Community Detection Using Time-Dependent Personalized PageRank

Haim Avron
Joint work with Lior Horesh

IBM T. J. Watson Research Center

32nd International Conference on Machine Learning
(ICML 2015)
July 7, 2015

Problem

- Input: an n node undirected, unweighted graph.
- Community detection: find a set of nodes that are both internally cohesive and well separated
- We consider *community detection using seed node*:
  - Given a seed node, find a community around it

Community Detection Using Local Diffusions

A popular framework:

1. Compute a diffusion vector
2. Reweight the vector based on degrees
3. Sort the nodes in decreasing order
4. Select a prefix that maximizes (or minimizes) some scoring function, e.g. *conductance* $\phi(S) \equiv \frac{\partial S}{\min(\mathbf{vol}(S), \mathbf{vol}(\bar{S}))}$

Can be rigorously analyzed (Andersen et al., 2006; Chung, 2009)...

... but in practice multiple diffusions/parameters/scores are used to generate "interesting points" in the combinatorial search space.

PageRank and Heat Kernel Diffusion Vectors

- Notation:
  - $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the adjacency matrix
  - $\mathbf{D} \in \mathbb{R}^{n \times n}$ is diagonal matrix of degrees
  - $\mathbf{P} \equiv \mathbf{A}\mathbf{D}^{-1}$ (random walk transition matrix)
  - $\mathsf{s} \in \mathbb{R}^{n}$ with 1 at seed, 0 elsewhere
- Personalized PageRank:

$$\mathsf{p} \equiv (1-\alpha)(\mathbf{I}_n - \alpha\mathbf{P})^{-1}\mathsf{s}$$

- Heat Kernel:

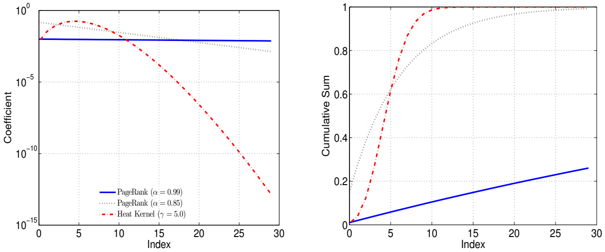
$$\mathsf{h} \equiv \exp\left\{-\gamma(\mathbf{I}_n - \mathbf{P})\right\}\mathsf{s}$$

Diffusion Coefficients

Diffusion vectors can be written as a series:

$$\mathbf{f} = \sum_{k=0}^{\infty} \alpha_k \mathbf{P}^k \mathbf{s}$$

$$\alpha_k^{pr} = (1-\alpha)\alpha^k \qquad \alpha_k^{hk} = e^{-\gamma}\frac{\gamma^k}{k!}$$

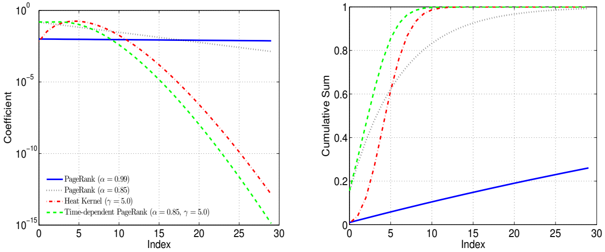


Time-Dependent PageRank

$$\mathbf{x} \equiv (1-\alpha)(\mathbf{I}_n-\alpha\mathbf{P})^{-1}\mathbf{s}+\exp\left\{-\gamma(\mathbf{I}_n-\alpha\mathbf{P})\right\}\left(\mathbf{s}-(1-\alpha)(\mathbf{I}_n-\alpha\mathbf{P})^{-1}\mathbf{s}\right)$$

$$\alpha_k^{tpr} = \left[\left(1-\sum_{r=0}^{k}\alpha_r^{hk}\right)\alpha_k^{pr}+\alpha^k\alpha_k^{hk}\right]$$

Generalized both: $\alpha = 1$ is heat kernel, $\gamma \to \infty$ is PageRank



The Importance of Being Local

- Even if the graph is huge, communities tend to be local
  - Running time should be proportional to community size
- Diffusion vectors are typically dense
  - $\Omega(n)$ to compute them exactly
- However most values are tiny
- Literature: A reasonably approximation $\mathbf{f}^\star$ to $\mathbf{f}$ is one that

$$\|\mathbf{D}^{-1}(\mathbf{f}^\star - \mathbf{f})\|_\infty < \epsilon$$

- There are sparse vectors that uphold this (when ϵ is not too small)

Local Algorithms for Local Diffusions

- Coordinate relaxation approach:
  - Write diffusion vector as solution to a linear system
  - Use a semi-greedy coordinate relaxation iteration (related to Gauss-Southwell rule)
  - General method for sparse approx of linear system solution
- Algorithms based on this approach:
  - Andersen et al. (2006) - PageRank (“push” alg.)
  - Kloster & Gleich (2014) - Heat Kernel (hkgrow)
- **Our contribution: a local algorithm for Time-Dependent Personalized PageRank**
- Main challenge: it does not translate to linear system
  - Do coordinate relaxation on a system of ODEs instead
  - Local approximation to system of ODE solution

Underlying Observations

1. $\mathbf{x} = \mathbf{x}(\gamma)$ where $\mathbf{x}(\cdot)$ is the solution to

$$\mathbf{x}'(t) = (1-\alpha)\mathbf{s} - (\mathbf{I}_n - \alpha\mathbf{P})\mathbf{x}(t) \quad \mathbf{x}(0) = \mathbf{s}$$

2. Let $\mathbf{y}(\cdot)$ be an approx solution. The *residual* of $\mathbf{y}(\cdot)$ is

$$\mathbf{r}(t) \equiv (1-\alpha)\mathbf{s} - (\mathbf{I}_n - \alpha\mathbf{P})\mathbf{y}(t) - \mathbf{y}'(t)$$

Proposition: $\mathbf{y}(\cdot)$ is good enough if for all i

$$\|r_i(\cdot)\|_\infty < \frac{(1-\alpha)d_i\epsilon}{1-\exp((\alpha-1)\gamma)}$$

Coordinate Relaxation "Algorithm"

Suggests a coordinate relaxation type algorithm:

1. Initialize $\mathbf{y}(t) = s$.
2. While there exists a violating i
   1. Select such an i arbitrarily.
   2. Set $y_i(\cdot)$ to the solution of the ODE

$$y'(t) = -y(t) + \alpha \sum_{j=1}^{n} \mathbf{P}_{ij} y_j(t) + (1-\alpha) s_i \quad y(0) = s_i$$

However, this "algorithm" requires operations on infinite dimensional objects, and so is not viable.

From Infinite to Finite Dimension

- Use degree N polynomials
- $N = O(\gamma + \log(1+\epsilon))$ (see paper for details)
- Polynomials are represented as samples on $N+1$ scaled and shifted Chebyshev points

$$\mathbb{S}_N[p(\cdot)] \equiv \begin{bmatrix} p(p_0) & \cdots & p(t_N) \end{bmatrix}^T \quad t_j = (\cos(j\pi/N) + 1)\gamma/2$$

- Need to map operations of the "algorithm":
  - Computing derivative (for residual computation)
  - Computing infinity norm of iterates (testing convergence)
  - Solving the ODE

Computing Residual (aka Computing Derivative)

- The residual is a degree N polynomial as well (derivative of a polynomial is a reduced degree polynomial)
- Derivative is a linear operation, so exists $\Xi \in \mathbb{R}^{(N+1)\times(N+1)}$ s.t.
$$\mathbb{S}_N[p'(\cdot)] = \Xi \mathbb{S}_N[p(\cdot)]$$
- Formulas for Ξ is easily derived from well known formulas

$$\Xi_{ij} = \begin{cases} \gamma(1+2N^2)/12 & i=j=0 \\ -\gamma(1+2N^2)/12 & i=j=N \\ \gamma x_j/(4-4x_j^2) & i=j;\ 0<j<N \\ (-1)^{i+j} p_i/(2p_j x_i - p_j x_j) & i \neq j \end{cases}$$

where $x_i = \cos(\pi i/N)$, $p_0 = p_N = 2$, and $p_j = 1$ otherwise.

Testing Convergence

- Amounts to bounding the infinity norm of polynomials
- Can be computed exactly, but expensive ($O(N^3)$)
- **Proposition:**

$$\|p(\cdot)\|_\infty \leq (1 + \frac{2}{\pi} \log N) \|\mathbb{S}_N[p(\cdot)]\|_\infty \, .$$

- Convergence test: for all i

$$\|\mathbf{r}_i\|_\infty < \frac{(1-\alpha) d_i \epsilon}{(1 - \exp((\alpha - 1)\gamma))(1 + \frac{2}{\pi} \log N)}$$

Solving the ODE

- The solution of the ODE is normally not a polynomial
    - So ODE can only be solved approximately
- Write the update as:

$$\mathbb{S}_N[y_i(\cdot)] \leftarrow \mathbb{S}_N[y_i(\cdot)] + \mathbf{d}$$

  with boundary condition $d_{N+1} = 0$
- Residual update is:

$$\mathbb{S}_N[r_i(\cdot)] \leftarrow \mathbb{S}_N[r_i(\cdot)] - (\Xi + \mathbf{I}_{N+1})\mathbf{d}$$

- Leads to:

$$\min_{\mathbf{d}} \|\mathbb{S}_N[r_i(\cdot)] - (\Xi + \mathbf{I}_{N+1})\mathbf{d}\|_2 \text{ s.t. } d_{N+1} = 0\,.$$

- Solution is

$$\mathbf{d} = \begin{pmatrix} \Xi_1^+ \mathbb{S}_N[r_i(\cdot)] \\ 0 \end{pmatrix}$$

Putting It All Together (sketch, pseudo code in paper)

- Keep a queue of violating indices. Initialize with seed.
- In each iteration,
  - pop an index i
  - Update $\mathbf{y}_i \equiv \mathbb{S}_N[y_i(\cdot)]$ and $\mathbf{r}_i \equiv \mathbb{S}_N[r_i(\cdot)]$
  - Update $\mathbf{r}_j \equiv \mathbb{S}_N[r_j(\cdot)]$ for neighboring js
  - If any violate: add to queue
- Most of the $\mathbf{y}_i$ and $\mathbf{r}_i$ are zero; do not keep in memory
- Only access to graph is via degree and adjacency queries
- Discounting hash operations, and with some preprocessing, $O(N^2 + deg \cdot N)$ per iteration.

Summary of Experimental Results

- Alg. tends to produce smaller communities (so presumably more realistic), with slightly higher conductance.
- Different results even when $\alpha = 1.0$ (hk) and $\gamma \to \infty$ (ppr)
  - There are many vectors that are "good enough" approx
- Comparable results to **hkgrow** (Kloster & Gleich, 2014) on datasets with ground truth.
- Running time:
  - Slower than **hkgrow** for heat-kernel ($\alpha = 1.0$)
  - Faster with non-degenerate parameters (e.g. $\gamma = 5.0, \alpha = 0.85$).
- Fewer access to edge lists – important for out-of-core.

Conclusions

- Efficient local algorithm for Time Dependent PageRank
- Also another local algorithm for PageRank and heat kernel.
- Experimentally, rankings that are distinct and competitive, thus a useful addition to the toolset
- Core technique: local solution of system of ODEs. Other uses?
- Open question: improved approximation bounds using time-dependent PageRank?
    - It generalizes both PageRank and heat kernel...