Machine Learning on graphs. Link Prediction

I. Makarov & L.E. Zhukov

BigData Academy MADE from Mail.ru Group

Network Science

Lecture outline

1. Link Prediction
   - Similarity-based
   - Matrix Factorization
   - Random walks
   - Other approaches and challenges
2. Graph Embeddings
   - Problem statement
   - Structural graph embeddings (simple models)

Graph machine learning

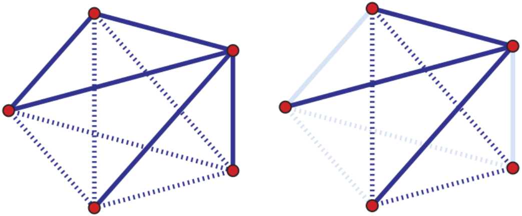
- Node classification (attribute inference)
- Link prediction (missing/hidden links inference)
- Community detection (clustering nodes in graph)
- Graph visualization (cluster projections)

Link prediction

- **Link prediction**. A network is changing over time. Given a snapshot of a network at time t, predict edges added in the interval (t, t')
- **Link completion** (missing links identification). Given a network, infer links that are consistent with the structure, but missing (find unobserved edges)
- **Link reliability**. Estimate the reliability of given links in the graph.

- Predictions: link existence, link weight, link type

Link prediction



- Graph G(V,E)
- Number of "missing edges": $|V|(|V|-1)/2-|E|$
- In sparse graphs $|E| \ll |V|^2$, Prob. of correct random guess $O(\frac{1}{|V|^2})$

Similarity based algorithms - unsupervised

Link prediction by proximity scoring

1. For each pair of nodes compute proximity (similarity) score $c(v_1, v_2)$
2. Sort all pairs by the decreasing score
3. Select top n pairs (or above some threshold) as new links
4. Quality measurements - precision $TP/(TP + FP)$, precision at top N

Local similarity indices

Local neighborhood of v_i and v_j

- Number of common neighbors:

$$s_{ij} = |\mathcal{N}(v_i) \cap \mathcal{N}(v_j)|$$

- Jaccard's coefficient:

$$s_{ij} = \frac{|\mathcal{N}(v_i) \cap \mathcal{N}(v_j)|}{|\mathcal{N}(v_i) \cup \mathcal{N}(v_j)|}$$

- Resource allocation:

$$s_{ij} = \sum_{w \in \mathcal{N}(v_i) \cap \mathcal{N}(v_j)} \frac{1}{|\mathcal{N}(w)|}$$

Adamic/Adar:

$$s_{ij} = \sum_{w \in \mathcal{N}(v_i) \cap \mathcal{N}(v_j)} \frac{1}{\log |\mathcal{N}(w)|}$$

Liben-Nowell and Kleinberg, 2003

Local similarity indices

- Preferential attachment:

$$s_{ij} = k_i \cdot k_j = |\mathcal{N}(v_i)| \cdot |\mathcal{N}(v_j)|$$

or

$$s_{ij} = k_i + k_j = |\mathcal{N}(v_i)| + |\mathcal{N}(v_j)|$$

- Clustering coefficient:

$$s_{ij} = CC(v_i) \cdot CC(v_j)$$

or

$$s_{ij} = CC(v_i) + CC(v_j)$$

Quasi-Local similarity indices

- Local Path Index:

$$s_{lp} = A^2 + \alpha A^3$$

- High-order LPI:

$$s_{lp(n)} = \sum_{i=2}^{n} \alpha^{i-2} A^i$$

or

$$s_{ij} = CC(v_i) + CC(v_j)$$

Path based methods

Paths and ensembles of paths between v_i and v_j

- Shortest path:

$$s_{ij} = -\min_s \{path^s_{ij} > 0\}$$

- Katz score:

$$s_{ij} = \sum_{s=1}^{\infty} \beta^s |paths^{(s)}(v_i, v_j)| = \sum_{s=1}^{\infty} (\beta A)^s_{ij} = (I - \beta A)^{-1} - I$$

- Personalized (rooted) PageRank:

$$PR = \alpha (D^{-1}A)^T PR + (1-\alpha) \cdot (e_i + e_j)$$

Liben-Nowell and Kleinberg, 2003

Path based indeces

- Expected number of random walk steps:
  hitting time: $s_{ij} = -H_{ij}$
  commute time $s_{ij} = -(H_{ij} + H_{ji})$
  normalized hitting/commute time $s_{ij} = -(H_{ij}\pi_j + H_{ji}\pi_i)$

- SimRank:

$$SimRank(v_i, v_j) = \frac{C}{|\mathcal{N}(v_i)| \cdot |\mathcal{N}(v_j)|} \sum_{m \in \mathcal{N}(v_i)} \sum_{n \in \mathcal{N}(v_j)} SimRank(m, n)$$

Liben-Nowell and Kleinberg, 2003

Community based methods

- Within-inter community/cluster of $v_i, v_j \in C$

$$\sum_{w \in \mathcal{N}(v_i) \cap \mathcal{N}(v_j)} \frac{|w \in C|}{|w \notin C|}$$

- Common neighbors with community information, $v_i, v_j \in C$, $f(w) = 1$ if $w \in C$

$$|\mathcal{N}(v_i) \cap \mathcal{N}(v_j)| + \sum_{w \in \mathcal{N}(v_i) \cap \mathcal{N}(v_j)} f(w)$$

- Resource allocation index with community information (soundarajan-hopcroft), $v_i, v_j \in C$, $f(w) = 1$ if $w \in C$

$$\sum_{w \in \mathcal{N}(v_i) \cap \mathcal{N}(v_j)} \frac{f(w)}{|\mathcal{N}(w)|}$$

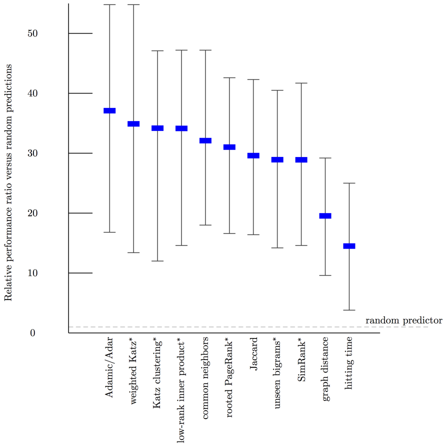
Low-rank approximations

- Low-rank approximation (truncated SVD)

$$A = \sum_{k}^{n} U_k S_k V_k^T \rightarrow \sum_{k}^{r} U_k S_k V_k^T = A', r < n$$

$$\underset{m \times n}{\overset{\hat{X}}{\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix}}} \approx \underset{m \times r}{\overset{U}{\begin{pmatrix} u_{11} & \cdots & u_{1r} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{pmatrix}}} \underset{r \times r}{\overset{S}{\begin{pmatrix} s_{11} & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{pmatrix}}} \underset{r \times n}{\overset{V^{\mathsf{T}}}{\begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix}}}$$

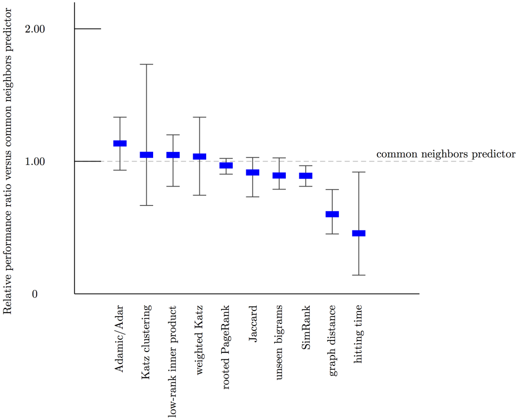
Evaluation of scoring prediction



Ratio of predictor performance over the baseline, averaged 5 datasets

Liben-Nowell and Kleinberg, 2007

Evaluation of scoring prediction



Ratio of predictor performance over the baseline, averaged 5 datasets

Liben-Nowell and Kleinberg, 2007

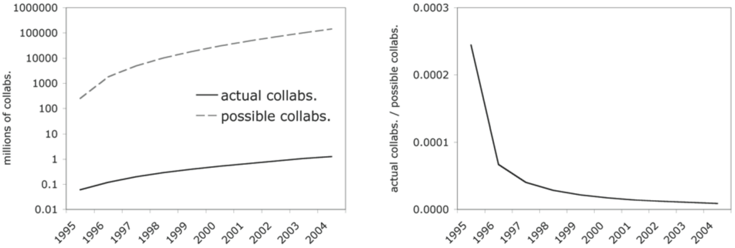
Classification for link prediction

Challenging classification problem:

- Computational cost of evaluating of very large number of possible edges (quadratic in number of nodes)
- Highly imbalanced class distribution: number of positive examples (existing edges) grows linearly and negative quadratically with number on nodes

Prediction difficulty

Actual and possible collaborations between DBLP authors



Extreme class imbalance

from Rattigan and Jensen, 2005

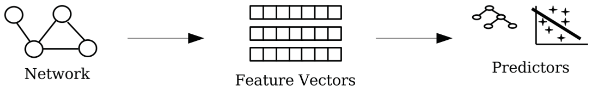
Link prediction with supervised learning

Supervised learning:

1. Features generation
2. Model training
3. Testing (model application)

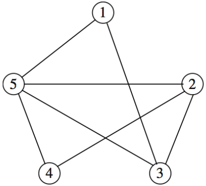
Features:

- Topological proximity features
- Aggregated features
- Content based node proximity features

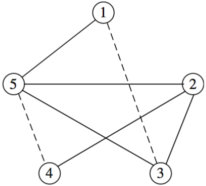


Simple evaluation

Simple "hold out set" evaluation



Whole graph



Training graph

Evaluation metrics

- Precision and Recall, F-measure

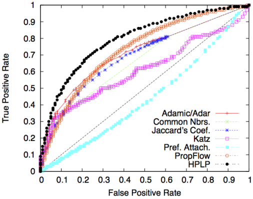
$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN}$$

$$F = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

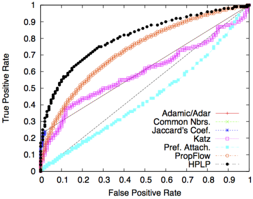
- True positive rate (TPR), False positive rate (FPR), ROC curve, AUC

$$TPR = \frac{TP}{TP + FN}, \quad FPR = \frac{FP}{FP + TN}$$

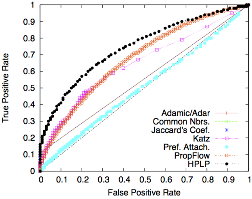
ROC curves



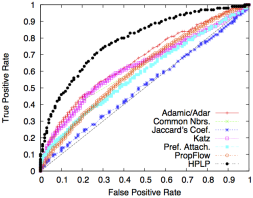
(a) phone $n = 2$



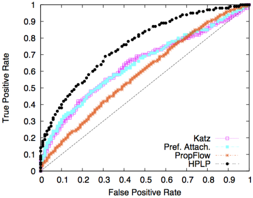
(b) phone $n = 3$



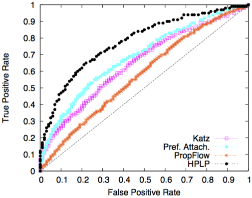
(c) phone $n = 4$



(d) condmat $n = 2$



(e) condmat $n = 3$



(f) condmat $n = 4$

from Lichtenwalter, 2010

Training and testing

Evaluation for evolving networks

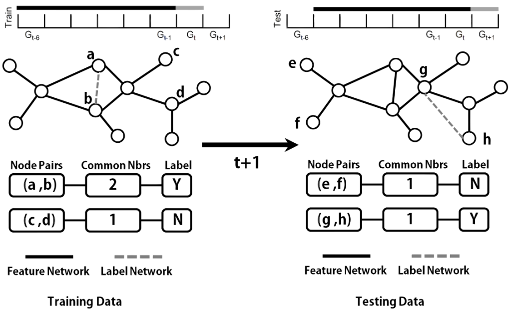


image from Y. Yang et.al, 2014

Probabilistic models

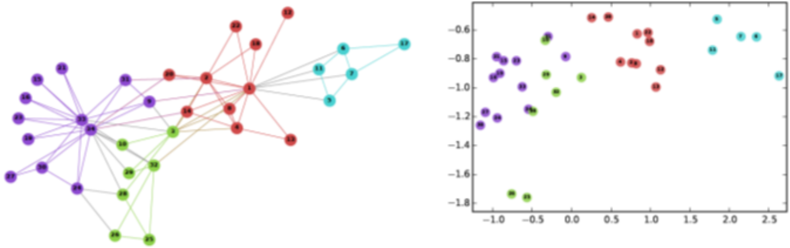
- Local model, Markov random fields [Wang, 2007]
- Hierarchical probabilistic model [Clauset, 2008]
- Probabilistic relations models:
  – Bayesian networks [Getoor, 2002]
  – relational Markov networks [Tasker, 2003, 2007]

Lecture outline

1. Link Prediction
   - Similarity-based
   - Matrix Factorization
   - Random walks
   - Other approaches and challenges
2. Graph Embeddings
   - Problem statement
   - Structural graph embeddings (simple models)

Graph Embeddings

- Necessity to automatically select features
- Reduce domain- and task- specific bias
- Unified framework to vectorize network
- Preserve graph properties in vector space
- Similar nodes $\rightarrow$ close embeddings

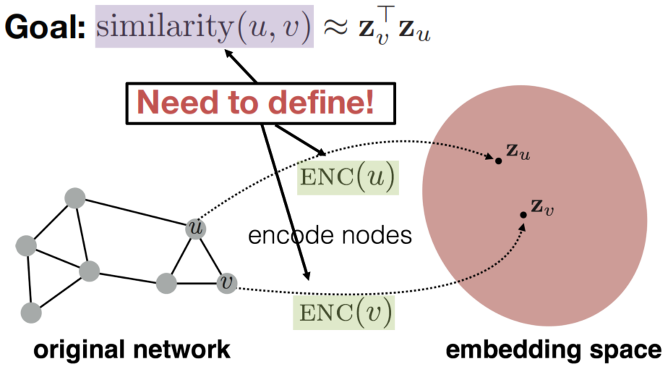


from Lescovec et al., 2018[1]

[1] `http://snap.stanford.edu/proj/embeddings-www/`

Graph Embeddings

- Define **Encoder**
- Define **Similarity**/graph feature to preserve graph properties
- Define similarity/distance in the embedding space
- **Optimize** loss to fit embedding with similarity computed on graph



from Lescovec et al., 2018

Structural Graph Embeddings

- Embedding look-up (each node - separate vector)
- Different similarity measures (adjacency, common neighbours, distances, exact function, etc.)
- Quadratic optimization for MSE loss
- Fast models via random walks

First-order Proximity

- Similarity between u and v is A_{uv}
- MSE Loss
- Variant of Matrix Decomposition

$$\boxed{\mathcal{L}} = \boxed{\sum_{(u,v)\in V\times V}} \|\boxed{\mathbf{z}_u^\top \mathbf{z}_v} - \boxed{\mathbf{A}}_{u,v}\|^2$$

loss (what we want to minimize)

sum over all node pairs

embedding similarity

(weighted) adjacency matrix for the graph

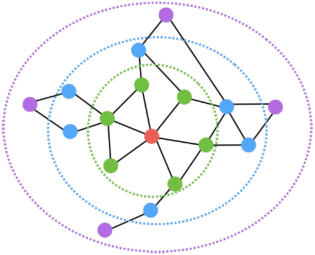
from Lescovec et al., 2018

First-order Proximity

- Pros:
  - Use SGD for scalable optimization
  - Matrix factorization (SVD) or decomposition (QR) may be applicable
- Cons:
  - Quadratic complexity
  - Large embeddings space
  - No indirect graph properties are preserved

Multi-order Proximity

- Similarity of neighborhoods of u and v via indices or k-hop paths
- Direct optimization of exact similarity metric



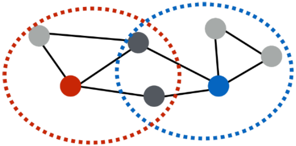
- **Red:** Target node
- **Green**: 1-hop neighbors
  - $\mathbf{A}$ (i.e., adjacency matrix)
- **Blue:** 2-hop neighbors
  - $\mathbf{A}^2$
- **Purple:** 3-hop neighbors
  - $\mathbf{A}^3$

$$\mathcal{L} = \sum_{(u,v)\in V\times V} \|\mathbf{z}_u^\top \mathbf{z}_v - \mathbf{A}_{u,v}^k\|^2$$

from Lescovec et al., 2018

Multi-order Proximity

- Similarity score S_{uv} as Jaccard/Common Neighbours, etc. (HOPE)



$$\mathcal{L} = \sum_{(u,v)\in V\times V} \left\| \boxed{\mathbf{z}_u^\top \mathbf{z}_v} - \boxed{\mathbf{S}_{u,v}} \right\|^2$$

embedding similarity

multi-hop network similarity (i.e., any neighborhood overlap measure)

- Weighted k-hop paths with different k (GraRep)

$$\tilde{\mathbf{A}}^k_{i,j} = \max\left(\log\left(\frac{(\mathbf{A}_{i,j}/d_i)}{\sum_{l\in V}(\mathbf{A}_{l,j}/d_l)^k}\right)^k - \alpha, 0\right)$$

node degree

constant shift

from Lescovec et al., 2018

- Even worse complexity

Random Walks

- Similarity between u and v is probability to co-occur on a random walk
- Sample each vertex u neighborhood $N_R(u)$ (multiset) by short random walks via strategy R
- Optimize similarity considering independent neighbor samples via MLE (remind Word2Vec)

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log(P(v|\mathbf{z}_u))$$

from Lescovec et al., 2018

Random Walks

- $P(v|z_u)$ is approximated via softmax over similarity $z_u^T \cdot z_v$

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log\left(\frac{\exp(\mathbf{z}_u^\top \mathbf{z}_v)}{\sum_{n \in V} \exp(\mathbf{z}_u^\top \mathbf{z}_n)}\right)$$

- Problem in second Σ over all nodes
- Hard to find optimal solution

Negative Sampling

- Use *Negative Sampling* to approximate denominator

$$\log\left(\frac{\exp(\mathbf{z}_u^\top \mathbf{z}_v)}{\sum_{n\in V}\exp(\mathbf{z}_u^\top \mathbf{z}_n)}\right)$$

random distribution over all nodes

$$\approx \log(\sigma(\mathbf{z}_u^\top \mathbf{z}_v)) - \sum_{i=1}^{k} \log(\sigma(\mathbf{z}_u^\top \mathbf{z}_{n_i})), n_i \sim P_V$$

from Lescovec et al., 2018

- Sample in proportion to node degree
- Experiment with k to impact negative prior and robustness
- No need to sample non-connected edges — same as random

Feature representation

- How to construct pair of nodes representation having node embeddings?
- Will it be more efficient than $\sigma(z_i^t \cdot z_j)$

Symmetry operator	Definition
Average	$\frac{f_i(u)+f_i(v)}{2}$
Hadamard	$f_i(u) \cdot f_i(v)$
Weighted-L_1	$\lvert f_i(u) - f_i(v) \rvert$
Weighted-L_2	$(f_i(u) - f_i(v))^2$
Neighbor Weighted-L_1	$\left\lvert \frac{\sum_{w \in N(u) \cup \{u\}} f_i(w)}{\lvert N(u) \rvert + 1} - \frac{\sum_{t \in N(v) \cup \{v\}} f_i(t)}{\lvert N(v) \rvert + 1} \right\rvert$
Neighbor Weighted-L_2	$\left(\frac{\sum_{w \in N(u) \cup \{u\}} f_i(w)}{\lvert N(u) \rvert + 1} - \frac{\sum_{t \in N(v) \cup \{v\}} f_i(t)}{\lvert N(v) \rvert + 1} \right)^2$

DOI: 10.7717/peerj-cs.172/table-2

from Makarov et al., 2019

Feature representation

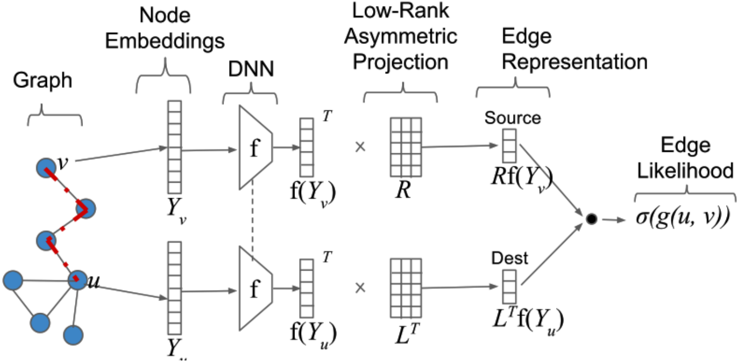
- How efficient simple solution?

- Works for undirected networks
- Samples neighbor information for low cost
- Not stable across different datasets (L_1 works in general better than L_2)
- For weighted networks it is better to solve binary classification stacked with regression rather then directly solve link regression problem

from Makarov et al., 2019

Directed network link prediction

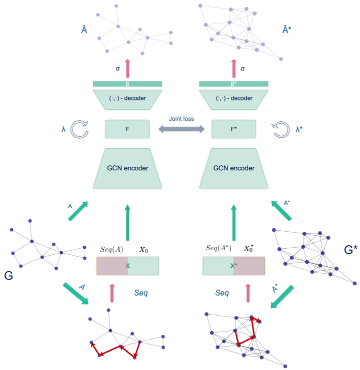
- When order matters, how to build classifier (see HOPE also)?
- Concat works not good probably - use asymmetric encoding via bi-linear form of compressed embeddings
  $M = LR,\ g(u,v) = f(Y_u)^t M f(Y_v)$



from Abu-El-Haija et al., 2018

Self-supervised learning via Line graph

- Edge-vertex dual (Line) graph allows to build dual representation and learn any edge embedding function
- Joint constraints on original and Line graph under bijective closure with agglutination of nodes embeddings in dual representation



from Makarov et al., 2021

References

- D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. Journal of the American Society for Information Science and Technology, 58(7):1019?1031, 2007
- R. Lichtenwalter, J.Lussier, and N. Chawla. New perspectives and methods in link prediction. KDD 10: Proceedings of the 16th ACM SIGKDD, 2010
- M. Al Hasan, V. Chaoji, S. Salem, M. Zaki, Link prediction using supervised learning. Proceedings of SDM workshop on link analysis, 2006
- M. Rattigan, D. Jensen. The case for anomalous link discovery. ACM SIGKDD Explorations Newsletter. v 7, n 2, pp 41-47, 2005
- M. Al. Hasan, M. Zaki. A survey of link prediction in social networks. In Social Networks Data Analytics, Eds C. Aggarwal, 2011.

References

- B. Perozzi, R. Al-Rfou, and S. Skiena. "Deepwalk: Online learning of social representations." In Proceedings of the 20th ACM SIGKDD international conference , pp. 701-710. 2014.
- Mutlu, Ece C., and Toktam A. Oghaz. "Review on graph feature learning and feature extraction techniques for link prediction." arXiv preprint arXiv:1901.03425 (2019).
- Makarov, Ilya, Olga Gerasimova, Pavel Sulimov, and Leonid E. Zhukov. "Dual network embedding for representing research interests in the link prediction problem on co-authorship networks." PeerJ Computer Science 5 (2019): e172.
- S. Abu-El-Haija, B. Perozzi, and R. Al-Rfou. "Learning edge representations via low-rank asymmetric projections." In Proceedings of the 2017 ACM CIKM conference, pp. 1787-1796. 2017.
- H. Cai, V.W. Zheng, and K.C.C. Chang. "A comprehensive survey of graph embedding: Problems, techniques, and applications." IEEE Transactions on Knowledge and Data Engineering 30, no. 9: 1616-1637, 2018