Pseudo-Euclidean Attract-Repel Embeddings for Undirected Graphs

Alexander Peysakhovich [1] **Anna Klimovskaia Susmelj** [2] **Leon Bottou** [1]

Abstract

Dot product embeddings take a graph and construct vectors for nodes such that dot products between two vectors give the strength of the edge. Dot products make a strong transitivity assumption, however, many important forces generating graphs in the real world lead to non-transitive relationships. We remove the transitivity assumption by embedding nodes into a pseudo-Euclidean space - giving each node an attract and a repel vector. The inner product between two nodes is defined by taking the dot product in attract vectors and subtracting the dot product in repel vectors. Pseudo-Euclidean embeddings can compress networks efficiently, allow for multiple notions of nearest neighbors each with their own interpretation, and can be 'slotted' into existing models such as exponential family embeddings or graph neural networks for better link prediction.

1. Introduction

Network analysis is ubiquitous across many disciplines ranging from the natural (Jeong et al., 2001; Barabasi & Oltvai, 2004) to the social sciences (Granovetter, 1985; Easley et al., 2010; Jackson, 2010). In general, graphs are high dimensional objects and can be difficult to work with. Finding easy representations is thus an important problem in applied machine learning. In the symmetric (or undirected) case a workhorse method are node embedding models (Lovász & Vesztergombi, 1999; Ng et al., 2002; Perozzi et al., 2014; Tang et al., 2015; Grover & Leskovec, 2016; Athreya et al., 2017; Lerer et al., 2019). In these models each node in a network is associated with an embedding (a.k.a latent vector) and dot products between vectors reflect the strength of an edge between two nodes.

We consider situations where a network exhibits a lack of 'transitivity' - A is strongly connected to B, B is strongly connected to C, but A is *not* connected to C. Such unclosed triangles are sometimes called 'forbidden triads' (Granovetter, 1973). Despite being forbidden, forces such as heterophily or role similarity in real networks give rise to such triads. A major weakness of dot product embeddings is their inability to easily represent networks that contain many such examples (Seshadhri et al., 2020).

Our key contribution is to consider embedding nodes into a pseudo-Euclidean space. Nodes still receive real vector embeddings, however, there is a modified 'inner product' where the latent vectors are split into two parts: one on which nodes attract (similar are more likely to connect) and one on which they repel (similar are less likely to connect). The total strength of an edge is modeled as the dot product of the attract sub-vector minus the dot product of the repel subvectors. We refer to this as an attract-repel (AR) embedding. We discuss multiplicity of solutions to an AR decomposition and give a give a method for constructing 'minimal' AR embeddings from an adjacency matrix using a combination of convex optimization and eigendecomposition.

We show that AR embeddings have much better representation capability in real-world graphs than Euclidean ones. We then show that AR embeddings can be used to understand the structure of graphs. In social networks, the relative contribution of A vs R components can be used measure heterophily both at the graph and node level. This measurement is fully latent and does not use any label information.

Pseudo-Euclidean space admit multiple notions of "nearest neighbors." Different notions each have interpretable properties in different types of networks. In co-occurrence graphs neighbors in R space map well onto the notion of 'substitutability'. In biological co-activation networks different notions of nearest neighbors appear to map well onto the notions of activation and inhibition.

Finally, we focus on the task of link prediction. The AR decomposition can be slotted into the loss function of any model which outputs node embedding vectors. We show that this can lead to increases in model performance for exponential family embeddings (Rudolph et al., 2016) as well as graph convolutional networks (Kipf & Welling, 2016) in intransitive graphs.

There is recent interest in using hyperbolic geometry to perform graph embeddings as many graphs of interest have

*Equal contribution [1]Meta AI Research [2]ETH Zurich. Correspondence to: Alexander Peysakhovich <alex.peys (at) gmail.com>.

Pre-print

hierarchical structure and Euclidean embeddings have a hard time representing hierarchies (Nickel & Kiela, 2017; 2018; Liu et al., 2019). We show that hyperbolic space is a manifold in pseudo-Euclidean space and thus AR embeddings are also able to represent hierarchies.

2. Related Work

2.1. VV' vs. VU' Factorizations

A large literature in graph embeddings considers factorizing the adjacency matrix as $M \sim VV'$ (Athreya et al., 2017) or as or as $M \sim f(VV')$ for some choices of f (Hoff et al., 2002; Rudolph et al., 2016). Other methods work indirectly on the adjacency matrix by constructing the embeddings from co-occurrences in random walks (Perozzi et al., 2014; Grover & Leskovec, 2016; Tang et al., 2015).

Recently Seshadhri et al. (2020) show that Euclidean dot product models cannot reproduce the distribution of triangles and degrees in real world social networks, no matter what algorithm is used to construct the embeddings. Chanpuriya et al. (2020) respond and argue for factorizing adjacency matrices as $M \sim VU'$. Our results show that the $M \sim VU'$ formalization is 'too general' for undirected graphs since symmetry implies $u_i v_j = u_j v_i$, so VU' can be written in a pseudo-Euclidean form $M \sim AA' - RR'$.

2.2. Non-Metric Visualization

Van der Maaten & Hinton (2012) considers extending t-SNE (Van der Maaten & Hinton, 2008) to intransitive similarity matrices by embedding objects in multiple t-SNE maps simultaneously - i.e. in multiple vector spaces each of dimension 2 (because of the interest in visualization). Constructing visualization techniques that take advantage of the unique geometry of pseudo-Euclidean space is an extremely interesting future direction.

2.3. The Eigenmodel

The 'Eigenmodel' is the closest work to our own (Hoff, 2007). It decomposes the adjacency matrix into a learned VDV' where D is a diagonal matrix. Our work builds upon this in several ways. First, we formalize much of the intuition in that paper. Second, we give a method for guaranteed computation of the 'simplest' AR decomposition. Third, we study properties of the model beyond just better out of sample fit (e.g. interpretation of 'neighbors').

2.4. Directed Graph Embeddings

Our work intersects a large literature on *directed* graphs. The ComplEx approach approximates knowledge graphs as $A \sim EVE'$ where V is the diagonal matrix of eigenvalues which can take complex values (Trouillon et al., 2016). When the matrix is symmetric V is guaranteed real so it is very close to the Eigenmodel above. Sim et al. (2021) considers the intransitivity problem in undirected graphs and performs a pseudo-Riemannian embedding to deal with it. The symmetry of undirected graphs gives us the additional structure that allows us to use off-the-shelf optimization techniques as well as good interpretability properties.

3. Dot Product Embeddings

We consider the general problem of embedding a weighted undirected graph $G = (N, E)$. Nodes N are generically indexed by i, j, edges e_{ij}, with $e_{ij} = e_{ji}$. We ignore self-edges in the graph, so e_{ii} is not defined.

We consider embeddings into $\mathbb{R}^D$ endowed with the dot product $x \cdot y = \sum_{j=1}^{d} x_j y_j$. We refer to a set of vectors, one for each node, $\mathcal{V} = \{v_1, \ldots, v_N\} \subset \mathbb{R}^D$ with the dot product as an Euclidean embedding of the graph. Let say that D is the dimension of the embedding.

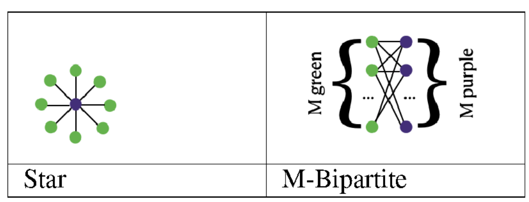
Definition 3.1. We say that a Euclidean embedding $\mathcal{V}$ **represents the graph** if for all $i \neq j$ we have

$$v_i \cdot v_j = e_{ij}.$$

Our interest will be the dimension of the embedding $\mathcal{V}$.

Proposition 3.2. *Let G be an arbitrary graph. There exists an infinitely sized family of dot product embeddings $\{\mathcal{V}\}$ that represents G*

We leave the proof to the Appendix. However, while the Proposition shows that Euclidean embeddings always exist, we will now see that this embedding may not be low dimensional even when the underlying graph is 'simple'.



Consider the graphs above with edge weights $e_{ij} \in \{0, 1\}$ with colors only added for visualization purposes. For the dot product to represent any star graph any two green nodes must have $v_i \cdot v_j = 0$, but this means they must be orthogonal. Thus, in the star graphs, we need at least as many dimensions as peripheral nodes. The M-bipartite graph is another example: this graph has a very simple structure but it is maximally intransitive (if i is linked to j, it is never linked to any neighbor of j). We will now formally show that any embedding of this graph does not really compress it. We will see later that both of these graphs have 2 dimensional pseudo-Euclidean embeddings for any choice of M.

Proposition 3.3. *If $\mathcal{V}$ is a Euclidean embedding that represents the M-bipartite graph then it has dimension* $\geq 2M-1$.

Again, we relegate the proof to the Appendix.

4. Pseudo-Euclidean Embeddings and Attract-Repel

Let us consider the counterexample above more. In the M-bipartite graph all green nodes have the same connectivity pattern, so for maximum compression we would like to give them the same representation, however, such a construction means they will have high dot product with each other. So, such an efficient encoding is impossible in the dot product model. We will now work with pseudo-Euclidean space where vectors can have inner product 0 with themselves and where the triangle inequality does not apply, thus allowing for efficient coding of the M-bipartite and other intransitive graphs.

Given $\mathbb{R}^M$ we endow it with $\cdot^k$ with $k > 0$ which is a symmetric bilinear form (which is not technically an inner product but we will refer to as one) defined as

$$v_i \cdot^k v_j = \sum_{p=1}^{M-k} v_{ip} v_{jp} - \sum_{q=M-k+1}^{M} v_{iq} v_{jq}.$$

Setting $k = 0$ we have standard Euclidean space. The special case of $M = 4$ and $k = 1$ is known as Minkowski spacetime (Naber, 2012). We consider the general (M, k) case here.

For simplicity of notation, instead of considering a single vector v_i per node, we split the vectors into a_i and r_i so that we can write the score as $a_i \cdot a_j - r_i \cdot r_j$ where $\cdot$ is the standard dot product. We refer to this as an attract-repel (AR) embedding. We can still define an embedding as representing a graph:

Definition 4.1. We say that **a AR embedding represents** G if for any two i, j with $i \neq j$ we have that

$$e_{ij} = a_i \cdot a_j - r_i \cdot r_j.$$

Both the star and M-bipartite graphs have high dimensional Euclidean-only representations but a simple AR decomposition where $a_i = 1$ for all i and $r_i = 1$ if i is green and $r_i = -1$ if i is purple. We know from the prior section that a Euclidean (and thus an AR embedding with R empty) always exists so the AR problem is over-parametrized in a non-trivial way.

4.1. Minimal Pseudo-Euclidean Embeddings

We now discuss how 'minimal' AR embeddings can be found. Let $\mathcal{AR}$ be the set of all AR embeddings representing a graph G and let A and R be the stacked embedding vectors for each node. We will look for the solution with the smallest Frobenius norm, a solution to $\min_{(A,R)\in\mathcal{AR}} ||A||_F^2 + ||R||_F^2$. We first show that this solution can be found using convex optimization and eigendecomposition which gives better guarantees than a local search.

Start with a graph G and consider the adjacency matrix of G. Recall that we do not look at self-edges in our graph, therefore any choice of diagonal for the matrix makes it a valid adjacency matrix for all $i \neq j$. Let D be an M-dimensional vector and denote by M_D as the matrix that is the adjacency matrix for G on the off-diagonal and has arbitrary diagonal D.

We now show that finding the simplest AR decomposition is strongly related to finding an appropriate choice for D. In particular, we choose D to minimize the nuclear norm of M_D (Candès & Recht, 2009). More formally:

Proposition 4.2. *Let A, R be a solution to $\min_{(A,R)\in\mathcal{AR}} ||A||_F^2 + ||R||_F^2$. Let M_D be the solution to $\min_D ||M_D||_*$. Then $M_D = A'A - R'R$.*

We leave the proof to the Appendix as it uses standard techniques from the literature. However, we use this equivalence to construct the lowest norm AR embedding:

Algorithm 1 Construct Minimal AR Decomposition

Solve the convex problem:

$$\min_{\hat{M}} ||\hat{M}||_* \text{ s.t. } \hat{M}_{ij} = e_{ij} \forall i \neq j$$

Compute the eigendecomposition of $\hat{M} = Q'DQ$.
if low rank is desired **then**
  Truncate the $n-k$ smallest in absolute value eigenvalues to 0
end if
Let D^- be the strictly negative eigenvalues
D^+ be the strictly positive ones
Let Q^- correspond to the eigenvectors with negative eigenvalues and Q^+ be the eigenvectors with positive eigenvalues.
Set $A = Q^+\sqrt{D^+}$ and set $R = Q^-\sqrt{-D^-}$
Rows of A are a_i, rows of R are r_i

For relatively small matrices, we can solve nuclear norm minimization directly. However, it scales poorly with matrix size. A popular solution for medium size approximate solutions to the nuclear norm is SVT (Cai et al. (2010)). We use code implemented in the R package *filling* (You, 2020).

Often we are willing to take a lossy compression of our data - i.e. a low rank representation. To select the 'natural' rank we use generalized Gabriel bi-cross-validation (BCV)

(Owen et al., 2009). In BCV the row and column indices are split into folds, one fold of the matrix, is held out while the rest of the matrix is used to fit a low-rank factorization. The estimated 'natural' rank of the matrix is the one which minimizes average held out loss. We point the readers to the exposition in (Owen et al., 2009) which discusses the guarantees of BCV as well as advantages of this method over many other choices. However, any method for rank selection for can be used in the procedure above.

For cases where SVT cannot be applied, we will use gradient descent methods and add an explicit regularizer on $||A||, ||R||$ when we deal with link prediction.

5. Hierarchical or 'Tree-Like' Graphs and AR

A recent literature focuses on another failure point of Euclidean embeddings: they are poor at representing hiearchical graphs. A proposed solution to this problem is instead embedding graphs into hyperbolic space which has better representation capacity for such graphs (Nickel & Kiela, 2017). Note that hyperbolic models continue to be metric, yet we will see there is a deep relationship between pseudo-Euclidean embeddings and hyperbolic ones.

We follow the exposition in Nickel & Kiela (2017) to introduce this model. The Poincare model of d-dimensional hyperbolic space is given by the open ball $\mathcal{B}^d = \{x \in \mathbb{R}^d | ||x|| < 1\}$ endowed with the metric tensor $g_x = (\frac{2}{1-||x||^2})g^E$ where g^E is the Euclidean metric tensor.

The distance between any two points in the poincare model is given by

$$d(x,y) = \text{arccosh}(1 + 2\frac{||x-y||^2}{(1-||x||^2)(1-||y||^2)}).$$

While so far we have focused on exact representations that require $e_{ij} = v_i \cdot v_j$, the experimental measures in Nickel & Kiela (2017) use a slightly different criterion, easier to apply in unweighted ($e_{ij} \in \{1, 0\}$) graphs:

Definition 5.1. A set of vectors $\mathcal{V}$ with metric d order-represents an unweighted graph if for any i, j with $e_{ij} = 1$ and k with $e_{ik} = 0$ we have $d(i,j) < d(i,k)$. In other words, for any node any other nodes it is connected to are closer to it in embedding space than any non-connected nodes.

For our comparison purposes we will work with this weaker requirement. Adapting the definition to the AR product means simply replacing the $d(i, \cdot)$ with the AR product and reversing the inequality (since in distance more similar is smaller but in inner product more similar is larger). Given these definitions we can now show the following result:

Proposition 5.2. *Let G be a graph that is order-represented by a hyperbolic embedding $\mathcal{V}$ of dimension d. Then there exists a $d+1$ dimensional AR embedding with R dimension 1 that also order-represents the graph.*

The proof of this Proposition is relatively straighforward and uses fact that there is a diffeomorphic model of hyperbolic space called the Lorenz model (Nickel & Kiela, 2018). We relegate it to the Appendix.

6. "Neighbors" in Pseudo-Euclidean Space

One of the most common uses of embeddings in practice is clustering or nearest neighbor lookup. Formally, the problem is: given a query vector q and a database of node embeddings for all nodes $j \neq q$ we want to find the nearest neighbor of q.[1] Normal Euclidean embeddings have a single notion of nearest neighbor given by $\text{argmax}_j v_q \cdot v_j$.

In this section we will show 1) there are at least 4 interesting notions of 'neighbor' in pseudo-Euclidean space, each has a different interpretation, 2) in practice we can still use Euclidean nearest neighbor libraries by appropriately formatting the query vector.

First, we write the database of vectors $\mathcal{D}$ to be searched as the concatenation of vectors $[a_j, r_j]$ for each node j. We can represent the query q as the concatenation $[a_q, -r_q]$. If we compute dot product neighbors of q we get back nodes with high values of $a_q \cdot a_j - r_q \cdot r_j$. This corresponds to nodes j with high e_{qj} in the original graph. We will refer to this as 'first order' similarity of nodes and denote it by $F(q,j)$.

We can also represent q as $[a_q, r_q]$. Finding standard dot product nearest neighbors in this case results in nodes j with high values of $a_q \cdot a_j + r_q \cdot r_j$. If q and j have a very high value of this product, it means that for any other node k, e_{ij} and e_{ik} are very close. Thus, this returns nodes with similar neighborhoods to q, which is sometimes called 'second order' similarity (Tang et al., 2015), we denote by $S(q,j)$.

We consider the difference $S(q,j) - F(q,j)$. Given a query node q a node j scores high on this composite metric if it has the same neighbors but is not connected to q. In other words, if q and j are part of many forbidden triads. We will later see that this corresponds to 'substitute' pairs in co-ocurrence graphs. Replacing F, S by their definitions gives $a_q \cdot a_j + r_q \cdot r_j - a_q \cdot a_j - r_q \cdot r_j = 2r_q \cdot r_j$. Since distances here are dimensionless we can replace this with $r_q \cdot r_j$ - i.e. the dot product nearest neighbor in R space or a lookup using the $[0, r_q]$ as the query vector.

[1]This is used in many real world machine learning pipelines under the name of 'vector databases' (Raghavan & Wong, 1986) with recent interest in constructing fast, large-scale, nearest neighbor lookup libraries (Johnson et al., 2019).

Proximity	Interpretation
$a_i \cdot a_j - r_i \cdot r_j$	Measures whether i, j are directly connected
$a_i \cdot a_j + r_i \cdot r_j$	Measures whether i, j connected to same other nodes
$r_i \cdot r_j$	Measures whether i, j are part of forbidden triads - i.e. connected to similar others but not to each other - useful in finding 'substitute' nodes
$a_i \cdot a_j$	Measures whether i, j are part of many triangles - connected to same other nodes and to each other

Figure 1. While Euclidean space has a single notion of proximity, pseudo-Euclidean embeddings admit multiple notions of proximity between nodes. Here we summarize four different ways of computing node proximity and their interpretation in practice.

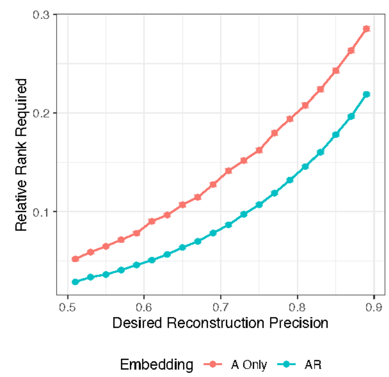


Figure 2. AR embeddings are much more efficient at compressing social networks than dot product embeddings. Error bars (very small) reflect standard errors.

The final notion we consider are nodes j that score high on $S(q, j) + F(q, j)$. These nodes are nodes which have similar neighborhoods *and* are strongly connected to each other. In other words, these are nodes that are part of many triangles with q. The same argument as the paragraph above gives that these are nodes with high similarity in A space, $a_q \cdot a_j$. This corresponds to standard nearest neighbors in A space or lookups using the query vector $[a_q, 0]$.

7. Empirical Evaluation

We now turn to an empirical evaluation of pseudo-Euclidean embeddings. We will evaluate representational capacity (experiments 1, 2), ability to learn about graphs from different notions of neighbors (experiments 3A, 3B, 4A, 4B, 5), and finally generalization capacity when slotted into commonly used link prediction models (experiment 6).

7.1. Experiment 1: AR vs Euclidean on Social Graphs

Our theoretical results show that AR embeddings require fewer, and sometimes drastically fewer, dimensions to perfectly represent the same graph as a dot product embedding. We first ask: does this hold for approximate representation?

We consider the anonymized ego-networks (an ego network takes a focal ego, takes all of their friends, and maps the friendships between them) of 627 users of a music social network (Rozemberczki et al., 2020). We consider users with at least 50 friends (mean ego network size = 81.6).

We construct minimal AR embeddings as described above. Letting e_{ij} be the true edges and $\hat{e}_{ij}$ be the model estimated edges we first consider the variance explained in e by $\hat{e}$ (reconstruction precision). We consider what dimension of embedding is required to achieve a given reconstruction quality across our 627 networks in Figure 2. The standard dot product requires a $\sim 50\%$ higher dimensionality to recover the network with the same fidelity as the AR decomposition.

7.2. Experiment 2: Representing Hierarchical Graphs

Theory guarantees that if hyperbolic embeddings can represent a graph, an AR representation also exists. However, just because a solution exists doesn't mean that gradient descent - the most common way that embeddings are trained in practice - can actually find it. This is what we study here using an experiment similar to Nickel & Kiela (2017) looking at the *transitive closure* of the mammal subtree of WordNet (Miller, 1995).

For the hyperbolic embeddings we use code directly from the paper repository.

A standard method for embedding unweighted (i.e. $e_{ij} \in \{0, 1\}$) graphs into Euclidean space is using an exponential family link function (Hoff et al., 2002; Rudolph et al., 2016) which we will refer to as a **logistic node embedding** (LNE). In the LNE we model $p(e_{ij} = 1) = \sigma(v_i \cdot v_j)$ where σ is the sigmoid function. Another way to think about LNE is that it is a Euclidean embedding of the matrix of logits (rather than the original binary edges).

LNE is trained with binary cross entropy loss on the binary edge labels. Since the graphs are sparse, we need to use negative sampling. We use the strategy introduced in Lerer et al. (2019) - for every positive sample e_{ij} we consider two 'corruptions' e_{ik} where k is not a true neighbor of i in the graph. In one we take k sampled uniformly from the set of non-neighbors, in the other we take k sampled proportional to its degree. This means that more common nodes are represented more highly, but for graphs with fat tailed degree distributions they do not completely dominate the set of negative samples.

We consider the AR extension of this model (LNE-AR) by using the AR product instead of the dot product giving $p(e_{ij}=1)=\sigma(a_i \cdot a_j - r_i \cdot r_j)$. In this experiment we use a single R dimension as suggested by our theorem, we refer to this as LNE-AR1.

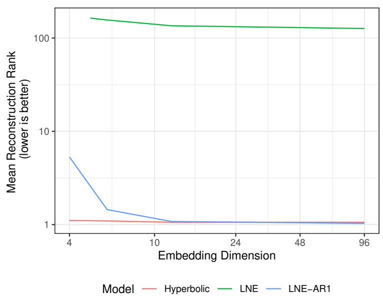


Figure 3. Pseudo-Euclidean with a single R dimension (LNE-AR1) embeddings are able to represent hierarchical relations while Euclidean (LNE) embeddings are not.

As in Nickel & Kiela (2017) we use mean reconstruction rank which takes every real edge (i, j) and all corresponding negative edges (i, k) where $e_{ik} = 0$ and asks how many k rank above j in terms of dot product/distance. In essence, we ask: are true neighbors of i closer in embedding space than non-neighbors?

Figure 3 shows our results. LNE fails completely to represent the graph even in high dimensions. We also see that the optimization does not quite reproduce the bound of our theorem: a 4 dimensional hyperbolic embedding represents the graph perfectly but a 5 dimensional LNE-AR1 still is slightly behind in terms of representation accuracy. This small discrepancy is likely due to many factors including the fact that while the hyperbolic code optimizes directly for the contrastive loss, the LNE optimizes for the classification loss and that the negative sampling and optimization procedures are off the shelf and not tuned. Nevertheless, the experiment shows that AR embeddings can work well in real world graphs that exhibit both intransitivity and hierarchy.

7.3. Experiment 3: Measuring Homophily and Heterophily using AR

We now begin to ask whether there are gains from using AR embeddings from an interpretability standpoint. That is, can we learn interesting things about the graph directly from the embeddings?

In the case of social networks, there is a straightforward interpretation of the AR decomposition. There are latent attributes A on which birds of a feather flock together (i.e. the homophily in the network) and there are latent attributes R where opposites attract (the heterophily in the network).

In the AR decomposition we can see how much of a network is explained by the R component by looking at

Network	R-Fraction	Node Assortativity
Wisconsin	.59	.15
Texas	.66	0.05
Cornell	.68	0.11
Citeseer	.81	0.72
Cora	.81	0.82
EU	.87	0.46

Table 1. Graph level R-fraction predicts node assortativity across graphs. Node assortativity is a commonly used heuristic for identifying a graph as homophilous or heterophilous.

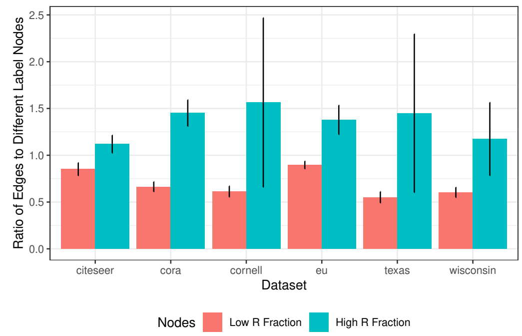


Figure 4. Nodes with higher R-fractions than the median in the graph have more links to nodes with different labels. Error bars reflect standard errors computed at the node level.

$\frac{||R||_F^2}{||A||_F^2 + ||R||_F^2}$. We call this the R-fraction of the network. We can think of this as a latent measure of heterophily.

To evaluate whether R-fraction is a useful measure we consider symmetric versions of standard datasets from the graph literature: Cora (McCallum et al., 2000), Citeseer (Giles et al., 1998), WebKB-Wisconsin, WebKB-Cornell, WebKB-Texas (Craven et al., 1998). We also include the EU e-mail dataset (Leskovec et al., 2007; Yin et al., 2017). Each dataset has a label for each node and thus we can use node assortativity (fraction of neighbors sharing focal node's label) as an observed proxy for heterophily in the dataset as is defined in the recent literature on heterophilic GNNs (Zhu et al., 2021; Zheng et al., 2022).

We begin by seeing whether the R-fraction of a network predicts its label assortativity. We use the algorithm outlined in Section 4 to build low rank AR representations. In Figure 1 we see that the R fraction of these low rank representations indeed predicts label assortativity at the network level.

7.4. Experiment 3B: Local Measures of Heterophily

The analysis above looks at the graph as a whole, but the same idea can be applied to each node. Given an AR embedding, we can compute the R-fraction for each node individually $\frac{||r_i||}{||r_i|| + ||a_i||}$. We ask whether individuals that have relatively high R-fractions are more likely than low-R fraction nodes to be connected across labels. In Figure 4 we take each graph, consider nodes with at least 5 neighbors, and median split these nodes by R-fraction. We then look at the number of edges they have to nodes with labels not the same as their own. To make comparisons across networks with very different degree distributions we normalize by the average number of edges that a node has to other nodes of different labels - in other words, we ask: do nodes whose R-fraction is above the median have more edges to nodes with different labels than average? In Figure 4 see that the answer is yes. Again, label data is *not used* at all during AR embedding construction, so we are reading per-node 'heterophily' purely from the graph.

7.5. Experiment 4: Finding Substitutes using AR

There is recent interest in using embedding techniques to find substitutable products (Ruiz et al., 2020). Substitutes in this case are defined as products which fulfill the same need - or, in the case of co-purchase graphs, are purchased with the same items but rarely together. For example, both Pepsi and Coke may be purchased with Hamburgers and Fries, but a purchase which contains Pepsi usually does not also contain Coke. Section 6 shows that, in theory, we can find such pairs by looking at neighbors in R space. We now ask whether this yields meaningful substitutes in practice.

7.5.1. Experiment 4A: Roles on Teams

We begin by looking at data from the online game DotA2. In this game individuals are placed in a team of 5, each individual chooses one of 115 (as the time of this analysis) 'heroes', and the team competes against another team. As with many team sports, there are different roles on a team that need to be covered and so real world teams are unlikely to include multiple copies of the same role. Heroes in DotA are different and specialized, each able to play only a subset of roles.

We use a publicly available Kaggle dataset of $39,675$ DotA matches. From this data we construct a co-occurrence matrix for the heroes. Letting c_{ij} be the co-occurrence between i and j. Because the co-occurences are extremely right skewed, we consider the matrix of $\log(c_{ij} + 1)$ though qualitatively all our results go through using the raw co-occurrence counts as well. We take the low rank (k=10) exact AR decomposition of this co-occurrence matrix.

For each possible hero, we take the list of 'official roles' the hero can play from the the DotA wiki. We construct a vector for each hero where a 0 in a dimension indicates that the hero cannot play that role and 1 indicates they can. We then ask whether, given two heroes, the similarity in these 'true role vectors' is predicted by their similarity in A or R space. Since similar roles are substitutes, we should expect to see R but not A similarity to be related to role similarity, which is precisely what we see in Figure 5. Again, the embeddings A and R do not use any role labels in their construction, only co-occurrence counts.

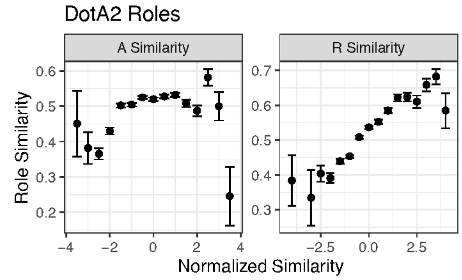


Figure 5. In our DotA data we see that similarity in A vectors does not predict similarity in roles very well but similarity in R vectors (produced without knowing roles) does. Error bars reflect standard errors computed at the bin level.

target	substitute	score
baking mix	bisquick	0.78
baking powder	baking soda	0.85
beer	apple juice	0.40
brown sugar	sugar	0.74
buttermilk	skim milk	0.52
chicken broth	vegetable broth	0.63
lemon	fresh lemon juice	0.76
onion	scallion	0.71
orange juice	honey	0.61
parmesan cheese	mozzarella	0.64
parsley	dried parsley	0.59
pecan	walnut	0.85
pecan	sliced almond	0.65
red wine	dry white wine	0.68
unsalted butter	margarine	0.67
unswtd chocolate	baking cocoa	0.74
vegetable oil	canola oil	0.88
vinegar	cider vinegar	0.89
yogurt	greek yogurt	0.70

Figure 6. Substitutes for various focal ingredients found by looking at R neighbors.

7.5.2. Experiment 4B: Substitutes in Ingredients

We now look at a different substitute task. We use a dataset of $180,000+$ cooking recipes (Majumder et al., 2019). We construct the log co-occurrence matrix of the 1000 most common ingredients in these recipes. We compute the exact low rank (k=125) AR decomposition of this matrix.

We then look at some commonly substituted cooking ingredients. We restrict to focal ingredients that appear in the 1000 most commonly used ingredients and have exact 1-1 substitutes rather than mixtures of items. In Table 2 we take some focal ingredients and show their nearest R neighbors using the cosine similarity. We use cosine similarity as the length of an R or A vector encodes a node's commonality. We include only the top neighbor for space here, in the Appendix we include an expanded version of the table including the top 3 suggested substitutes per target ingredient. We see that using R-similarity as a substitutability metric seems to yield qualitatively good results in this dataset.

7.6. Experiment 5: Inhibition and Activation in Biological Networks

Systems biology is a field focusing on study of interactions between genes or proteins. Deterministic or stochastic dynamical systems are usually used to model these interactions. However, the topology of the governing equations is quite often partially or fully unknown. We ask whether AR embeddings can help researchers recover information about the directed graph of structural equations governing interactions from observed co-occurrence relationships.

Most real gene regulatory networks are poorly understood, so simulations of a gene regulatory networks are often used. In our example we use a commonly used, simplified model of hematopoietic stem cell differentiation (Krumsiek et al., 2011). This network consists of 11 transcription factors with 28 directed regulatory interactions between them. Some exhibit activation relationships (x makes y more likely) and some of which exhibit inhibition (x makes y less likely), see the Appendix A.2 for a full description. We sampled snapshots of expressions from the system. From these snapshots we construct the co-occurrence matrix of transcription factors. We compute the AR decomposition of this matrix using the same methodology as the experiments above (rank = 8).

We then compare the similarity in A and R components across inhibitor and activator pairs. Importantly, while inhibition/activation are directed relationships, we only observe undirected correlations. In Figure 7 we see that activators are closer in A space while inhibitors are closer in R space. Looking at the Euclidean embedding of the correlation between two nodes does not display as clean of a pattern. See the Appendix A.2 for another analysis visualizing the embeddings.

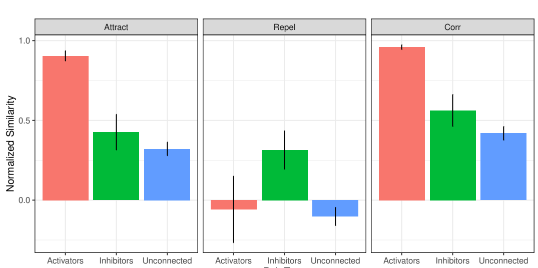


Figure 7. Activators are closer in A space than average while inhibitors are close in R space. Euclidean embeddings are less clear. Error bars reflect standard errors.

7.7. Experiment 6: Pseudo-Euclidean Embeddings in Other Models

In this section, we move from the problem of reconstruction (asking how well models can express certain data) and consider the task of link prediction - in other words, generalization to unseen edges.

The first models we consider are the LNE/LNE-AR from above. An increasingly popular method for dealing with graph data are graph neural networks (GNN) (Zhou et al., 2020). Typically GNNs are used in graphs where nodes have feature vectors to do inductive classification tasks, however since GNNs construct a vector for each node (call this v_i^{GNN}) we can also use these vectors for link prediction tasks (Zhang & Chen, 2018) by using $\sigma(v_i^{GNN} \cdot v_j^{GNN})$ as our edge probabilities. We can convert the GNN to do the AR embeddings simply by splitting the v^{GNN} vectors and using the LNE-AR formula for edge probabilities.

Our goal is to investigate bonuses provided by AR rather than trying to achieve perfect state of the art performance. Thus, we focus on the simple graph convolutional network (GCN) (Kipf & Welling, 2016). We train LNE, LNE-AR, GCN, and GCN-AR models on Cora, Citeseer, Wisconsin, Texas, Cornell which all have feature vectors for all nodes. We do not use the EU data or the ego network data as they do have node features and so are less interesting from the GCN standpoint.

We split each dataset into $80/10/10$ train/validation/test. We adapt the hyperparameters, training, evaluation, and error bar construction code from Chami et al. (2019) for all of our experiments. See Appendix A.3 for more details.

In Figure 8 we plot the test set AUC for these models. In graphs which had high intransitivity in Table 1 we see a large gain from using AR as opposed to Euclidean embeddings in both GCN and LNE models.

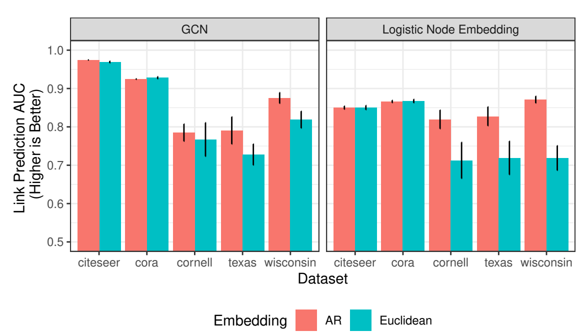


Figure 8. Pseudo-Euclidean embeddings peform better in link prediction in both GCN and LNE models when underlying graphs are intransitive. Error bars reflect multiple iterations with different random seeds as in Chami et al. (2019).

References

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

Barabasi, A.-L. and Oltvai, Z. N. Network biology: understanding the cell's functional organization. *Nature reviews genetics*, 5(2):101–113, 2004.

Cai, J.-F., Candès, E. J., and Shen, Z. A singular value thresholding algorithm for matrix completion. *SIAM Journal on optimization*, 20(4):1956–1982, 2010.

Candès, E. J. and Recht, B. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

Chami, I., Ying, Z., Ré, C., and Leskovec, J. Hyperbolic graph convolutional neural networks. *Advances in neural information processing systems*, 32, 2019.

Chanpuriya, S., Musco, C., Sotiropoulos, K., and Tsourakakis, C. Node embeddings and exact low-rank representations of complex networks. *Advances in Neural Information Processing Systems*, 33, 2020.

Craven, M., McCallum, A., PiPasquo, D., Mitchell, T., and Freitag, D. Learning to extract symbolic knowledge from the world wide web. Technical report, Carnegie-mellon univ pittsburgh pa school of computer Science, 1998.

Easley, D., Kleinberg, J., et al. *Networks, crowds, and markets*, volume 8. Cambridge university press Cambridge, 2010.

Giles, C. L., Bollacker, K. D., and Lawrence, S. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, pp. 89–98, 1998.

Granovetter, M. Economic action and social structure: The problem of embeddedness. *American journal of sociology*, 91(3):481–510, 1985.

Granovetter, M. S. The strength of weak ties. *American journal of sociology*, 78(6):1360–1380, 1973.

Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

Hoff, P. Modeling homophily and stochastic equivalence in symmetric relational data. *Advances in Neural Information Processing Systems*, 20:657–664, 2007.

Hoff, P. D., Raftery, A. E., and Handcock, M. S. Latent space approaches to social network analysis. *Journal of the american Statistical association*, 97(460):1090–1098, 2002.

Jackson, M. O. *Social and economic networks*. Princeton university press, 2010.

Jeong, H., Mason, S. P., Barabási, A.-L., and Oltvai, Z. N. Lethality and centrality in protein networks. *Nature*, 411 (6833):41–42, 2001.

Johnson, J., Douze, M., and Jégou, H. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7 (3):535–547, 2019.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Krumsiek, J., Marr, C., Schroeder, T., and Theis, F. J. Hierarchical differentiation of myeloid progenitors is encoded in the transcription factor network. *PloS one*, 6(8):e22649, 2011.

Lerer, A., Wu, L., Shen, J., Lacroix, T., Wehrstedt, L., Bose, A., and Peysakhovich, A. Pytorch-biggraph: A large-scale graph embedding system. *arXiv preprint arXiv:1903.12287*, 2019.

Leskovec, J., Kleinberg, J., and Faloutsos, C. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)*, 1 (1):2–es, 2007.

Liu, Q., Nickel, M., and Kiela, D. Hyperbolic graph neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Lovász, L. and Vesztergombi, K. Geometric representations of graphs. *Paul Erdos and his Mathematics*, 2, 1999.

Majumder, B. P., Li, S., Ni, J., and McAuley, J. Generating personalized recipes from historical user preferences. *EMNLP*, 2019.

Mazumder, R., Hastie, T., and Tibshirani, R. Spectral regularization algorithms for learning large incomplete matrices. *The Journal of Machine Learning Research*, 11: 2287–2322, 2010.

McCallum, A. K., Nigam, K., Rennie, J., and Seymore, K. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

Miller, G. A. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

Naber, G. L. *The geometry of Minkowski spacetime: An introduction to the mathematics of the special theory of relativity*, volume 92. Springer Science & Business Media, 2012.

Ng, A. Y., Jordan, M. I., Weiss, Y., et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

Nickel, M. and Kiela, D. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30, 2017.

Nickel, M. and Kiela, D. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International Conference on Machine Learning*, pp. 3779–3788. PMLR, 2018.

Owen, A. B., Perry, P. O., et al. Bi-cross-validation of the svd and the nonnegative matrix factorization. *The annals of applied statistics*, 3(2):564–594, 2009.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

Raghavan, V. V. and Wong, S. M. A critical analysis of vector space model for information retrieval. *Journal of the American Society for information Science*, 37(5): 279–287, 1986.

Rozemberczki, B., Kiss, O., and Sarkar, R. Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*, pp. 3125–3132. ACM, 2020.

Rudolph, M., Ruiz, F., Mandt, S., and Blei, D. Exponential family embeddings. *Advances in Neural Information Processing Systems*, 29, 2016.

Ruiz, F. J., Athey, S., Blei, D. M., et al. Shopper: A probabilistic model of consumer choice with substitutes and complements. *Annals of Applied Statistics*, 14(1):1–27, 2020.

Seshadhri, C., Sharma, A., Stolman, A., and Goel, A. The impossibility of low-rank representations for triangle-rich complex networks. *Proceedings of the National Academy of Sciences*, 117(11):5631–5637, 2020.

Sim, A., Wiatrak, M. L., Brayne, A., Creed, P., and Paliwal, S. Directed graph embeddings in pseudo-riemannian manifolds. In *International Conference on Machine Learning*, pp. 9681–9690. PMLR, 2021.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pp. 1067–1077, 2015.

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. Complex embeddings for simple link prediction. In *International conference on machine learning*, pp. 2071–2080. PMLR, 2016.

Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Van der Maaten, L. and Hinton, G. Visualizing non-metric similarities in multiple maps. *Machine learning*, 87(1): 33–55, 2012.

Yin, H., Benson, A. R., Leskovec, J., and Gleich, D. F. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 555–564, 2017.

You, K. *filling: Matrix Completion, Imputation, and Inpainting Methods*, 2020. URL `https://CRAN.R-project.org/package=filling`. R package version 0.2.2.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

Zheng, X., Liu, Y., Pan, S., Zhang, M., Jin, D., and Yu, P. S. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082*, 2022.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

Zhu, J., Rossi, R. A., Rao, A., Mai, T., Lipka, N., Ahmed, N. K., and Koutra, D. Graph neural networks with heterophily. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11168–11176, 2021.

A. Appendix

Proof of Proposition 3.2. Let G be an arbitrary graph. We will construct an embedding that represents it.

Let N be the number of nodes, let D be a vector in $\mathbb{R}^N$. Let M_D be an $N \times N$ matrix where $m_{ij} = e_{ij}$ for $i \neq j$ and D is the matrix diagonal. If M_D is positive semi-definite then there exists a factorization $M_D = VV'$.

Row vectors of V are embeddings for each node that represent the graph G. We can see now that any D which makes M_D positive semi-definite gives us these required embeddings. Such a D always exists since we can always construct one by taking D arbitrary, computing the largest negative eigenvalue λ_{min} and then the matrix $M_{D+\lambda_{min}}$ will be positive semi-definite. Clearly the family of embeddings that represent the graph can be put into $1-1$ correspondence with the set of matrix diagonals that make M_D positive semi-definite. $\square$

Proof of Proposition 3.3. In essence, we want to determine the rank, that is, determine the dimension of the nullspace of a matrix

$$M = \begin{pmatrix} A & R \\ R & B \end{pmatrix}$$

where A is an $n \times n$ diagonal matrix with coefficients $a_1 \dots a_n > 0$, B is an $n \times n$ diagonal matrix with coefficients $b_2 \dots b_n > 0$, and R is an $n \times n$ matrix of all 1. Let's solve!

$$M \times \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} = 0 \quad \Longleftrightarrow \quad \forall i \begin{cases} a_i u_i + \sum_j v_j = 0 \\ b_i v_i + \sum_j u_j = 0 \end{cases}$$

Since $a_i > 0$ and $b_i > 0$, this implies

$$\forall i \quad u_i = -\frac{1}{a_i} \sum_j v_j \quad v_i = -\frac{1}{b_i} \sum_j u_j \tag{1}$$

If we were free to set $\sum_j v_j$ and $\sum_j u_j$ as we please, the two equations (1) would describe a 2-dimensional space. Therefore the nullspace of M has dimension at most 2. However we can also use the first of these equations to write

$$\sum_i u_i = -\sum_i \frac{1}{a_i} \sum_j v_j \tag{2}$$

Therefore our nullspace has dimension at most 1. But we can continue and use the second equations from (1) to replace v_j above:

$$\sum_i u_i = -\left(\sum_i \frac{1}{a_i}\right)\left(\sum_j v_]\right) = \left(\sum_i \frac{1}{a_i}\right)\left(\sum_j \frac{1}{b_j}\right) \sum_k u_k$$

Therefore, if $r = \left(\sum_i \frac{1}{a_i}\right)\left(\sum_j \frac{1}{b_j}\right) \neq 1$, then we must have $\sum_i u_i = \sum_j v_j = 0$ which means that $u_i = v_j = 0$: the matrix is nonsingular. On the other hand, if $r = 1$, then I can choose $\sum_j v_j$ equal to any non zero value, deduce $\sum_j u_i$ using (2), compute u_i and v_i using (1), and verify that we have described a one-dimensional nullspace.

In conclusion: if $r = \left(\sum_i \frac{1}{a_i}\right)\left(\sum_j \frac{1}{b_j}\right) \neq 1$, the matrix has full rank. If $r = 1$ the matrix has rank $2n - 1$. This is the case, for instance, when $a_i = b_j = n$. $\square$

Proof of Proposition 4.2. Let D be a solution of problem $\min_D \|M_D\|_*$. Matrix M_D is real and symmetric and therefore diagonalizable. For each eigenvector u associated with a positive eigenvalue λ, we can form a vector $a = u\sqrt{\lambda}$. For each eigenvector u associated with a negative eigenvalue, we can form a vector $r = u\sqrt{-\lambda}$. Collecting these a and r vectors into matrices A and R, we obtain $M_D = A\,A^\top - R\,R^\top$ and $\|M_D\|_* = ||A||_F^2 + ||R||_F^2$. Then, (A, R) is also a solution of problem $\min_{(A,R)\in\mathcal{AR}} \; ||A||_F^2 + ||R||_F^2$ because, for any $(A', R') \in \mathcal{AR}$,

$$||A||_F^2 + ||R||_F^2 = \|M_D\|_* \leq \|A'\,A'^\top - R'\,R'^\top\|_* \leq ||A'||_F^2 + ||R'||_F^2$$

where the first inequality holds because $\|M_D\|_*$ is optimal, and the second inequuality results from Lemma 6 of (Mazumder et al., 2010) after noticing that $A'\,A'^\top - R'\,R'^\top = [A', R']\,[A', -R']^\top$.

Conversely, let A, R be a solution of problem $\min_{(A,R)\in\mathcal{AR}} \ ||A||_F^2+||R||_F^2$. Then matrix $M_D = A\,A^\top - R\,R^\top$ is a solution of problem $\min_D \|M_D\|_*$, because, for any D', we can diagonalize $M_{D'}$ as above and form $(A', R') \in \mathcal{AR}$ such that

$$\|M_D\|_* = \|A\,A^\top - R\,R^\top\|_* \le ||A||_F^2 + ||R||_F^2 \le ||A'||_F^2 + ||R'||_F^2 = \|M_{D'}\|_*$$

where the first inequality results again from Lemma 6 of (Mazumder et al., 2010). □

Proof of Proposition 5.2. The Lorenz model works as follows: we take a vector in $\mathbb{R}^{d+1}$ written as $(x_0, x_1, \ldots, x_d)$ and define the inner product

$$\mathcal{L}(x, y) = -x_0 y_0 + \sum_{i=1}^{d} x_i y_i.$$

This is clearly the pseudo-Euclidean inner product with R dimension 1. However, while pseudo-Euclidean space is not a metric space, we can take the manifold defined by

$$\mathcal{H}^d = \{x \in \mathbb{R}^{d+1} \mid \mathcal{L}(x, x) = -1, x_0 > 0\}$$

and endow it with the metric

$$d(x, y) = \text{arccosh}(-\mathcal{L}(x, y)).$$

This defined manifold and metric is diffeomorphic to the Poincare ball. Given a set of vectors on the Poincare ball, we can take the inverse of this diffeomorphism on these vectors to get their Lorenz counterparts. Then since $d(x, y)$ here is a monotone transformation of $\mathcal{L}(x, y)$, we have that if $d(x, y) > d(x, z)$ then $\mathcal{L}(x, y) < \mathcal{L}(x, z)$. Thus we have constructed an AR embedding that order-represents the graph. □

A.1. Expanded Ingredient Substitution List

In the main text we reported the top R neighbor for each focal ingredient to save space. Here we report the top 3 neighbors per each focal ingredient.

A.2. Experiment 5 Supplement

We show the governing equations of the Krumsiek et al. (2011) model in 9. The original network is represented by boolean rules, which we translated into a system of ODEs to allow us to sample from the model.

In addition to the analysis in the main text, we use Kernel PCA to visualize the 2 dimensional projection of the PSD factorization of the Spearman correlation matrix of transcription factors (panel B) compared to the projections of the A (panel C) and R (panel D) components of the AR decomposition.

We see that mutual inhibitors have the highest R-similarity scores (and so are close together in the Kernel PCA representation). One way inhibitions have lower scores, which is not surprising, because the inhibitions doesn't happen immediately and at some points of time both transcription factors can still be observed together. A similar story is obvious in the A-similarities showing mutual and one-way activations. However, looking at the embedding of the correlations, such relationships are not obvious. This is partially driven by the fact that in the correlations it is hard to differentiate between two items which have similar contexts but do not appear together (i.e. inhibitors) from items which have low correlation because they are on very different pathways.

A.3. Experiment 6 Supplement

The architecture of the GCN works as follows: let X be the feature vectors of nodes stacked, a single layer GCN is written as $R(\hat{A}XW^0)$ where $\hat{A}$ is the normalized adjacency matrix and R is some non-linearity. The GCN takes the node features, maps them into a hidden space by the learned W^0, takes neighbor averages, and passes them through a non-linearity. This outputs an embedding vector for each node. The graph convolution process can be repeated on this vector again if desired, still outputting one vector per node.

When training the link prediction models of dimension d we always use $\frac{d}{2}$ dimensions for A and R in the LNE-AR/GCN-AR cases. Because of this we found it is important to have different regularization rates for each of the subspaces with the full r

target	substitute	R score
baking mix	bisquick	0.78
baking mix	biscuit mix	0.73
baking mix	bisquick mix	0.70
baking powder	baking soda	0.85
baking powder	whole wheat flour	0.51
baking powder	all-purpose flour	0.42
beer	apple juice	0.40
beer	mango	0.39
beer	corn oil	0.39
brown sugar	sugar	0.74
brown sugar	honey	0.69
brown sugar	light brown sugar	0.68
buttermilk	skim milk	0.52
buttermilk	soymilk	0.48
buttermilk	chickpea	0.39
chicken broth	chicken stock	0.85
chicken broth	vegetable broth	0.63
chicken broth	vegetable stock	0.61
lemon	fresh lemon juice	0.76
lemon	lemon, juice of	0.71
lemon	lemon juice	0.66
onion	red onion	0.71
onion	scallion	0.71
onion	yellow onion	0.68
orange juice	honey	0.61
orange juice	orange	0.50
orange juice	lemon	0.47
parmesan cheese	mozzarella	0.64
parmesan cheese	cheddar	0.62
parmesan cheese	olive oil	0.53
parsley	fresh parsley	0.93
parsley	flat leaf parsley	0.65
parsley	dried parsley	0.59
pecan	walnut	0.85
pecan	nut	0.75
pecan	sliced almond	0.65
red wine	dry red wine	0.79
red wine	dry white wine	0.68
red wine	white wine	0.61
unsalted butter	butter	0.74
unsalted butter	margarine	0.67
unsalted butter	heavy cream	0.48
unswtd chocolate	unswtd choc square	0.83
unswtd chocolate	baking cocoa	0.74
unswtd chocolate	unswtd cocoa	0.71
vegetable oil	oil	0.96
vegetable oil	canola oil	0.88
vegetable oil	olive oil	0.67
vinegar	cider vinegar	0.89
vinegar	white vinegar	0.87
vinegar	apple cider vinegar	0.82
yogurt	plain yogurt	0.73
yogurt	greek yogurt	0.70
yogurt	vanilla yogurt	0.53

Table 2. Substitutes for various focal ingredients found by looking at cosine similarity neighbors in the R component.

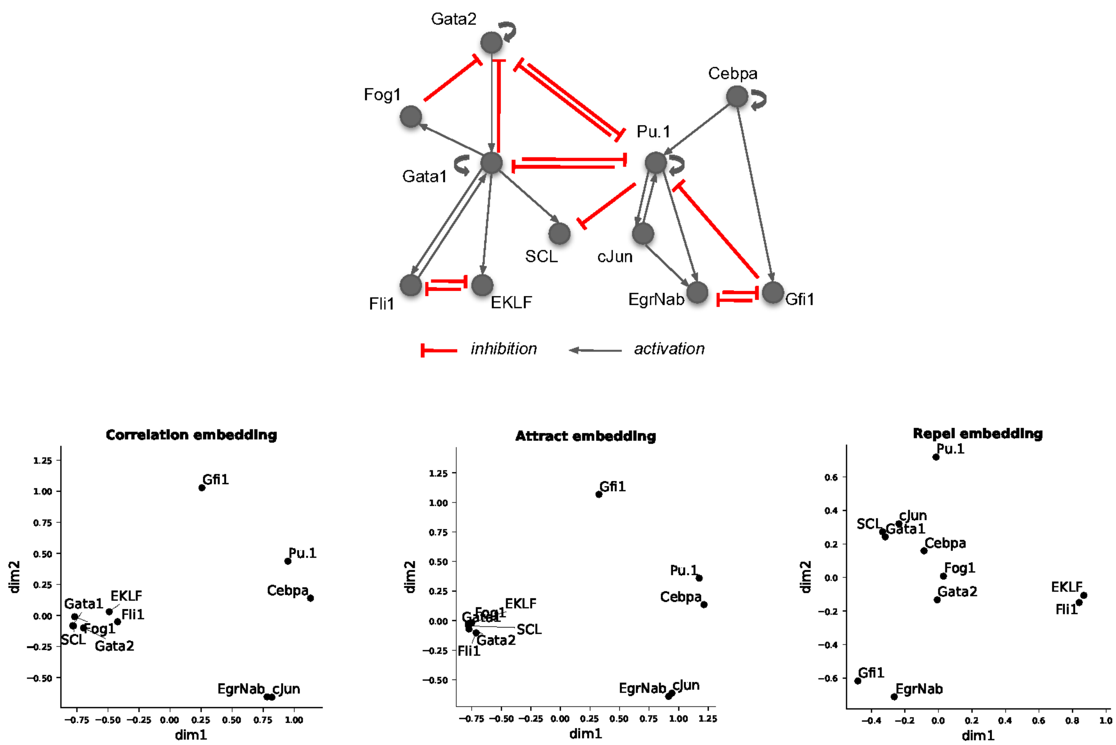


Figure 9. First panel shows the true directed network generating the symmetric co-occurrence patterns we observe with red indicating inhibition and black denoting activation. Other panels show a kernel PCA of the correlation matrix dot product mode as well as the A and R components respectively. While the inhibition/activation structure is well preserved in the AR embeddings, it is not nearly as clear in the standard 'attract only' decomposition.

regularizer being$\lambda_A \sum_i ||a_i|| + \lambda_R \sum_i ||r_i||$. An alternative is to keep a single $L2$ regularization weight over all parameters but fix a dimension d and sweep k so that $d-k$ dimensions are A and k are R.

We use an $80/10/10$ train/validation/test split. We vary hyperparameters of $d \in \{12, 24, 48, 96\}$ and regularization rates $\lambda_i \in \{1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1\}$ for all models. For GCNs we also vary the number of convolution layers (1 or 2) as well as dropout $\in \{0, .1\}$, though we did not find dropout or convolution beyond a single layer to be useful in our datasets.