Lecture 18: Application of EigenProblems: Markov Matrices and the page-rank algorithm

Outline:

1) Markov-Matrices and the Iterative map $p_{k+1} = Mp_k$
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

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

2x2 Example: $M=\begin{pmatrix} .6 & .8 \\ .4 & .2 \end{pmatrix}$;

Markov Matrices: $M$ describes the discrete transitional probabilities between states in a Markov Chain given by the iterative map $B_{k+1} = MB_k$

where $B$ is a vector containing the discrete probability of being in state $p_i$

2 state example: with $p_0=\begin{pmatrix} 1 & 0 \end{pmatrix}$, $M=\begin{pmatrix} .6 & .8 \\ .4 & .2 \end{pmatrix}$

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

Markov Matrices: EigenValues and EigenVectors

Example: $M=\begin{pmatrix} .6 & .8 \\ .4 & .2 \end{pmatrix}$ (can show for $M=\begin{pmatrix} a & b \\ (1-a) & (1-b) \end{pmatrix}$)
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 \( w = Mw \) with \( \sum w_i = 1 \)
3) in the limit \( k \to \infty \), \( M^k = w^T \) (rank 1 matrix with \( w \) for columns)
4) i.e. the iterative map converges to \( w \) from any \( \delta_0 \)
Example: \( M = \begin{bmatrix} a & b \\ (1-a) & (1-b) \end{bmatrix} \), \( 0 < a, b < 1 \)

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...

The Probabilistic Surfer Matrix \( H \)

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

Understanding the Google Page-Rank algorithm

The Adjacency Matrix (outlinks)

Understanding the Google Page-Rank algorithm

A fix for Dead Nodes: the random teleporter Matrix \( S \)

\[ S = H + \frac{1}{m} \mathbf{1} \mathbf{1}^T \] (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)

\[ G = \alpha (H + \frac{\mathbf{v} \mathbf{v}^T}{N}) + (1-\alpha) \frac{1}{N} \mathbf{1}^T = \]

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

Page Rank: G is now a convergent Markov Matrix, just find its steady solution \(G = \mathbf{Gp}\)

Approach 1) Solve \(G\mathbf{p} = \mathbf{p}\) or \((G-I)\mathbf{p} = \mathbf{0}\); (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

\[ \mathbf{p}_{k+1} = G\mathbf{p}_k \]

\[ \mathbf{p}_0 = \mathbf{v} \]

Still not good as G is dense...however

Approach 3) Power method with

\[ \mathbf{p}_{k+1} = \mathbf{H}\mathbf{p}_k \]

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 \(Ax = b\)