Scaling Attributed Network Embedding to Massive Graphs

by: R. Yang, J. Shi, X. Xiao, Y. Yang, J. Liu, and S. Bhowmick
Basic data analytics is easy.

| Stock | Profit | Revenue | Market share | Overvalued? | Buy? |
|-------|--------|---------|--------------|-------------|------|
| TSLA  | $721m  | $31B    | 3.6%         | Yes         | ???  |

Target attribute
Graph data analytics is more powerful.

| Stock | Profit | Revenue | Market share | Overvalued? | Buy? |
|-------|--------|---------|--------------|-------------|------|
| TSLA  | $721m  | $31B    | 3.6%         | Yes         | ???  |
Graph data analytics is powerful but difficult.

| Tools          | Single table data analytics | Attributed graph data analytics |
|----------------|----------------------------|---------------------------------|
| **Tools**      | **dmlearn** + **XGBoost**   | Deep graph neural network       |
| **Difficulty** | 🌟 🌟                     | 🌟 🌟 🌟 🌟 🌟 🌟                   |
| **Req. Skill level** | ![Image](image1.png) | ![Image](image2.png) |
We present Practical Attributed Network Embedding (PANE).
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

attribute $r$ → $u$ → node → attribute $r$ → node → $v$

Backward: attribute-to-node

attribute $r$ → $v$ → node → attribute $r$ → $u$ → node → attribute $r$
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**

$B[r, v] \leftarrow$ normalized random walk probability from attribute $r$ 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: \( O(n^2) \to O(nd), d \ll n \)
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
- $$Y^T \cdot Y = I$$
- $$X_b = X_b \cdot Y^T \cdot Y = B \cdot Y$$

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

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

- Default embedding dimensionality: $k = 128$

| 6 neural-network-based methods | 3 factorization-based methods | 1 other method |
|-------------------------------|-------------------------------|----------------|
| STNE [KDD 2018]               | TADW [IJCAI 2015]             | PRRE [CIKM 2018] |
| ARGA [IJCAI 2018]             | BANE [ICDM 2018]              |                |
| LQANR [IJCAI 2019]            | NRP [VLDB 2020]               |                |
| CAN [WSDM 2019]               |                               |                |
| DGI [ICLR 2019]               |                               |                |
| GATNE [KDD 2019]              |                               |                |
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

1. Random walks
2. Joint matrix factorization
3. Parallelization
Scaling Attributed Network Embedding to Massive Graphs

by: R. Yang, J. Shi, X. Xiao, Y. Yang, J. Liu, and S. Bhowmick

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