Intuitive Network Visualization Via the Convex Linear Visualization Method

Gabriel Augustynowicz

Honors Thesis
Department of Physics and Astronomy
Northwestern University
Spring 2023

Thesis Advisor: István Kovács

Abstract

Network visualization is a powerful tool for understanding complex data. However, current network visualization techniques have limitations in how they accurately portray the intricate relationships between entities in large, real-world networks. We present a new method for visualizing networks, the Convex Linear Visualization, that respects the relationship of linear combination between nodes. We achieve this by using a relative entropy optimization method to get positions in the embedding space for node attributes, or features, and linearly combine those into positions for the nodes themselves. This provides an aesthetic visualization where the position of a node gives direct insight into shared features. We apply the method to a real-world dataset as a demonstration.

1 Introduction and Background

Networks appear in a variety of domains, ranging from social networks to biological systems to communication networks. In each of these, we often find the network is incredibly complex, which presents a problem with comprehension of the intricate relationships between entities. The problem only compounds as the number of nodes and links grows: real-world networks routinely have million or billions of nodes, and the edges possible between these grow combinatorially. This calls attention to the need for clever network representations that both enhance intelligibility and scale well with network size.

Network embedding attempts to address this problem by mapping a network into a low-dimensional vector space. Ideally, the representation maintains the properties of the network: topological structure, node centrality, community structure, and connectivity patterns, to name a few [1]. This way, the distance between two nodes can be taken as a measure of similarity or connectedness, and so the structural characteristics of a node are encoded in its embedding vector. In the representation, extraneous information is reduced thanks to lower dimensionality and the dependence of embedding vectors. This embedding can then more easily be used for further analysis of the network, e.g. link prediction, visualization, and clustering.

Currently there are a variety of methods for developing meaningful embeddings. The simplest rely on matrix factorization of the adjacency matrix, which is often accomplished using the Singular Value Decomposition (SVD)

[9]. There are also force-directed methods, which model attractive and repulsive forces between nodes and find a least-energy configuration; the physical analogy is to the equilibrium seen in systems of springs or eletric particles [5]. Some techniques focus on altering the embedding space, such as a hyperbolic geometry [4]. Other approaches include random walks (DeepWalk, Node2Vec) and deep learning, the crux of which involves the optimization of some objective function that rewards similar embeddings for similar nodes, and differing embeddings for dissimilar nodes. Often these can take into account side information, like node attributes, to supplement the network's inherent topological information [1] [12].

Once an embedding has been determined, a natural next step is transforming it into a visualization of the embedding vectors. Visualizing networks is key for representing data in a comprehensible and aesthetic manner, with prioritizations like minimal overlaps between links and preservation of symmetry. By doing this, physicists can identity important features and patterns of interactions in the complex systems they study, aiding analysis of otherwise intractable information. Additionally, visualization is key to the development and testing of models, and for its role in efficient communication of data.

Current visualization methods, however, prove insufficient to meet all the challenges innate to complex networks. For instance, weighted networks can have negative edges, which many current methods ignore or can't properly represent. Real-world networks don't always follow the principle of triangle closure, i.e. the idea that two nodes that share a neighbor should also be neighbors; however, commonly used embeddings tend to succumb to this principle, which can weaken potential for downstream analysis like link prediction [2].

To deal with this gap in meaningful visualization of network data, we present a new approach that builds on the work of Kovács et al. in their 2015 paper [7]. There they present an information theoretic heuristic and algorithm for finding a representation based on treating individual nodes as probability distributions. We implement this method and use its output to produce a new network visualization that doesn't impose triangle closure and shows the similarity between nodes that share common attributes. We call this Convex Linear Visualization. As a demonstration of the method's advantages, we then run it on a biological dataset of fly neuron synaptic connections.

2 Results

2.1 Conditions on SVD Embedding

The singular value decomposition (SVD) of a matrix is a powerful linear algebraic tool that finds use in many applications. Its properties make it ideal for signal processing and pattern recognition, for instance, and in the realm of network science it can provide an optimal low-rank embedding. However, in these and other applications we are also interested in taking functions of these matrices, and one may potentially only apply the SVD after a complex operation has been performed on a matrix.

The question naturally arises, then, if for a matrix A and a function f, there is some simple relation between the SVD of A and the SVD of $f(A)$. In general, the answer is no. Even a simple function like $f(A) = A^{-1}$ proves this: the singular values of $f(A)$ are the inverses of the singular values of A, and the rest of the decomposition has no correspondence.

However, in some special cases, a relation can be proved. We present a result in the realm of embedding via SVD matrix factorization.

Theorem: Assume $A \in \mathbb{R}^{n\times n}$ is a real symmetric n-by-n matrix with SVD given by $A = U\Sigma V^T$. Let f be a continuous, monotonic increasing, and non-negative function with $f(0) = 0$; furthermore, f is either even or odd. If f has a Taylor series given by $p(x)$, then the best rank-k approximation to the matrix $f(A)$ is given by

$$[f(A)]_k = \begin{cases} U[f(\Sigma)]_k V^T & \text{f odd} \\ U[f(\Sigma)]_k U^T & \text{f even} \end{cases} \tag{1}$$

where $[B]_k$ denotes the best rank-k approximation to matrix B given by the truncated SVD.

This theorem gives conditions for the interchange of functions and rank-reduction in network analysis. Both steps are crucial in the analysis of networks. Low-rank representations are used to capture the most essential characteristics of a network while reducing computational and storage load, or filtering out input noise. Meanwhile, power series of matrices are used for processes such as similarity calculation between nodes, link prediction, or continuous-time evolution of a network. The commuting of these two steps can expedite more complex calculations and highlight important structural properties.

2.2 Convex Linear Visualization

We present here a new network visualization method, the Convex Cluster Visualization. The motivation for this method arises from the need to better display and understand single cell sequencing data for cells. Visualizing the complex relationships between cells at different stages of development can be challenging due to the high-dimensional and noisy nature of the data. Existing methods can distinguish cells of different types with variable success, but often struggle to meaningfully capture the relationships between cells in intermediate stages of development. Our method brings new insight by representing mixed or intermediate nodes based on their feature profiles, which could reveal previously unrecognized developmental trajectories in cells, or subtler relationships between mapped nodes in the general case.

We consider here a network given by a symmetric $n \times n$ adjacency matrix A with positive entries $a_{ij} \geq 0$. We treat each node as having certain attributes or "features," with nodes connected if they share a feature. Call the nodes of the network $\{v_1, \ldots, v_n\}$.

The intuitive idea behind our approach is to develop a method of network visualization where node positions are relevant to which features are shared between nodes. In the representation, if node C is connected to nodes A and B, then its embedding should be in between the embeddings of A and B in proportion to how similar it is to either, i.e. how many shared features they have.

To achieve this, we first make use of the EntOpt algorithm, explained in the Methods section and originally presented in [7]. Each node is considered to be a Gaussian distribution centered on some point in our embedding space $\mathbb{R}^2$. By using this approach, we apply the heuristic of treating each point of space as a feature inherent to the node; the value of each node's distribution at a point informs the strength of that feature attribute for said node.

We take $S = A^T A$ to be our similarity matrix, which counts how many times a pair of features appears together. This will in generality be weighted. Using the EntOpt algorithm on S gives us what we call the "feature positions" of the network, $\{\vec{x}_1, \ldots, \vec{x}_n\}$. We can think of this as assigning an embedding for each feature, instead of the usual paradigm of doing so initially for each node.

Using these feature positions, our method can calculate meaningful node positions that represent similarity to each other based on common features. The straightforward way to do this is as a linear combination of feature

positions. We row-normalize A with the absolute-value norm: define matrix $\widetilde{A}$ with entries $\widetilde{a_{ij}} = \frac{a_{ij}}{\sum_{i=1}^{n} a_{ik}}$. This row normalization scales the adjacency matrix, whose elements represent how strongly a node possesses a feature, such that each node's (row's) feature weights sum to 1 but the proportion of expression is the same.

From here, we define the desired embedding as given by the following linear combination: for each node v_i its position will be $\vec{n}_i = \sum_{j=1}^{n} \widetilde{a_{ij}} \vec{x}_j$. In essence, a node's position is given by a weighted sum of the positions of each of its features, with weights in proportion to the expression.

We note that this method has some very desirable characteristics:

Convexness

Suppose one row of the adjacency matrix A is a linear combination of two others: we have, in terms of row vectors, $\vec{a_i} = \alpha\vec{a}_j + \beta\vec{a}_k$ for $\alpha, \beta > 0$. In network terms, we interpret this to say that node i expresses all the features of nodes j and k in proportion. Under the Convex Linear Visualization, the calculation of node positions comes from a normalized linear combination of feature positions; due to the choice of norm, we will find that the node position $\vec{n}_i$ will be on the line induced by node positions $\vec{n}_j$ and $\vec{n}_k$. Moreover, this mapping is sensitive to the proportions α and β, as well as the total feature strength of row $\vec{a}_j$ versus $\vec{a}_k$:

$$\vec{n}_i = \frac{\alpha\|\vec{a}_j\|}{\|\alpha\vec{a}_j + \beta\vec{a}_k\|}\vec{n}_j + \frac{\beta\|\vec{a}_k\|}{\alpha\vec{a}_j + \beta\vec{a}_k}\vec{n}_k \tag{2}$$

For $\alpha \in (0, 1)$ and $\beta = 1 - \alpha$, this will be on the line directly between $\vec{n}_j$ and $\vec{n}_k$, which is only possible by our choice of norm and the restriction that the entries of A be positive. This convex nature is a desired property in our visualization: it makes sense that if a node is a direct mix of two others, it should appear between them.

The statement extends when $\vec{a}_i$ is a linear combination of more than two rows, so long as the combination coefficients sum to 1: the node position $\vec{n}_i$ maps into the convex subspace between the corresponding other node positions.

Breaks TCP

The idea behind the Triangle Closure Principle (TCP), also referred to as network transitivity, is the interconnectedness of adjacent nodes. In many

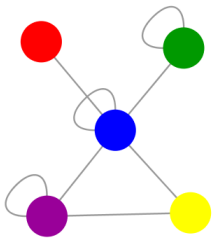
complex networks, we find that there is a high propensity of triangle structures; that is, if two nodes share a neighbor, they are likely to be neighbors themselves [2]. Many embedding methods can capture this property; however, just as important is the ability to break TCP and consider models where nodes sharing a neighbor are unlikely to be connected (e.g. protein-protein interaction networks, where a "lock-and-key" understanding of protein interactions hints that shared neighbors have similar structure and thus won't interact).

The EntOpt algorithm is unlikely to break TCP alone. However, the linear combination of its output feature positions in the Convex Linear Visualization *can* break TCP. Wherever a triad of nodes in the original network satisfies triangle closure, the positions of the three nodes under our technique will be strongly interdependent due to the linear combination taken, and so they are likely to have similar embeddings. However, if a triad of nodes doesn't satisfy TCP, i.e. we have two nodes A, B with a common neighbor C that aren't adjacent to each other, then the positions of A and B will be independently determined by their features, and a dissimilar embedding is possible.

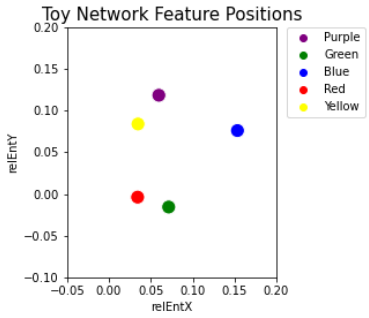
Rectangular Matrices

Although the method is presented above for symmetric $n \times n$ matrix A, this is actually a stronger condition than necessary. A can be a rectangular $n \times m$ matrix with positive entries $a_{ij} \geq 0$. The similarity matrix $S = A^T A$ will then be a symmetric square $m \times m$ matrix, which is a valid input for EntOpt. The output will give m feature positions, which can then still be linearly combined using the m normalized entries of each row of A to get n node positions.

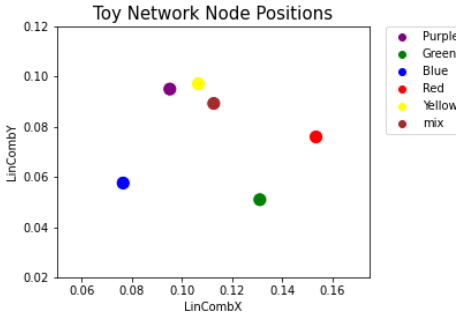
This interpretation is particularly helpful where we have a distinct set of features we can attribute to each node. For instance, in a biological application we may have a matrix A showing cell-gene correspondence. The rows represent each cell and the columns each gene; a nonzero entry occurs in the case that a cell presents a gene. We would first use the gene-gene similarity matrix given by $A^T A$ to get positions for each gene, and then take a linear combination based on gene expression to get a visualization for the cells.



(a)



(b)



(c)

Figure 1: (a) A toy feature network for testing our method. This network represents the connections possible between nodes of different features; e.g. purple nodes will connect to other purple nodes, yellow nodes, and blue nodes. (b) Embedding of a 30-node network created by using the network in (a) as part of the Stochasitc Block Model detailed in Methods. Although only five distinct nodes appear, this is a result of nodes of the same color feature being mapped to the same point. This embedding gives the feature positions from applying EntOpt on a similarity matrix. (c) Embedding of the Convex Linear Visualization, representing the node positions. The brown node has mixed features: it was made as a linear combination of a red node and purple node, with coefficients 0.3 and 0.7 respectively. As a consequence, it falls directly on the line between the red and purple clusters, demonstrating convexness. Additionally, although both red and purple nodes connect to blue nodes in our toy network, they themselves are not connected, and so have dissimilar node embeddings under our visualization, which demonstrates the deviation from TCP.

2.3 Application to Datasets

2.3.1 Fly Synaptic Network

We demonstrate the method on a biological dataset comprising a fly neuron connectome. This is compared to the visualizations that the t-SNE algorithm would produce.

The Fly Cell Atlas is presented by [10] as a high-quality cellular sequence of fruit fly *Drosophila melanogaster* at single-cell resolution. The filtered version of the dataset used here is comprised of 1703 nodes and 269968 edges, the former corresponding only to neurons and the latter to synaptic connections between them. The neurons are characterized by their type as Kenyon cells, which we use as the feature classification in our embedding.

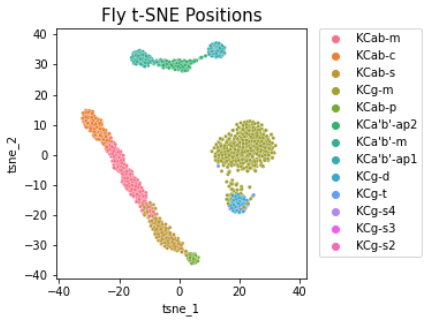
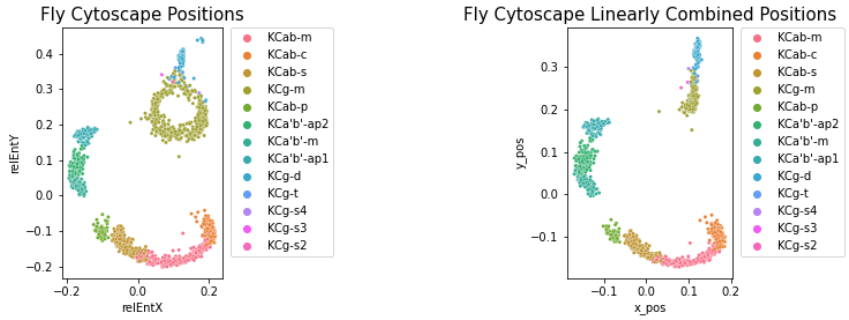


Figure 2: t-SNE embedding of the adjacency matrix of the fly synaptic data. This embedding represent node positions.

Figure 2 is the result of the popular t-SNE (t-distributed Stochastic Neighbor Embedding) algorithm, which is a statistical layout method for visualizing high-dimensional data [11]. The mathematics behind t-SNE is similar to EntOpt as described in the Methods section, as it also minimizes the Kullback-Leibler divergence of a low-dimension embedding; however, the distributions and distance measures it uses differ.

Figure 3 is the result of our Convex Linear Visualization. We display both the feature positions and the node positions as a demonstration of how this method compares to application of only EntOpt. All three methodologies (t-SNE, EntOpt, and Convex Linear Visualization) succeed in clustering together neurons with similar feature, and even tend to place the same sets of features closer together (e.g. the pairing of KCg-m and KCg-d). However,



(a) (b)

Figure 3: (a) Embedding of feature positions achieved by applying EntOpt on the fly similarity matrix $S = A^T A$. For the embedding, we choose to ignore self-similarity, i.e. the diagonal entries of S. (b) The full result of our Convex Linear Visualization, which is the weighted linear combination of the result seen in (a). This represents the node positions.

the t-SNE visualization is sensitive to the parameters used for input: namely, different choices of perplexity or early exaggeration, while retaining much of the correct clustering, will assign wildly different positions. Thus t-SNE may be suitable for identifying clusters, but not for drawing conclusions about the relationship between nodes with mixed feature profiles. Our method, conversely, has no input parameters and will always return the same relationship between nodes. Additionally, the Convex Cluster Visualization improves on both t-SNE and EntOpt in this case by narrowing the spread of each cluster, highlighting the localization inherent to each feature under this paradigm.

3 Methodology

3.1 SVD Embedding Theorems

In this section we present a short proof of the theorem presented in section 2.1.

Any real $n \times m$ matrix X admits an SVD of the form $X = U\Sigma V^T$, where U and V are real orthogonal matrices and Σ is a diagonal matrix whose entries σ_i are called the singular values. The singular values are non-negative, and the number of which are non-zero is equal to the rank of X. The SVD is

generally not unique, but if we impose the condition that the singular values appear in decreasing order, then Σ is uniquely determined [9].

The SVD is useful for giving low-rank approximations to matrices. Given a matrix X, denote its best rank-k approximation by $[X]_k$. The quality of an approximation is measured by minimizing the Frobenius norm of their difference, $\|X - [X]_k\| = \sqrt{\sum_{i,j}(x_{ij} - [x]_{ij})^2}$. By the well-known Eckart-Young theorem, the best rank-k approximation is given by the truncated SVD of X:

$$[X]_k = U[\Sigma]_k V^T \tag{3}$$

where $[\Sigma]_k$ contains only the k largest singular values of X [9].

Proof.

Consider the case that f is an odd function. Then its Taylor series $p(x)$ will also be odd.

$\implies p(x) = c_1 x + c_3 x^3 + \dots$

We utilize that fact that if $A = U\Sigma V^T$, then $A^n = U\Sigma^n V^T$ for n odd. This is true by writing $A^n = AA^TA \dots A^TA$ (recall that A is symmetric) and simplifying the products of orthogonal matrices.

$f(A) \equiv p(A) \equiv c_1 A + c_3 A^3 + \dots$
$= c_1 U\Sigma V^T + c_3 U\Sigma^3 V^T + \dots$
$= U(c_1\Sigma + c_3\Sigma + \dots)V^T$

We know Σ is diagonal; suppose it takes the form $\begin{pmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \dots & \dots & & \dots \\ 0 & 0 & \dots & \sigma_r \end{pmatrix}$

Then the middle portion $c_1\Sigma + c_3\Sigma + \dots$ is calculated nicely as

$$\begin{pmatrix} c_1\sigma_1 + c_3\sigma_1^3 + \dots & 0 & \dots & 0 \\ 0 & c_1\sigma_2 + c_3\sigma_2^3 + \dots & \dots & 0 \\ \dots & \dots & & \dots \\ 0 & 0 & \dots & c_1\sigma_r + c_3\sigma_r^3 + \dots \end{pmatrix}$$

Therefore we have $f(A) = U \begin{pmatrix} f(\sigma_1) & 0 & \dots \\ 0 & f(\sigma_2) & \dots \\ \dots & & \end{pmatrix} V^T = Uf(\Sigma)V^T$.

By the fact that f is non-negative and monotonic, we have $f(\sigma_i) \geq 0$ and $f(\sigma_i) \geq f(\sigma_{i+1})$. These properties mean that $Uf(\Sigma)V^T$ is a valid SVD of $f(A)$.

Finally we apply the Eckart-Young Theorem to state that the best rank-k approximation of $f(A)$ will be $U[f(\Sigma)]_k V^T = Uf([\Sigma]_k)V^T$.

The evens case proceeds almost identically, except that $A^n = U\Sigma^n U^T$ for n even. □

3.2 Feature Embedding Via EntOpt Algorithm

3.2.1 EntOpt

In this section we describe the Relative Entropy Optimization (EntOpt) algorithm developed in [7], which we use for finding an optimal representation of the network features before applying our Convex Linear Visualization.

The EntOpt method attempts to find a representation which is hardest to distinguish from the original input matrix A (the adjacency matrix or a similarity matrix) in terms of relative entropy, also known as Kullback–Leibler divergence. This is an information theoretic measure of the extra description length of a representation B when compared to the original input matrix [8]. The less distinguishable they are, the lower the relative entropy. The value is given as follows:

$$D(A\|B) = \sum_{i,j} a_{ij} \ln \frac{a_{ij} b_{**}}{b_{ij} a_{**}} \tag{4}$$

where $a_{**} = \sum_{i,j} a_{ij}$, $b_{**} = \sum_{i,j} b_{ij}$ are the sums of all matrix elements, used to properly normalize the entropy and subsequent probability distributions.

To get the values of the representation matrix B, we treat each node as a probability distribution over a d-dimensional ambient space, which for the purposes of visualization will be $\mathbb{R}^2$. A natural choice of distribution is the Gaussian

$$\rho_i(\vec{x}) = \frac{h}{(2\pi)^{d/2}} \exp(-\frac{(\vec{x_0} - \vec{x}) \cdot (\vec{x_0} - \vec{x})}{2\sigma^2}) \tag{5}$$

where the parameters $\vec{x_0}, \sigma, h$ correspond to the mean, width, and a normalization factor, respectively. The entries b_{ij} correspond to the overlap of the Gaussians; for $d = 2$, this is given by

$$b_{ij} = \frac{h_i h_j}{2\pi(\sigma_i^2 + \sigma_j^2) \exp(-\frac{(x_i - x_j)^2 + (y_i - y_j)^2}{2(\sigma_i^2 + \sigma_j^2)})} \tag{6}$$

3.2.2 Newton-Raphson

For the actual procedure of optimizing the distribution, we minimize the entropy between the input A and the representation B as described above. This is done via the multidimensional Newton-Raphson algorithm [7]. In analogy to the familiar single-variable Newton's method, the process goes as follows:

1. Identify the node k whose gradient vector J has the largest magnitude $\|J\|^2 = (\frac{\partial D}{\partial x_k})^2 + (\frac{\partial D}{\partial y_k})^2$.

2. Calculate the second derivative matrix $\mathcal{F}$ with entries $\frac{\partial^2 D}{\partial x_k^2}, \frac{\partial^2 D}{\partial y_k^2}, \frac{\partial^2 D}{\partial x_k \partial y_k}$.

3. Update the position of node k by $-\mathcal{F}^{-1}J$.

4. Iterate.

We intend for this to run until a global minimum of the relative entropy D is reached. In the case that a single update to the position of node k as prescribed would actually increase D, that step is replaced by gradient descent with a sufficiently small step size that D decreases.

To initialize the Newton-Raphson algorithm, one can use random values within a reasonable range for each of the values defining the probability distributions: node position, width, and normalization. Then an alternating optimization of each of the three would eventually converge to ideal values. We have outlined the procedure for minimizing D with respect to position, but analogous steps work for finding best σ and h. In practice, however, the position optimization is most important. Assuming that the initial positions chosen are close enough that their distributions are likely to overlap, we can fix reasonable values for width and normalization. We found that for node positions within the unit square $[0, 1] \times [0, 1]$, values of $\sigma = 0.1$ and $h \sim 10^{-4}$ sufficed. Furthermore, the results for EntOpt can be improved and expedited by substituting random initial positions for those given by another visualization method, preferably a force-directed layout.

3.2.3 Stochastic Block Model

For the purposes of testing both our implementation in Python of the EntOpt algorithm and the plausibility of our Convex Linear Visualization, we created toy networks by a Stochastic Block Model (SBM) [3]. This is a probabilistic

generative model for networks, which finds wide use for community detection and network clustering.

In the SBM, the set of nodes is partitioned into communities, and edges are assigned based on community membership. The probability of an edge between two nodes is determined by which communities each belongs to; there is generative control over how likely two groups of nodes are to be connected.

To generate our toy networks for testing, we let our communities represent node features, and assign each node a feature as a method of partition. This can be represented as a matrix X whose rows represent nodes and each column a feature; each row of X will have a single nonzero entry corresponding to the selected feature. We also choose a structure of community connections. Treating the communities as their own graph, we can create a weighted adjacency matrix O whose entries o_{ij} represent the desired probability that a node in community i and a node in community j will have an edge between them [6].

The matrix product $\widetilde{A} = XOX^T$ gives a matrix whose entries represent the probability that an edge should exist between any pair of nodes. Applying said probability gives an adjacency matrix A.

The SBM provides a probabilistic framework for hypothesis testing, model selection, and uncertainty quantification, which was necessary to get numerous nontrivial examples to test the Convex Linear Visualization. In particular, this model was advantageous for its control over cluster generation. The partition into communities possessing only a single feature meant that the feature positions derived from EntOpt translated very naturally when our visualization method was applied; additionally the connectedness between communities could be leveraged to get clear examples of how linear combinations of nodes map in a convex manner.

4 Discussion and Conclusion

We have introduced a simple solution to the problem of network visualization based on the principle of assigning an embedding position from a linear combination of feature embeddings. Our framework extends the algorithm of relative entropy optimization [7] with a post-processing step which adds further meaning to the derived positions. We input a similarity matrix of the given network and interpret the output of EntOpt as an embedding of fea-

tures. Subsequently, a row-normalized version of the adjacency matrix gives the coefficients we use for linear combination of the set of feature positions, resulting in a node embedding for each element of the network. We call this algorithm the Convex Linear Visualization.

This method is advantageous for its properties and broad applicability. As shown, node positions are influenced by the linear dependence of adjacency matrix rows. If a row is a linear combination of other rows in the matrix, that node's embedding can be determined as a linear combination of the embeddings of the corresponding other nodes; furthermore, if this linear combination is a proper mix, in the sense of combination coefficients summing to 1, then the embedding is convex, which is a very nice visual property when displaying data or trying to infer community patterns. By using EntOpt for feature positions, Convex Linear Visualization is able to mitigate the triangle closure principle for networks with more complex structure. The theory behind our method is very simple and generalizable. It depends on no arbitrary parameters, unlike commonly used force-directed methods or the t-SNE algorithm; all the information for linear combination comes from the values in the adjacency matrix itself, and our initial choice of entropy function and row normalization determines everything. The method is detailed for square symmetric adjacency matrices, but can be used for rectangular inputs without alteration or loss of meaning.

The method still admits some potential for improvement and further testing to build upon the foundation detailed here. We have yet to test whether the Kullback-Leibler divergence used for EntOpt can be improved upon. Instead of using that entropy, a heuristic making use of the Frobenius norm $L = \|A - B\|_{\mathcal{F}}^2 = \sum_{i,j}(a_{ij} - b_{ij})^2$, also referred to as squared error between the input matrix A and representation B, may offer some computational advantages. This function has simpler partial derivatives when using the Newton-Raphson method to solve for a minimum, and doesn't need global normalization in the same way relative entropy calculation does.

$$\frac{\partial L}{\partial x_k} = 4\sum_j \frac{x_k - x_j}{\sigma_k^2 + \sigma_j^2} b_{kj}(a_{kj} - b_{kj}) \tag{7}$$

$$\frac{\partial^2 L}{\partial x_k^2} = 4\sum_j [\frac{1}{\sigma_k^2 + \sigma_j^2} b_{kj}(a_{kj} - b_{kj}) + (\frac{x_k - x_j}{\sigma_k^2 + \sigma_j^2})^2 b_{kj}(2b_{kj} - a_{kj})] \tag{8}$$

$$\frac{\partial^2 L}{\partial x_k \partial y_k} = 4\sum_j \frac{(x_k - x_j)(y_k - y_j)}{\sigma_k^2 + \sigma_j^2} b_{kj}(2b_{kj} - a_{kj}) \tag{9}$$

Equations 7 and 8 hold similarly for differentiation by y_k.

Furthermore, such minimization problems subject to the constraint of $\text{rank}(B) \leq r$ have analytic solutions given by the singular value decomposition, which offers a natural comparison for our fit and may help expedite calculations further. Another untested prescription is the form of the similarity matrix used to calculate feature positions for our Convex Linear Visualization. The current model uses $S = A^T A$ as an intuitive base, but variations like $S = \sqrt{A^T A}$ are likewise meaningful, and not so difficult to calculate — again using the SVD. These applications of the SVD may benefit from use of the included theorem in rank-reduction of matrix functions.

5 Acknowledgements

Portions of this work were supported via a Baker Faculty Grant issued by Northwestern University's Weinberg College of Arts and Sciences. The author wishes to thank the Kovács lab group, and in particular their thesis advisor István Kovács for constant support, motivation, and patience.

6 References

References

[1] P. Cui, X. Wang, J. Pei and W. Zhu, "A Survey on Network Embedding," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833-852, 1 May 2019, doi: 10.1109/TKDE.2018.2849727.

[2] Hasan Mohammad Al and Dave Vachik S.. 2018. Triangle counting in large networks: A review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8, 2 (2018), e1226.

[3] Holland Paul W., Laskey Kathryn Blackmond, Leinhardt Samuel, "Stochastic blockmodels: First steps," *Social Networks*, Volume 5, Issue 2, 1983, Pages 109-137, ISSN 0378-8733, https://doi.org/10.1016/0378-8733(83)90021-7.

[4] Kitsak M., Voitalov I., Krioukov D., "Link prediction with hyperbolic geometry," *Phys. Rev. Res.*, 2 (2020), Article 043113

[5] Kobourov, Stephen G, "Spring Embedders and Force Directed Graph Drawing Algorithms," *arXiv*, 2012; 1201.3011

[6] Kovács, I. A., Barabási, D. L., & Barabási, A.-L. "Uncovering the genetic blueprint of the *C. elegans* nervous system. *Proceedings of the National Academy of Sciences*, 117(52), 33570–33577. https://doi.org/10.1073/pnas.2009093117

[7] Kovács, I. A. *et al.* A unified data representation theory for network visualization, ordering and coarse-graining. *Sci. Rep.* **5**, 13786; doi: 10.1038/srep13786 (2015)

[8] S. Kullback, R. A. Leibler "On Information and Sufficiency," The Annals of Mathematical Statistics, Ann. Math. Statist. 22(1), 79-86, (March, 1951)

[9] N. Kishore Kumar & J. Schneider (2016): Literature survey on low rank approximation of matrices, Linear and Multilinear Algebra, DOI: 10.1080/03081087.2016.1267104

[10] Li, Hongjie *et al.*, Fly Cell Atlas: A single-nucleus transcriptomic atlas of the adult fruit fly. *Science* **375**, eabk2432(2022). DOI:10.1126/science.abk2432

[11] van der Maaten, L.J.P.; Hinton, G.E. (Nov 2008). "Visualizing Data Using t-SNE." *Journal of Machine Learning Research.* **9**: 2579–2605.

[12] Z. Zhang, P. Cui and W. Zhu, "Deep Learning on Graphs: A Survey," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 249-270, 1 Jan. 2022, doi: 10.1109/TKDE.2020.2981333.