Hyperedge2vec: Distributed Representations for Hyperedges

Ankit Sharma*, Shafiq R. Joty**,
Himanshu Kharakwal# & Jaideep Srivastava*

This work has been supported in part by the NSF Award IIS-1422802.

May 15, 2018

Outline

Motivation

- Group structured data is more abundant than studied
- Examples:
  - Social: MMO Game or Software Teams, research collaborations, communication tools like Skype, etc.
  - Others: NLP (sentences), Biology (protein complexes), e-commerce (item-sets) and Chemistry (reaction species).

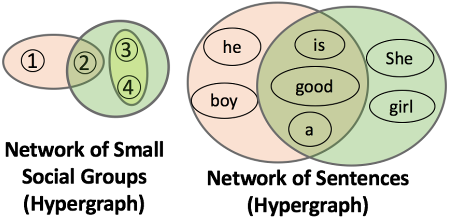


Figure 1: Left is a collaboration network between four individuals and on right is a network resulting from two sentences: "He is a good boy" & "She is a good girl"; and eight word nodes.

Representing Group Structure

- *Hypergraph* is a generalization of graphs
- Naturally captures higher-order relationships between sets of objects
- Hypergraph also has a corresponding *Hasse Diagram*

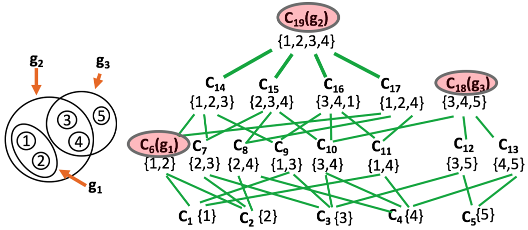


Figure 2: Example of a hypergraph (left) and the hasse diagram (right) corresponding to this hypergraph's simplicial complex.

Problem Statement

- Input:
  - $V = \{v_1, v_2, ..., v_n\}$ represents n elements (actors or words)
  - $G = \{g_1, g_2, ..., g_m\}$, where $g_i \subseteq V$ is a set (group or sentence)
  - Each $g_i \in G$ has occurred $R(g_i)$ times
  - As these sets are overlapping we consider them as a hypergraph $N_g = (V, G)$
  - Incidence matrix $\mathbf{H} \in \{0, 1\}^{|G| \times |V|}$ associated to N_g, with $\mathbf{H}(g_i, v) = 1$ if $v \in g_i$ else 0.
- Goal: Learn the mapping $\mathbf{Z} : G \to \mathbb{R}^d$ from hyperedges to feature representations (i.e., embeddings)

Research Gaps

- A Good *Hyperedge* Embedding Method should:

Research Gaps

- A Good *Hyperedge* Embedding Method should:
  1. Learn hyperedge embeddings directly

Research Gaps

- A Good *Hyperedge* Embedding Method should:
  1. Learn hyperedge embeddings directly
  2. Leverage the hypergraph topology

Research Gaps

- A Good *Hyperedge* Embedding Method should:
  1. Learn hyperedge embeddings directly
  2. Leverage the hypergraph topology
  3. Not loose the hyperedge-level joint information

Research Gaps

- A Good *Hyperedge* Embedding Method should:
  1. Learn hyperedge embeddings directly
  2. Leverage the hypergraph topology
  3. Not loose the hyperedge-level joint information
- *Node* Embeddings for Graphs: Obvious limitation is that they are limited to graphs and not principally designed for set-level embedding. (**not addresses 1, 2 or 3**) [Perozzi et al., 2014] [Tang et al., 2015, Grover and Leskovec, 2016, Cai et al., 2017]

Research Gaps

- A Good *Hyperedge* Embedding Method should:
  1. Learn hyperedge embeddings directly
  2. Leverage the hypergraph topology
  3. Not loose the hyperedge-level joint information
- *Node* Embeddings for Graphs: Obvious limitation is that they are limited to graphs and not principally designed for set-level embedding. (**not addresses 1, 2 or 3**) [Perozzi et al., 2014] [Tang et al., 2015, Grover and Leskovec, 2016, Cai et al., 2017]
- *Node* Embeddings for Hypergraphs:

Research Gaps

- A Good *Hyperedge* Embedding Method should:
  1. Learn hyperedge embeddings directly
  2. Leverage the hypergraph topology
  3. Not loose the hyperedge-level joint information
- *Node* Embeddings for Graphs: Obvious limitation is that they are limited to graphs and not principally designed for set-level embedding. (**not addresses 1, 2 or 3**) [Perozzi et al., 2014] [Tang et al., 2015, Grover and Leskovec, 2016, Cai et al., 2017]
- *Node* Embeddings for Hypergraphs:
  - *Using proxy graphs:* Leverage hypergraph topology but lossy (**Addresses 2**) [Zhou et al., 2006, Hwang et al., 2008] [Agarwal et al., 2006]

Research Gaps

- A Good *Hyperedge* Embedding Method should:
  1. Learn hyperedge embeddings directly
  2. Leverage the hypergraph topology
  3. Not loose the hyperedge-level joint information
- *Node* Embeddings for Graphs: Obvious limitation is that they are limited to graphs and not principally designed for set-level embedding. (**not addresses 1, 2 or 3**) [Perozzi et al., 2014] [Tang et al., 2015, Grover and Leskovec, 2016, Cai et al., 2017]
- *Node* Embeddings for Hypergraphs:
  - *Using proxy graphs:* Leverage hypergraph topology but lossy (**Addresses 2**) [Zhou et al., 2006, Hwang et al., 2008] [Agarwal et al., 2006]
  - *Preserving set-level info.:* Capture hypergraph topology in a loss-less manner (**Addresses 2 & 3**) [Shashua et al., 2006, Bulò and Pelillo, 2009, Ghoshdastidar and Dukkipati, 2017].

Research Gaps (contd.)

- A Good *Hyperedge* Embedding Method should:
  1. Learn hyperedge embeddings directly
  2. Leverage the hypergraph topology
  3. Not loose the hyperedge-level joint information

Research Gaps (contd.)

- A Good *Hyperedge* Embedding Method should:
  1. Learn hyperedge embeddings directly
  2. Leverage the hypergraph topology
  3. Not loose the hyperedge-level joint information
- *Sequence* Embeddings: Limited to sequences (**Addresses 3**) [Bengio and Bengio, 2000, Le and Mikolov, 2014]

Research Gaps (contd.)

- A Good *Hyperedge* Embedding Method should:
  1. Learn hyperedge embeddings directly
  2. Leverage the hypergraph topology
  3. Not loose the hyperedge-level joint information
- *Sequence* Embeddings: Limited to sequences (**Addresses 3**) [Bengio and Bengio, 2000, Le and Mikolov, 2014]
- *Set* Embeddings: Deep Learning and Random Vector Theory based methods (**Addresses 1 & 3**) [Vinyals et al., 2016] [Rezatofighi et al., 2016]

Research Gaps (contd.)

- A Good *Hyperedge* Embedding Method should:
  1. Learn hyperedge embeddings directly
  2. Leverage the hypergraph topology
  3. Not loose the hyperedge-level joint information
- *Sequence* Embeddings: Limited to sequences (**Addresses 3**) [Bengio and Bengio, 2000, Le and Mikolov, 2014]
- *Set* Embeddings: Deep Learning and Random Vector Theory based methods (**Addresses 1 & 3**) [Vinyals et al., 2016] [Rezatofighi et al., 2016]
- But, both completely ignore hypergraph structure (**unaware!**)

Methods

- We propose two methods in order to:
  1. Directly learn hyperedge embeddings
  2. For different cardinality hyperedges simultaneously (i.e. non-uniform hypergraphs)
  3. Capture the hypergraph structure in a principled manner
  4. Retain the hyperedge-level higher-order information

Methods

- We propose two methods in order to:
  1. Directly learn hyperedge embeddings
  2. For different cardinality hyperedges simultaneously (i.e. non-uniform hypergraphs)
  3. Capture the hypergraph structure in a principled manner
  4. Retain the hyperedge-level higher-order information
- **First**, is an **algebraic method** which is based on the novel *dual* (**addresses 1**) tensors (**addresses 4**) of different sizes decomposed simultaneously (**addresses 2**) while regularized by the hypergraph topology (**addresses 3**)

Methods

- We propose two methods in order to:
  1. Directly learn hyperedge embeddings
  2. For different cardinality hyperedges simultaneously (i.e. non-uniform hypergraphs)
  3. Capture the hypergraph structure in a principled manner
  4. Retain the hyperedge-level higher-order information
- **First**, is an **algebraic method** which is based on the novel *dual* (**addresses 1**) tensors (**addresses 4**) of different sizes decomposed simultaneously (**addresses 2**) while regularized by the hypergraph topology (**addresses 3**)
- **Second**, is **neural network** based deep (nonlinear, possibly **addresses 4**) auto-encoder which embeds each hyperedge vector (naturally **addresses 1 & 2**) with the noise generated from hypergraph's *hasse* topology (**addresses 3**)

An Information Theoretic Result from Combinatorics

Proposition

Given a set of random variables $X_1, ...X_c$ $(c \geq 2)$, and $H(.)$ as the information entropy, we have ([Chung et al., 1986, p. 34]):

$$H(X_1, ..., X_c) \leq (\frac{1}{c-1}) \sum_{(i,j) \subseteq 2^{[c]}} H(X_i, X_j)$$

Therefore, the joint probability distribution over c cardinality hyperedges is more informative (lower entropy) than the sum total of information attained from probability distributions over each of the $\binom{c}{2}$ dyadic edges. $\Rightarrow$ Tensors should retain set-level information

Hyperedge2vec using Hypergraph Tensor Decomposition

- Hypergraph $N_g \xrightarrow{\text{extract}} k$-uniform sub-hypergraph or k-graph

Hyperedge2vec using Hypergraph Tensor Decomposition

- Hypergraph $N_g \xrightarrow{\text{extract}}$ k-uniform sub-hypergraph or k-graph
- k-graph $\Rightarrow$ k^{th} order n-dimensional symmetric tensor $\mathcal{A}^k_{\textbf{hyp}} = (a_{p_1,p_2,..,p_k}) \in \mathbb{R}^{[k,n]}$ with $a_{p_1,p_2,..,p_k} = R(g_i)$, where $\{v_{p_1}, v_{p_2}, ..., v_{p_k}\} \in g_i$ and $|g_i| = k, \forall i \in \{1, ..., m\}$.

Hyperedge2vec using Hypergraph Tensor Decomposition

- Hypergraph $N_g \xrightarrow{\text{extract}} k$-uniform sub-hypergraph or k-graph
- k-graph $\Rightarrow k^{th}$ order n-dimensional symmetric tensor $\mathcal{A}^k_{\mathbf{hyp}} = (a_{p_1,p_2,..,p_k}) \in \mathbb{R}^{[k,n]}$ with $a_{p_1,p_2,..,p_k} = R(g_i)$, where $\{v_{p_1}, v_{p_2}, ..., v_{p_k}\} \in g_i$ and $|g_i| = k, \forall i \in \{1, ..., m\}$.
- *Symmetry* $\Rightarrow a_{p_1,p_2,..,p_k}$ is invariant under any permutation of its indices $(p_1, p_2, .., p_k)$.

Hyperedge2vec using Hypergraph Tensor Decomposition

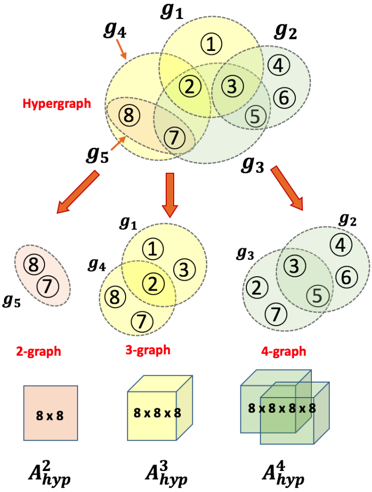
- Hypergraph $N_g \xrightarrow{\text{extract}}$ k-uniform sub-hypergraph or k-graph
- k-graph $\Rightarrow$ k^{th} order n-dimensional symmetric tensor $\mathcal{A}^k_{\mathbf{hyp}} = (a_{p_1,p_2,..,p_k}) \in \mathbb{R}^{[k,n]}$ with $a_{p_1,p_2,..,p_k} = R(g_i)$, where $\{v_{p_1}, v_{p_2}, ..., v_{p_k}\} \in g_i$ and $|g_i| = k, \forall i \in \{1, ..., m\}$.
- *Symmetry* $\Rightarrow$ $a_{p_1,p_2,..,p_k}$ is invariant under any permutation of its indices $(p_1, p_2, .., p_k)$.
- Define the lexicographically ordered index set for hyperedges:

$$\mathcal{P}^k = \{\mathbf{p}|\mathbf{p} = (p_1, p_2, .., p_k) \text{ where } \{v_{p_1}, v_{p_2}, ..., v_{p_k}\} \in g_i, \\ \forall g_i \in G \text{ s.t. } |g_i| = k \text{ and } p_1 < p_2 < ... < p_k\},$$

and we have different sets $\{\mathcal{P}^k\}, \forall k \in \{c_{min}, .., c_{max}\}$ (cardinality range). Also, we have $|\mathcal{P}^k| = |\{g_i : |g_i| = k\}|$.

Example Hypergraph Tensor

- $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$
- $G = \{g_1, g_2, g_3, g_4, g_5\}$
- $\mathcal{A}^2_{\textbf{hyp}}$, $\mathcal{P}^2 = \{(7, 8)\}$
- $\mathcal{A}^3_{\textbf{hyp}}$, $\mathcal{P}^3 = \{(2, 7, 8), (1, 2, 3)\}$
- $\mathcal{A}^4_{\textbf{hyp}}$, $\mathcal{P}^4 = \{(3, 4, 5, 6), (2, 3, 5, 7)\}$



Hyperedge2vec using Hypergraph Tensor Decomposition

- Dual Hypergraph $N_d \xrightarrow{\text{extract}} k$-uniform dual or k-dual

Hyperedge2vec using Hypergraph Tensor Decomposition

- Dual Hypergraph $N_d \xrightarrow{\text{extract}} k$-uniform dual or k-dual
- k-dual $\Rightarrow k^{th}$ order m-dimensional *dual* symmetric tensor $\mathcal{A}^k_{\mathbf{dual}} = (a_{q_1,q_2,..,q_k}) \in \mathbb{R}^{[k,m]}$ with $a_{q_1,q_2,..,q_k} = 1$, where $\{g_{q_1}, g_{q_2}, ..., g_{q_k}\} \ni v_j$ and $d(v_j) = k, \forall j \in \{1, ..., n\}$.

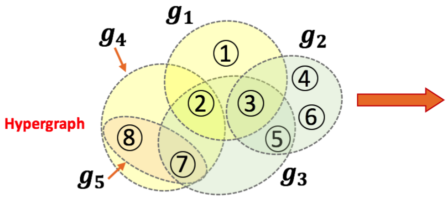
Hyperedge2vec using Hypergraph Tensor Decomposition

- Dual Hypergraph $N_d \xrightarrow{\text{extract}}$ k-uniform dual or k-dual
- k-dual $\Rightarrow$ k^{th} order m-dimensional *dual* symmetric tensor $\mathcal{A}^k_{\textbf{dual}} = (a_{q_1,q_2,..,q_k}) \in \mathbb{R}^{[k,m]}$ with $a_{q_1,q_2,..,q_k} = 1$, where $\{g_{q_1}, g_{q_2}, ..., g_{q_k}\} \ni v_j$ and $d(v_j) = k, \forall j \in \{1, ..., n\}$.
- Lexicographically ordered index set for *dual* hyperedges (vertices in the *original* hypegraph):

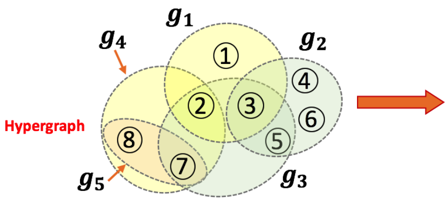
$$\mathcal{Q}^k = \{\mathbf{q}|\mathbf{q} = (q_1, q_2, .., q_k) \text{ where } v_j \in \{g_{q_1}, g_{q_2}, ..., g_{q_k}\}, \\ \forall v_j \in V \text{ s.t. } |d(v_j)| = k \text{ and } q_1 < q_2 < ... < q_k\},$$

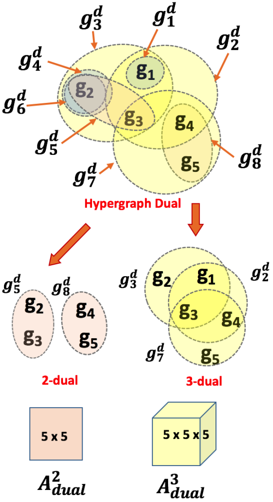
and we have different sets $\{\mathcal{Q}^k\} \forall k \in \{d_{min}, .., d_{max}\}$ (vertex degree range in the *original* hypergraph). We have $|\mathcal{Q}^k| = |\{v_i : d(v_i) = k\}|$

Example Dual Tensor



Example Dual Tensor





- $V_{dual} = \{g_1, g_2, g_3, g_4, g_5\}$
- $G_{dual} = \{g_1^d, g_2^d, g_3^d, g_4^d, g_5^d, g_6^d, g_7^d, g_8^d\}$
- $\mathcal{A}^2_{\mathbf{dual}}$, $\mathcal{P}^2 = \{(2,3), (4,5)\}$
- $\mathcal{A}^3_{\mathbf{dual}}$, $\mathcal{P}^3 = \{(1,2,3), (1,3,4), (3,4,5)\}$

Hyperedge2vec using Hypergraph Tensor Decomposition

- For the hyperedge embeddings we consider the following optimization formulation:

$$f(\lambda, \mathbf{Z}) = \sum_{k=\alpha_1}^{\alpha_2} D_{\mathrm{KL}}\left(\mathcal{M}^k \middle\| \mathcal{A}^k_{\textbf{dual}}\right)$$

where,

$$\mathcal{M}^k = \sum_{r=1}^{d} \lambda_r (\underbrace{\mathbf{z}_r \otimes \mathbf{z}_r \otimes \ldots \otimes \mathbf{z}_r}_{k \text{ times}}) \equiv \sum_{r=1}^{d} \lambda_r \mathbf{z}_r^{\otimes k}$$

with $\otimes$ is the Kronecker product (generalized outer product), $\mathbf{z}_r \in \mathbb{R}^m$, $\lambda_r \in \mathbb{R}$, $\mathbf{Z} \in \mathbb{R}^{m \times d}$ with $\mathbf{Z}(:, r) = \mathbf{z}_r$.

Hyperedge2vec using Hypergraph Tensor Decomposition

- For the hyperedge embeddings we consider the following optimization formulation:

Embeds all cardinality hyperedges

$$f(\lambda, \mathbf{Z}) = \sum_{k=\alpha_1}^{\alpha_2} D_{\mathrm{KL}} \left(\mathcal{M}^k \middle\| \mathcal{A}^k_{\mathbf{dual}} \right)$$

where,

$$\mathcal{M}^k = \sum_{r=1}^{d} \lambda_r (\underbrace{\mathbf{z}_r \otimes \mathbf{z}_r \otimes \ldots \otimes \mathbf{z}_r}_{k \text{ times}}) \equiv \sum_{r=1}^{d} \lambda_r \mathbf{z}_r^{\otimes k}$$

with $\otimes$ is the Kronecker product (generalized outer product), $\mathbf{z}_r \in \mathbb{R}^m$, $\lambda_r \in \mathbb{R}$, $\mathbf{Z} \in \mathbb{R}^{m \times d}$ with $\mathbf{Z}(:, r) = \mathbf{z}_r$.

Hyperedge2vec using Hypergraph Tensor Decomposition

- For the hyperedge embeddings we consider the following optimization formulation:

Embeds all cardinality hyperedges

$$f(\lambda, \mathbf{Z}) = \sum_{k=\alpha_1}^{\alpha_2} D_{\mathrm{KL}}\left(\mathcal{M}^k \middle\| \mathcal{A}^k_{\textbf{dual}}\right)$$

where,

Considers higher-order information, can express HO terms like $z_{r1}z_{r2}z_{r3}$, not just linear mapping

$$\mathcal{M}^k = \sum_{r=1}^{d} \lambda_r (\underbrace{\mathbf{z}_r \otimes \mathbf{z}_r \otimes \ldots \otimes \mathbf{z}_r}_{k \text{ times}}) \equiv \sum_{r=1}^{d} \lambda_r \mathbf{z}_r^{\otimes k}$$

with $\otimes$ is the Kronecker product (generalized outer product), $\mathbf{z}_r \in \mathbb{R}^m$, $\lambda_r \in \mathbb{R}$, $\mathbf{Z} \in \mathbb{R}^{m \times d}$ with $\mathbf{Z}(:, r) = \mathbf{z}_r$.

Hyperedge2vec using Hypergraph Tensor Decomposition

- For the hyperedge embeddings we consider the following optimization formulation:

Embeds all cardinality hyperedges

Employs topology to (1) address cold start issues (2) gluing embeddings from various k-duals

$$f(\lambda, \mathbf{Z}) = \sum_{k=\alpha_1}^{\alpha_2} D_{\mathrm{KL}}\left(\mathcal{M}^k \middle\| \mathcal{A}^k_{\mathbf{dual}}\right) + \eta \sum_{r=1}^{d} \mathbf{z}_r^{\mathsf{T}} \mathbf{L}_{\mathbf{dual}} \mathbf{z}_r$$

where,

Considers higher-order information, can express HO terms like $z_{r1} z_{r2} z_{r3}$, not just linear mapping

$$\mathcal{M}^k = \sum_{r=1}^{d} \lambda_r (\underbrace{\mathbf{z}_r \otimes \mathbf{z}_r \otimes \ldots \otimes \mathbf{z}_r}_{k \text{ times}}) \equiv \sum_{r=1}^{d} \lambda_r \mathbf{z}_r^{\otimes k}$$

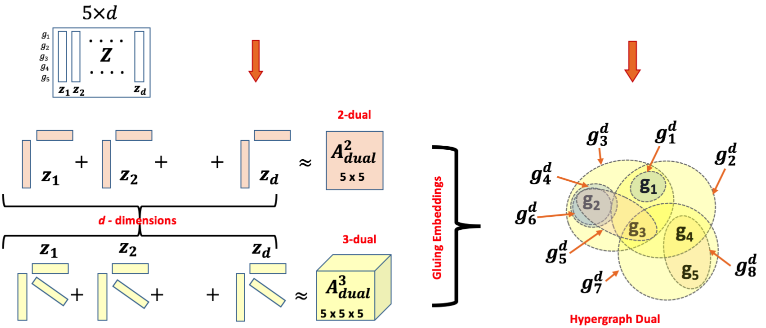
with $\otimes$ is the Kronecker product (generalized outer product), $\mathbf{z}_r \in \mathbb{R}^m$, $\lambda_r \in \mathbb{R}$, $\mathbf{Z} \in \mathbb{R}^{m \times d}$ with $\mathbf{Z}(:, r) = \mathbf{z}_r$.

Example Hypergraph Tensor Decomposition

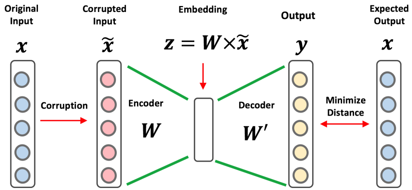
$$f(\lambda,\mathbf{Z}) = \underbrace{D_{\mathrm{KL}}\left(\sum_{r=1}^{d} \lambda_r \mathbf{z}_r^{\otimes 2} \,\middle\|\, \mathcal{A}^2_{\mathbf{dual}}\right) + D_{\mathrm{KL}}\left(\sum_{r=1}^{d} \lambda_r \mathbf{z}_r^{\otimes 3} \,\middle\|\, \mathcal{A}^3_{\mathbf{dual}}\right)} + \eta \underbrace{\sum_{r=1}^{d} \mathbf{z}_r^{\intercal} \mathbf{L}_{\mathbf{dual}} \mathbf{z}_r}$$

Example Hypergraph Tensor Decomposition

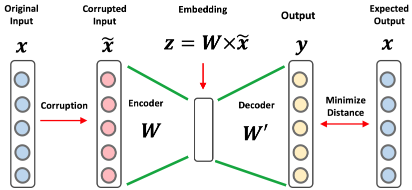
$$f(\lambda,\mathbf{Z}) = \underbrace{D_{\mathrm{KL}}\left(\sum_{r=1}^{d} \lambda_r \mathbf{z}_r^{\otimes 2} \,\middle\|\, \mathcal{A}^2_{\mathbf{dual}}\right) + D_{\mathrm{KL}}\left(\sum_{r=1}^{d} \lambda_r \mathbf{z}_r^{\otimes 3} \,\middle\|\, \mathcal{A}^3_{\mathbf{dual}}\right)} + \underbrace{\eta \sum_{r=1}^{d} \mathbf{z}_r^{\intercal} \mathbf{L}_{\mathbf{dual}} \mathbf{z}_r}$$



Hyperedge2Vec using Hasse De-noising Auto-encoder



Hyperedge2Vec using Hasse De-noising Auto-encoder

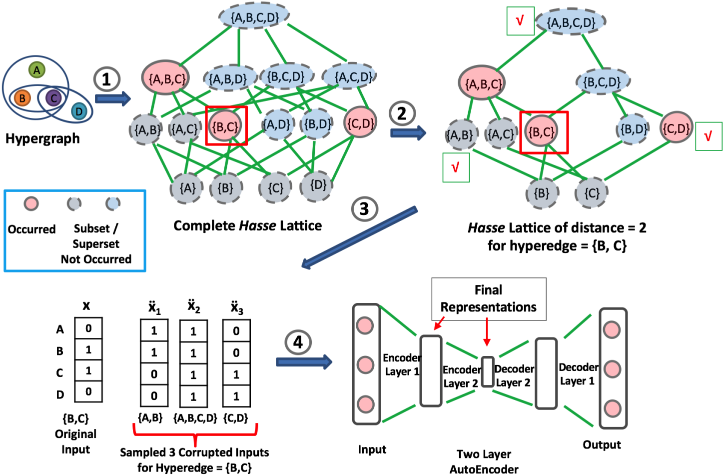


$$\mathbf{W}^*, \mathbf{W}'^* = \underset{\mathbf{W},\mathbf{W}'}{\arg\min} \frac{1}{m} \sum_{i=1}^{m} L(\mathbf{x}_i, \mathbf{y}_i) = \underset{\mathbf{W},\mathbf{W}'}{\arg\min} \frac{1}{m} \sum_{i=1}^{m} L\{\mathbf{x}_i, \sigma(\mathbf{W}'(\sigma(\mathbf{W}\tilde{\mathbf{x}}_i)))\}$$

where $\sigma(x) = 1/(1+e^{-x})$ is sigmoid function, L is the *cross-entropy loss*:

$$L(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{n} [\mathbf{x}(j) \log \mathbf{y}(j) + (1 - \mathbf{x}(j)) \log(1 - \mathbf{y}(j))]$$

Hyperedge2Vec using Hasse De-noising Auto-encoder



Datasets and Baselines

- Two Datasets:
  - *EverQuest II* (**EQ II**): 5964 hyperedges (teams) among 6536 nodes (players)
  - *Stanford Sentiment Treebank* (**LangNet**): 141,410 hyperedges (phrases) and 21,122 nodes (words)
- Two Proposed Methods:
  - *hypergraph tensor decomposition* (**t2v**)
  - *hypergraph auto-encoder* (**h2v-auto**)
- Six Baselines:
  - *Language Embeddings*: (1) **h2v-DM** (2) **h2v-DBOW**
  - *Spectral Embeddings*: (1) **h2v-inv** (2) **h2v-dual**
  - *Graph Embeddings*: (1) **e2v** (2) **e2v-hyp**
- Except for **h2v-auto** all the baselines and **t2v** can generate both vertex as well as hyperedge embeddings.

Performance of Hypergraph Tensor Decomposition

	Baselines						Hypergraph
	Sentence Embed based		Node2Vec based		Spectral methods		Tensor Decomp.
Embed Combination	**h2v-DM**	**h2v-DBOW**	**h2v-inv**	**h2v-dual**	**e2v**	**e2v-hyp**	**(t2v)**
Node Embed Sum	0.79308	0.79567	0.80418	0.79956	0.81183	0.81405	0.81341
Node Embed Sum + Hyperedge Embed	0.79651	0.80241	0.81362	0.80636	0.8113	0.81652	0.81299
Node Embed Average	0.81584	0.81733	0.82407	0.82281	0.81234	0.81369	0.81303
Node Embed Avg + Hyperedge Embed	0.8182	0.82077	0.83378	0.82896	0.81223	0.81608	0.8127
Only Hyperedge Embed	0.81203	0.81522	0.82189	0.81984	0.81233	0.81608	0.81341

RMSE Scores of **(t2v)** compared to baselines for **EQ II** Team Performance Analysis

	Baselines						Hypergraph
	Sentence Embed based		Node2Vec based		Spectral methods		Tensor Decomp.
Embed Combination	**h2v-DM**	**h2v-DBOW**	**h2v-inv**	**h2v-dual**	**e2v**	**e2v-hyp**	**(t2v)**
Node Embed Sum	0.14081	0.14029	N/A	N/A	0.14633	0.14854	0.14194
Node Embed Sum + Hyperedge Embed	0.14028	0.13883	N/A	N/A	0.14627	0.14845	0.14144
Node Embed Average	0.14245	0.14115	N/A	N/A	0.14665	0.14852	0.14381
Node Embed Avg + Hyperedge Embed	0.14178	0.14007	N/A	N/A	0.14661	0.14845	0.14333
Only Hyperedge Embed	0.14194	0.14147	N/A	N/A	0.14744	0.14844	0.1482

RMSE Scores of **(t2v)** compared to baselines for **LangNet** Sentiment Analysis

Performance of Hypergraph Autoencoder and Run-times

	EQ II			LangNet		
Layer Sizes	**L1:128**	**L1:96/L2:32**	**L1:512/L2:128**	**L1:128**	**L1:96/L2:32**	**L1:512/L2:128**
RMSE	0.81104	0.81512	0.81635	0.14568	0.14529	0.14784
Run Time	52 min	40 min	1 hr 20 min	2 hr 10 min	3 hr 20 min	6 hr

RMSE Scores & Run-times of (**h2v-auto**)

	Baselines						Hypergraph
	Sentence Embed based		Node2Vec based		Spectral methods		Tensor Decomp.
Dataset	**h2v-DM**	**h2v-DBOW**	**h2v-inv**	**h2v-dual**	**e2v**	**e2v-hyp**	**(t2v)**
EQ2	455.84	103.47	90.05	93.41	128.03	12.01	213.37
LangNet	80.61	62.31	211.97*	207.86*	221.46	47.12	483.81

* these are average time taken for learning vertex embeddings only

Average Runtime (seconds) of **(t2v)** compared to baselines across datasets

Choice of Method

	Baselines						Proposed	
	Language Embed		Graph Embed		Spectral Embed		Tensor Embed	Auto-encoder Embed
Property	**h2v-DM**	**h2v-DBOW**	**h2v-inv**	**h2v-dual**	**e2v**	**e2v-hyp**	**t2v**	**h2v-auto**
Interpret-ability	NO	YES	YES	YES	YES	YES	YES	NO
Information Loss	NO	YES	YES	YES	YES	YES	NO	NO
Use Hyp. Topology	NO	NO	YES	YES	NO	YES	YES	YES

Comparing methods

Conclusion

- Propose two hyperedge embedding methods designed specifically for hypergraph data
- Proposed methods embed general hypergraphs, unlike uniform hypergraph which have been the focus in past research
- Introduce the idea of *dual tensors*
- Propose a novel idea of joint decomposition of hypergraph tensors across cardinalities
- Introduce the use of auto-encoder in context of hypergraphs
- Highlight: Leverage the existing structure present in network data as the auxiliary contextual information

Acknowledgements

This work has been supported in part by the NSF Award IIS-1422802.

References I

Agarwal, S., Branson, K., and Belongie, S. (2006).
Higher order learning with graphs.
In *Proceedings of the 23rd international conference on Machine learning*, pages 17–24. ACM.

Bengio, Y. and Bengio, S. (2000).
Modeling high-dimensional discrete data with multi-layer neural networks.
In *Advances in Neural Information Processing Systems*, pages 400–406.

Bretto, A. (2013).
Hypergraph theory.
An introduction. Mathematical Engineering. Cham: Springer.

References II

Bulò, S. R. and Pelillo, M. (2009).
A game-theoretic approach to hypergraph clustering.
In *Advances in neural information processing systems*, pages 1571–1579.

Cai, H., Zheng, V. W., and Chang, K. C.-C. (2017).
A comprehensive survey of graph embedding: Problems, techniques and applications.
arXiv preprint arXiv:1709.07604.

Chung, F. R., Graham, R. L., Frankl, P., and Shearer, J. B. (1986).
Some intersection theorems for ordered sets and graphs.
Journal of Combinatorial Theory, Series A, 43(1):23–37.

References III

Ghoshdastidar, D. and Dukkipati, A. (2017).
Uniform hypergraph partitioning: Provable tensor methods and sampling techniques.
The Journal of Machine Learning Research, 18(1):1638–1678.

Grover, A. and Leskovec, J. (2016).
node2vec: Scalable feature learning for networks.
In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 855–864.

References IV

Hwang, T., Tian, Z., Kuangy, R., and Kocher, J.-P. (2008).
Learning on weighted hypergraphs to integrate protein interactions and gene expressions for cancer outcome prediction.
In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 293–302. IEEE.

Le, Q. V. and Mikolov, T. (2014).
Distributed representations of sentences and documents.
In *ICML*, volume 14, pages 1188–1196.

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

References V

- Rezatofighi, S. H., Milan, A., Abbasnejad, E., Dick, A., Reid, I., et al. (2016).
  Deepsetnet: Predicting sets with deep neural networks.
  arXiv preprint arXiv:1611.08998.

- Shashua, A., Zass, R., and Hazan, T. (2006).
  Multi-way clustering using super-symmetric non-negative tensor factorization.
  Computer Vision–ECCV 2006, pages 595–608.

- Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. (2015).
  Line: Large-scale information network embedding.
  In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. ACM.

References VI

Vinyals, O., Bengio, S., and Kudlur, M. (2016).
Order matters: Sequence to sequence for sets.
In *ICLR*.

Zhou, D., Huang, J., and Schölkopf, B. (2006).
Learning with hypergraphs: Clustering, classification, and embedding.
In *Advances in neural information processing systems*, pages 1601–1608.