Social Graph Learning

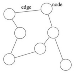
Steve Poulson

Supervisor: Dell Zhang, Mark Levine

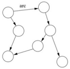
28 May 2013

Types of social graphs

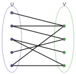
Unipartite, Undirected e.g. LinkedIn



Unipartite, Directed e.g. Google Plus

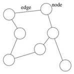


Bipartite, UnDirected e.g. Netflix

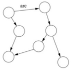


Types of social graphs

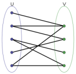
Unipartite, Undirected e.g. LinkedIn



Unipartite, Directed e.g. Google Plus

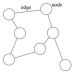


Bipartite, UnDirected e.g. Netflix

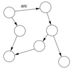


Types of social graphs

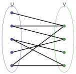
Unipartite, Undirected e.g. LinkedIn



Unipartite, Directed e.g. Google Plus



Bipartite, UnDirected e.g. Netflix



Sampling

We need to get the data, we could sample a real graph

- Breadth First
- Depth First
- Snowball
- Forest Fire

In practise we need to simulate the growth of the graph

Sampling

We need to get the data, we could sample a real graph

- Breadth First
- Depth First
- Snowball
- Forest Fire

In practise we need to simulate the growth of the graph

Sampling

We need to get the data, we could sample a real graph

- Breadth First
- Depth First
- Snowball
- Forest Fire

In practise we need to simulate the growth of the graph

Sampling

We need to get the data, we could sample a real graph

- Breadth First
- Depth First
- Snowball
- Forest Fire

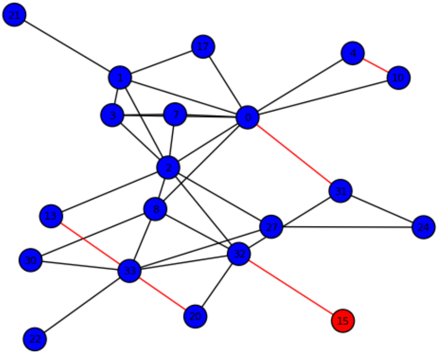
In practise we need to simulate the growth of the graph

Algorithm

Data: Directed Graph A
Result: Digraph B with n new vertices, m new edges
while *not n vertices* **do**
    pick random edge (v_1, v_2) from A where $v_1 \in$ B
    add v_2 add (v_1, v_2) to graph B
end
while *not m edges added* **do**
    pick random edge (v_1, v_2) from A where $v_1, v_2 \in$ B
    add (v_1, v_2) to graph B
end

Algorithm 1: Grow

Karate Dataset grown to t=4, n=10



Supervised learning problem

Formally

- $\mathbf{Y} \leftarrow \mathbf{A^{t+1}} - \mathbf{A^{t}}$
- $\min(L(f(\phi(\mathbf{A^{t}})), \mathbf{Y}))$

Supervised learning problem

Formally

- $\mathbf{Y} \leftarrow \mathbf{A^{t+1}} - \mathbf{A^t}$
- $\min(L(f(\phi(\mathbf{A^t})), \mathbf{Y}))$

Supervised learning problem

Formally

- $\mathbf{Y} \leftarrow \mathbf{A^{t+1}} - \mathbf{A^t}$
- $\min(L(f(\phi(\mathbf{A^t})), \mathbf{Y}))$

Feature Function ϕ

Maps $\mathbf{A}_{i,j} \rightarrow (x_1 \ldots x_n)$

- Random Walk between nodes
- Modularity
- Jaccard Distance of neighbour list
- Betweeneness Centrality

Feature Function ϕ

Maps $\mathbf{A}_{i,j} \rightarrow (x_1 \ldots x_n)$

- Random Walk between nodes
- Modularity
- Jaccard Distance of neighbour list
- Betweeneness Centrality

Feature Function ϕ

Maps $\mathbf{A}_{i,j} \rightarrow (x_1 \ldots x_n)$

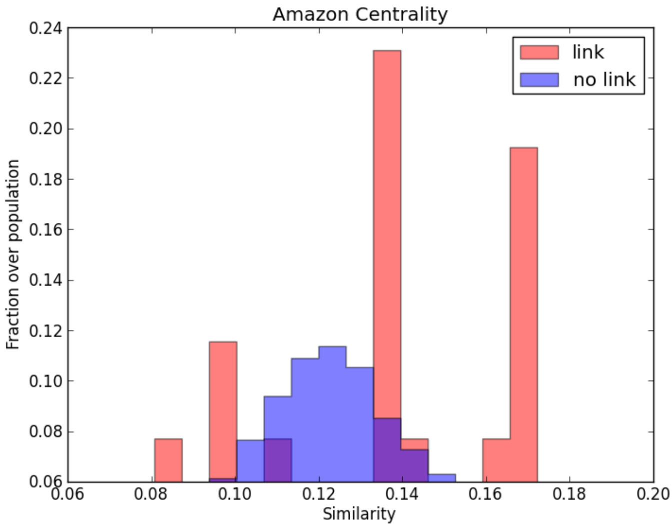
- Random Walk between nodes
- Modularity
- Jaccard Distance of neighbour list
- Betweeness Centrality

Feature Function ϕ

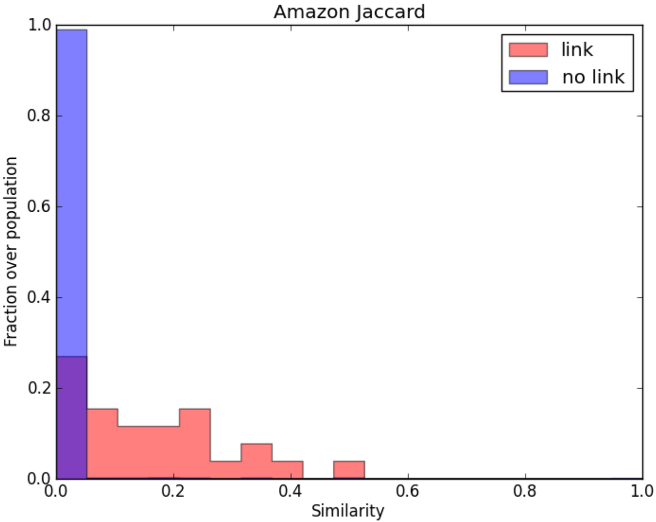
Maps $\mathbf{A}_{i,j} \rightarrow (x_1 \ldots x_n)$

- Random Walk between nodes
- Modularity
- Jaccard Distance of neighbour list
- Betweeness Centrality

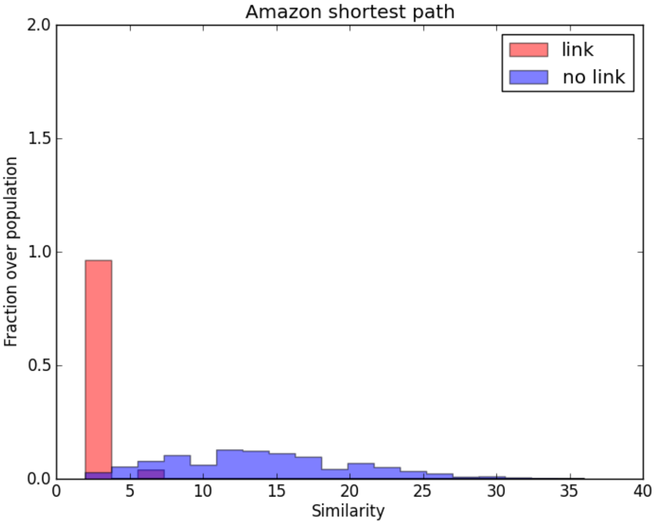
Betweeness


Amazon Centrality
link
no link
Fraction over population
0.24
0.22
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.06
0.08
0.10
0.12
0.14
0.16
0.18
0.20
Similarity

Jaccard Distance



Shortest Path



Local vs Global

- Global features from a 1 million node graph unfeasible
- Power Law graphs scale invariant so local features should characterize graph
- Can use Hadoop Map calculates subset of local features, reduce assembles training set which classifier runs on

Local vs Global

- Global features from a 1 million node graph unfeasible
- Power Law graphs scale invariant so local features should characterize graph
- Can use Hadoop Map calculates subset of local features, reduce assembles training set which classifier runs on

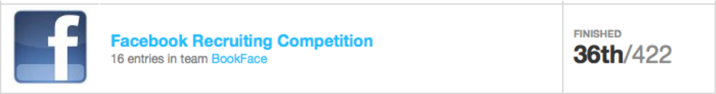
Local vs Global

- Global features from a 1 million node graph unfeasible
- Power Law graphs scale invariant so local features should characterize graph
- Can use Hadoop Map calculates subset of local features, reduce assembles training set which classifier runs on

Let's try it on a Kaggle competition

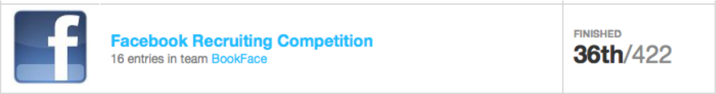
- Prototyped on sklearn / networkx
- Hadoop cluster on AWS
- Java: Weka / Hadoop / cern.colt.matrix / Jung
- Random Forrest classifer beat rest

Kaggle Evaluation



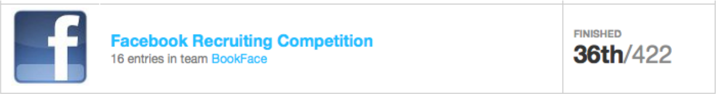
- A bit of fun :)
- Mean Average Precision @ 10 = 0.71371
- Within 2% of winner

Kaggle Evaluation



- A bit of fun :)
- Mean Average Precision @ 10 = 0.71371
- Within 2% of winner

Kaggle Evaluation



- A bit of fun :)
- Mean Average Precision @ 10 = 0.71371
- Within 2% of winner

Summary

- Power law Sampling gives a time varying dataset
- Random Forrest best
- Pretty good results

Summary

- Power law Sampling gives a time varying dataset
- Random Forrest best
- Pretty good results

Summary

- Power law Sampling gives a time varying dataset
- Random Forrest best
- Pretty good results

Future work

- Factorial Machines
- Deep Learning
- Sample real data Twitter / Google+
- Bipartite Graphs