Lecture 18:

Application of EigenProblems: Markov Matrices and the Google page-rank algorithm

Outline:

1) Markov-Matrices and the Iterative map $\underline{p}_{k+1}=M\underline{p}_k$
2) Example: 2x2 Markov chain
3) Eigenvalues and eigenvectors of Markov Matrices
4) Google as world's largest Eigen problem: The PageRank Algorithm
    A) Overview of Search Engine requirements
    B) Construction of the Google Matrix G (a really big Markov Matrix)
    C) Iteration and the power method
5) Matlab Demo

Markov Matrices:

Definition: a Markov Matrix M contains all positive elements ($M_{ij}>0$) and has column sums=1.

2x2 Example: M=[.6 .8 ; .4 .2];

Markov Matrices:

M describes the discrete transitional probabilities between states in a Markov Chain given by the iterative map

$$\underline{p}_{k+1}=M\underline{p}_k$$

where $\underline{p}$ is a vector containing the discrete probability of being in state p_i

2 state example: with p_0=[1 0]', M=[.6 .8 ; .4 .2]=[3 4 ; 2 1]/5

Question: What is the most probable state after n iterations? Is there a steady-state "fixed point" such that $\underline{p}=M\underline{p}$?

Markov Matrices: EigenValues and EigenVectors

Example: M=[.6 .8 ; .4 .2] (can show for M=[a b ; (1-a) (1-b)])

Markov Matrices: EigenValues and EigenVectors

Properties of Markov Matrices (Perron-Frobenius Theorem)

1) *the largest eigenvalue =1*
   (All other eigenvalues have $|\lambda|<1$)
2) *there is a unique corresponding "steady-state" eigenvector that satisfies $\underline{w}=M\underline{w}$ with $\Sigma\, w_i = 1$*
3) *in the limit $k \rightarrow \infty$ $M^k = \underline{w}\ \underline{1}^T$ (rank 1 matrix with $\underline{w}$ for columns)*
4) *i.e. the iterative map converges to $\underline{w}$ from any p_0*

Example: M=[a b ; (1-a) (1-b)], 0<a,b<1

Why Care?:

This idea is worth a Bajillion dollars

The Google Page Rank algorithm as a giant Markov Matrix Eigen Problem. (From Amy Langville: Link Analysis by Web Search engines)

Understanding the Google Page-Rank algorithm

Preliminaries: Elements of a Web Search Engine

1) *Web Crawlers: search pages and tabulate content*
2) *Indexing: Create an Inverted Index that returns all pages that contain a given word:*

   Example

   aardvark: pages 3, 54, 1990, 34000
   ...
   aztec : pages 3, 15, 16, 200, 765 ...
   baby : pages 3, 12, 20, 195, 765....
   etc...
3) *Querying: User inputs query (e.g. aztec baby) and search engine returns all pages that includes the query*

   Example: 3, 765, ??? (Google gives ~221,000 pages that match aztec baby)

Understanding the Google Page-Rank algorithm

Preliminaries: Elements of a Web Search Engine

4) *Ranking algorithms: How to decide which pages are the "ones you want"*

The Web pre and post-google!

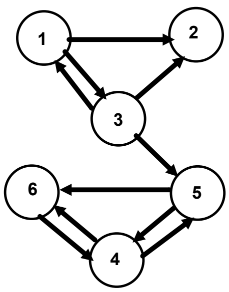
There are multiple ranking algorithms and this is a very hot topic of research (for obvious reasons). But the original Brin and Page "PageRank" algorithm is a lovely application of linear algebra...

So where's the matrix?

Understanding the Google Page-Rank algorithm

The web as a graph:
pages are nodes
links are arcs (or edges)

Example: a 6 page web (from Amy Langville)

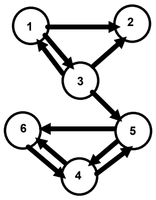


Page rank is essentially the probability that a random surfer will land on a page after traversing the web an infinite number of times!

Page Rank depends only on structure of outlinks...

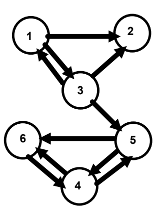
Understanding the Google Page-Rank algorithm

The Adjacency Matrix (outlinks)



Understanding the Google Page-Rank algorithm

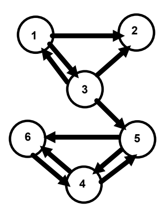
The Probabalistic Surfer Matrix H



Almost a Markov Matrix: but need to handle the "dead nodes"

Understanding the Google Page-Rank algorithm

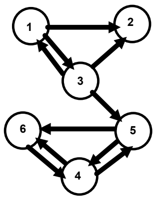
A fix for Dead Nodes: the random teleporter Matrix S



$S = H + \underline{v}\,\underline{a}^T$ (rank one update)...Still need to fix for cycles

Understanding the Google Page-Rank algorithm

Guaranteeing convergence: Need to add global connectivity (so the surfer can eventually traverse the entire web)



The Google Matrix G: add a random teleporter everywhere with probability (1-α)/N i.e.

$G = \alpha(H + \underline{v}\,\underline{a}^T) + (1-\alpha)\,\underline{v}\,\underline{1}^T$ =

Understanding the Google Page-Rank algorithm

Page Rank: G is now a convergent Markov Matrix, just find its steady solution $\underline{p}=G\underline{p}$

Approach 1) Solve $G\underline{p}=\underline{p}$ or $(G-I)p=0$; (find null space of G-I)
(Bad idea: G is dense 8 billion by 8 billion matrix)

Approach 2) Use the power method to solve

$\underline{p}_{k+1}=G\underline{p}_k \quad \underline{p}_0=\underline{v}$

Still not good as G is dense....however

Approach 3) Power method with

$\underline{p}_{k+1}=$

Good idea, H is very sparse and rank 1 update is cheapish

Understanding the Google Page-Rank algorithm

Convergence still depends on size of second eigenvalue (and alpha)

unclear how much work it takes to converge but in 2002 order

report 50-100 iterations
days to converge

(and there's probably quite a bit more living in page rank these days)

But lots of good math and Computational Science in Link Analysis

Eigenvalues and Eigenvectors

Applications of Iterative maps

Example #2: Iterative methods for solving $A\underline{x}=\underline{b}$