Lecture 21

Google and Markov Chains, Power Method, SVD

David Semeraro

University of Illinois at Urbana-Champaign

November 7, 2013

Randomly Walking with Google

- start at any webpage
- randomly select a link and follow
- repeat
- what are the outcomes?

The outcomes of such a random walk are:

- a dead end on a page with no outgoing links
- a cycle where you end up where you began: known as a *Markov chain* or *Markov process*.
- The limiting probability that an infinitely dedicated random surfer visits any particular page is its PageRank.
- A page has high rank if other pages with high rank link to it.

Markov Chains

- Markov chains can model the behavior of a system that depends only on the previous experiment or state.
- That is, the next state of the system depends only on the current state where the outcome of each experiment is one of a discrete set of states.
- Markov chains require a transition matrix, P, where $P(j, i)$ equals the probability of going from state i to state j.

Back to Google

- Let W be the set of Web pages that can reached by following a chain of hyperlinks starting from a page at Google.
- Let n be the number of pages in W.
- The set W actually varies with time, by the end of 2005, n was over 10 billion.
- Let G be the $n \times n$ connectivity matrix of W, that is, $G_{i,j}$ is 1 if there is a hyperlink from page i to page j and 0 otherwise.
- Let H be G with each row i divided by the number of outgoing links from node i.
- The matrix H is huge, but very sparse; its number of nonzeros is the total number of hyperlinks in the pages in W.

Google and Probability

- Let c_j and r_i be the column and row sums of G, respectively. That is,

$$c_j = \sum_i G_{i,j}, \qquad r_i = \sum_j G_{i,j}$$

- Then c_k and r_k are the indegree and outdegree of the k-th page. In other words, c_k is the number of links into page k and r_k is the number of links from page k.
- Let p be the fraction of time that the random walk follows a link.
- Google typically takes this to be $p = 0.85$.
- Then $1 - p$ is the fraction of time that an arbitrary page is chosen.

Google meets Markov

- Let A be an $n \times n$ matrix whose elements are $A_{i,j} = pG_{i,j}/c_j + \delta$ where $\delta = (1-p)/n$.
- This matrix is the transition matrix of the Markov chain of a random walk!
- Notice that A comes from scaling the connectivity matrix by is column sums.
- The j-th column is the probability of jumping from the j-th page to the other pages on the Web.

The problem

Can write A, the transition matrix, as

$$A = pGD + ez^T$$

where e is the vector of all ones and where ez^T account for dead linked pages and

$$D_{jj} = 1/c_j \text{ (or 0)} \quad z_j = \delta \text{ (or } 1/n)$$

Then $x = Ax$ can be written

$$(I - pGD)x = (z^T x)e = \gamma e$$

and we can scale x such that $\gamma = 1$

Eigenvectors and Google

Find $x = Ax$ and the elements of x are Google's PageRank. Remember $n > 10^{10}$ (as of 2005) and growing (a Google blog post claimed $n > 10^{12}$ in 2008) .

For any particular query, Google finds pages on the Web that match the query. The pages are then listed in the order of their PageRank.

Goal

- Find $x = Ax$ and the elements of x are Google's PageRank.
- For a matrix A, the scalar-vector pairs (λ, v) such that $Av = \lambda v$ are eigenvalue-eigenvectors.
- Topic #1: Power Method
- Topic #2: Singular Value Decomposition (SVD)

Power Method

Suppose that A is $n \times n$ and that the eigenvalues are ordered:

$$|\lambda_1| > |\lambda_2| \geqslant |\lambda_3| \geqslant \cdots \geqslant |\lambda_n|$$

Assuming A is nonsingular, we have a linearly independent set of v_i such that $Av_i = \lambda_i v_i$.

Goal

Computing the value of the largest (in magnitude) eigenvalue, λ_1.

Power Method

Take a guess at the associated eigenvector, x_0. We know

$$x^{(0)} = c_1 v_1 + \cdots + c_n v_n$$

Since the guess was random, start with all $c_j = 1$:

$$x^{(0)} = v_1 + \cdots + v_n$$

Then compute

$$\begin{aligned} x^{(1)} &= Ax^{(0)} \\ x^{(2)} &= Ax^{(1)} \\ x^{(3)} &= Ax^{(2)} \\ &\vdots \\ x^{(k+1)} &= Ax^{(k)} \end{aligned}$$

Power Method

Or $x^{(k)} = A^k x^{(0)}$. Or

$$\begin{aligned} x^{(k)} &= A^k x^{(0)} \\ &= A^k v_1 + \cdots + A^k v_n \\ &= \lambda_1^k v_1 + \ldots \lambda_n^k v_n \end{aligned}$$

And this can be written as

$$x^{(k)} = \lambda_1^k \left(v_1 + \left(\frac{\lambda_2}{\lambda_1} \right)^k v_2 + \cdots + \left(\frac{\lambda_n}{\lambda_1} \right)^k v_n \right)$$

So as $k \to \infty$, we are left with

$$x^{(k)} \to \lambda^k v_1$$

The Power Method (with normalization)

```
1 for k = 1 to kmax
2     y = Ax
3     r = φ(y)/φ(x)
4     x = y/‖y‖∞
```

- often $\phi(x) = x_1$ is sufficient
- r is an estimate of the eigenvalue; x the eigenvector

Inverse Power Method

- We now want to find the smallest eigenvalue
- $Av = \lambda v \quad \Rightarrow \quad A^{-1}v = \frac{1}{\lambda}v$
- So "apply" power method to A^{-1} (assuming a distinct smallest eigenvalue)
- $x^{(k+1)} = A^{-1}x^{(k)}$
- Easier with $A = LU$
- Update RHS and backsolve with U:

$$Ux^{(k+1)} = L^{-1}x^{(k)}$$

SVD: motivation

SVD uses in practice:

1. Search Technology: find closely related documents or images in a database
2. Clustering: aggregate documents or images into similar groups
3. Compression: efficient image storage
4. Principal axis: find the main axis of a solid (engineering/graphics)
5. Summaries: Given a textual document, ascertain the most representative tags
6. Graphs: partition graphs into subgraphs (graphics, analysis)

SVD: Singular Value Decomposition

SVD takes an $m \times n$ matrix A and factors it:

$$A = USV^T$$

where U $(m \times m)$ and V $(n \times n)$ are orthogonal and S $(m \times n)$ is diagonal.

Definition

A is orthogonal if $A^TA = AA^T = I$.

S is made up of "singular values":

$$\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_r \geqslant \sigma_{r+1} = \cdots = \sigma_p = 0$$

Here, $r = rank(A)$ and $p = min(m, n)$.

Diagonalizing a matrix

We want to factorize A into U, S, and V^T. First step: find V. Consider

$$A = USV^T$$

and multiply by A^T

$$A^TA = (USV^T)^T(USV^T) = VS^TU^TUSV^T$$

Since U is orthogonal

$$A^TA = VS^2V^T$$

This is called a similarity transformation.

Definition

Matrices A and B are similar if there is an invertible matrix Q such that

$$Q^{-1}AQ = B$$

Theorem

Similar matrices have the same eigenvalues.

Proof

$$
\begin{aligned}
Bv &= \lambda v \\
Q^{-1}AQv &= \lambda v \\
AQv &= \lambda Qv \\
Aw &= \lambda w.
\end{aligned}
$$

Further, if v is an eigenvector of B, Qv is an eigenvector of A.

So far...

Need $A = USV^T$

Look for V such that $A^TA = VS^2V^T$. Here S^2 is diagonal.

If A^TA and S^2 are similar, then they have the same eigenvalues. So the diagonal matrix S^2 is just the eigenvalues of A^TA and V is the matrix of eigenvectors. To see the latter, note that since S^2 is diagonal, the eigenvectors

are e_i, and V^Te_i is just the ith column of V^T.

Similarly...

Now consider

$$A = USV^T$$

and multiply by A^T from the right

$$AA^T = (USV^T)(USV^T)^T = USV^TVS^TU^T$$

Since V is orthogonal

$$AA^T = US^2U^T$$

Now U is the matrix of eigenvectors of AA^T.

In the end...

We get

$$A = \begin{bmatrix} \vdots & \vdots & \vdots \\ u_1 & \dots & u_m \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_r & \\ & & & \ddots \\ & & & & 0 \end{bmatrix} \begin{bmatrix} \dots & v_1^T & \dots \\ \dots & \vdots & \dots \\ \dots & v_n^T & \dots \end{bmatrix}$$

Example

Decompose

$$A = \begin{bmatrix} 2 & -2 \\ 1 & 1 \end{bmatrix}$$

First construct $A^T A$:

$$A^T A = \begin{bmatrix} 2 & 1 \\ -2 & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 5 & -3 \\ -3 & 5 \end{bmatrix}$$

Eigenvalues: $\lambda_1 = 8$ and $\lambda_2 = 2$. So

$$S^2 = \begin{bmatrix} 8 & 0 \\ 0 & 2 \end{bmatrix} \quad \Rightarrow \quad S = \begin{bmatrix} 2\sqrt{2} & 0 \\ 0 & \sqrt{2} \end{bmatrix}$$

Example

Now find V^T and U. The columns of V^T are the eigenvectors of A^TA.

- $\lambda_1 = 8$: $(A^TA - \lambda_1 I)v_1 = 0$

$$\Rightarrow \begin{bmatrix} -3 & -3 \\ -3 & -3 \end{bmatrix} v_1 = 0 \quad \Rightarrow \quad \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} v_1 = 0 \quad \Rightarrow \quad v_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -\sqrt{2}/2 \\ \sqrt{2}/2 \end{bmatrix}$$

- $\lambda_2 = 2$: $(A^TA - \lambda_2 I)v_2 = 0$

$$\Rightarrow \begin{bmatrix} 3 & -3 \\ -3 & 3 \end{bmatrix} v_2 = 0 \quad \Rightarrow \quad \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix} v_2 = 0 \quad \Rightarrow \quad v_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{bmatrix}$$

- Finally:

$$V = \begin{bmatrix} -\sqrt{2}/2 & \sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix}$$

Example

Now find U. The columns of U are the eigenvectors of AA^T.

- $\lambda_1 = 8$: $(AA^T - \lambda_1 I)u_1 = 0$

$$\Rightarrow \begin{bmatrix} 0 & 0 \\ 0 & -6 \end{bmatrix} u_1 = 0 \quad \Rightarrow \quad \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} u_1 = 0 \quad \Rightarrow \quad u_1 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

- $\lambda_2 = 2$: $(AA^T - \lambda_2 I)u_2 = 0$

$$\Rightarrow \begin{bmatrix} 6 & 0 \\ 0 & 0 \end{bmatrix} u_2 = 0 \quad \Rightarrow \quad \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} u_2 = 0 \quad \Rightarrow \quad u_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

- Finally:

$$U = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

- Together:

$$A = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2\sqrt{2} & 0 \\ 0 & \sqrt{2} \end{bmatrix} \begin{bmatrix} -\sqrt{2}/2 & \sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix}$$

SVD: who cares?

How can we actually *use* $A = USV^T$? We can use this to represent A with far fewer entries...

Notice what $A = USV^T$ looks like:

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T + 0 u_{r+1} v_{r+1}^T + \cdots + 0 u_p v_p^T$$

This is easily truncated to

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T$$

see `svd_test.py`
What are the savings?

- A takes $m \times n$ storage
- using k terms of U and V takes $k(1 + m + n)$ storage