californica data with 5 clusters. In these Figures, the stimuli were given during graph indices 5.45, 13, 20.7, and 28.36, corresponding to minutes 5, 12, 19, and 26.

and then cluster them using the R-package **kml** [20] an implementation of K-means, but for longitudinal data. Figure 4 shows the mean trajectory for each cluster of neurons, when we consider 5 clusters in our algorithm. These time series help us extract important trends of neuron activity over time, specifically the way each neuron group reacts to the sequential stimuli given, and the desensitization that occurs as the second and third stimuli are applied.

5 Discussion and Future Work

Adversarial attacks on networks are phenomena that unfortunately happen frequently and thus, trimming methods and HitL regularization methods that help mitigate the affect of the contamination are much sought after. While Stochastic Block Models and RDPG models are simple, yet rich models to work with, a natural next step would be to develop the analogous theory for more general structured noise settings. One can also consider local regularization methods (for more localized noise sources), which is another open area of research. For contamination procedures that act locally, regularization methods performed at a local level might prove to be effective in many settings. Another path to explore is lifting the above ranking problems to higher dimensions, including multilayered graphs and time series. In this case, tensors are a natural framework to work in and provide the relevant theory that will support a thorough exploration of the subject. Lastly, it is natural to consider implementing these algorithm in the case where we are given multiple sets of queries/vertices of interest and sets S and T for each query.

In our work using the DCRDPG model to analyze embeddings of neuron spike trains, our goal is to further understand the biological relevance of our methodology. To this end, we have compared our results to consensus clustering [21, 44, 46], a tool commonly used by neuroscientists to cluster and analyse the

neurons in the spike-train data under consideration. While our obtained clusters capture signal common to the consensus clusterings, our method also sheds light on neuron behaviour that the consensus clustering does not necessarily pick up. Work is currently underway with neuroscience colleagues to better understand the biological relevance of our obtained clusters especially as compared to the better understood consensus clusters. Further work includes analyzing a setting where the strength and number of stimuli applied changes in order to better understand how this changes our clustering's effectiveness.

References

[1] Tobias Achterberg. What's new in gurobi 9.0. *Webinar Talk url: https://www. gurobi. com/wp-content/uploads/2019/12/Gurobi-90-Overview-Webinar-Slides-1. pdf*, 2019.

[2] J. Agterberg, Y. Park, J. Larson, C. White, C. E. Priebe, and V. Lyzinski. Vertex nomination, consistent estimation, and adversarial modification. *Electronic Journal of Statistics*, 14(2):3230–3267, 2020.

[3] A. Athreya, D. E. Fishkind, M. Tang, C. E. Priebe, Y. Park, J. T. Vogelstein, K. Levin, V. Lyzinski, and Y. Qin. Statistical inference on random dot product graphs: a survey. *Journal of Machine Learning Research*, 18(1):8393–8484, 2017.

[4] M. Awad and R. Khanna. Support vector machines for classification. *Efficient Learning Machines*, pages 39–66, 2015.

[5] T. T. Cai and X. Li. Robust and computationally feasible community detection in the presence of arbitrary outlier nodes. *Annals of Statistics*, 43(3):1027–1059, 2015.

[6] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408:189–215, 2020.

[7] O. Chapelle and Y. Chang. Yahoo! learning to rank challenge overview. *ser. Proceedings of Ma- chine Learning Research*, 14, 2011.

[8] Sourav Chatterjee. Matrix estimation by Universal Singular Value Thresholding. *The Annals of Statistics*, 43(1):177 – 214, 2015.

[9] G. Coppersmith. Vertex nomination. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(2):144–153, 2014.

[10] G. A. Coppersmith and C. E. Priebe. Vertex nomination via content and context. *arXiv preprint arXiv:1201.4118*, 2012.

[11] Catherine S Cutts and Stephen J Eglen. Detecting pairwise correlations in spike trains: an objective comparison of methods and application to the study of retinal waves. *Journal of Neuroscience*, 34(43):14288–14303, 2014.

[12] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song. Adversarial attack on graph structured data. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1115–1124, 2018.

[13] D. Edge, J. Larson, M. Mobius, and C. White. Trimming the hairball: Edge cutting strategies for making dense graphs usable. In *IEEE International Conference on Big Data*, pages 3951–3958, 2018.

[14] K Eichler, F Li, AL Kumar, Y Park, I Andrade, C Schneider-Mizell, T Saumweber, A Huser, D Bonnery, B Gerber, et al. The complete wiring diagram of a high-order learning and memory center, the insect mushroom body. *Nature*, 548(175-182):23, 2017.

[15] N. Entezari, S. A. Al-Sayouri, A. Darvishzadeh, and E. E. Papalexakis. All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 169–177, 2020.

[16] D. E. Fishkind, V. Lyzinski, H. Pao, L. Chen, and C. E. Priebe. Vertex nomination schemes for membership prediction. *Annals of Applied Statistics*, 9(3):1510–1532, 09 2015.

[17] D. E. Fishkind, V. Lyzinski, H. Pao, L. Chen, and C. E. et al. Priebe. Letor: A benchmark collection for research on learning to rank for information retrieval. *Inf. Retr.*, 13(4):346–374, 2010.

[18] D. E. Fishkind, V. Lyzinski, H. Pao, L. Chen, and C. E. et al. Priebe. Vertex nomination schemes for membership prediction. *The Annals of Applied Statistics*, 9(3):1510–1532, 2015.

[19] Gerald Gamrath, Daniel Anderson, Ksenia Bestuzheva, Wei-Kun Chen, Leon Eifler, Maxime Gasse, Patrick Gemander, Ambros Gleixner, Leona Gottwald, Katrin Halbig, et al. The scip optimization suite 7.0. 2020.

[20] Ch. Genolini and B. Falissard. Kml: k-means for longitudinal data. *computational Statistics*, 25:317–328, 2010.

[21] A. Goder and V. Filkov. Consensus clustering algorithms: Comparison and refinement. *2008 Proceedings of the Tenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, 2008.

[22] H. S. Helm, M. Abdin, B. D. Pedigo, S. Mahajan, V. Lyzinski, Y. Park, A. Basu, P. Choudhury, C. M. White, W. Yang, and C. E. Priebe. Leveraging semantically similar queries for ranking via combining representations. *arXiv preprint https://arxiv.org/pdf/2106.12621.pdf*, 2021.

[23] H. S. Helm, A. Basu, A. Athreya, Y. Park, J. T. Vogelstein, M. Winding, M. Zlatic, A. Cardona, P. Bourke, J. Larson, C. White, and C. E. Priebe. Learning to rank via combining representations. *arXiv preprint https://arxiv.org/pdf/2005.10700.pdf*, 2020.

[24] P. W. Holland, K. B. Laskey, and S. Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

[25] R. Hu, X. Zhu, Y. Zhu, and J. Gan. Robust svm with adaptive graph learning. *https://doi.org/10.1007/s11280-019-00766-x*, 2020.

[26] P. J. Huber. *Robust statistics*. John Wiley & Sons, 2004.

[27] J. Jia, B. Wang, X. Cao, and N. Z. Gong. Certified robustness of community detection against adversarial structural perturbation via randomized smoothing. In *Proceedings of The Web Conference 2020*, pages 2718–2724, 2020.

[28] W. Jin, Y. Li, H. Xu, Y. Wang, Sh. Ji, Ch. Aggarwal, and J. Tang. Adversarial attacks and defenses on graphs: A review, a tool and empirical studies. *ACM SIGKDD Explorations Newsletter Volume 22 Issue 2 December, pp 19*, 2020.

[29] J. Li, T. Xie, L. Chen, F. Xie, X. He, and Z. Zheng. Adversarial attack on large scale graph. *arXiv:2009.03488v2*, 2021.

[30] Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

[31] V. Lyzinski, K. Levin, and C. E. Priebe. On consistent vertex nomination schemes. *Journal of Machine Learning Research*, 20(69):1–39, 2019.

[32] R. Manrique, F. Cueto-Ramirez, and O. Mariño. Comparing graph similarity measures for semantic representations of documents. *CCIS*, 885, 2018.

[33] D. Marchette, C. E. Priebe, and G. Coppersmith. Vertex nomination via attributed random dot product graphs. In *Proceedings of the 57th ISI World Statistics Congress*, volume 6, 2011.

[34] A. Mele. A structural model of segregation in social networks. *doi:10.1920/wp.cem.2010.3210*, 2013.

[35] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[36] H. G. Patsolic, Y. Park, V. Lyzinski, and C. E. Priebe. Vertex nomination via seeded graph matching. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 13(3):229–244, 2020.

[37] Sheyda Peyman, Minh Tang, and Vince Lyzinski. Adversarial contamination of networks in the setting of vertex nomination: a new trimming method. *arXiv preprint arXiv:2208.09710*, 2022.

[38] P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[39] M. Roy, S. Schmid, and G. Tredan. Modeling and measuring graph similarity: The case for centrality distance. *arXiv preprint https://arxiv.org/abs/1406.5481*, 2014.

[40] P. Rubin-Delanchy, J. Cape, M. Tang, and C. E. Priebe. A statistical interpretation of spectral embedding: the generalised random dot product graph. *Journal of the Royal Statistical Society, Series B*, 2022+.

[41] S Saha, A. K. Halder, S. S. Bandyopadhyay, P. Chatterjee, M. Nasipuri, and S Basu. Computational modeling of human-ncov protein-protein interaction network. *arXiv preprint arXiv:2005.04108v1*, 2020.

[42] L. Scrucca, M. Fop, T. B. Murphy, and A. E. Raftery. mclust 5: Clustering, classification and density estimation using gaussian finite mixture models. *The R Journal*, 8/1:289–317, 2016.

[43] S. M. Stigler. The asymptotic distribution of the trimmed mean. *Annals of Statistics*, 1(3):472–477, 1973.

[44] A. Strehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining multiple partition. *Journal of Machine Learning Research*, 2002.

[45] Daniel L Sussman, Minh Tang, Donniell E Fishkind, and Carey E Priebe. A consistent adjacency spectral embedding for stochastic blockmodel graphs. *Journal of the American Statistical Association*, 107(499):1119–1128, 2012.

[46] A. Topchy, A. K. Jain, and W. Punch. Clustering ensembles: models of consensus and weak partitions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27, 2005.

[47] A. L. Traud, P. J Mucha1, and M. A. Porter. Social structure of facebook networks. *arXiv preprint arXiv:1102.2166v1*, 2011.

[48] J. T. Vogelstein, H. S. Helm, R. D. Mehta, J. Dey, W. Levine, W. Yang, B. Tower, J. Larson, C. White, and C. E. Priebe. A general approach to progressive learning. 2020.

[49] M. Wang and W. Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312(10):135–153, 2018.

[50] M. Zhu and A. Ghodsi. Automatic dimensionality selection from the scree plot via the use of profile likelihood. *Computational Statistics & Data Analysis*, 51(2):918–930, 2006.

[51] D. Zügner, A. Akbarnejad, and S. Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856, 2018.