Scaling Attributed Network Embedding to Massive Graphs

Basic data analytics is easy.

<table>
<thead>
<tr>
<th>Stock</th>
<th>Profit</th>
<th>Revenue</th>
<th>Market share</th>
<th>Overvalued?</th>
<th>Buy?</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSLA</td>
<td>$721m</td>
<td>$31B</td>
<td>3.6%</td>
<td>Yes</td>
<td>???</td>
</tr>
</tbody>
</table>

Attributes:
- $X_1$
- $X_2$
- $X_3$
- $X_4$

Target attribute: $Y$
Graph data analytics is more powerful.

<table>
<thead>
<tr>
<th>Stock</th>
<th>Profit</th>
<th>Revenue</th>
<th>Market share</th>
<th>Overvalued?</th>
<th>Buy?</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSLA</td>
<td>$721m</td>
<td>$31B</td>
<td>3.6%</td>
<td>Yes</td>
<td>???</td>
</tr>
</tbody>
</table>
Graph data analytics is powerful but difficult.

<table>
<thead>
<tr>
<th></th>
<th>Single table data analytics</th>
<th>Attributed graph data analytics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tools</strong></td>
<td><img src="xgboost.png" alt="XGBoost" /> + <a href="https://www.google.com">google</a></td>
<td><img src="deep-graph-neural-network.png" alt="Deep graph neural network" /></td>
</tr>
<tr>
<td><strong>Difficulty</strong></td>
<td>★★</td>
<td>★★★★★★</td>
</tr>
<tr>
<td><strong>Req. Skill level</strong></td>
<td><img src="skill-level.png" alt="Skill level" /></td>
<td><img src="skill-level.png" alt="Skill level" /></td>
</tr>
</tbody>
</table>
We present Practical Attributed Network Embedding (PANE).

Embedding vector

### Attributed graph data analytics

Embedding vector data analytics
Performance of PANE

**Effective**

Accuracy (F1): up to \(+17.2\%\)

Compared to SOTA Neural Network methods

**Efficient**

Single-server: \(~12\) hours

Computing all embeddings on a HUGE graph with 59m nodes, 0.98b edges, 2k attributes
Applications of PANE

Link Prediction

Attribute Inference

Node Classification
PANE is based on mostly novel database technologies (with a bit of machine learning flavor).

1. PANE measures Node-Attribute affinity via random walks.

2. PANE computes embeddings with joint matrix factorization.

3. PANE makes full use of multi-core parallel computation.
Types of Random Walks in PANE

Forward: node-to-attribute

Backward: attribute-to-node
Forward Random Walks

- Forward random walk from node $u$:
  - Start from $u$
  - At each step, stop with $\alpha$ probability
  - After stopping at a node $v$, pick an attribute $r$ with probability $\propto w(v, r)$

- Intuition: it samples an attribute $r$ from the vicinity of $u$
Node-Attribute Affinity

Node-attribute affinity:

$$F[u, r] = \text{normalized probability that a forward random walk from } u \text{ samples } r \text{ in the end}$$
Backward random walk from attribute \( r \)
- Randomly pick a node \( s \) with probability proportional to the weight of \((s, r)\)
- Start a random walk from \( s \)
- At each step, stop with \( \alpha \) probability
- Let \( v \) be the stopping point of the walk

Attribute-node affinity
\[
B[r, v] \leftarrow \text{normalized random walk probability from attribute } r \text{ to node } v
\]
Node-to-Node affinity is derived.

\[ p(u, v) = \sum_{r \in R} F[u, r] \cdot B[r, v] \]

This saves a LOT of space: \( O(n^2) \rightarrow O(nd), d \ll n \)
Embedding Matrices in PANE

- We construct
  - two embedding matrices $X_f$ and $X_b$ for the nodes, and
  - one embedding matrix $Y$ for attributes

- Optimization objective:
  - $X_f \cdot Y^T \approx F$, to capture node-attribute affinity
  - $Y \cdot X_b^T \approx B$, to capture attribute-node affinity
Solving the optimization program

- Jointly factorize $F$ and $B$ to obtain $X_f$, $X_b$, and $Y$
  - Formulate the joint factorization as a least square problem
  - Solve it using gradient descent
  - Use randomized SVD to obtain a good initial solution

- Time complexity: $O(mdt + ndkt)$
  - $k$ is the embedding size
  - $t$ is the number of iterations
    ($t = 5$ in our experiments)
Greedy Initialization + SGD

\[
F \approx U \cdot \Sigma \cdot V^T
\]

- \(X_f = U \cdot \Sigma, Y = V\)
- \(V = Y\) is unitary
- \(Y^T \cdot Y = I\)
- \(X_b = X_b Y^T, Y = B \cdot Y\)

Greedy initialization of embeddings via randomized SVD and the unitary property

For \(t\) iterations:
- Update \(X_f, X_b\) via SGD;
- Update \(Y\) via SGD;

\[
F \approx X_f \cdot Y
\]

\[
B \approx X_b \cdot Y
\]

Only a few iterations are needed!
PANE is fully parallelized on multi-core computers.

Explained in Section 4 of our paper.
## Experiments: 8 Datasets

<table>
<thead>
<tr>
<th>Name</th>
<th># of nodes</th>
<th># of edges</th>
<th># of distinct attributes</th>
<th># of attributes per node</th>
<th># of distinct labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>2.7k</td>
<td>5.4k</td>
<td>1.4k</td>
<td>18.2</td>
<td>7</td>
</tr>
<tr>
<td>Citeseer</td>
<td>3.3k</td>
<td>4.7k</td>
<td>3.7k</td>
<td>31.9</td>
<td>6</td>
</tr>
<tr>
<td>Facebook</td>
<td>4k</td>
<td>88.2k</td>
<td>1.3k</td>
<td>8.3</td>
<td>193</td>
</tr>
<tr>
<td>Pubmed</td>
<td>19.7k</td>
<td>44.3k</td>
<td>0.5k</td>
<td>50.2</td>
<td>3</td>
</tr>
<tr>
<td>Flickr</td>
<td>7.6k</td>
<td>479.5k</td>
<td>12.1k</td>
<td>24.0</td>
<td>9</td>
</tr>
<tr>
<td>Google+</td>
<td>107.6k</td>
<td>13.7M</td>
<td>15.9k</td>
<td>2793.7</td>
<td>468</td>
</tr>
<tr>
<td>TWeibo</td>
<td>2.3M</td>
<td>50.7M</td>
<td>1.7k</td>
<td>7.3</td>
<td>8</td>
</tr>
<tr>
<td>MAG</td>
<td>59.3M</td>
<td>978.2M</td>
<td>2.0k</td>
<td>7.3</td>
<td>100</td>
</tr>
</tbody>
</table>
Experiments: 10 Competitors

- Default embedding dimensionality: $k = 128$

<table>
<thead>
<tr>
<th>6 neural-network-based methods</th>
<th>3 factorization-based methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>STNE [KDD 2018]</td>
<td>TADW [IJCAI 2015]</td>
</tr>
<tr>
<td>ARGA [IJCAI 2018]</td>
<td>BANE [ICDM 2018]</td>
</tr>
<tr>
<td>LQANR [IJCAI 2019]</td>
<td>NRP [VLDB 2020]</td>
</tr>
<tr>
<td>CAN [WSDM 2019]</td>
<td></td>
</tr>
<tr>
<td>DGI [ICLR 2019]</td>
<td></td>
</tr>
<tr>
<td>GATNE [KDD 2019]</td>
<td></td>
</tr>
</tbody>
</table>

- 1 other method
  - PRRE [CIKM 2018]
Results: Node Classification

- Percentage of nodes used for training: 10% ~ 90%
- PANE vs. SOTA: improvements of 3.4%-17.2% in terms of F1 measure
THANK YOU

Random walks

Joint matrix factorization

Parallelization

Best Paper Award

PANE
Scaling Attributed Network Embedding to Massive Graphs


Code: https://github.com/AnryYang/PANE