## Abstract

Hypergraph used to explain higher-order relationships, and the paper use GCN based NHP for link prediction in hypergraphs, can be used for both directed and undirected.

Introduction

Besides the higher-order relationships, the directions modelling is also important.

Related work:

1. Learning Representations on Graph:
    a) Matrix factorization-based method: learn low-dimensional representations of vertices by factorizing metrices that capture the relationships between nodes
    b) Random-work based algorithm: use random walks to learn representations that capture the neighborhood structure around nodes.
    c) Deep learning on graphs: use GCNs handle higher-order relationships in hypergraphs
2. Link Prediction on Hypergraphs:
    a) Spectral Clustering: first **compute the Laplacian matrix L** of the graph(D – A, where D(i, i) is the degree of vertex i, A is the adjacency matrix), then **calculate the eigenvalues and eigenvectors of L** (because the smallest non-zero eigenvalues carry important information about the structure), then **use the eigenvectors to map** each vertex to a point in a low-dimensional space and **cluster the points.**
    b) Coordinated Matrix Minimization (CMM): use non-negative matrix factorization(NMF) and lest squares matching, optimized through the Expectation-Maximization (EM) algorithm, allow use to predict these hyperlinks by learning from the observed patterns in the data and using them to infer the missing parts. (So, usually begin with an incomplete adjacency matrix of the hypergraph)

Two different hypergraphs may result in the same weighed clique expansion (hyperedge is transformed into a clique which is a set of vertices where every pair of vertices is connected).

## Setup:

1. Undirected hyperlink prediction: H = (V, E)
    V = $\{v_1, \cdots, v_n\}$ is a set of n vertices
    E = $\{e_1, \cdots, e_m\} \subseteq 2^V$ is a set of m hyperlinks
    Hyperlink prediction: predict missing hyperlinks from $\bar{E} = 2^V - E$
    Usually using matrix minimization algorithm and uses the expectation-maximization
2. Directed hyperlink prediction
    V = $\{v_1, \cdots, v_n\}$ is a set of n vertices
    E = $\{(t_1, h_1), \cdots, (t_m, h_m)\} \subseteq 2^V \times 2^V$ is a set of m directed hyperlinks.
    Each $e \in E$ is denoted by $(t, h)$, $t \subseteq V$ is the tail and $h \subseteq V$ is the head.

## Neural Hyperlink Predictor

**NHP-U:**

1. Hyperlink-aware GCN layer:
   Problem: each hyperlink connects arbitrary number of vertices (variable cardinality).
   Embedding is refined by aggregating features from its neighbors with that hyperlink.

$$h_v^{(e)} = \mathrm{ReLU}\left(W_{GCN} \sum_{u \in e - \{v\}} x_u + b_{GCN}\right) \qquad (1)$$

   $h_v^{(e)}$: the embedding of vertex v for a particular hyperlink e
   $W_{GCN}$: weight matrix of GCN, transforms the features into a new space
   $x_u$: initial features of the neighboring vertices u in hyperlink e excluding v itself
   $b_{GCN}$: bias term for GCN
   ReLU: $\mathrm{ReLU}(x) = \max(0, x)$

2. Hyperlink scoring layer:
   Problem: to preserve the higher order relationships among the vertices of a given hyperlink

$$I_e = \sigma\left(W \cdot g\left(\{h^{(e)}\}_{v \in e}\right) + b\right) \qquad (2)$$

   σ: sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ , map input to a value between 0 – 1, indicating the likelihood of the link's existence
   W: weight matrix of dimension 1 × d (d is the dimensionality of the vertex embeddings)
   g: aggregation function, mentioned below
   b: bias term
   $I_e$: the score for hyperlink e, higher score suggests a higher likelihood the hyperlink exists

   Equation 3 for g:

$$I_e := \sigma\left(\frac{1}{|e|} W \cdot \sum_{v \in e} h_v^{(e)} + b\right). \qquad (3)$$

   Does not explicitly guarantee any priors (prior knowledge or assumption that are made before observing the data) on the vertex embeddings of a hyperlink.
   Equation 4 for g (chosen):

$$I_e := \sigma\left(W \cdot \mathrm{maxmin}\{h_v^{(e)}\}_{v \in e} + b\right), \qquad (4)$$

   where given a set of vectors $x_1, \cdots, x_k \in \mathbb{R}^d$, the function

$$\mathrm{maxmin}\{x_j : j \in [k]\} = (\max_{s \in [k]} x_{sl} - \min_{i \in [k]} x_{il})_{l=1,\cdots,d} \qquad (5)$$

   Which is the element-wise difference of maximum and the minimum values of the vectors.

3. Optimization (loss function): ranking objective

$$\mathcal{L} = \frac{1}{|E|} \sum_{e \in E} \Lambda\left(\left(\frac{1}{|F|} \sum_{f \in F} I_f\right) - I_e\right), \qquad (6)$$

E: set of existing hyperlinks in the hypergraph

F: set of sampled vertex sets from $2^V$ – E (not in current hyperlinks), used as negative samples for training

$I_e$: scoring function for existing hyperlinks

$I_f$: scoring function applied to the sets in F

$\Lambda(x)$: a non-decreasing function, $\Lambda(x) = \log(1 + e^x)$, used to ensure the loss is non-decreasing with respect to its input

$\mathcal{L}$: loss function is designed to encourage the model to assign higher scores to actual hyperlinks in E compared to the average score of non-hyperlinks in F

4. $f \in F$ Sampling method:
   For each hyperlink $e \in E$:

   (1). create $f \in F$ by having half of the vertices ($\frac{|e|}{2}$) sampled from $e$

   (2). and remaining half from $V - e$.

   Keep the same size to **avoid any possible bias**

**NHP-D:**

1. Hyperlink-aware GCN layer: The same as NHP-U.
2. Hyperlink scoring layer: Still using equation (2), but use the following $g$ function
   $e = (t, h)$, $I_e = \sigma(W \cdot \text{maxmin}_+\{h_v^{(e)}\}_{v \in e} + b)$ where

$$\text{maxmin}_+\{x_v\}_{v \in e} = (\max_{s \in t} x_{sl} - \min_{i \in h} x_{il})_+, l=1,\cdots,d$$

   Where $t$ represents the tail and $h$ represents the head of the hyperlink

   $g$: taking the element-wise maximum in the tail and the element-wise minimum in the head

   $(m)_+$: equals $m$ if $m > 0$ and 0 otherwise, ensuring only positive differences contribute to the score
3. Direction scoring layer:
   Bilinear form:

$$D_{pq} = \sigma\left(\mathbf{p}^T \ W_{BL} \ \mathbf{q} + b_{BL}\right), \qquad (7)$$

   $D_{pq}$: the direction score between hyperlinks $p$ (tail) and $q$ (head)

   $p$ and $q$: the embeddings for the hyperlinks $p$ and $q$, vectors in $\mathbb{R}^d$, getting by the mean of the embeddings of a hyperlink i.e. $\mathrm{p} = \Sigma_{v \in p} h_v^{(p)}$

   $W_{BL}$: the bilinear weight matrix, in $\mathbb{R}^{d \times d}$

   $b_{BL}$: the bias term for the bilinear form

   σ: sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$
4. Ranking objective to predict direction:

$$\mathcal{L}_{dir} = \frac{1}{|E|} \sum_{(t,h) \in E} \Lambda\left(\frac{1}{|F|}\left(\sum_{(p,q) \in F} D_{pq}\right) - D_{th}\right). \qquad (8)$$

   We use a joint optimization strategy: for an ordered pair $e = (t, h)$, we minimize:
   $\mathcal{L} + \lambda \mathcal{L}_{dir}$

**Computational complexity:**

For the given input hypergraph (V, E), let

$$N_1 = \sum_{e \in E} |e| \quad \text{and} \quad N_2 = \sum_{e \in E} \frac{1}{2} \cdot |e| \cdot \Big( |e| - 1 \Big)$$

Computation complexity of NHP:

$$O(N_1 + N_2 + |E|)$$

In sparse real-world hypergraphs, NHP takes

$$O(|V|)$$

## Metrics:

**AUC:**

Area under ROC Curve:

| | Real Labels | | |
|---|---|---|---|
| Predictions | | Positive | Negative |
| | Positive | TP | FP |
| | Negative | FN | TN |

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN}$$

**Recall@k:**

$$Recall@k = \frac{Number\ of\ true\ positive\ in\ top\ k\ scores}{Total\ number\ of\ positive}$$