Linear Algebra Methods for Data Mining

Saara Hyvönen, Saara.Hyvonen@cs.helsinki.fi

Spring 2007

Mining the web: PageRank and Power Iteration

LSI and related methods

Given a set of k basis vectors $(\mathbf{x}_1 \ldots \mathbf{x}_k) = \mathbf{X}_k$, express the data matrix $\mathbf{A}$ in terms of the columns of this basis.. That is, solve the least squares problem:

$$\min_{\mathbf{Y}} \|\mathbf{A} - \mathbf{X}_k\mathbf{Y}\|.$$

The basis vectors $\mathbf{x}_j$ may be the left singular vectors, or the centroids of some clustering of the data vectors into k clusters.

Also express the query in the new basis, and perform query matching in the new subspace spanned by the $\mathbf{x}_j$, $j = 1 \ldots k$.

Note:

- SVD gives the best *approximation* of $\mathbf{A}$ in terms of minimizing distance between $\mathbf{A}$ and $\mathbf{X}_k\mathbf{Y}$ (in the euclidean norm).
- But: centroids seem to give a better (reduced dimensional) *representation* of the original documents (at least for well clustered data).
- That is, with centroids it seems possible to use even smaller reduced dimension than SVD to get results of similar quality.
- Centroids play a very similar role as do singular vectors in LSI.
- **BUT** :Centroids/Concept vectors are sparse. Singular vectors are essentially full.

Query matching

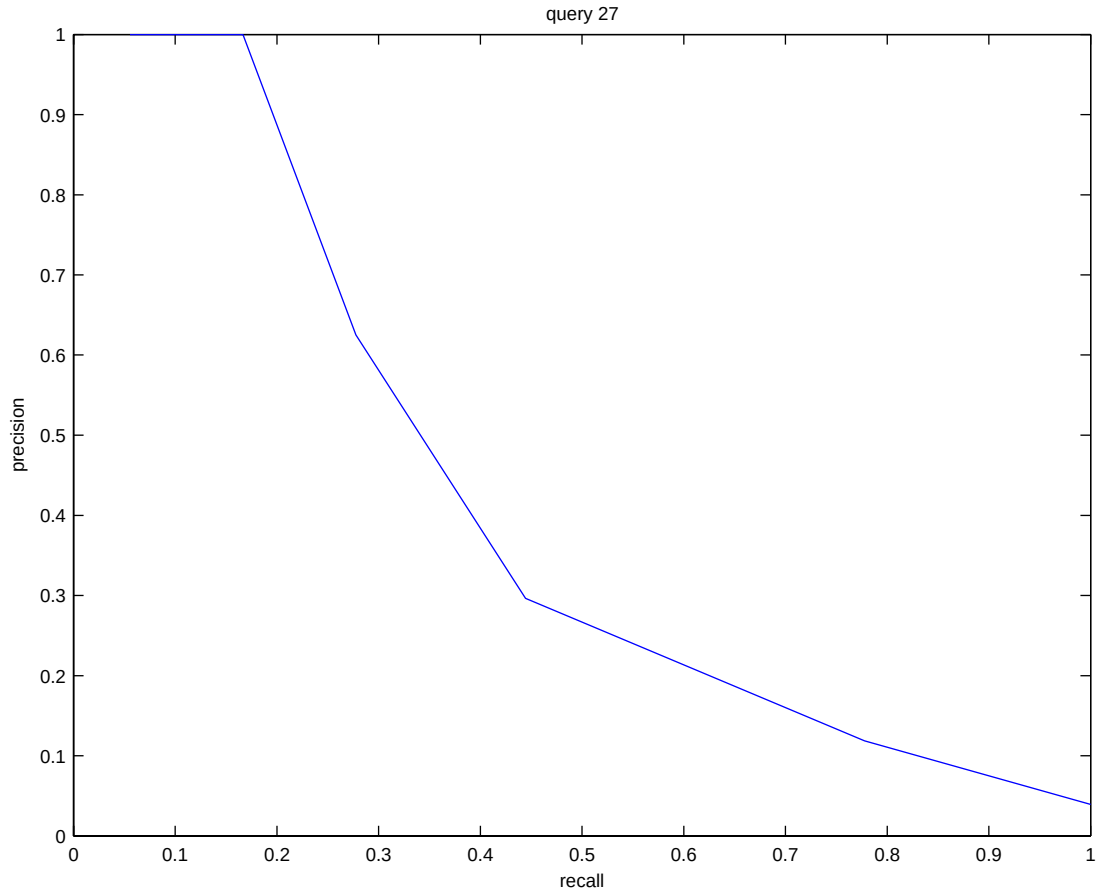
Query matching is usually done using cosine similarity:

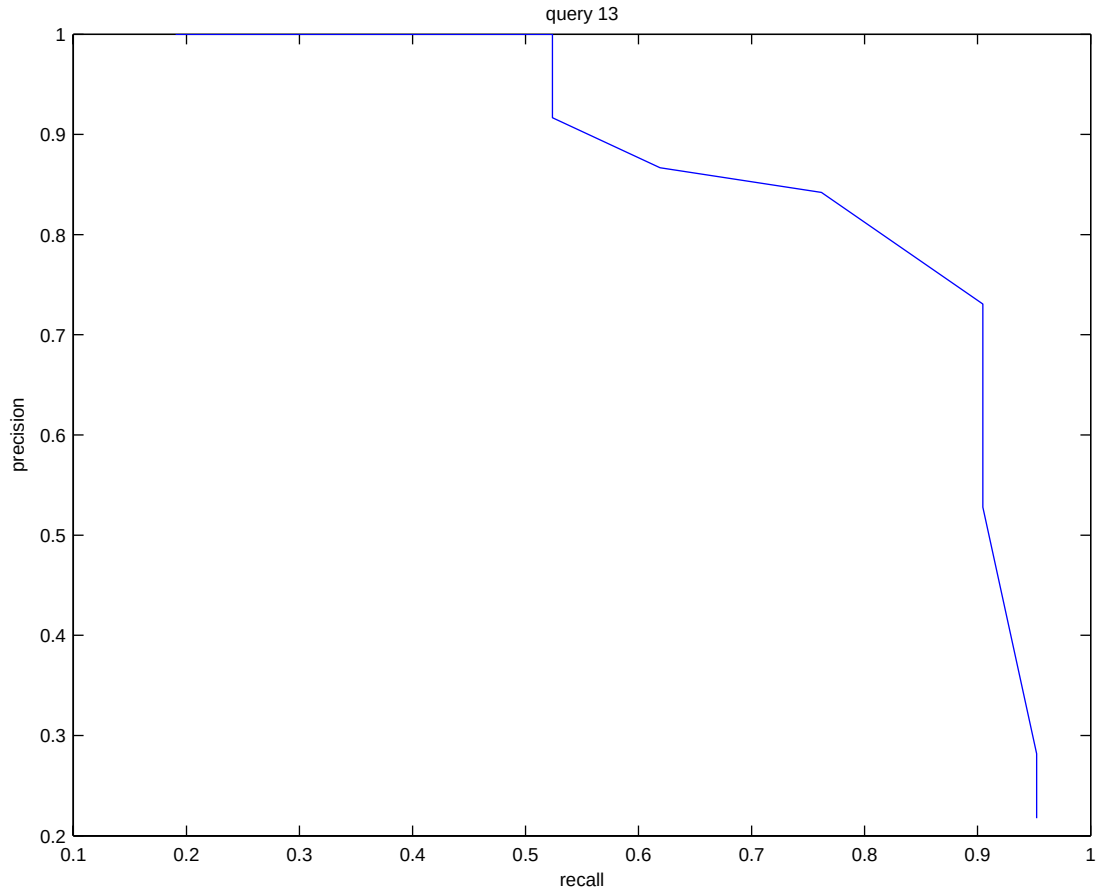
$$\cos(\theta(\mathbf{q}, \mathbf{a}_j)) = \frac{\mathbf{q}^T \mathbf{a}_j}{\|\mathbf{q}\| \|\mathbf{a}_j\|} \geq tol$$

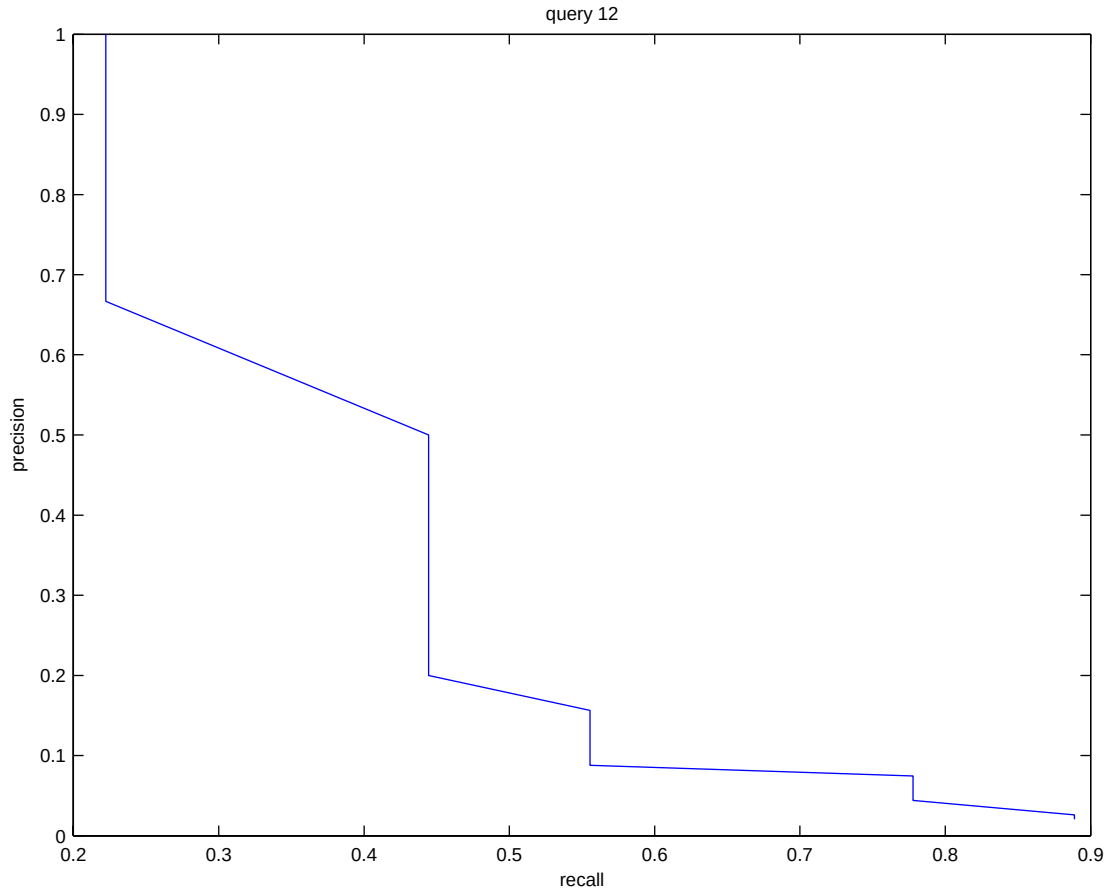
Varying tolerance you vary the *precision* and *recall*:

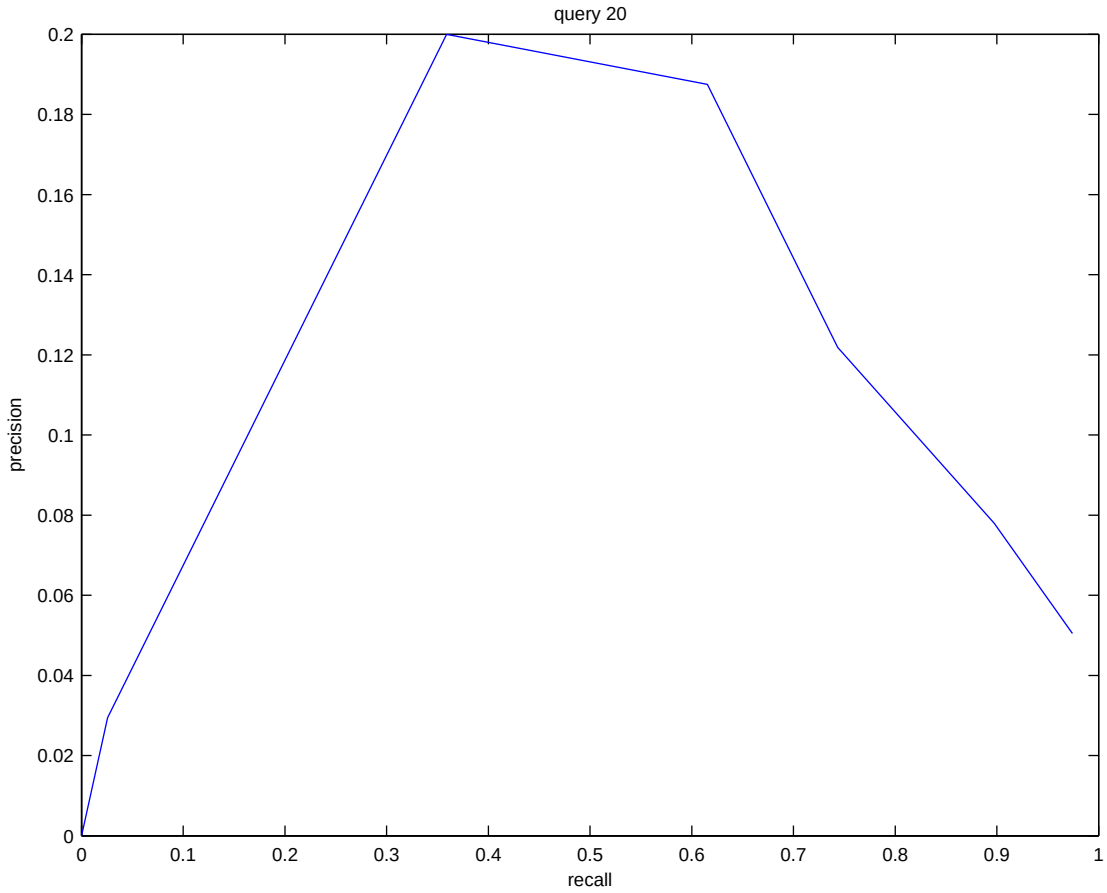
$$P = \frac{D_r}{D_t} \qquad R = \frac{D_r}{N_r},$$

D_r is the number of relevant documents retrieved, D_t is the total number of documents retrieved, N_r is the total number of relevant documents in the database. **Wanted:** high precision and high recall!


query 27
precision
recall
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


query 13
precision
recall
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


query 12
precision
recall
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9


query 20
precision
recall
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Searching the Internet

- The Internet is huge: billions of web pages!
- E.g. google makes searches among 8 058 044 651 web pages.
- Typical search phrase (query) is under-specified: lots of matches.
- Not all of them are interesting. How to decide which are?

Example: search for university on the Internet

www.google.com
Results 1 - 10 of about 384,000,000
for university:

Welcome to Harvard University

Stanford University

University of Cambridge

University of Michigan

University of Toronto -- Home Page

Yale University | Welcome to Yale Univer

University of Washington

www.yahoo.com
Results 1 - 10 of about 307,000,000
for university

World Wide Colleges and Universites

American Universities

Universities Worldwide

U.S. Universities, by State

College and University Home Pages

Canadian Universities

Universities UK

Searching the Internet

Step 1: traditional text processing: find all web pages containing the words of the query.

Step 2: Order the pages according to relevance. PageRank, HITS.

PageRank

Each page has a measure of *prestige* (or *pagerank*) that is independent of any query or textual content.

Idea: number of links to and from a page give information about the importance of a page.

Roughly, the prestige of a page is proportional to the sum of the prestige scores of pages linking to it.

All web pages ordered from 1 to n. Consider page i.

Denote by

- O_i: the set of pages that i is linked to, the *outlinks*.
- N_i: the number of outlinks.
- I_i: *inlinks*, i.e. the pages that have an outlink to i.

The more inlinks, the more important the page? Easy to manipulate!

Define rank of i in such a way, that if a highly ranked page j is linked to i, $j \in I_i$, then this adds to the importance of i:

$$r_i = \sum_{j \in I_i} \frac{r_j}{N_j}$$

Note: rank of page j is divided evenly among its outlinks.

Definition of rank is recursive: cannot be solved directly. Instead, use fixed point iteration:

Guess an initial ranking vector r^0, then iterate:

$$r_i^{k+1} = \sum_{j \in I_i} \frac{r_j^k}{N_j}, \quad k = 0, 1, ...$$

Problem

Start with intial guess for page ranks r^0 and use fixed point iteration:

$$r_i^{k+1} = \sum_{j \in I_i} \frac{r_j^k}{N_j}, \quad k = 0, 1, ...$$

Question: will this converge?

Matrix formulation

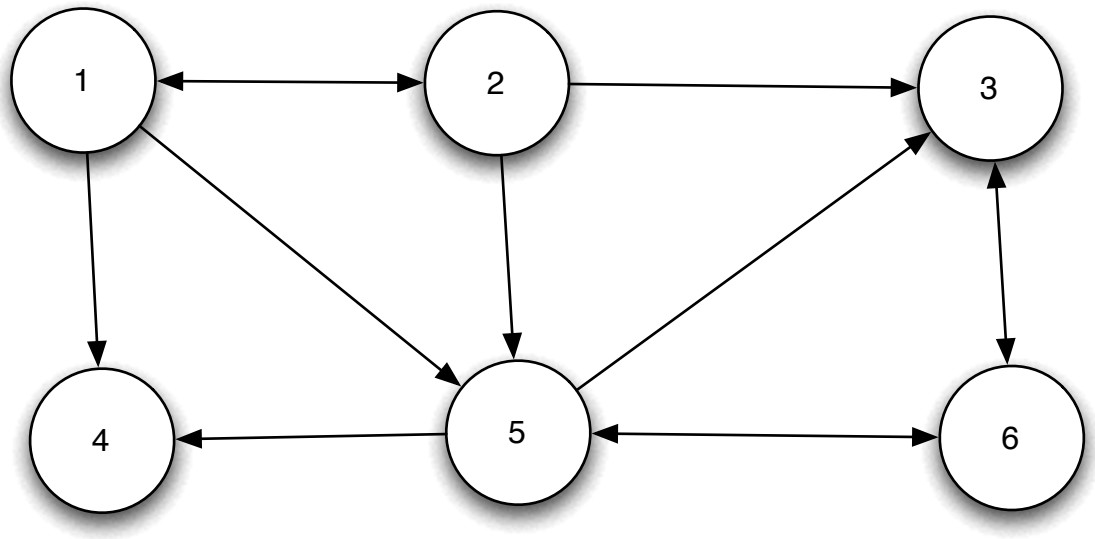
Reformulate the problem as an eigenvalue problem for a matrix representing the graph of the Internet:

Let $\mathbf{Q}$ be a $n \times n$ matrix such that

$$\mathbf{Q}_{ij} = \begin{cases} 1/N_j & \text{it there is a link from } j \text{ to } i, \\ 0 & \text{otherwise.} \end{cases}$$

- row i has nonzero element in positions corresponding to inlinks I_i.
- column j has nonzero elements in positions corresponding to outlinks of j (N_j in total)
- if page j has outlinks, the sum of the elements in the j^{th} column $=1$.

Example



The corresponding matrix is

$$\mathbf{Q} = \begin{pmatrix} 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 & \frac{1}{3} & 0 \end{pmatrix}$$

Now

$$r_i = \sum_{j \in I_i} \frac{r_j}{N_j}$$

becomes

$$\lambda \mathbf{r} = \mathbf{Q}\mathbf{r},$$

so $\mathbf{r}$ is the eigenvector of $\mathbf{Q}$ corresponding to the eigenvalue $\lambda = 1$, and the iteration

$$r_i^{k+1} = \sum_{j \in I_i} \frac{r_j^k}{N_j}, \quad k = 0, 1, ...$$

becomes

$$\mathbf{r}^{k+1} = \mathbf{Q}\mathbf{r}^k, \quad k = 0, 1, ...$$

But is $\lambda = 1$ an eigenvalue of $\mathbf{Q}$? Does the iteration converge?

Random walks and Markov Chains

Consider a web surfer clicking on hyperlinks forever, choosing the next page among the outlinks of the current page with equal probability.

If the surfer is on page i, then what is the probability that he will continue to page j?

If there is no link from page i to page j, the probability is zero.

If there is a link from page i to page j, then the probability is $1/N_i$.

Assume the initial probability distribution $\mathbf{p}^0$,
i.e. the probability we start from page i is $\mathbf{p}_i^0$.

Then after one step the probability we are on page j is $\mathbf{p}_j^1$, where

$$\mathbf{p}^1 = \mathbf{Q}^T \mathbf{p}^0.$$

After k steps we get

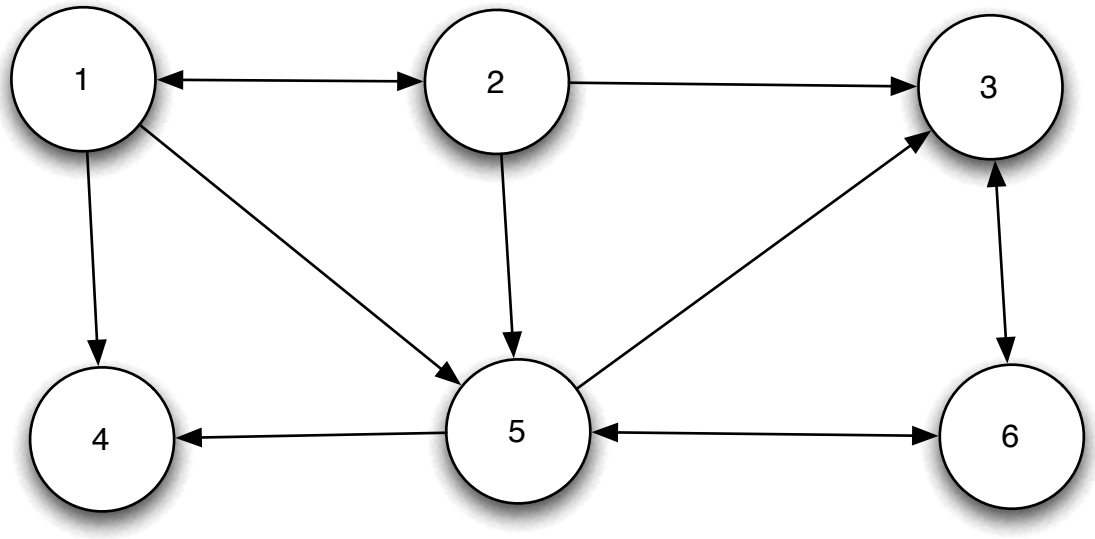
$$\mathbf{p}^k = \mathbf{Q}^T \mathbf{p}^{k-1}.$$

Our random walk has induces a Markov chain with transition matrix $\mathbf{Q}^T$.

Markov chain model

Random process, where next state is determined completely from present state. Process has no memory.

What about the nodes with not outlinks? A random surfer should never get stuck! So these are not allowed in random walk model.



Solution: if there are no outlinks, equal probability to move to any other page in the net:

Define the vector $\mathbf{d}$, the elements of which are

$$\mathbf{d}_j = \begin{cases} 1 & \text{if } N_j = 0, \\ 0 & \text{otherwise,} \end{cases} \qquad j = 1, ..., n,$$

and the vector $\mathbf{e} = (1\ 1\ ...\ 1)^T \in \mathbb{R}^n$. The modified matrix is defined

$$\mathbf{P} = \mathbf{Q} + \frac{1}{n}\mathbf{e}\mathbf{d}^T.$$

What we have now is a *column-stochastic matrix* $\mathbf{P}$.

Column-stochastic matrix

Matrix with non-negative elements such that the elements of each column sum up to 1.

Proposition. A column-stochastic matrix $\mathbf{P}$ satisfies

$$\mathbf{e}^T\mathbf{P} = \mathbf{e}^T,$$

where $\mathbf{e} = (1\ 1\ ...\ 1)^T \in \mathbb{R}^n$.

Example

The matrix in our previous example is modified to

$$\mathbf{P} = \begin{pmatrix} 0 & \frac{1}{3} & 0 & \frac{1}{6} & 0 & 0 \\ \frac{1}{3} & 0 & 0 & \frac{1}{6} & 0 & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{6} & \frac{1}{3} & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & \frac{1}{6} & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{6} & 0 & \frac{1}{2} \\ 0 & 0 & 1 & \frac{1}{6} & \frac{1}{3} & 0 \end{pmatrix}$$

Remember, that we wanted to define the rank of each page by

$$\mathbf{r} = \mathbf{Q}\mathbf{r},$$

but we didn't know if $\mathbf{Q}$ had the eigenvalue $\lambda = 1$.

For our modified matrix $\mathbf{P}$ the rank of each page would be defined by

$$\mathbf{r} = \mathbf{P}\mathbf{r}.$$

Now we know that $\mathbf{P}$ has the eigenvalue 1 (why?), but we want the eigenvector corresponding to one to be *unique.* Alas, this is still not guaranteed!

Why not?

Think again of $\mathbf{P}$ as the transition matrix of a Markov chain model. Now the eigenvector $\mathbf{r}$ corresponding to the eigenvalue $\lambda = 1$:

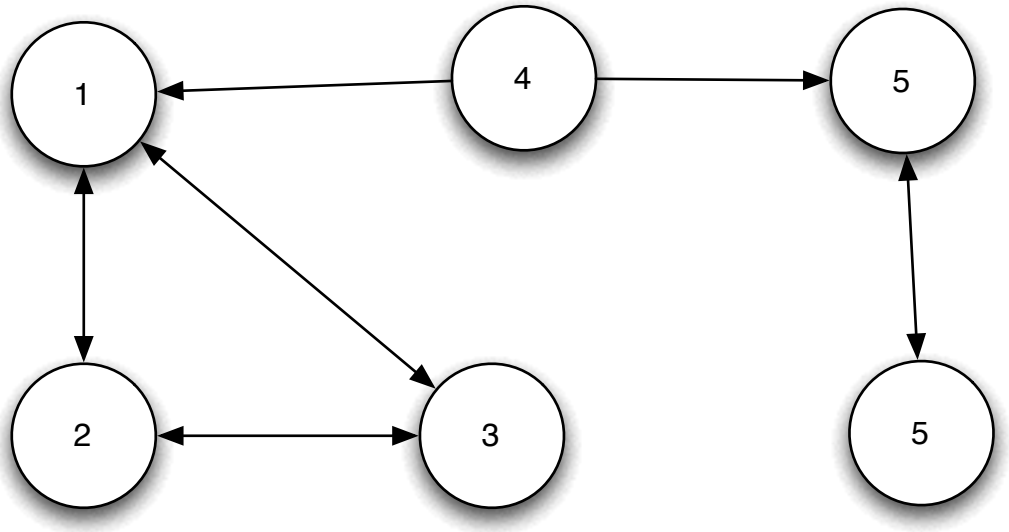
$$\mathbf{Pr} = \mathbf{r}$$

corresponds to a stationary probability distribution for the Markov chain.

That is, the element $\mathbf{r}_i$ in position i is the probability that after a large number of steps the randow walker is at web page i.

But there might be more than one such stationary distribution! Consider the following example:

Example: reducible matrix



In this example, the random walker who entered the left part of the link graph will never get out of it, and the same is true for the random walker who entered the right part.

The matrix corresponding to this graph is

$$\mathbf{P} = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

has the eigenvalue $\lambda = 1$ with the eigenvector $\mathbf{v} = (1\ 1\ 1\ 0\ 0\ 0)^T$...

...and the eigenvector $\mathbf{v} = (0\ 0\ 0\ 0\ 1\ 1)^T$!!

Irreducibility

To ensure existence of a unique eigenvector corresponding to the eigenvalue 1 the matrix must be *irreducible*: there must be a directed path from every node to every other node of the graph.

That is, the random walker must never get stuck.

Hardly true for the internet!

Solution: teleportation.

To ensure the random walker does not get stuck one inserts a fake low-probability transition all over the place.

At each node,

1. With probability $1 - \alpha$ the surfer jumps to a random place on the web,

2. With probability α the surfer decides to choose, uniformly at random, an outlink of the current node.

In matrix terms, we get the (further) modified matrix

$$\mathbf{A} = \alpha\mathbf{P} + (1-\alpha)\frac{1}{n}\mathbf{e}\mathbf{e}^T$$

for some $0 \leq \alpha \leq 1$.

The probability $1-\alpha$ of the surfer to jump to a random place is sometimes referred to as teleportation.

The matrix $\mathbf{A} = \alpha\mathbf{P} + (1-\alpha)\frac{1}{n}\mathbf{e}\mathbf{e}^T$ is also column-stochastic:

$$\mathbf{e}^T\mathbf{A} = \ \ldots\ = \mathbf{e}^T.$$

(Fill in the missing steps.)

Again look at the solution of

$$\mathbf{A}\mathbf{r} = \mathbf{r}.$$

This time it is well defined!

Theorem 1. The largest eigenvalue in magnitude λ_1 of a column-stochastic matrix $\mathbf{P}$ is equal to 1. The corresponding eigenvector has all nonzero elements, and is the only eigenvector with this property.

Doesn't tell about multiplicity of eigenvalue $\lambda = 1$! For it to be simple, more assumptions are needed. That is why we formed $\mathbf{A}$. Thankfully, the eigenvalues of $\mathbf{A}$ can be expressed in terms of the eigenvalues of $\mathbf{P}$:

Theorem 2. Let the eigenvalues of the column-stochastic matrix $\mathbf{P}$ be $\{1,\ \lambda_2,\ \lambda_3,\ ...,\ \lambda_n\}$. Then the eigenvalues of $\mathbf{A} = \alpha\mathbf{P} + (1-\alpha)\frac{1}{n}\mathbf{e}\mathbf{e}^T$ are $\{1,\ \alpha\lambda_2,\ \alpha\lambda_3,\ ...,\ \alpha\lambda_n\}$.

Consequences

Even if $\mathbf{P}$ has a multiple eigenvalue equal to 1 (which is the case for the Google matrix), the second largest eigenvalue in mangitude of $\mathbf{A}$ is α!

This result

- Guarantees the uniqueness of the pagerank.
- Also tells about how fast it can be computed numerically.

Example

Consider the reducible example

$$\mathbf{P} = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

with the two eigenvectors corresponding to the eigenvalue 1. Form $\mathbf{A} = \alpha\mathbf{P} + (1-\alpha)\frac{1}{n}\mathbf{e}\mathbf{e}^T$, where $\alpha = 0.85$.

```
eP=eig(P)'
   -0.5000    1.0000   -0.5000    1.0000   -1.0000         0

e=ones(6,1);alfa=0.85;A=alfa*P+(1-alfa)/6*e*e';
[vA,eA]=eig(A);
vA =
    0.4468   -0.3651   -0.3536    0.0000    0.8165    0.0440
    0.4297   -0.3651    0.3536   -0.0000   -0.4082   -0.7281
    0.4297   -0.3651    0.3536   -0.0000   -0.4082    0.6841
    0.0572   -0.0000   -0.7071   -0.0000    0.0000    0.0000
    0.4690    0.5477    0.0000   -0.7071   -0.0000    0.0000
    0.4559    0.5477    0.3536    0.7071    0.0000   -0.0000

diag(eA)'
    1.0000    0.8500   -0.0000   -0.8500   -0.4250   -0.4250
```

Note

In $\mathbf{A} = \alpha\mathbf{P} + (1-\alpha)\frac{1}{n}\mathbf{e}\mathbf{e}^T$, the vector $\frac{1}{n}\mathbf{e}$ can be replaced by any non-negative vector $\mathbf{v}$ with norm $\|\mathbf{v}\|_1 = 1$:

$$\mathbf{A} = \alpha\mathbf{P} + (1-\alpha)\mathbf{v}\mathbf{e}^T$$

The vector $\mathbf{v}$ can be chosen to be biased towards (or against) certain kinds of web pages.

Often referred to as a *personalization vector*.

(Check that $\mathbf{A}$ is a column-stochastic matrix!)

How to compute the pagerank vector?

Task: solve the eigenvalue problem

$$\mathbf{A}\mathbf{r} = \mathbf{r},$$

where $\mathbf{r}$ is normalized $\|\mathbf{r}\|_1 = 1$.

(Note: in dealing with stochastic matrices and vectors that are probability distributions, it is natural to use the 1-norm $\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$ for vectors.)

$\mathbf{A}$ large ($\sim$ 8 billion) and sparse: computing the eigenvalues and -vectors using standard methods for dense cases as discussed before is impossible.

The only viable method is the *Power method*.

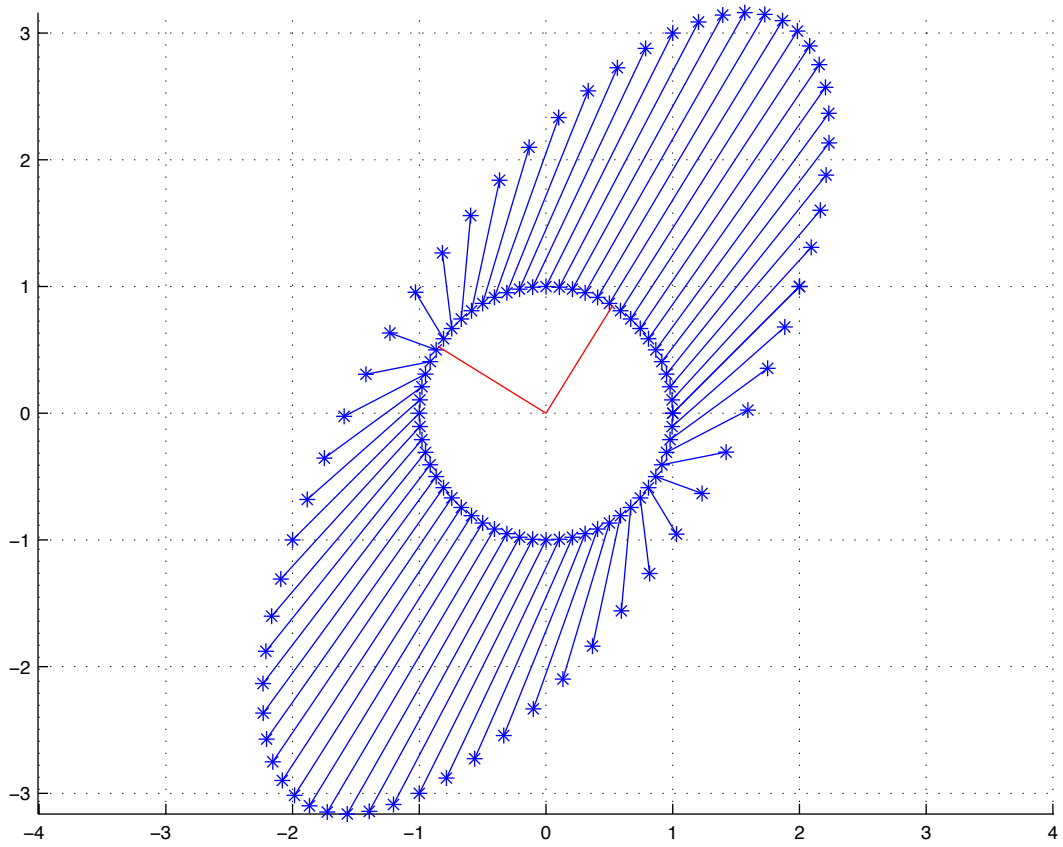
Power method

Given a initial vector $\mathbf{r}^0$ with unit norm, the *power method* produces a sequence of vectors $\mathbf{r}^k$ as follows:

for $k = 1, 2, ...$

$$\mathbf{q}^k = \mathbf{A}\mathbf{r}^{k-1}$$
$$\mathbf{r}^k = \mathbf{q}^k / \|\mathbf{q}^k\|$$
$$\lambda^k = \frac{(\mathbf{r}^k)^T \mathbf{A}\mathbf{r}^k}{(\mathbf{r}^k)^T \mathbf{r}^k}.$$

The λ^k converge to the largest eigenvalue in absolute value, and the vectors $\mathbf{r}^k$ converge to the corresponding eigenvector.


3
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3
4

We are not interested in finding an estimate for the largest eigenvalue in magnitude, as we already know it to be one. So we just compute:

for $k = 1, 2, ...$

$$\mathbf{q}^k = \mathbf{A}\mathbf{r}^{k-1}$$
$$\mathbf{r}^k = \mathbf{q}^k / \|\mathbf{q}^k\|_1$$

The convergence of the power iteration depends on the distribution of eigenvalues.

References

[1] Lars Eldén: Matrix Methods in Data Mining and Pattern Recognition, SIAM 2007.

[2] Soumen Chakrabarti: Mining the Web, Morgan Kaufmann Publishers, 2003.

[3] T. Haveliwala, S. Kamvar, G. Jeh, An Analytic Comparison of Approaches to Personalizing PageRank, Stanford University Technical Report, July 2003.