Survey of Popular Link Prediction Algorithms

CS 768 Course Project

Abhinav Kumar 180050003
Nimay Gupta- 180050068
Niraj Mahajan - 180050069

Introduction

- Due to the increasing surge in the research on complex graphical models, a considerable attention is devoted to Social Networks and their properties
- A social network is a graph where the nodes represent entities or people, and the edges represent a link or connection between them
- Link Prediction is defined as: Given a snapshot of a social network at time t, we seek to accurately predict the edges that will be added to the network during the interval from time t to a given future time t’
- In other words, we need to use the intrinsic features of a social network to define a measure of proximity or similarity between any two nodes, and thus predict the probability of a link between the two.

Heuristic-Based methods

- Common Neighbours: Score(u,v) = $|\Gamma(u) \cap \Gamma(v)|$
- Preferential Attachment: Score(u,v) = $|\Gamma(u)| . |\Gamma(v)|$
- Adamic Adar:
  - Instead of simply computing the number of common neighbors, we compute a weighted sum, weighing rarer features more heavily.
  - Score: $s(u,v) = \sum_{w \epsilon N(u) \cap N(v)} \frac{1}{\log(d(w))}$

- Jaccard coefficient: score(u, v) = $|\Gamma(u) \cap \Gamma(v)| \ / \ |\Gamma(u) \cup \Gamma(v)|$

Heuristic-Based methods (ctd)

- Katz measure:
  - Katz defines a measure that directly sums over the collection of paths between two nodes, exponentially damped by path length.
  - For nodes u,v, define paths u,v as the set of all l-length paths between nodes u,v.
  - $$score(u, v) = \sum_{l=1}^{\inf} \beta^l . |\mathbf{paths}_{\mathbf{u,v}}^{\langle \mathbf{l} \rangle}|$$
  - where β > 0 is a dampening parameter. Using the adjacency matrix A, we can get the Katz matrix by solving the infinite geometric progression, as K = (I − βA) −1 − I.

Node2vec

- Random walk

$$P(c_i = x | c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z}, & \text{if}(v, x) \in E \\ 0, & \text{otherwise} \end{cases}$$

- Search bias (alpha)

$$\alpha(t, x) = \begin{cases} \frac{1}{p}, & \text{if} d_{tx} = 0 \\ 1, & \text{if } d_{tx} = 1 \\ \frac{1}{q}, & \text{if } d_{tx} = 2 \end{cases}$$

Node2vec (ctd)

Algorithm 1 The *node2vec* algorithm.

LearnFeatures (Graph $G = (V, E, W)$, Dimensions d, Walks per node r, Walk length l, Context size k, Return p, In-out q)
π = PreprocessModifiedWeights(G, p, q)
$G' = (V, E, \pi)$
Initialize $walks$ to Empty
for $iter = 1$ **to** r **do**
  for all nodes $u \in V$ **do**
    $walk$ = node2vecWalk(G', u, l)
    Append $walk$ to $walks$
f = StochasticGradientDescent(k, d, $walks$)
return f

Node2vec (ctd)

node2vecWalk (Graph $G' = (V, E, \pi)$, Start node u, Length l)

Inititalize $walk$ to $[u]$

for $walk\_iter = 1$ **to** l **do**

$curr = walk[-1]$

$V_{curr} =$ GetNeighbors($curr$, G')

$s =$ AliasSample(V_{curr}, π)

Append s to $walk$

return $walk$

Graphsage

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

Experimental Results

Dataset	Evaluation Method	Algorithm						
		Adamic Adar	Common Neighbor	Preferential Attachment	Katz Measure	Logistic Regression	node2vec	GraphSage
Facebook	MAP	**0.726**	0.719	0.112	0.698	0.694	0.663	———
	MRR	**0.875**	0.872	0.242	0.860	0.849	0.813	———
Celegans	MAP	0.453	**0.525**	0.304	0.446	0.453	0.400	———
	MRR	0.584	**0.653**	0.443	0.583	0.587	0.530	———
arXiv mini	MAP	0.814	**0.831**	0.077	0.759	0.212	0.741	———
	MRR	0.922	**0.936**	0.173	0.886	0.319	0.863	———
cora	MAP	0.637	**0.684**	0.169	0.500	0.466	0.405	0.01
	MRR	0.664	**0.713**	0.187	0.549	0.522	0.447	0.02

Performance of Heuristics

- Common Neighbors gives us the best results for cora, arXiv and C-elegans graph
- Adamic Adar gives the best results for the facebook graph.
- Adamic Adar weights the score according to the degree of the common neighbor and this shields our score from being influenced by celebrity nodes.
- Such nodes can be frequently occurring in the facebook graph, but will be less frequent in citation graphs or C-elegans graph

Performance of Heuristics

- Preferential Attachment performs poorly as it is based on linking highly (socially) active nodes.
- This is a bad measure since it is improbable that an edge can be established between any two random nodes, just because they are socially active.
- For example it is absurd to say that two researchers, one in Computer Vision and one in Earth Science, will collaborate on a project

Performance of node2vec and GraphSage

- Node2vec gives sub optimal results as compared to the heuristic based approaches
- GraphSage gives really bad results.
- The embeddings obtained using GraphSage give an 86.20% accuracy(F1-score) for node classification.
- Since the graph is extremely sparse (0.04 test fraction) the embeddings contain only the nodes' intrinsic information, and fail to extract the neighborhood information.
- This makes it difficult to train a logistic regression on the embeddings obtained from such sparse graphs. Another reason could be that embeddings learnt are not transferable to other task hence poor results