Linear Algebra Methods for Data Mining

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

Spring 2007

Mining the web: PageRank and Power Iteration, HITS

PageRank: matrix formulation

PageRank can be formulated 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.

Modify this slightly: 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* $\mathbf{P}$.

In random walk terms this was done to ensure we do not get stuck at a node with no outlinks. But we might still get stuck in some subgraph, if $\mathbf{P}$ is reducible. To avoid this we insert 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 $0 \leq \alpha < 1$.

Now the pagerank is the unique eigenvector of the matrix $\mathbf{A}$ corresponding to the eigenvalue $\lambda = 1$:

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

This can be solved using the power iteration:

for $k = 1, 2, \ldots$

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

Convergence of the power iteration

Assume $\mathbf{A}$ is diagonalizable, i.e. there exists a nonsingular matrix $\mathbf{V}$ of eigenvectors, $\mathbf{V}^{-1}\mathbf{A}\mathbf{V} = \text{diag}(\lambda_1,\ \lambda_2,\ \lambda_3,\ ...,\ \lambda_n)$.

Furthermore, assume that the eigenvalues are ordered $1 = \lambda_1 > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_n|$.

Expand the initial approximation $\mathbf{r}^0$ in terms of the eigenvectors

$$\mathbf{r}^0 = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \ldots + c_n\mathbf{v}_n,$$

where $c_1 \neq 0$ is assumed.

Then we have

$$\mathbf{A}^k\mathbf{r}^0 = \lambda_1^k(c_1\mathbf{v}_1 + \sum_{j=2}^{n} c_j\big(\frac{\lambda_j}{\lambda_1}\big)^k\mathbf{v}_j)$$

For $j = 2, \ , \ldots$ we have $|\lambda_j| < 1$, so the second term tends to zero, and the power method converges to the eigenvector $\mathbf{r}_1$ corresponding the the eigenvalue 1.

The rate of convergence is determined by the ratio $|\lambda_2/\lambda_1|$.

If this is close to one, convergence is very slow. But we know that this is not the case: $\lambda_2 = \alpha$.

Examples

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix}, \quad \mathbf{r}^0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & \alpha \end{pmatrix}, \quad \mathbf{r}^0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad 0 < \alpha < 1.$$

How does the power method perform?

Stopping criterion

We want the computed approximations $\hat{\lambda}$ and $\hat{\mathbf{r}}$ of the eigenvalue and eigenvector to be close to the real ones.

It can be shown, that the optimal error matrix $\mathbf{E}$ for which

$$(\mathbf{A}+\mathbf{E})\hat{\mathbf{r}} = \hat{\lambda}\hat{\mathbf{r}}$$

exactly, satisfies

$$\|\mathbf{E}\|_2 = \|\mathbf{s}\|_2, \quad \mathbf{s} = \mathbf{A}\hat{\mathbf{r}} - \hat{\lambda}\hat{\mathbf{r}}.$$

In pagerank computations we use the 1-norm of the residual instead:

$$\text{Stop when } \|\mathbf{A}\hat{\mathbf{r}} - \hat{\mathbf{r}}\|_1 < tol.$$

Computational aspects

We are considering a huge, sparse column-stochastic matrix $\mathbf{A}$.

The power method is the way to compute the eigenvector corresponding to the eigenvalue 1.

But: with the size of the matrix (dimension in billions), this is still cumbersome!

Computing $\mathbf{y} = \mathbf{A}\mathbf{z}$ where $\mathbf{A} = \alpha \mathbf{P} + (1-\alpha)\frac{1}{n}\mathbf{e}\mathbf{e}^T$ is nontrivial.

We take some steps to make computations easier.

First, note that if $\|\mathbf{z}\|_1 = \mathbf{e}^T\mathbf{z} = 1$, then, since $\mathbf{A}$ is column-stochastic,

$$\|\mathbf{y}\|_1 = \mathbf{e}^T\mathbf{y} = \mathbf{e}^T\mathbf{A}\mathbf{z} = \mathbf{e}^T\mathbf{z} = 1.$$

Therfore, no need to normalize vectors produced by power iteration!

Then recall:

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

where $\mathbf{d}$ has an element 1 in all those positions that correspond to pages with no outlinks.

$\mathbf{Q}$ sparse, but to form $\mathbf{P}$ we insert full vectors into $\mathbf{Q}$! And these full vectors have dimension = number of web pages on the web = billions.

Cannot form $\mathbf{P}$ expicitly!

We have

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

and

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

Using these, take a closer look at $\mathbf{y} = \mathbf{A}\mathbf{z}$:

$$\begin{aligned}\mathbf{y} = \mathbf{A}\mathbf{z} &= \alpha(\mathbf{Q} + \frac{1}{n}\mathbf{e}\mathbf{d}^T)\mathbf{z} + \frac{(1-\alpha)}{n}\mathbf{e}\mathbf{e}^T\mathbf{z} \\ &= \alpha\mathbf{Q}\mathbf{z} + \beta\frac{1}{n}\mathbf{e},\end{aligned}$$

where

$$\beta = \alpha\mathbf{d}^T\mathbf{z} + (1-\alpha)\mathbf{e}^T\mathbf{z}.$$

Now, combine this with

$$\|\mathbf{y}\|_1 = \mathbf{e}^T\mathbf{y} = \mathbf{e}^T\mathbf{A}\mathbf{z} = \mathbf{e}^T\mathbf{z} = 1 :$$

Combine

$$\|\mathbf{y}\|_1 = \mathbf{e}^T\mathbf{y} = \mathbf{e}^T\mathbf{A}\mathbf{z} = \mathbf{e}^T\mathbf{z} = 1$$

with

$$\mathbf{y} = \mathbf{A}\mathbf{z} = \alpha\mathbf{Q}\mathbf{z} + \beta\frac{1}{n}\mathbf{e}$$

to get

$$1 = \mathbf{e}^T(\alpha\mathbf{Q}\mathbf{z}) + \beta\mathbf{e}^T(\frac{1}{n}\mathbf{e}) = \mathbf{e}^T(\alpha\mathbf{Q}\mathbf{z}) + \beta.$$

So

$$\beta = 1 - \|(\alpha\mathbf{Q}\mathbf{z})\|_1.$$

Note: we don't need $\mathbf{d}$!

Power method algorithm for pagerank

Choose an initial guess $\mathbf{z}$ at random, define $\mathbf{v} = \frac{1}{n}\mathbf{e}$, and compute

$$\hat{\mathbf{y}} = \alpha \mathbf{Q}\mathbf{z}$$

$$\beta = 1 - \|\hat{\mathbf{y}}\|_1$$

$$\mathbf{y} = \hat{\mathbf{y}} + \beta \mathbf{v}$$

$$\mathbf{s} = \|\mathbf{y} - \mathbf{z}\|_1$$

Repeat until the residual $\mathbf{s}$ is small enough.

Note 1: $\mathbf{v}$ may be defined differently, to be any nonnegative vector with unit 1-norm.

Note 2: to save space, replace $\hat{\mathbf{y}}$ by $\mathbf{y}$.

Note 1

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*.

Matlab code

Given the $n \times n$ matrix $\mathbf{Q}$ and the vector personalization vector $\mathbf{v}$ (e.g. $\mathbf{v} = \frac{1}{n}(1,\ 1,\ \ldots,\ 1)^T$) iterate:

```
yhat=alfa*Q*z;
beta=1-norm(yhat,1);
y=yhat+beta*v;
residual=norm(y-z,1);
```

Rate of convergence

The rate of convergence depended on the factor $\|\lambda_2\|/\|\lambda_1\| = \alpha$.

A typical value for $\alpha = 0.85$.

To make 0.85^k equal to 10^{-4} we need approximately $k = 57$ iterations.

Google reportedly uses around this number of iterations (Page & Brin: The PageRank Citation Ranking: Bringing Order to the Web, 1998).

For a part of the web, it may be possible to use the matlab function eigs.

For the whole web, only the power method has any hope of succeeding.

Several adjustments have been proposed to accelerate convergence:

- Checking which components have converged.
- Using the *block structure* of the web.
- and other strategies...

How to find other eigenvalues?

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

for $k = 1, 2, \ldots$

$$\begin{aligned}\mathbf{q}^k &= \mathbf{A}\mathbf{r}^{k-1}\\ \mathbf{r}^k &= \mathbf{q}^k / \|\mathbf{q}^k\|_2\\ \lambda^k &= (\mathbf{r}^k)^T \mathbf{A}\mathbf{r}^k.\end{aligned}$$

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

But what if we want the second eigenvalue too?

Consider the symmetric case. If $\mathbf{A}$ is symmetric, then it has an eigenvalue decomposition:

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T,$$

where $\mathbf{U}$ and $\mathbf{\Lambda}$ contain the eigenvectors and eigenvalues.

We can write this as

$$\mathbf{A} = \sum_{i=1}^{n} \lambda_i \mathbf{u}_i \mathbf{u}_i^T.$$

Now define a new matrix by *deflating* the original one: $\hat{\mathbf{A}} = \mathbf{A} - \lambda_1 \mathbf{u}_1 \mathbf{u}_1^T$. This has the same eigenvalues and eigenvectors as $\mathbf{A}$, except that the eigenvalue corresponding to the eigenvector $\mathbf{u}_1$ is now zero!

Use the power iteration on this new matrix to the the second largest eigenvalue.

What if $\mathbf{A}$ is not symmetric?

Suppose we have the first eigenvalue and eigenvector of $\mathbf{A}$. Normalize the eigenvector to have unit norm, and use e.g. Givens rotations to transform it into a unit vector:

$$\mathbf{G}^T \mathbf{u}_1 = \mathbf{e}_1.$$

We can write the orthogonal matrix $\mathbf{G}$ as

$$\mathbf{G} = [\mathbf{u}_1 \; \mathbf{V}].$$

Define $\hat{\mathbf{A}} = \mathbf{V}^T \mathbf{A} \mathbf{V}$ and use power iteration on this.

You get the second largest eigenvalue λ_2 and the corresponding eigenvector $\hat{\mathbf{u}}_2$ of $\hat{\mathbf{A}}$, from which you can compute the eigenvector of $\mathbf{A}$ corresponding to λ_2: $\mathbf{u}_2 = \mathbf{V}\hat{\mathbf{u}_2} + \beta \mathbf{u}_1$, $\beta = \mathbf{u}_1^T \mathbf{A} \mathbf{V} \hat{\mathbf{u}_2} / (\lambda_2 - \lambda_1)$.

Reminder: Our Problem was 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?

Ranking web pages according to relevance

- One answer to this question is posed by the PageRank algorithm, used by e.g. Google. (Note: Google also uses e.g. keywords, phrase matches, anchor text etc.)
- PageRank does not depend on query, it only ranks the pages.
- Good: no need to do computations every time a query is posed.
- Bad: even a high quality page (in terms of pagerank or prestige) may not be the most relevant page with respect to the query!
- Alternative answer to problem: HITS.

HITS (hyperlink induced topic search)

Idea: Web includes two flavors of prominent pages:

- authorities contain high-quality information,
- hubs are comprehensive lists of links to authorities

A page is a good authority, if many hubs point to it.

A page is a good hub if it points to many authorities.

Reminder

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.

Good authorities are pointed to by good hubs and good hubs point to good authorities.

Each page i has both a hub score h_i and an authority score a_i.

HITS successively refines these scores by computing

$$a_i^{(k)} = \sum_{j \in I_i} h_j^{(k-1)}, \quad h_i^{(k)} = \sum_{j \in O_i} a_j^{(k-1)}.$$

This can be written in matrix form:

Define the adjacency matrix $\mathbf{L}$ of the direced web graph:

$$\mathbf{L} = \begin{cases} 1, & \text{if there exists an edge from node } i \text{ to node } j, \\ 0, & \text{otherwise.} \end{cases}$$

Now

$$\mathbf{a}^{(k)} = \mathbf{L}^T \mathbf{h}^{(k-1)} \quad \text{and} \quad \mathbf{h}^{(k)} = \mathbf{L}\mathbf{a}^{(k-1)}$$

Note: we need to normalize the authority and hub vectors for this to converge!

Iterative computation of $\mathbf{a}$ and $\mathbf{h}$

1. Make an initial guess: $\mathbf{h}^{(0)}$, e.g. $\mathbf{h}^{(0)} = \mathbf{e} = (1\ 1\ \ldots\ 1)^T$. Normalize $\mathbf{h}$.

- Iterate:

$$\mathbf{a}^{(k)} = \mathbf{L}^T \mathbf{h}^{(k-1)}$$
$$\mathbf{h}^{(k)} = \mathbf{L}\mathbf{a}^{(k)}$$

Normalize $\mathbf{a}$ and $\mathbf{h}$.

This is a form of power iteration.

Example

```
L =

     0     0     0     1     1     1
     0     0     0     1     1     0
     0     0     0     0     1     1
     0     0     0     0     1     0
     0     0     0     0     0     0
     0     0     1     0     0     0
```

```
h=rand(6,1);h=h/norm(h);
a=L'*h;a=a/norm(a)

         0
         0
    0.2463
    0.3818
    0.7350
    0.5032

h=L*a;h=h/norm(h)

    0.6611
    0.4557
    0.5053
    0.2999
         0
    0.1005
```

```
a=L'*h;a=a/norm(a)

         0
         0
    0.0400
    0.4445
    0.7650
    0.4642

h=L*a;h=h/norm(h)

    0.6636
    0.4795
    0.4873
    0.3033
         0
    0.0159
```

```
a=L'*h;a=a/norm(a);
h=L*a;h=h/norm(h);
...
a=L'*h;a=a/norm(a);
h=L*a;h=h/norm(h);

a =                                  h =


         0                          0.6635
         0                          0.4835
    0.0000                          0.4835
    0.4544                          0.3035
    0.7662                          0
    0.4544                          0.0000
```

Note:

The pair of equations

$$\mathbf{a}^{(k)} = \mathbf{L}^T \mathbf{h}^{(k-1)}$$
$$\mathbf{h}^{(k)} = \mathbf{L}\mathbf{a}^{(k)}$$

can be written as

$$\mathbf{a}^{(k)} = \mathbf{L}^T \mathbf{L}\mathbf{a}^{(k-1)}$$
$$\mathbf{h}^{(k)} = \mathbf{L}\mathbf{L}^T \mathbf{h}^{(k-1)}$$

- $\mathbf{L}^T\mathbf{L}$ is the authority matrix because it determines the authority scores.
- $\mathbf{L}\mathbf{L}^T$ is the hub matrix, because it determines the hub scores.
- We are in fact performing power iteration on the authority matrix and the hub matrix.
- What power method will find are the dominating eigenvectors of the authority and hub matrices.
- But these are in fact the first left and right singular vectors of $\mathbf{L}$!!

HITS and SVD

- When we use power iteration on the authority matrix $\mathbf{L}^T\mathbf{L}$, or the hub matrix $\mathbf{L}\mathbf{L}^T$, we get the dominant eigenvector of the matrices.
- The eigenvalues of $\mathbf{L}^T\mathbf{L}$ and $\mathbf{L}\mathbf{L}^T$ are the squares of the singular values of $\mathbf{L}$ and eigenvectors are the left and right singular vectors of $\mathbf{L}$.
- So we are in fact running SVD on the adjacency matrix.
- The authority vector and the hub vector are thus the first left and right singular vectors of $\mathbf{L}$.
- In some cases it is useful to look at other singular vectors as well (ambiguous or polarized queries)

Do $\mathbf{a}$ and $\mathbf{h}$ depend on the query??

Yes. Because we do not compute them on the whole web.

Instead, we first find all the pages with the query, the so called root set.

Then we include the pages with links to or from the pages in the root set. Now we have the base set.

Now we perform the HITS algorithm on this set.

So the authority and hub scores are always computed with respect to the query!

Good: result tuned to match the query.

Bad: Need a power iteration per query!

Convergence of HITS

Power iteration converges to the dominating eigenvector.

Here we power iterate the matrices $\mathbf{L}^T\mathbf{L}$ and $\mathbf{L}\mathbf{L}^T$, the eigenvalues of which are the squares of the singular values of $\mathbf{L}$.

Note that they are nonnegative!

The speed of covergence depends on the ratio of the largest (in magnitude) and second largest eigenvalues.

If this is is close to one, convergence is slow... and here nothing guarantees it will not be.

Furthermore, we might even have a multiple largest eigenvalue (that is, the two largest singular values $\sigma_1 = \sigma_2$)...

Remember, when we investigated the convergence of the power iteration earlier, we expanded the initial approximation $\mathbf{r}^0$ in terms of the eigenvectors $\mathbf{v}_j$ (here singular vector of $\mathbf{L}$):

$$\mathbf{r}^0 = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \ldots + c_n\mathbf{v}_n,$$

where $c_1 \neq 0$ was assumed.

Consider the case when $\lambda_1 = \sigma_1^2 = \sigma_2^2 > \sigma_3^2$ (denote by λ_j the eigenvalues of $\mathbf{L}^T\mathbf{L}$ and by σ_j the singular values of $\mathbf{L}$).

Then

$$\mathbf{A}^k\mathbf{r}^0 = \lambda_1^k(c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \sum_{j=3}^{n} c_j\big(\frac{\lambda_j}{\lambda_1}\big)^k\mathbf{v}_j)$$

This tends to $c_1\mathbf{v}_1 + c_2\mathbf{v}_2$, which is not unique, but depends on the initial guess, which fixes c_1 and c_2. This is again related to the (ir)reducibility of the matrix.

Because all eigenvalues are nonnegative, we do not have the oscillation problem demonstrated in a previous example.

The rate of convergence in this case is determined by λ_3/λ_1.

HITS vs PageRank

- PageRank may be computed once, HITS is computed per query.
- HITS takes query into account, PageRank doesn't.
- PageRank has no concept of hubs: not a big problem, good hubs soon get links and become good authorities and in general prestigious pages too.
- HITS is sensitive to local topology: insertion or deletion of a small number of nodes may change the scores alot.
- PageRank more stable, because of its random jump step.

Lots of variants exist

- SALSA (stochastic algorithm for link structure analysis)
- different kinds of hybrids: hub PageRank, randomized HITS...
- for a survey, see e.g. A. Langville, C. Meyer: A Survey of Eigenvector Methods of Web Information Retrieval

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] A. Langville, C. Meyer: A Survey of Eigenvector Methods of Web Information Retrieval, 2003.