Updating the Stationary Vector of a Markov Chain

Amy Langville
Carl Meyer

Department of Mathematics
North Carolina State University
Raleigh, NC

NSMC 9/4/2003

Outline

- Updating and Pagerank
- Aggregation
- Partitioning
- Iterative Aggregation Algorithm
- Experiments

The Updating Problem

Given: Original chain's P and π^T

and new chain's $\widetilde{P}$

Find: new chain's $\widetilde{\pi}^T$

PageRank Application

P

Google uses hyperlink structure of Web + fudge factor matrix to form irreducible, aperiodic Markov chain

π_i

long-run proportion of time a random surfer spends on webpage i

π^T

gives ranking of relative importance of webpages

How Google Uses π^T

To rank importance of thousands of pages containing a query phrase and list only the most "important" of those relevant pages to users.

Need for Updating PageRank vector π^T

Fact:

Currently $\boldsymbol{\pi}^T$ for immense Web Markov chain is computed monthly.

Fact:

Web changes much more frequently. (hourly on news sites)

Fact:

Computing $\boldsymbol{\pi}^T$ takes days. (power method used)

Need:

Update $\boldsymbol{\pi}^T$ $\underline{more\ frequently}$ with $\underline{less\ work}$.

Computing π^T

A Big Problem

Solve $\pi^T = \pi^T \mathbf{P}$ (stationary distribution vector)

$\pi^T(\mathbf{I} - \mathbf{P}) = 0$ (too big for direct solves)

Start with $\pi_0^T = \mathbf{e}/n$ and iterate $\pi_{j+1}^T = \pi_j^T \mathbf{P}$ (power method)

Google's solution to updating problem

Full recomputation — run power method from scratch

Start with $\pi_0^T = \mathbf{e}/n$ and iterate $\pi_{j+1}^T = \pi_j^T \widetilde{\mathbf{P}}$

Don't use old PageRank vector to find new PageRank faster.

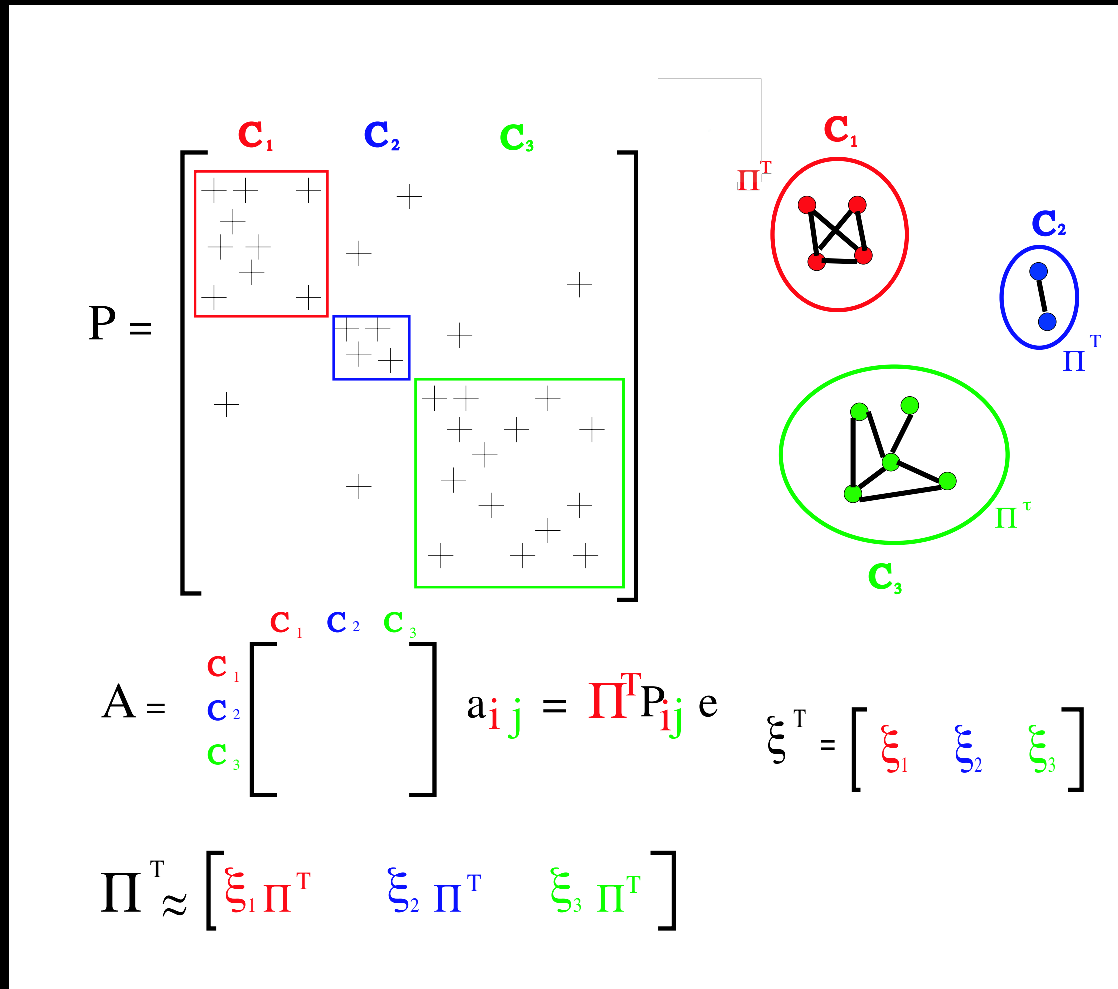
Our goal

Use iterative aggregation to find $\tilde{\pi}^T$ faster, with less work, than full recomputation.

Idea behind Aggregation

Best for NCD systems

(Simon and Ando (1960s), Courtois (1970s))



Pro

exploits structure to reduce work

Con

produces an approximation, quality is dependent on degree of coupling

Iterative Aggregation

- Problem: repeated aggregation leads to fixed point.
- Solution: Do a power step to move off fixed point.
- Do this iteratively. Approximations improve and approach exact solution.
- Success with NCD systems, not in general.

Input: approximation to $\boldsymbol{\pi}^T$

get censored distributions $\boldsymbol{\pi}^T$ $\boldsymbol{\pi}^T$ $\boldsymbol{\pi}^T$

get coupling constants $\boldsymbol{\xi}_i$

Output: get approximate global stationary distribution $\boldsymbol{\pi}^T = \left[\boldsymbol{\xi}_1\boldsymbol{\pi}^T \quad \boldsymbol{\xi}_2\boldsymbol{\pi}^T \quad \boldsymbol{\xi}_3\boldsymbol{\pi}^T\right]$

Output: move off fixed point with power step

How to Partition for Updating Problem?

Intuition

- There are some bad states (G) and some good states ($\overline{G}$).
- Give more attention to bad states. Each state in G forms a partitioning level.
- Lump good states into 1 superstate.

Aggregation Matrix

$$\mathbf{A} = \begin{array}{c} \\ G_1 \\ G_2 \\ \vdots \\ \\ \overline{G} \end{array} \begin{array}{c} \begin{array}{cccc} G_1 & G_2 & \cdots & \overline{G} \end{array} \\ \left(\begin{array}{ccc|c} & & & \\ & & & \\ & & & \\ \text{---} & \text{---} & \text{---} & \text{---} \\ & & & \end{array} \right)_{(|G|+1)\times(|G|+1)} \end{array}$$

Definitions for "Good" and "Bad"

1. Good = states most likely to have π_i change
   Bad = states least likely to have π_i change

2. Good = states with smallest π_i after k transient steps
   Bad = states "nearby", with largest π_i after k transient steps

3. Good = smallest π_i from old PageRank vector
   Bad = largest π_i from old PageRank vector

4. Good = fast–converging states
   Bad = slow–converging states

Determining "Fast" and "Slow"

Consider power method and its rate of convergence

$$\boldsymbol{\pi}_{k+1}^T = \boldsymbol{\pi}_k^T \mathbf{P} = \boldsymbol{\pi}_k^T \mathbf{e} \boldsymbol{\pi}^T + \lambda_2^k \boldsymbol{\pi}_k^T \mathbf{x}_2 \mathbf{y}_2^T + \lambda_3^k \boldsymbol{\pi}_k^T \mathbf{x}_3 \mathbf{y}_3^T + \cdots + \lambda_n^k \boldsymbol{\pi}_k^T \mathbf{x}_n \mathbf{y}_n^T$$

Asymptotic rate of convergence is rate at which $\lambda_2^k \to 0$

Consider convergence of elements

Some states converge to stationary value faster than λ_2–rate, due to LH e–vector $\mathbf{y}_2^T$.

Partitioning Rule

Put states with largest $|\mathbf{y}_2^T|_i$ values in bad group G, where they receive more individual attention in aggregation method.

Practicality

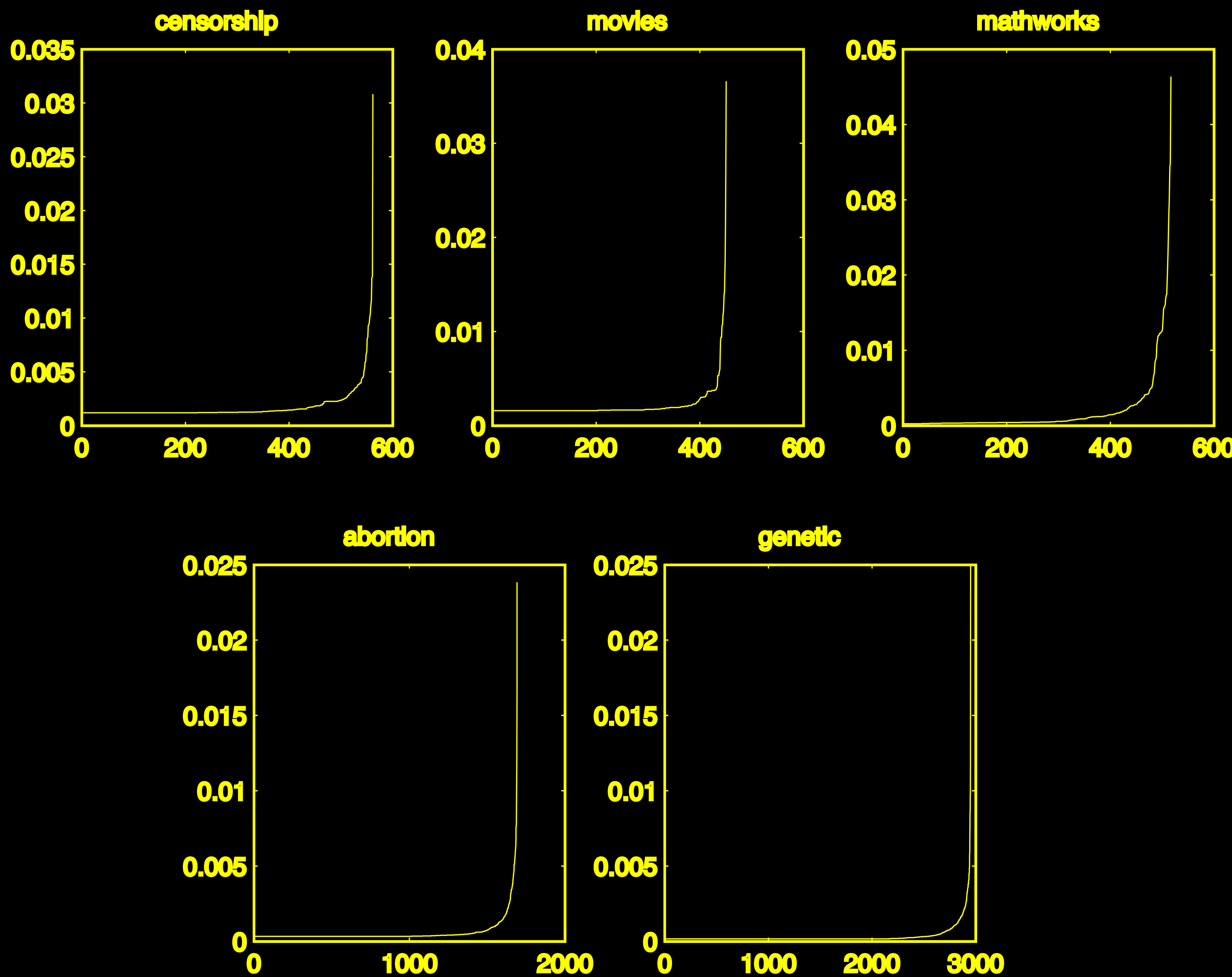
$\mathbf{y}_2^T$ expensive, but for PageRank problem, Kamvar et al. show states with large π_i are slow-converging. $\Rightarrow$ inexpensive, use old $\boldsymbol{\pi}^T$ to determine G.

(adaptively approximate $\mathbf{y}_2^T$)

Power law for PageRank

Scale-free Model of Web network creates power laws

(Kamvar, Barabasi, Raghavan)



Experiments

Test Networks From Crawl Of Web (Supplied by Ronny Lempel)

Censorship (Sites concerning "censorship on the net")
562 nodes 736 links

Movies (Sites concerning "movies")
451 nodes 713 links

MathWorks (Supplied by Cleve Moler)
517 nodes 13,531 links

Abortion (Sites concerning "abortion")
1,693 nodes 4,325 links

Genetics (Sites concerning "genetics")
2,952 nodes 6,485 links

Parameters

Number Of Nodes (States) Added

$\mathbf{3}$

Number Of Nodes (States) Removed

$\mathbf{5}$

Number Of Links Added (Different values have little effect on results)

$\mathbf{10}$

Number Of Links Removed

$\mathbf{20}$

Stopping Criterion

1-norm of residual $< 10^{-10}$

The Partition

Intuition

− Prefer to use $\mathbf{y}_2^T$ to find slow–converging states, but expensive.

\+ Slow–converging components tend to be high PageRank pages

The G Set

New states go into G

States corresponding to large entries in

$$\phi^T = (\phi_1, \phi_2, \ldots, \phi_m) \longrightarrow G$$

States corresponding to small entries $\longrightarrow \overline{G}$

Censorship

Power Method

Iterations	Time
38	1.40

Iterative Aggregation

$\|G\|$	Iterations	Time
5	38	1.68
10	38	1.66
15	38	1.56
20	20	1.06
25	20	1.05
50	10	.69
100	8	.55
300	6	.65
400	5	.70

$nodes$ = **562** $links$ = **736**

Censorship

Power Method

Iterations	Time
38	1.40

Iterative Aggregation

$\|G\|$	Iterations	Time
5	38	1.68
10	38	1.66
15	38	1.56
20	20	1.06
25	20	1.05
50	10	.69
100	8	.55
200	6	.53
300	6	.65
400	5	.70

$nodes = \mathbf{562}$ $links = \mathbf{736}$

Movies

Power Method

Iterations	Time
17	.40

Iterative Aggregation

$\|G\|$	Iterations	Time
5	12	.39
10	12	.37
15	11	.36
20	11	.35
100	9	.33
200	8	.35
300	7	.39
400	6	.47

$nodes = \mathbf{451} \quad links = \mathbf{713}$

Movies

Power Method

Iterations	Time
17	.40

Iterative Aggregation

$\lvert G \rvert$	Iterations	Time
5	12	.39
10	12	.37
15	11	.36
20	11	.35
25	11	.31
50	9	.31
100	9	.33
200	8	.35
300	7	.39
400	6	.47

$nodes = 451 \quad links = 713$

MathWorks

Power Method

Iterations	Time
54	1.25

Iterative Aggregation

$\lvert G \rvert$	Iterations	Time
5	53	1.18
10	52	1.29
15	52	1.23
20	42	1.05
25	20	1.13
300	11	.83
400	10	1.01

$nodes = \mathbf{517} \quad links = \mathbf{13,531}$

MathWorks

Power Method

Iterations	Time
54	1.25

Iterative Aggregation

$\lvert G \rvert$	Iterations	Time
5	53	1.18
10	52	1.29
15	52	1.23
20	42	1.05
25	20	1.13
50	18	.70
100	16	.70
200	13	.70
300	11	.83
400	10	1.01

$nodes = \mathbf{517} \quad links = \mathbf{13,531}$

Abortion

Power Method

Iterations	Time
$\mathbf{106}$	$\mathbf{37.08}$

Iterative Aggregation

$\vert G\vert$	Iterations	Time
$\mathbf{5}$	$\mathbf{109}$	$\mathbf{38.56}$
$\mathbf{10}$	$\mathbf{105}$	$\mathbf{36.02}$
$\mathbf{15}$	$\mathbf{107}$	$\mathbf{38.05}$
$\mathbf{20}$	$\mathbf{107}$	$\mathbf{38.45}$
$\mathbf{25}$	$\mathbf{97}$	$\mathbf{34.81}$
$\mathbf{50}$	$\mathbf{53}$	$\mathbf{18.80}$
$\mathbf{250}$	$\mathbf{12}$	$\mathbf{5.62}$
$\mathbf{500}$	$\mathbf{6}$	$\mathbf{5.21}$
$\mathbf{750}$	$\mathbf{5}$	$\mathbf{10.22}$
$\mathbf{1000}$	$\mathbf{5}$	$\mathbf{14.61}$

$nodes = \mathbf{1,693} \quad links = \mathbf{4,325}$

Abortion

Power Method

Iterations	Time
106	37.08

Iterative Aggregation

$\|G\|$	Iterations	Time
5	109	38.56
10	105	36.02
15	107	38.05
20	107	38.45
25	97	34.81
50	53	18.80
100	13	5.18
250	12	5.62
500	6	5.21
750	5	10.22
1000	5	14.61

$nodes = \mathbf{1,693} \quad links = \mathbf{4,325}$

Genetics

Power Method

Iterations	Time
$\mathbf{92}$	$\mathbf{91.78}$

Iterative Aggregation

$\|G\|$	Iterations	Time
$\mathbf{5}$	$\mathbf{91}$	$\mathbf{88.22}$
$\mathbf{10}$	$\mathbf{92}$	$\mathbf{92.12}$
$\mathbf{20}$	$\mathbf{71}$	$\mathbf{72.53}$
$\mathbf{50}$	$\mathbf{25}$	$\mathbf{25.42}$
$\mathbf{100}$	$\mathbf{19}$	$\mathbf{20.72}$
$\mathbf{250}$	$\mathbf{13}$	$\mathbf{14.97}$
$\mathbf{1000}$	$\mathbf{5}$	$\mathbf{17.76}$
$\mathbf{1500}$	$\mathbf{5}$	$\mathbf{31.84}$

$nodes = \mathbf{2,952} \quad links = \mathbf{6,485}$

Genetics

Power Method

Iterations	Time
92	91.78

Iterative Aggregation

$\|G\|$	Iterations	Time
5	91	88.22
10	92	92.12
20	71	72.53
50	25	25.42
100	19	20.72
250	13	14.97
500	7	11.14
1000	5	17.76
1500	5	31.84

$nodes = \mathbf{2,952} \quad links = \mathbf{6,485}$

Conclusions

- First updating algorithm to handle both element– and state–updates.
- Algorithm is very sensitive to partition.
- For PageRank problem, partition can be determined cheaply from old PageRanks.
- For general Markov updating, use $\mathbf{y}_2^T$ to determine partition. When too expensive, approximate adaptively with Aitken's δ^2 or difference of successive iterates.
- Improvements
  - Practical
    - Optimize G-set
    - Accelerate Smoothing
  - Theoretical
    - Relationship between partitioning by $\mathbf{y}_2^T$ and $\lambda_2(\mathbf{S}_2)$ not well-understood.
- Predict algorithm and partitioning by old $\boldsymbol{\pi}^T$ will work very well on other scale-free networks.