Link Prediction on the Patent Citation Network

Samuel Chen¹  David Dang¹  Robert Macy²  Chris Rockwell²
¹ Department of Aerospace Engineering  ² Electrical Engineering and Computer Science

Problem Definition
- Link prediction (LP) is cast as a binary classification problem
  - Does a link exist between any two pair of nodes?
- INPUT: Adjacency list
- OUTPUT: Ranked list of most likely edges

Motivation
- Generate citation recommendations on new patents
- LP on the Patent Citation Dataset is unprecedented
  - LP on large and temporal graphs is less well studied
- Compare the performance of SDNE to other classical network embedding methods on different graph types

Datasets
Patent Citation Dataset
- 1988-1989 subgraph: 40K nodes, 30K edges
- 1990-1996 subgraph: 580K nodes, 1.2M edges
  - Yielded poor results because of size
- "Future" test set: 500 proceeding nodes with 2 or more outward edges, inward edges removed

Blog Catalog: 10K nodes, 30K edges
- Undirected graph

Our Approach

Data Partitioning
- Subgraphs are divided into training and test sets
- 15% of the links in the sub-graph are hidden from the training set

Link Prediction from Node Embeddings
- Methods embed similar nodes close together
- Use node embeddings to predict hidden links

Approach 1: SDNE
Nodes embedded by optimizing weighted loss from:
- Supervised "first-order proximity"
  - Influenced by Laplacian Eigenmaps
- Captures pairwise similarities i.e. common neighbors
- Unsupervised autoencoder "second-order proximity"
  - Mimics Graph Convolutional Network
  - Captures global structure i.e. role in network
  - High unsupervised weight should help in sparse graphs

Approach 2: node2vec
- Focused on learning low-dimensional representation learning
- Node embeddings generated through random walk approach
- Random walks can be biased
  - Uses unbiased in this work

Approach 3: Jaccard Coefficient
- Predicts whether two nodes share a relationship, in this case a link, based on the intersection over union of their neighborhoods
- No learning involved, but very expensive at inference time

Approach 4: PyTorch BigGraph
- Learns node embedding by minimizing distance between adjacent nodes
- Gains efficiency by augmenting negative sampling with uniform samples used as negatives

Experimental Results

Data Catalog

Link Prediction from Node Embeddings

Approach 2: node2vec

SDNE
- Much more computationally efficient than SDNE
- Performs better than SDNE, particularly on sparse graphs

node2vec: Prediction on future is better than baseline

Approach 3: Jaccard Coefficient

1. Lowest asymptotic complexity

Approach 4: PyTorch BigGraph

1. Performance increases with k
2. Highly scalable

Conclusions
- LP can work well on temporal graphs. However, sparse graphs makes LP difficult
- RW methods do disproportionately better than nn in this situation
- Scalability is challenging for all methods studied, Pytorch handles best

Future directions: link prediction on more dense temporal graphs

References
1. Grover and Leskovec. node2vec: Scalable Feature Learning for Networks. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2016.