NetVec: A Scalable Hypergraph Embedding System

Sepideh Maleki¹, Donya Sales⁵, Dennis P Wall³, and Keshav Pingali¹

¹The University of Texas at Austin, ²University of Tehran, ³Stanford University

Motivation: Prediction tasks on hypergraphs.
- Hypergraphs arise in many application domains.
- Many important problems can be formulated as classification and prediction problems for hyperedges.

Our work:
- NetVec, a novel hierarchical framework for scalable, unsupervised embeddings of hyperedges and nodes.
- NetVec can be coupled with any existing network embedding algorithm to reduce the amount of compute time.

Background

Previous Approaches vs. NetVec

- Hyperedges are not represented explicitly
- Takes days to find embedding of hypergraphs with millions of nodes and hyperedges
- Hyperedges are represented explicitly
- Finds embedding of hypergraphs with millions of nodes and hyperedges in just a few minutes

A hypergraph with 3 hyperedges and 6 nodes.

(i) At each level of coarsening, AssignHyperedge assigns each node v in the current hypergraph to a coarser level i as defined below where \( w(e) \) is the weight of hyperedge e and \( \delta_i(v) \) is the weight of node v in hyperedge e.

\[
\delta_i(v) = \arg \max_{e \in E} \omega(e) \cdot \delta_i(v)
\]

(ii) NetVec coarsens the hypergraph until it is small enough that any unsupervised embedding method can generate the embedding of the coarsest graph in just a few seconds.

(iii) The goal of refinement is to improve embedding obtained from the initial embedding algorithm. If a set of node S in a hypergraph \( H_n \) were merged to form a node \( n \) in the coarser hypergraph \( H_{n-1} \), the embedding of \( n \) in \( H_{n-1} \) is assigned to all the nodes of \( S \) in \( H_n \), at the beginning of refinement.

Algorithm 2: Refinement

Input: bipartite graph representation \( H = (V, E, W) \) of hypergraph \( H = (V, E, W) \), vector representation \( u \), for all \( v \in V \). neighborhood function \( N(u), \) depth \( K \), parameter \( \omega \).

Output: final vector representation \( u_n \) of \( V \).

Algorithm 2: Edge Coarsening

\[
\begin{align*}
\text{for } n & = 1 \text{ to } N \text{ do} \\
\text{for } e & = 1 \text{ to } K \text{ do} \\
& \text{AssignHyperedge}(v) \\
& \text{end for} \\
& \text{end for} \\
\end{align*}
\]

Abstractly, the refinement algorithm uses Jacobi over-relaxation to solve the linear system, \( Lz = 0 \) using the relaxation parameter \( \omega \), where \( L \) is the Laplacian matrix of \( H \). It is defined as \( D - W \) where \( D \) is the diagonal matrix with diagonal elements \( d_i = \sum_{j} W_{ij} \) is the weighted adjacency matrix of \( H \).

Algorithm 3: Node Coarsening

Input: fineGraph \( G = (V, E, W) \), neighborhood function \( N(u), \) depth \( K \)

Output: coarseGraph \( G' = (V', E', W') \)

\[
\begin{align*}
\text{for } c & = 1 \text{ to } K \text{ do} \\
& \text{AssignHyperedge}(v) \\
& \text{end for} \\
& \text{end for} \\
\end{align*}
\]

For higher classification prediction task, we compare NetVec with Hyper-SAGNN(1), the state-of-the-art supervised hyperedge prediction framework and Node2Vec on 4 standard datasets listed below.

Results (hyperedge classification)

<table>
<thead>
<tr>
<th>Data set</th>
<th>NODES</th>
<th>HYPEREDGES</th>
<th>EDGES</th>
<th>CLASSES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>2,708</td>
<td>1,985</td>
<td>10,335</td>
<td>7</td>
</tr>
<tr>
<td>Citeseer</td>
<td>3,328</td>
<td>2,143</td>
<td>9,353</td>
<td>6</td>
</tr>
<tr>
<td>Cora</td>
<td>3,328</td>
<td>2,143</td>
<td>9,353</td>
<td>6</td>
</tr>
<tr>
<td>Cora</td>
<td>3,328</td>
<td>2,143</td>
<td>9,353</td>
<td>6</td>
</tr>
</tbody>
</table>

For hyperedge classification, we compare NetVec with Hyper-SAGNN(1), the state-of-the-art supervised hyperedge prediction framework and Node2Vec on 4 standard datasets listed below.

Results (hyperedge classification)

<table>
<thead>
<tr>
<th>Data set</th>
<th>NODES</th>
<th>HYPEREDGES</th>
<th>EDGES</th>
<th>CLASSES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>2,708</td>
<td>1,985</td>
<td>10,335</td>
<td>7</td>
</tr>
<tr>
<td>Citeseer</td>
<td>3,328</td>
<td>2,143</td>
<td>9,353</td>
<td>6</td>
</tr>
<tr>
<td>Cora</td>
<td>3,328</td>
<td>2,143</td>
<td>9,353</td>
<td>6</td>
</tr>
<tr>
<td>Cora</td>
<td>3,328</td>
<td>2,143</td>
<td>9,353</td>
<td>6</td>
</tr>
</tbody>
</table>

For hyperedge classification, we compare NetVec with Hyper-SAGNN(1), the state-of-the-art supervised hyperedge prediction framework and Node2Vec on 4 standard datasets listed below.

References


Results (hyperedge classification)

<table>
<thead>
<tr>
<th>Data set</th>
<th>NODES</th>
<th>HYPEREDGES</th>
<th>EDGES</th>
<th>CLASSES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>2,708</td>
<td>1,985</td>
<td>10,335</td>
<td>7</td>
</tr>
<tr>
<td>Citeseer</td>
<td>3,328</td>
<td>2,143</td>
<td>9,353</td>
<td>6</td>
</tr>
<tr>
<td>Cora</td>
<td>3,328</td>
<td>2,143</td>
<td>9,353</td>
<td>6</td>
</tr>
<tr>
<td>Cora</td>
<td>3,328</td>
<td>2,143</td>
<td>9,353</td>
<td>6</td>
</tr>
</tbody>
</table>

For hyperedge classification, we compare NetVec with Hyper-SAGNN(1), the state-of-the-art supervised hyperedge prediction framework and Node2Vec on 4 standard datasets listed below.

Contact

Sepideh Maleki: smaleki@cs.utexas.edu