Link Analysis & Ranking

CS 224W
How to Organize the Web?

How to organize the Web?

First try: Human curated Web directories
- Yahoo, DMOZ, LookSmart

Second try: Web Search
- Information Retrieval attempts to find relevant docs in a small and trusted set
  - Newspaper articles, Patents, etc.
- But: Web is huge, full of untrusted documents, random things, web spam, etc.
- So we need a good way to rank webpages!

2 challenges of web search:

(1) Web contains many sources of information. Who to “trust”?
   - **Insight:** Trustworthy pages may point to each other!

(2) What is the “best” answer to query “newspaper”?
   - No single right answer
   - **Insight:** Pages that actually know about newspapers might all be pointing to many newspapers
All web pages are not equally “important”

www.joe-schmoe.com vs. www.stanford.edu

We already know:
There is large diversity in the web-graph node connectivity.

So, let’s rank the pages using the web graph link structure!
We will cover the following Link Analysis approaches to computing importance of nodes in a graph:

- Hubs and Authorities (HITS)
- Page Rank
- Topic-Specific (Personalized) Page Rank

Sidenote: Various notions of node centrality: Node $u$

- Degree centrality = degree of $u$
- Betweenness centrality = #shortest paths passing through $u$
- Closeness centrality = avg. length of shortest paths from $u$ to all other nodes of the network
- Eigenvector centrality = like PageRank
Hubs and Authorities

Goal (back to the newspaper example):
- Don’t just find newspapers. Find “experts” – pages that link in a coordinated way to good newspapers.

Idea: Links as votes
- Page is more important if it has more links
  - In-coming links? Out-going links?

Hubs and Authorities
Each page has 2 scores:
- Quality as an expert (hub):
  - Total sum of votes of pages pointed to
- Quality as an content (authority):
  - Total sum of votes of experts

Principle of repeated improvement
Interesting pages fall into two classes:

1. **Authorities** are pages containing useful information
   - Newspaper home pages
   - Course home pages
   - Home pages of auto manufacturers

2. **Hubs** are pages that link to authorities
   - List of newspapers
   - Course bulletin
   - List of U.S. auto manufacturers

NYT: 10
Ebay: 3
Yahoo: 3
CNN: 8
WSJ: 9
Counting in-links: Authority

Each page starts with **hub score 1**
Authorities collect their votes

(Note this is idealized example. In reality graph is not bipartite and each page has both a hub and the authority score)
Expert Quality: Hub

Hubs collect authority scores

(Note this is idealized example. In reality graph is not bipartite and each page has both a hub and authority score)
Reweighting

Authorities collect hub scores

(Note this is idealized example. In reality graph is not bipartite and each page has both a hub and authority score)
Mutually Recursive Definition

- A good hub links to many good authorities
- A good authority is linked from many good hubs
  - Note a self-reinforcing recursive definition

Model using two scores for each node:
- **Hub** score and **Authority** score
- Represented as vectors $h$ and $a$, where the $i$-th element is the hub/authority score of the $i$-th node
Hubs and Authorities

Each page \( i \) has 2 scores:
- Authority score: \( a_i \)
- Hub score: \( h_i \)

**HITS algorithm:**

Initialize: \( a_j^{(0)} = 1/\sqrt{n}, \ h_j^{(0)} = 1/\sqrt{n} \)

Then keep iterating until convergence:

\( \forall i: \) Authority: \( a_i^{(t+1)} = \sum_{j \rightarrow i} h_j^{(t)} \)

\( \forall i: \) Hub: \( h_i^{(t+1)} = \sum_{i \rightarrow j} a_j^{(t)} \)

\( \forall i: \) Normalize:
\[
\sum_i \left( a_i^{(t+1)} \right)^2 = 1, \ \sum_j \left( h_j^{(t+1)} \right)^2 = 1
\]

Convergence criteria:
\[
\sum_i \left( h_i^{(t)} - h_i^{(t+1)} \right)^2 < \varepsilon \\
\sum_i \left( a_i^{(t)} - a_i^{(t+1)} \right)^2 < \varepsilon
\]
Hubs and Authorities

- Hits in the vector notation:
  - Vector \( a = (a_1 \ldots, a_n) \), \( h = (h_1 \ldots, h_n) \)
  - Adjacency matrix \( A \) \((n \times n)\): \( A_{ij} = 1 \) if \( i \rightarrow j \)

- Can rewrite \( h_i = \sum_{i \rightarrow j} a_j \) as \( h_i = \sum_j A_{ij} \cdot a_j \)

- So: \( h = A \cdot a \) And similarly: \( a = A^T \cdot h \)

- Repeat until convergence:
  - \( h^{(t+1)} = A \cdot a^{(t)} \)
  - \( a^{(t+1)} = A^T \cdot h^{(t)} \)
  - Normalize \( a^{(t+1)} \) and \( h^{(t+1)} \)
What is \( a = A^T \cdot h \)?

Then: \( a = A^T \cdot (A \cdot a) \)

\( a \) is updated (in 2 steps):
\[
a = A^T (A \ a) = (A^T A) \ a
\]

\( h \) is updated (in 2 steps)
\[
h = A (A^T h) = (A \ A^T) \ h
\]

Thus, in \( 2k \) steps:
\[
a = (A^T \cdot A)^k \cdot a
\]
\[
h = (A \cdot A^T)^k \cdot h
\]

Repeated matrix powering
Hubs and Authorities

**Definition:** Eigenvectors & Eigenvalues

Let $R \cdot x = \lambda \cdot x$

for some scalar $\lambda$, vector $x$, matrix $R$

Then $x$ is an eigenvector, and $\lambda$ is its eigenvalue

**The steady state (HITS has converged):**

$A^T \cdot A \cdot a = c' \cdot a$

$A \cdot A^T \cdot h = c'' \cdot h$

So, authority $a$ is eigenvector of $A^T A$

(associated with the largest eigenvalue)

Similarly: hub $h$ is eigenvector of $AA^T$

Note constants $c', c''$ don’t matter as we normalize them out every step of HITS
PageRank
Still the same idea: Links as votes
- Page is more important if it has more links
  - In-coming links? Out-going links?

Think of in-links as votes:
- [www.stanford.edu](http://www.stanford.edu) has 23,400 in-links
- [www.joe-schmoe.com](http://www.joe-schmoe.com) has 1 in-link

Are all in-links equal?
- Links from important pages count more
- Recursive question!
A “vote” from an important page is worth more:
- Each link’s vote is proportional to the importance of its source page.
- If page $i$ with importance $r_i$ has $d_i$ out-links, each link gets $r_i / d_i$ votes.
- Page $j$’s own importance $r_j$ is the sum of the votes on its in-links.

$$r_j = \frac{r_i}{3} + \frac{r_k}{4}$$
PageRank: The “Flow” Model

- A page is important if it is pointed to by other important pages

- Define a “rank” $r_j$ for node $j$

$$r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$$

$d_i$ … out-degree of node $i$

“Flow” equations:

- $r_y = \frac{r_y}{2} + \frac{r_a}{2}$
- $r_a = \frac{r_y}{2} + r_m$
- $r_m = \frac{r_a}{2}$
Why do we divide by out-degree?
PageRank: Matrix Formulation

- **Stochastic adjacency matrix** $M$
  - Let page $j$ have $d_j$ out-links
  - If $j \rightarrow i$, then $M_{ij} = \frac{1}{d_j}$
  - $M$ is a column stochastic matrix
    - Columns sum to 1

- **Rank vector** $r$: An entry per page
  - $r_i$ is the importance score of page $i$
  - $\sum_i r_i = 1$

- The flow equations can be written
  \[ r = M \cdot r \]
Imagine a random web surfer:
- At any time $t$, surfer is on some page $i$
- At time $t + 1$, the surfer follows an out-link from $i$ uniformly at random
- Ends up on some page $j$ linked from $i$
- Process repeats indefinitely

Let:
- $p(t)$ ... vector whose $i^{th}$ coordinate is the prob. that the surfer is at page $i$ at time $t$
- So, $p(t)$ is a probability distribution over pages
The Stationary Distribution

- Where is the surfer at time $t+1$?
  - Follows a link uniformly at random

\[ p(t + 1) = M \cdot p(t) \]

- Suppose the random walk reaches a state

\[ p(t + 1) = M \cdot p(t) = p(t) \]

then $p(t)$ is \textit{stationary distribution} of a random walk

- Our original rank vector $r$ satisfies $r = M \cdot r$

- So, $r$ is a stationary distribution for the random walk
Computing PageRank

Given a web graph with \( n \) nodes, where the nodes are pages and edges are hyperlinks:

- Assign each node an initial page rank
  - Repeat until convergence \( (\sum_i |r_i^{(t+1)} - r_i^{(t)}| < \epsilon) \)
- Calculate the page rank of each node

\[
  r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i}
\]

\( d_i \ldots \) out-degree of node \( i \)
Computing PageRank

- **Power Iteration:**
  - Set $r_j \leftarrow 1/N$
  - **1:** $r_j' \leftarrow \sum_{i \rightarrow j} \frac{r_i}{d_i}$
  - **2:** $r \leftarrow r'$
  - If $|r - r'| > \varepsilon$: goto 1

- **Example:**

\[
\begin{pmatrix}
  r_y \\
  r_a \\
  r_m
\end{pmatrix} = \begin{pmatrix}
  1/3 \\
  1/3 \\
  1/3
\end{pmatrix}
\]

*Iteration 0, 1, 2, …*
### Computing PageRank

#### Power Iteration:
- Set $r_j \leftarrow 1/N$
- **1:** $r'_j \leftarrow \sum_{i \rightarrow j} \frac{r_i}{d_i}$
- **2:** $r \leftarrow r'$
- If $|r - r'| > \varepsilon$: goto **1**

#### Example:

<table>
<thead>
<tr>
<th></th>
<th>y</th>
<th>a</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>m</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
</tr>
</tbody>
</table>

$r_y = \frac{r_y}{2} + \frac{r_a}{2}$

$r_a = \frac{r_y}{2} + \frac{r_m}{2}$

$r_m = \frac{r_a}{2}$

$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} \begin{pmatrix} 1/3 \\ 1/3 \\ 5/12 \\ 9/24 \\ 6/15 \\ 6/15 \end{pmatrix}$$

Iteration 0, 1, 2, …
PageRank: Three Questions

Does this converge?

Does it converge to what we want?

Are results reasonable?

\[ r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i} \]

or equivalently

\[ r = Mr \]
Does this converge?

- The “Spider trap” problem:

\[ r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{d_i} \]

- Example:

<table>
<thead>
<tr>
<th>Iteration: 0, 1, 2, 3...</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_a ) = 1</td>
</tr>
<tr>
<td>( r_b ) = 0</td>
</tr>
</tbody>
</table>
Does this converge?

The “Spider trap” problem:

Example:

\[
\begin{align*}
r_a^{(t+1)} &= \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i} \\
\end{align*}
\]

Iteration: 0, 1, 2, 3...

\[
\begin{align*}
 r_a &= 1 \quad 0 \quad 1 \quad 0 \\
 r_b &= 0 \quad 1 \quad 0 \quad 1
\end{align*}
\]
The “Dead end” problem:

\[ r^{(t+1)}_j = \sum_{i \rightarrow j} \frac{r^{(t)}_i}{d_i} \]

Example:

<table>
<thead>
<tr>
<th>Iteration</th>
<th>0, 1, 2, 3...</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_a )</td>
<td>1</td>
</tr>
<tr>
<td>( r_b )</td>
<td>0</td>
</tr>
</tbody>
</table>
Does it converge to what we want?

- The “Dead end” problem:

\[ r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i} \]

- Example:

<table>
<thead>
<tr>
<th>Iteration: 0, 1, 2, 3...</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_a )</td>
</tr>
<tr>
<td>( r_b )</td>
</tr>
</tbody>
</table>
RageRank: Problems

2 problems:

- **(1)** Some pages are **dead ends** (have no out-links)
  - Such pages cause importance to “leak out”

- **(2)** **Spider traps**
  - (all out-links are within the group)
  - Eventually spider traps absorb all importance
Problem: Spider Traps

- **Power Iteration:**
  - Set \( r_j = \frac{1}{N} \)
  - \( r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i} \)
  - And iterate

- **Example:**

\[
\begin{pmatrix}
    r_y \\
    r_a \\
    r_m
\end{pmatrix} =
\begin{pmatrix}
    1/3 \\
    1/3 \\
    1/3
\end{pmatrix}
\]

\[
\begin{array}{c|c|c}
  y & a & m \\
  \hline
  \frac{1}{2} & \frac{1}{2} & 0 \\
  \frac{1}{2} & 0 & 0 \\
  0 & \frac{1}{2} & 1 \\
\end{array}
\]

\[
\begin{align*}
r_y &= \frac{r_y}{2} + \frac{r_a}{2} \\
r_a &= \frac{r_y}{2} \\
r_m &= \frac{r_a}{2} + r_m
\end{align*}
\]

Iteration 0, 1, 2, …
**Problem: Spider Traps**

**Power Iteration:**
- Set \( r_j = \frac{1}{N} \)
- \( r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i} \)
- And iterate

**Example:**

\[
\begin{pmatrix}
  r_y \\
r_a \\
r_m
\end{pmatrix} =
\begin{pmatrix}
  1/3 & 2/6 & 3/12 & 5/24 & 0 \\
  1/3 & 1/6 & 2/12 & 3/24 & \ldots & 0 \\
  1/3 & 3/6 & 7/12 & 16/24 & 1
\end{pmatrix}
\]

Iteration 0, 1, 2, …

\[
\begin{array}{c|c|c|c|c}
& y & a & m \\
\hline
y & 1/2 & 1/2 & 0 \\
a & 1/2 & 0 & 0 \\
m & 0 & 1/2 & 1 \\
\end{array}
\]

\[
\begin{align*}
  r_y &= r_y / 2 + r_a / 2 \\
  r_a &= r_y / 2 \\
  r_m &= r_a / 2 + r_m
\end{align*}
\]
The Google solution for spider traps: At each time step, the random surfer has two options:

- With prob. $\beta$, follow a link at random
- With prob. $1 - \beta$, jump to a random page

Common values for $\beta$ are in the range 0.8 to 0.9.

Surfer will teleport out of spider trap within a few time steps.
Problem: Dead Ends

**Power Iteration:**
- Set $r_j = \frac{1}{N}$
- $r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$
- And iterate

**Example:**

\[
\begin{bmatrix}
    r_y \\
    r_a \\
    r_m
\end{bmatrix}
= \begin{bmatrix}
    1/3 & 2/6 & 3/12 & 5/24 & 0 \\
    1/3 & 1/6 & 2/12 & 3/24 & \ldots & 0 \\
    1/3 & 1/6 & 1/12 & 2/24 & 0
\end{bmatrix}
\]

Iteration 0, 1, 2, …
Solution: escape dead ends immediately

- **Teleports:** Follow random teleport links with probability 1.0 from dead-ends
- Adjust matrix accordingly
Google’s solution: At each step, random surfer has two options:
- With probability $\beta$, follow a link at random
- With probability $1-\beta$, jump to some random page

PageRank equation [Brin-Page, ‘98]

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1 - \beta) \frac{1}{n}$$

The above formulation assumes that $M$ has no dead ends. We can either preprocess matrix $M$ (bad!) or explicitly follow random teleport links with probability 1.0 from dead-ends. See P. Berkhin, A Survey on PageRank Computing, Internet Mathematics, 2005.
PageRank & Eigenvectors

- **PageRank as a principal eigenvector**

\[ r = M \cdot r \] or equivalently \[ r_j = \sum_i \frac{r_i}{d_i} \]

- **But we really want (**)**:

\[ r_j = \beta \sum_i \frac{r_i}{d_i} + (1 - \beta) \frac{1}{n} \]

- **Let’s define**:

\[ M'_{ij} = \beta M_{ij} + (1 - \beta) \frac{1}{n} \]

- **Now we get what we want**:

\[ r = M' \cdot r \]

- **What is \((1 - \beta)\)?**
  - In practice \(0.15\) (Jump approx. every 5-6 links)

\(d_i\) ... out-degree of node \(i\)

**Note:** \(M\) is a sparse matrix but \(M'\) is dense (all entries \(\neq 0\)). In practice we never “materialize” \(M\) but rather we use the “sum” formulation (**)
The PageRank Algorithm

**Input:** Graph $G$ and parameter $\beta$
- Directed graph $G$ with spider traps and dead ends
- Parameter $\beta$

**Output:** PageRank vector $r$
- Set: $r_j^{(0)} = \frac{1}{N}$, $t = 1$
- do:
  - $\forall j$: $r_j^{(t)} = \sum_{i \rightarrow j} \beta \frac{r_i^{(t-1)}}{d_i}$
  - $r_j^{(t)} = 0$ if in-deg. of $j$ is 0
- Now re-insert the leaked PageRank:
  - $\forall j$: $r_j^{(t)} = r_j^{(t)} + \frac{1-S}{N}$ where: $S = \sum_j r_j^{(t)}$
- $t = t + 1$
- while $\sum_j |r_j^{(t)} - r_j^{(t-1)}| > \varepsilon$
Example
PageRank and HITS

- PageRank and HITS are two solutions to the same problem:
  - What is the value of an in-link from $u$ to $v$?
  - In the PageRank model, the value of the link depends on the links into $u$.
  - In the HITS model, it depends on the value of the other links out of $u$.

- The destinies of PageRank and HITS post-1998 were very different.
Example application: expertise ranking
``Knowledge search is like oozing out knowledge in human brains to the Internet. People who know something better than others can present their know-how, skills or knowledge''

NHN CEO Chae Hwi-young

``(It is) the next generation of search… (it) is a kind of collective brain -- a searchable database of everything everyone knows. It's a culture of generosity. The fundamental belief is that everyone knows something.”

-- Eckart Walther (Yahoo Research)
Why identify expertise?

- The Response Time Gap

- The Expertise Gap
- Difficult to infer reliability of answers

Automatically ranking expertise may be helpful.
Java Forum

- 87 sub-forums
- 1,438,053 messages
- Community expertise network constructed:
  - 196,191 users
  - 796,270 edges
Constructing a community expertise network

Thread 1: Large Data, binary search or hashtable? user A
  Re: Large... user B
  Re: Large... user C

Thread 2: Binary file with ASCII data user A
  Re: File with... user C

unweighted

weighted by # threads

weighted by shared credit

weighted with backflow
Uneven participation

\[ \alpha = 1.87 \text{ fit, } R^2 = 0.9730 \]

- ‘answer people’ may reply to thousands of others
- ‘question people’ are also uneven in the number of repliers to their posts, but to a lesser extent

number of people one replied to

number of people one received replies from
Not Everyone Asks/Replies

- Core: A strongly connected component, in which everyone asks and answers
- IN: Mostly askers.
- OUT: Mostly Helpers

The Web is a bow tie

The Java Forum network is an uneven bow tie
fragment of the Java Forum
## Relating network structure to Java expertise

### Human-rated expertise levels

- **2 raters**
- **135 JavaForum users with >= 10 posts**
- **inter-rater agreement** \( (\tau = 0.74, \rho = 0.83) \)
- **for evaluation of algorithms, omit users where raters disagreed by more than 1 level** \( (\tau = 0.80, \rho = 0.83) \)

<table>
<thead>
<tr>
<th>L</th>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Top Java expert</td>
<td>Knows the core Java theory and related advanced topics deeply.</td>
</tr>
<tr>
<td>4</td>
<td>Java professional</td>
<td>Can answer all or most of Java concept questions. Also knows one or some sub topics very well,</td>
</tr>
<tr>
<td>3</td>
<td>Java user</td>
<td>Knows advanced Java concepts. Can program relatively well.</td>
</tr>
<tr>
<td>2</td>
<td>Java learner</td>
<td>Knows basic concepts and can program, but is not good at advanced topics of Java.</td>
</tr>
<tr>
<td>1</td>
<td>Newbie</td>
<td>Just starting to learn java.</td>
</tr>
</tbody>
</table>
simple local measures do as well (and better) than measures incorporating the wider network topology
automated vs. human ratings

- # answers
- indegree
- # answers
- indegree
- HITS authority
- PageRank
Modeling community structure to explain algorithm performance

**Control Parameters:**
- Distribution of expertise
- Who asks questions most often?
- Who answers questions most often?
  - best expert most likely
  - someone a bit more expert

ExpertiseNet Simulator
Simulating probability of expertise pairing

suppose:

- expertise is uniformly distributed
- probability of posing a question is inversely proportional to expertise
  \( p_{ij} = \text{probability a user with expertise } j \text{ replies to a user with expertise } i \)

2 models:

- ‘best’ preferred
  \[ p_{ij} \sim e^{\beta (j-i) / i} \]

- ‘just better’ preferred
  \[ p_{ij} \sim e^{\gamma (i-j) / i} \quad \text{for } j > i \]
Visualization

Best “preferred”

just better
Degree correlation profiles

Java Forum Network

best preferred (simulation)

just better (simulation)
Simulation can tell us when to use which algorithms

Preferred Helper: ‘best available’

Preferred Helper: ‘just better’
Different ranking algorithms perform differently

In the ‘just better’ model, a node is correctly ranked by PageRank but not by HITS. Can you explain why?
Coming soon...

Network Challenge

Social and Information Network Analysis, Fall 2015
Stanford University
SVD

Capturing network features
### Dimensionality Reduction

- **Compress / reduce dimensionality:**
  - 10^6 rows; 10^3 columns; no updates
  - Random access to any cell(s); **small error: OK**

<table>
<thead>
<tr>
<th>customer</th>
<th>We 7/10/96</th>
<th>Th 7/11/96</th>
<th>Fr 7/12/96</th>
<th>Sa 7/13/96</th>
<th>Su 7/14/96</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC Inc.</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DEF Ltd.</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GHI Inc.</td>
<td>1</td>
<td>1</td>
<td><strong>1</strong></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>KLM Co.</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Smith</td>
<td><strong>0</strong></td>
<td><strong>0</strong></td>
<td><strong>0</strong></td>
<td><strong>2</strong></td>
<td><strong>2</strong></td>
</tr>
<tr>
<td>Johnson</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td><strong>3</strong></td>
<td><strong>3</strong></td>
</tr>
<tr>
<td>Thompson</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The above matrix is really “2-dimensional.” All rows can be reconstructed by scaling [1 1 1 0 0] or [0 0 0 1 1].

Jure Leskovec, Stanford C246: Mining Massive Datasets
**Assumption:** Data lies on or near a low \( d \)-dimensional subspace

**Axes of this subspace are effective representation of the data**
SVD - Definition

\[ A_{[m \times n]} = U_{[m \times r]} \Sigma_{[r \times r]} (V_{[n \times r]})^T \]

- **A**: $m \times n$ matrix (e.g., $m$ documents, $n$ terms)
- **U**: $m \times r$ matrix ($m$ documents, $r$ concepts)
- **Σ**: $r \times r$ diagonal matrix (strength of each ‘concept’) ($r$ : rank of the matrix $A$)
- **V**: $n \times r$ matrix ($n$ terms, $r$ concepts)
\[ A \approx U \Sigma V^T = \sum_i \sigma_i u_i \circ v_i^T \]
\[ \mathbf{A} \approx \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T = \sum_i \sigma_i \mathbf{u}_i \odot \mathbf{v}_i^T \]
It is always possible to decompose a real matrix $A$ into $A = U \Sigma V^T$, where

- $U$, $\Sigma$, $V$: unique

- $U$, $V$: column orthonormal
  - $U^T U = I$; $V^T V = I$ ($I$: identity matrix)
  - (Columns are orthogonal unit vectors)

- $\Sigma$: diagonal
  - Entries (singular values) are positive, and sorted in decreasing order ($\sigma_1 \geq \sigma_2 \geq \ldots \geq 0$)
SVD – Example: Users-to-Movies

\[ A = U \Sigma V^T \] - example: Users to Movies

<table>
<thead>
<tr>
<th>Matrix</th>
<th>SciFi</th>
<th>Romance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 3 3 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 4 4 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 5 5 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 2 0 4 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 5 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 0 2 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

“Concepts”
AKA Latent dimensions
AKA Latent factors

Jure Leskovec, Stanford C246: Mining Massive Datasets
SVD – Example: Users-to-Movies

\[ A = U \Sigma V^T \] - example: Users to Movies

<table>
<thead>
<tr>
<th></th>
<th>Matrix</th>
<th>Alien</th>
<th>Serenity</th>
<th>Casablanca</th>
<th>Amelie</th>
</tr>
</thead>
<tbody>
<tr>
<td>SciFi</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Romance</td>
<td>1 1 1 0 0</td>
<td>3 3 3 0 0</td>
<td>4 4 4 0 0</td>
<td>5 5 5 0 0</td>
<td>0 2 0 4 4</td>
</tr>
<tr>
<td></td>
<td>0 0 0 5 5</td>
<td>0 1 0 2 2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
\begin{bmatrix}
0.13 & 0.02 & -0.01 \\
0.41 & 0.07 & -0.03 \\
0.55 & 0.09 & -0.04 \\
0.68 & 0.11 & -0.05 \\
0.15 & -0.59 & 0.65 \\
0.07 & -0.73 & -0.67 \\
0.07 & -0.29 & 0.32 \\
\end{bmatrix} \times \begin{bmatrix}
12.4 & 0 & 0 \\
0 & 9.5 & 0 \\
0 & 0 & 1.3 \\
\end{bmatrix} \times \begin{bmatrix}
0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\
0.40 & -0.80 & 0.40 & 0.09 & 0.09 \\
\end{bmatrix}
\]
SVD – Example: Users-to-Movies

\[ A = U \Sigma V^T \] - example: Users to Movies

\[ U \] is “user-to-concept” similarity matrix

\[ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{pmatrix} \]

\[ \begin{pmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{pmatrix} \]

\[ x = \begin{pmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{pmatrix} \]
SVD – Example: Users-to-Movies

\[ A = U \Sigma V^T \] example: Users to Movies

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2 \\
\end{bmatrix}
= 
\begin{bmatrix}
0.13 & 0.02 & -0.01 \\
0.41 & 0.07 & -0.03 \\
0.55 & 0.09 & -0.04 \\
0.68 & 0.11 & -0.05 \\
0.15 & -0.59 & 0.65 \\
0.07 & -0.73 & -0.67 \\
0.07 & -0.29 & 0.32 \\
\end{bmatrix}
\begin{bmatrix}
12.4 \\
0 \\
9.5 \\
0 \\
0 \\
0 \\
0 \\
\end{bmatrix}
\]
SVD – Example: Users-to-Movies

\[ A = U \Sigma V^T \] - example:

V is “movie-to-concept” similarity matrix

\[ \begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2
\end{bmatrix} \]

\[ \begin{bmatrix}
0.13 & 0.02 & -0.01 \\
0.41 & 0.07 & -0.03 \\
0.55 & 0.09 & -0.04 \\
0.68 & 0.11 & -0.05 \\
0.15 & -0.59 & 0.65 \\
0.07 & -0.73 & -0.67 \\
0.07 & -0.29 & 0.32
\end{bmatrix} \]
SVD - Interpretation #1

‘movies’, ‘users’ and ‘concepts’:

- $U$: user-to-concept similarity matrix
- $V$: movie-to-concept similarity matrix
- $\Sigma$: its diagonal elements: ‘strength’ of each concept
SVD - interpretation #2

- SVD gives ‘best’ axis to project on:
  - ‘best’ = min sum of squares of projection errors

- In other words, minimum reconstruction error

Movie 1 rating

Movie 2 rating

first right singular vector

v₁
SVD - Interpretation #2

\[ A = U \Sigma V^T \] - example:

- \( V \): “movie-to-concept” matrix
- \( U \): “user-to-concept” matrix

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2
\end{bmatrix}
= \begin{bmatrix}
0.13 & 0.02 & -0.01 \\
0.41 & 0.07 & -0.03 \\
0.55 & 0.09 & -0.04 \\
0.68 & 0.11 & -0.05 \\
0.15 & -0.59 & 0.65 \\
0.07 & -0.73 & -0.67 \\
0.07 & -0.29 & 0.32
\end{bmatrix}
\begin{bmatrix}
12.4 & 0 & 0 \\
0 & 9.5 & 0 \\
0 & 0 & 1.3
\end{bmatrix}
\begin{bmatrix}
0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\
0.40 & -0.80 & 0.40 & 0.09 & 0.40
\end{bmatrix}
\]
SVD – ingredient networks
Substitution network

- Extract substitution relationships from comments
  - e.g. “I replaced the butter with sour cream”
  - “replace $a$ with $b$”, “substitute $b$ for $a$”, “$b$ instead of $a$”

- Nodes: ingredients

- Edge weights = $p(b | a)$, which is the proportion of substitutions of ingredient $a$ that suggests ingredient $b$
Substitution network: communities

Ingredients:
- Chicken
- Tilapia
- Italian seasoning
- Seasoning
- Onion
- Garlic
- Chicken broth
- Milk
- Sour cream
- Honey
- Olive oil
- Spinach
- Bread
- Apple
- Sweet potato
- Cinnamon
- Black bean
- Flour
- Tomato
- Sauce
- Lemon juice
- Pepper
- Brown rice
- White wine
- Strawberry
- Spaghetti sauce
- Almond extract
- Vanilla
- Cheese
- Almond
- Chocolate chip
- Baking powder
- Cream of mushroom soup
- Egg
- Cranberry
- Pie crust
- Cabbage
- Celery
- Champagne
- Coconut milk
- Corn chip
- Sea scallop
- Apple juice
- Hoagie roll
- Iceberg lettuce
- Cottage cheese
- Golden syrup
- Black olive
- Pickle
- Red potato
- Quinoa
- Graham cracker
- Lemon cake mix
- Imitation crab meat
- Peach schnapp
- Hot
- Vegetable shortening
- Dijon mustard
- Almond milk
- Almond extract
- Cranberry
- Strawberry
- Lemon juice
- Apple juice
- White wine
- Champagne
- Corn chip
- Buttermilk
- Whole milk
- Skim milk
- Heavy cream
- Half and half
- Whipping cream
- Soy milk
- Evaporated milk
- Heavy whipping cream
- Dijon mustard
- Dijon mustard
- Sugar snap pea
- Curried lettuce
- Curry powder
- Pickle
- Smoked paprika
- Smoked paprika
- Cream of mushroom soup
- Cream of mushroom soup
- Cream
- Cinnamon
- Cinnamon
- Nutmeg
- Allspice
- Allspice
- Cardamom
- Ginger root
- Ginger root
- Clove
- Ginger
- Ginger
- Nutmeg
- Nutmeg
- Allspice
- Allspice
- Mace
- Mace
Preferences network

- Create an edge from ingredient $a$ to $b$ if $\text{rating}(a) < \text{rating}(b)$

**ex:**
- Recipe X contains
- Recipe Y contains
- Rating(X) > Rating(Y)
Substitute network and users’ preference

- Weight of preference network
  - PMI(a->b) = log(p(a->b)/p(a)p(b))
  - where p(a->b) = (# of recipe pairs from a to b)/(# of recipe pairs)

- Correlations between preference network and substitute network (\(\rho = 0.72, p<0.001\))
Prediction task

- Given a recipe pair with overlapped ingredients, determine which one has the higher rating.

![Recipe 1](image1.png)

![Recipe 2](image2.png)

Rating:

- Recipe 1: ★★★★★
- Recipe 2: ★★★★☆
Prediction task

Features
- Baseline
  - Cooking methods, preparation time, the number of servings
- 1000 popular ingredient list
  - Binary vector indicating the occurrence of ingredients
- Nutrition
  - Calories, carbohydrates, fat, etc.
- Ingredient networks
  - Network positions (centrality) and communities (SVD)
- Combined set
  - Everything listed above
Prediction task

- 62,031 recipe pairs \((X,Y)\)
  - where \(\text{rating}(X) > \text{rating}(Y)\)
  - \(\geq 10\) user reviews
  - \(\geq 50\%\) users have rated both recipes
  - Cosine similarity of ingredients \((X,Y) > 0.2\)

- Train with gradient boosting tree
  - balanced dataset
  - 2/3 for training, 1/3 for testing
  - Evaluate based on accuracy
Ingredient network features lead to improved performance.
Network structure can be used to rank nodes
- HITS
- PageRank

SVD can represent network features compactly

The application possibilities are endless