Random Walks on Graphs - Part II

Pawan Goyal

CSE, IITKGP

September 11, 2015

The problem of link prediction and recommendation

Link Prediction

- We are given a snapshot of a social network at time t
- We seek to predict the edges that will be added to the network during the interval from time t to a future time t'

e.g. we are given a large network, say Facebook, at time t and for each user we would like to predict what new edges (friendships) that particular user will create between t and t'

The problem of link prediction and recommendation

Link Prediction

- We are given a snapshot of a social network at time t
- We seek to predict the edges that will be added to the network during the interval from time t to a future time t'

e.g. we are given a large network, say Facebook, at time t and for each user we would like to predict what new edges (friendships) that particular user will create between t and t'

Link Recommendation Problem

The same problem can also be viewed as a *link recommendation problem*, where we aim to suggest to each user a list of people that the user is likely to create new connections to.

Challenges Involved

Sparsity

Real networks are really sparse, in Facebook, a typical user is connected to about 100-200 out of more than 500 million nodes

Can it be modeled using network features only?

New edges in Facebook social network

Creation of New Links: Important questions

How do network and node features interact?

- How important it is to have common interests and characteristics?
- How important it is to be in the same social circle and be “close” in the network in order to eventually connect.
- *Develop a method that combines the features of nodes (user profile) and edges (interaction) with the network structure*

Correlation Discovery by random walk

Problem definition

Estimate the importance/affinity of node "B" with respect to another node "A" in the graph.

Correlation Discovery by random walk

Problem definition

Estimate the importance/affinity of node “B” with respect to another node “A” in the graph.

Framework: Random walk with restarts

- **Goal:** Compute the importance of node “B” for node “A"
- Consider a random walker that starts from node “A”, choosing among the available edges every time
- Except that, before he makes a choice, with probability c_r, he goes back to node “A” (restart)

Random walk with restarts

- Let $u_A(B)$ denote the steady state probability that the random walker will find himself at node "B".
- $u_A(B)$ is what we want, the importance of "B" with respect to "A".
- $u_A = (u_A(1), \ldots, u_A(N))$
- Steady-state vector: $u_A = (1 - c_r)u_A A + c_r v_A$
- A: transition matrix, c_r: restart probability, v_A: restart vector with all its N elements zero except for the entry corresponding to node A.

Choice of restart probability c

- c_r controls how “far” the walk wanders from the seed node s before it restarts and jumps back to s
- High values of c_r give very short and local random walks, while low values allow the walk to go further away.

Choice of restart probability c

- c_r controls how "far" the walk wanders from the seed node s before it restarts and jumps back to s
- High values of c_r give very short and local random walks, while low values allow the walk to go further away.

A good choice

Depends on the diameter of the graph. A good choice would follow $(1 - c_r)^d = 0.045$, where d is the diameter.
$d = 6 \rightarrow c_r = 0.4$, $d = 19 \rightarrow c_r = 0.15$

Supervised Random Walks

Basic Idea

In a *supervised way*, learn how to bias a PageRank-like random walk on the network so that it visits given nodes (positive training examples) more often than the others.

- Use node and edge features to learn *edge strengths*.
- Random walk on such a weighted network will be more likely to visit “positive” than “negative” nodes.
- Link Prediction: ‘*positive*’: nodes to which new edges will be created in the future, *negative*: all other nodes.
- Link recommendation: ‘*positive*’: nodes to which user clicks on

Learning Task

Training data

A source node s is given, along with the training examples to which s will create links in the future.

Goal

Learn a function that assigns a strength (random walk probability) to each edge.

Link Prediction: Baseline Approaches

Link Prediction as a classification task

- Take nodes to which s has created edges as positive training examples, all other nodes as negative training examples
- Learn a classifier that predicts where node s is going to create links

Random walk with restarts

Start a random walk at node s and compute the proximity of each other node to node s.

Relation to personalized PageRank

- We are given a source node s and a set of destination nodes $d_1, \ldots, d_k \in D$ to which s will create edges in the future
- Aim is to bias the random walk such that it will visit nodes d_i more often than the other nodes in the network

Relation to personalized PageRank

- We are given a source node s and a set of destination nodes $d_1, \ldots, d_k \in D$ to which s will create edges in the future
- Aim is to bias the random walk such that it will visit nodes d_i more often than the other nodes in the network
- Can we directly set an arbitrary transition probability to each edge?

Relation to personalized PageRank

- We are given a source node s and a set of destination nodes $d_1, \ldots, d_k \in D$ to which s will create edges in the future
- Aim is to bias the random walk such that it will visit nodes d_i more often than the other nodes in the network
- Can we directly set an arbitrary transition probability to each edge?
- Would result in drastic over-fitting

Relation to personalized PageRank

- We are given a source node s and a set of destination nodes $d_1, \ldots, d_k \in D$ to which s will create edges in the future
- Aim is to bias the random walk such that it will visit nodes d_i more often than the other nodes in the network
- Can we directly set an arbitrary transition probability to each edge?
- Would result in drastic over-fitting
- Instead, we assign the transition probability for each edge (u, v) based on features of nodes u and v, as well as features of edge (u, v).

Problem Formulation

- Directed graph $G(V,E)$
- Node s, destination nodes $D = \{d_1, \ldots, d_k\}$ and no-link nodes $L = \{l_1, \ldots, l_n\}$

Problem Formulation

- Directed graph $G(V,E)$
- Node s, destination nodes $D = \{d_1, \ldots, d_k\}$ and no-link nodes $L = \{l_1, \ldots, l_n\}$
- Each edge (u,v) has a feature vector $\psi(u,v)$ that describes the nodes u and v (e.g., gender, age, hometown) and the interaction attributes (e.g., time of edge creation, messages exchanges, photos appeared together in)

Problem Formulation

- Directed graph $G(V,E)$
- Node s, destination nodes $D = \{d_1, \ldots, d_k\}$ and no-link nodes $L = \{l_1, \ldots, l_n\}$
- Each edge (u,v) has a feature vector $\psi(u,v)$ that describes the nodes u and v (e.g., gender, age, hometown) and the interaction attributes (e.g., time of edge creation, messages exchanges, photos appeared together in)
- Compute the strength $a_{uv} = f_w(\psi_{uv})$ for edge (u,v).

Problem Formulation

- Directed graph $G(V,E)$
- Node s, destination nodes $D = \{d_1, \ldots, d_k\}$ and no-link nodes $L = \{l_1, \ldots, l_n\}$
- Each edge (u,v) has a feature vector $\psi(u,v)$ that describes the nodes u and v (e.g., gender, age, hometown) and the interaction attributes (e.g., time of edge creation, messages exchanges, photos appeared together in)
- Compute the strength $a_{uv} = f_w(\psi_{uv})$ for edge (u,v).
- We want to learn the function $f_w(\psi)$ in the training phase of the algorithm

Predicting new edges using Edge Strength

- Edge strengths of all edges are calculated using f_w
- Random walk with restarts is run from s
- Stationary distribution p of the random walk assigns each node u a probability p_u
- Top ranked nodes are predicted as destinations of future links of s

Using edge weights

- Function $f_w(\psi_{uv})$ combines the attributes ψ_{uv} and the parameter vector w to output a non-negative weight a_{uv} for each edge

Using edge weights

- Function $f_w(\psi_{uv})$ combines the attributes ψ_{uv} and the parameter vector w to output a non-negative weight a_{uv} for each edge
- We use this to build the random walk stochastic transition matrix Q' such that

$$Q'_{uv} = \frac{a_{uv}}{\sum_w a_{uw}}, (u, v) \in E$$

Using edge weights

- Function $f_w(\psi_{uv})$ combines the attributes ψ_{uv} and the parameter vector w to output a non-negative weight a_{uv} for each edge
- We use this to build the random walk stochastic transition matrix Q' such that

$$Q'_{uv} = \frac{a_{uv}}{\sum_w a_{uw}}, (u,v) \in E$$

- Corresponding matrix for random walk with restart:

$$Q_{uv} = (1-c)Q'_{uv} + c1(v = s)$$

- Verify that Q is row stochastic

Using edge weights

- Function $f_w(\psi_{uv})$ combines the attributes ψ_{uv} and the parameter vector w to output a non-negative weight a_{uv} for each edge
- We use this to build the random walk stochastic transition matrix Q' such that

$$Q'_{uv} = \frac{a_{uv}}{\sum_w a_{uw}}, (u,v) \in E$$

- Corresponding matrix for random walk with restart:

$$Q_{uv} = (1-c)Q'_{uv} + c1(v=s)$$

- Verify that Q is row stochastic
- $P_{1\times n}$ is the stationary distribution of the Random walk with restarts, and is the solution of the following equation:

$$P = PQ$$

Optimization Problem

- Aim: Learn the parameters w of function $f_w(\psi_{uv})$ that assigns each edge a strength of a_{uv}
- Criterion: Assign the weights such that the random walk is more likely to visit nodes in D than L, i.e., $p_l < p_d$, for each $d \in D$ and $l \in L$

Optimization Problem

- Aim: Learn the parameters w of function $f_w(\psi_{uv})$ that assigns each edge a strength of a_{uv}
- Criterion: Assign the weights such that the random walk is more likely to visit nodes in D than L, i.e., $p_l < p_d$, for each $d \in D$ and $l \in L$

Optimization function

$min_w F(w) = ||w||^2$ such that $\forall d \in D, l \in L : p_l < p_d$
p_is are the pageRank scores
A smaller w is preferred simply for regularization

Optimization function: Softer version

$$min_w F(w) = ||w||^2 + \lambda \sum_{d \in D, l \in L} h(p_l - p_d)$$

$h(.)$: loss function such that $h(.) = 0$ as $p_l < p_d$ and $h(.) > 0$ for $p_l - p_d > 0$

References

- **Random Walk with Restarts:** Pan, Jia-Yu, Hyung-Jeong Yang, Christos Faloutsos, and Pinar Duygulu. "*Automatic multimedia cross-modal correlation discovery.*" In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 653-658. ACM, 2004.
- **Supervised Random Walks:** Backstrom, Lars, and Jure Leskovec. "*Supervised random walks: predicting and recommending links in social networks.*" In Proceedings of the fourth ACM international conference on Web search and data mining, pp. 635-644. ACM, 2011.