Unit 2:

Linear Algebra

Solving systems of linear equations -- the how and why.

LA-01: Google Page Rank

Goal: To introduce what Google page rank is, and one way to compute it.

Meet Randy, the random web surfer.

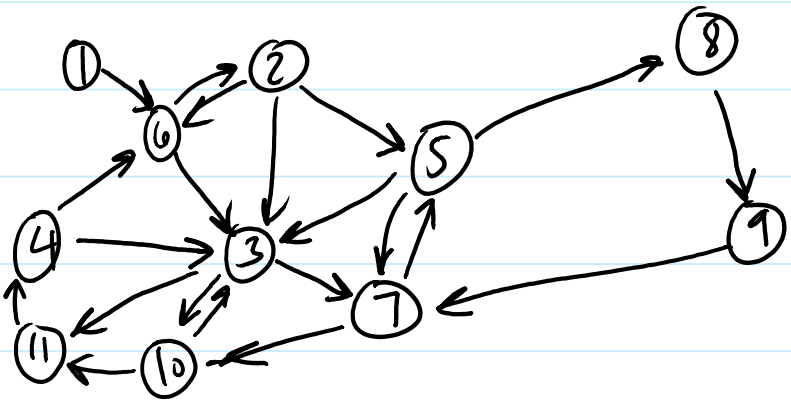
His job is to visit web page after web page… just surf the web.

Each time he clicks to a new web page, he looks at the links on the page, and chooses one randomly (hence his name).

If we give Randy enough time, he will eventually visit all the pages on the web… more than once if we let him continue.

But he will not visit all the pages with equal frequency. For example, he will probably load the UWaterloo CS home page more often than the web page "Enjoyable Dental Procedures". The dental page might only be linked to from Prof. Orchard's home page, but the UW CS web page is linked to by many pages.

Consider this tiny web, where arrows indicate outgoing links:



Intuitively, it seems that Randy will visit more often than .

Code for Randy

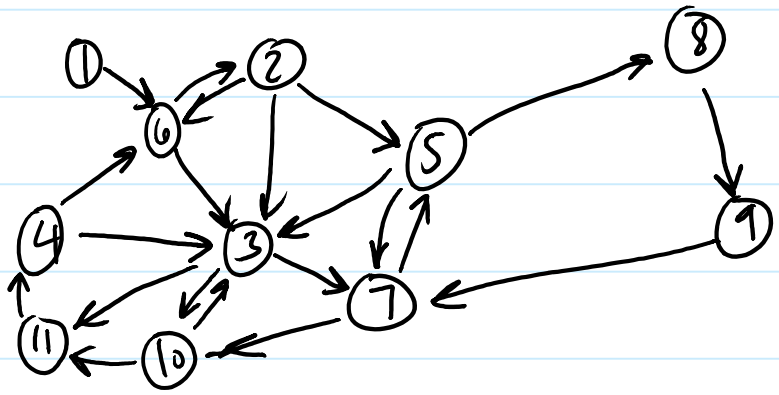
```
%% Random walk
Init; % Set up a graph adjacency matrix in G
N = size(G,1); % number of nodes
```

```
visits = zeros(N,1); % counts number of visits

numSteps = 10000; % number of steps to take
j = ceil(rand * N); % start at a randomly-choosen node

for t = 1:numSteps
   visits(j) = visits(j) + 1; % increment counter for node j
   outLinks = find(G(:,j)==1); % list outgoing links for node j
   numOutLinks = length(outLinks);

   j = outLinks( ceil(rand*numOutLinks) ); % choose one randomly
end

plot(1:N, visits);
```

One way to measure the value or importance of a web page is

Notice that it is better to be linked to by other important web pages since Randy will visit those more often, and hence visit your page that much more frequently.

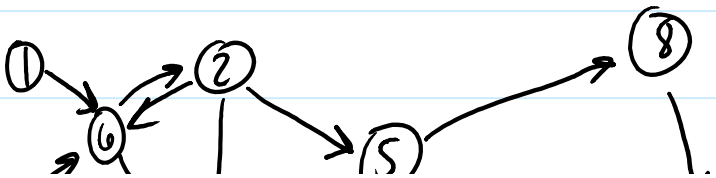
Try removing the link from , and replace it with

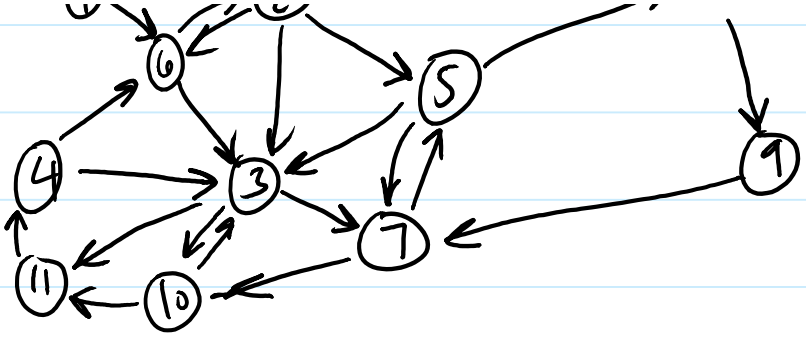


The importance of

Problem: Terminal Pages

What if Randy reaches a page that has no outgoing links?





Randy gets stuck and the random walk does not give a good ranking of web page importance.

Solution: Teleportation

When a web page has no out links,

Problem: The above issue is actually more insidious. Consider…

.
.
.
.
.

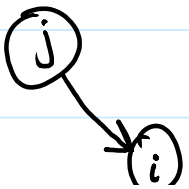
Solution: Random teleportation

At each page, we

Mathematical Formulation of Random Surfer Ranking

Recall matrix $G \in \{0,1\}^{R \times R}$

$$G_{ij} = \begin{cases} 1 & \text{if node } j \text{ links to node } i \\ 0 & \text{otherwise} \end{cases}$$



a_{ij} $\begin{cases} \\ 0 \text{ otherwise} \end{cases}$ → (i)

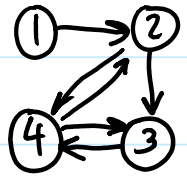
Consider, instead, matrix $P \in [0,1]^{R \times R}$

$$P_{ij} = \begin{cases} \text{if node } j \text{ links to node } i \\ 0 \quad \text{otherwise} \end{cases}$$

where deg(j) is the # of outlinks coming from page j.

Then P_{ij} is the probability of following a link to node i given that you are at node j.

$$P_{ij} = Pr(i|j) \quad \text{(conditional probability)}$$

eg. 

$P =$

Instead of following a single surfer, we can track the progress of an infinite number of surfers (sorry Randy) using the matrix P.

eg. If all our surfers start at ②...

i.e. $x =$

$Px =$

$P^2x = P(Px) =$

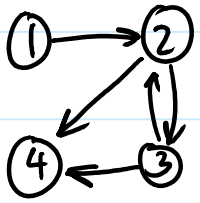
$P^3x =$

∴ after 3 clicks, of the surfers will be at node 3.

Notice that $\|P^n x\|_1 = \|x\|_1$. All the surfers are accounted for. This is because each column of P redistributes all the surfers. Note also that all entries of P are non-negative. Each column of P is a probability distribution, and hence each column adds to 1.

$$\sum_{i=1}^{R} P_{ij} = 1$$

Terminal Nodes: Consider



$P =$

The column corresponding to a terminal node does not add to 1; it adds to 0.

$Px = P[0\ 1\ 0\ 0]^T =$

$P^2x =$

$P^3x =$

$\vdots$

$P^{20}x =$

The matrix equivalent of the teleportation effect can be included…

$P' = P +$

In our example,

In our example,

$$P' = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} + \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix} = \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$$

To address the issue of terminal branches, we add a background randomness.

$$M = \alpha P' +$$

This guarantees that no surfer gets stuck in a subnet of the web.

M is called a Markov Transition Matrix.

LA-02: Markov Transition Matrices

Goal: To learn what a Markov Transition Matrix is, and what its properties are.

Markov Transition Matrices

Def'n A matrix Q is a Markov matrix if

and (sum down cols)

Clearly, M (from the previous lecture) is a Markov matrix.

Def'n A vector q is a probability vector if

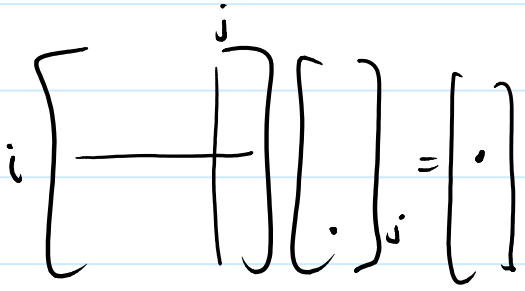
and

Multiplying a Markov matrix by a probability vector yields another probability vector.

Consider $p = Mx$

Markov mat. ↑ ↖ prob. vect.

$\sum_i p_i =$



$\therefore$ p is a prob vector.

Page Rank

Let v^n be a value vector indicating the value, or importance, of

Page Rank

Let p^n be a value vector indicating the value, or importance, of each page on the web. Without loss of generality (WLOG), we scale it so that

$$\sum_i [p^n]_i = 1$$

Thus, you can also think of it as a distribution of random surfers on the web.

If we allow the surfers one more click, then we get

$$p^{n+1} = M p^n \qquad (1)$$

We are looking for a steady-state solution… a fixed-point solution of (1)

i.e.

This is actually an eigenvalue problem,

Eigenvalues & Eigenvectors

The eigenvectors, x, of a square matrix Q satisfy

for some scalar $\lambda \in \mathbb{C}$. This λ is called an eigenvalue.

To find eigenvalues, we can form the charactersitic polynomial.

Any nontrivial (ie. not entirely zero) x that satisfies (2) is an eigenvector. For there to be a nontrivial solution, the matrix $\lambda I - Q$ has to be singular.

has to be singular.

$\Rightarrow$

The determinant is a polynomial in λ called the characteristic polynomial. The roots of the char. poly. are the eigenvalues. The x that solves $(\lambda I - Q)x = 0$ is the corresponding eigenvector.
In our case, we are looking for an eigenvetor corresponding to an eigenvalue of 1,

It is not feasible to use standard algebraic methods to solve

Instead, watch this… →

```
Init;

D = diag(1./sum(G));
P = G * D;
N = size(P,1);

x = ones(N,1) / N; % uniform distribution
for n = 1:20
   x = P * x;
   plot(x);
   hold on;
   pause(0.5);
end

hold off;
```

Goal: To develop the theory behind why the random surfer algorithm works.

Recall:

- We can encode the behaviour of a cohort of random surfers in the Markov matrix
- If p is a probability vector representing the distribution of surfers (or the value of each web page), then no matter what the initial p is, it seems that

$$\lim_{n\to\infty} M^n p \longrightarrow \text{steady-state}$$

i.e. if $p^\infty = \lim_{n\to\infty} M^n p$

$$\Rightarrow M p^\infty = p^\infty$$ (p^∞ is a fixed point of the mapping M)

How can we know for sure that this is the case?

Thm: Every Markov matrix Q has 1 as an eigenvalue.

Pf: The matrix Q^T has the same eigenvalues as Q since they have the same characteristic polynomial.

Let $e = [1\ 1\ \cdots\ 1]^T$.

Comment: OK, so now we know that there is a solution to $Mp = p$. But how do we explain convergence?

Thm: Every (possibly complex) e-value λ of a Markov matrix Q satisfies

$$|\lambda| \leq 1 .$$

Pf: To prove this, we'll use another theorem called the Gershgorin Circle Theorem.

Thm: Girshgorin Circle Thm.
Let B be an arbitrary matrix. Then the e-vals. λ of B are located in the union of the n disks

$$|\lambda - b_{kk}| \leq \sum_{j \neq k} |b_{jk}| .$$

eg. $B = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{4} \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & \frac{1}{4} \end{bmatrix}$

col 1 →
col 2 →
col 3 →

The Gershgorin thm. implies

$$|\lambda - m_{kk}| \leq \sum_{j \neq k} |m_{jk}|$$

Since $0 \leq m_{ij} \leq 1 \ldots$

$$|\lambda| - m_{kk} \leq \sum_{j \neq k} m_{jk}$$

$\Rightarrow |\lambda| \leq \sum_j m_{jk} = 1$ (sum down the k^{th} col) ∎

Def'n: A Markov matrix Q is **positive** if

$Q_{ij} > 0 \quad \forall i,j$ (i.e. no zero elements)

Thm: If Q is a positive Markov matrix, then there is only one eigenvector with $|\lambda| = 1$.

Pf: Isn't it obvious?!  ∎

Comment: Now we know that there is only one solution to $Mp = p$.

Thm: If M is a positive Markov matrix, then

$$\lim_{n\to\infty} M^n p^0$$

converges to a unique vector p^∞ for any initial probability vector p^0.

Pf: With relatively few assumptions, we can represent p^0 using an eigenvector basis

where we order the e-values by decreasing magnitude s.t. $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_R|$. Hence, x_1 is the unique e-vector of $\lambda_1 = 1$.

Notice that

$$M p^0 = \lambda_1 c_1 x_1 + \sum_{l=2}^{R} \lambda_l c_l x_l$$

$$\vdots$$

$$(M)^n p^0 =$$

and that , but

This is because $|\lambda_l| < 1$ for $l \neq 1$.

Thus, $\lim_{n\to\infty} (M)^n p^0 = c_1 x_1 = p^\infty$. ∎

Interpretation:

No matter what the initial distribution, simple iteration converges to the steady-state. Moreover, the rate of convergence depends on the second-largest eigenvalue; if λ_2 is close to 1, then convergence is slow. This is because the convergence results from waiting for the other eigen-components $\{\lambda_2, \ldots \lambda_R\}$ to decay down close to zero.

<u>Implementation</u>

To find the Page Rank "importance" score for each web page, we can start with some probability vector p^0 and iterate,

$$M^n p^0 \approx p^\infty \quad \text{for sufficiently large } n$$

Recall:

$$M = \alpha P' + (1-\alpha)\frac{1}{R} e e^T$$

$$=$$

$$M = \alpha P' + (1-\alpha)\frac{1}{R} e e^T$$

$$=$$

So, $M_p =$

$$=$$

Since P is sparse, it can be stored and applied in $O(R)$.
Thus, M_p takes $O(R)$ flops.

Solving Triangular Systems

A triangular matrix has zeros either above the diagonal, or below the diagonal.

eg. $\begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{bmatrix}$ is ...

$\begin{bmatrix} a & 0 & 0 & 0 \\ b & c & 0 & 0 \\ d & e & f & 0 \\ g & h & i & j \end{bmatrix}$ is...

$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$ is...

Triangular matrices occur in certain matrix factorizations, and are a useful type of matrix.

Solving Upper-Triangular Systems: Back Substitution

$Ux = z$ U is upper-Δ matrix

$$\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1N} \\ 0 & u_{22} & \cdots & u_{2N} \\ \vdots & & \ddots & \\ & & u_{N-1,N-1} & u_{N-1,N} \\ 0 & & \cdots & u_{N,N} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{bmatrix}$$

Start with the last row

Now it's easy to solve for x_{N-1} in the second-last row.

The i-th row…

Back Substitution Algorithm (a.k.a. Back Solve)
(see page 101 in the course notes)

Complexity
For each i, the j-loop performs $2(N-i)$ flops (floating-point operations). Together with $\div u_{ii}$ ⇒

Sum over i

$$\text{Total flops} = \sum_{i=1}^{N} \left(2(N-i)+1\right)$$

$$=$$

$$=$$

=

=

Forward Substitution

$Lx = z$ $\quad L$ is $N \times N$ lower-Δ

$$\begin{bmatrix} l_{11} & 0 & \cdots & \\ l_{21} & l_{22} & 0 & \cdots \\ \vdots & & & \\ l_{N1} & l_{N2} & \cdots & l_{NN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{bmatrix}$$

From 1st row,

i^{th} row

$$l_{i1}x_1 + l_{i2}x_2 + \cdots + l_{i,i-1}x_{i-1} + l_{ii}x_i = z_i$$

$$\Rightarrow x_i = \frac{z_i - (l_{i1}x_1 + \cdots + l_{i,i-1}x_{i-1})}{l_{ii}}$$

$$x_i = \frac{z_i - \sum_{j=1}^{i-1} l_{ij}x_j}{l_{ii}}$$

Gaussian Elimination

To solve a system of linear equations

$$Ax = b \qquad \text{eg. } \begin{bmatrix} 1 & 1 & 2 \\ -1 & -2 & 3 \\ 3 & -7 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 1 \\ 10 \end{bmatrix}$$

one can use Gaussian elimination.

1) Form the augmented matrix.

$$\left[\begin{array}{ccc|c} 1 & 1 & 2 & 8 \\ -1 & -2 & 3 & 1 \\ 3 & -7 & 4 & 10 \end{array}\right]$$

2) Perform linear row operations to get an upper-$\triangle$ form

3) You might have been taught to follow this with more row operations to get a diagonal matrix.

$$\Rightarrow \cdots \Rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 \end{array}\right] \quad \therefore \text{ solution is}$$

A better way is to use back substitution after step 2.

Solve $\begin{bmatrix} 1 & 1 & 2 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ -9 \\ 2 \end{bmatrix}$

LA-05: Gaussian Elimination

Goal: To review how row operations can be used to reduce and solve a linear system.

LU Factorization

Any square matrix A can be factored into a product of an upper-triangular and lower-triangular matrices such that

$$LU = PA$$

P is a permutation matrix used to swap rows.

LU factorization is essentially the same as Gaussian elimination.

eg. $\begin{bmatrix} 2 & 3 & -1 \\ 4 & 6 & -1 \\ -2 & 2 & -3 \end{bmatrix}$

In this case, there is no way to use the pivot element to get rid of the 5. So, swap rows 2 and 3.

$$\begin{bmatrix} 2 & 3 & -1 \\ 0 & 5 & -4 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \quad & \quad & \quad \\ \quad & \quad & \quad \\ \quad & \quad & \quad \end{bmatrix} \begin{bmatrix} 2 & 3 & -1 \\ 0 & 0 & 1 \\ 0 & 5 & -4 \end{bmatrix}$$

We will go over the details later, but for now I'll simply state that LU factorization takes $\mathcal{O}(N^3)$ flops.

Applications of LU Factorization

a) Solving $Ax = b$

Two steps to compute x

1) Solve
2) Solve

Gaussian elimination = LU factorization + forward sub
+ back sub

b) Solving $AX = B$

Suppose X and B each have M columns.

$$A\left[x_1 \mid \cdots \mid x_M\right] = \left[b_1 \mid \cdots \mid b_M\right]$$

1) Factor:
2) Solve:

Take-home message: Do the expensive LU factorization once, and use it for each of the M systems.

Gaussian Elimination (GE)

2x2 example

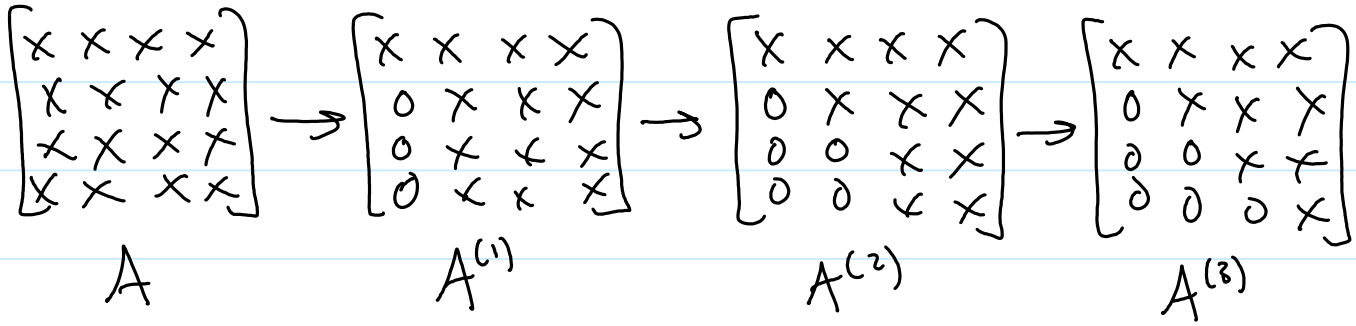
$$\begin{cases} a_{11}x_1 + a_{12}x_2 = b_1 & (1) \\ a_{21}x_1 + a_{22}x_2 = b_2 & (2) \end{cases}$$

$② - \frac{a_{21}}{a_{11}} ① \Rightarrow$

In matrix form: $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$

$② - \frac{a_{21}}{a_{11}} ① \Rightarrow$

For general N , the big picture of GE…

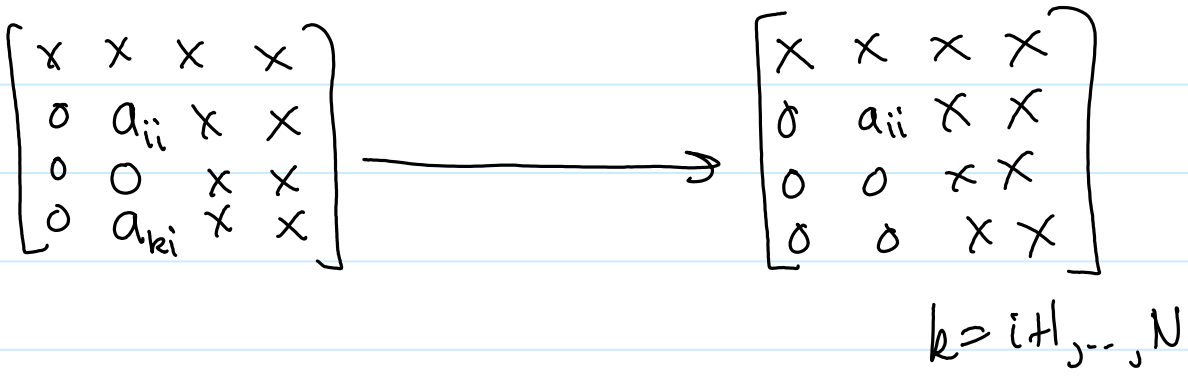


GE Version 1:

for i = 1:N-1

end

At the i-th stage of GE…

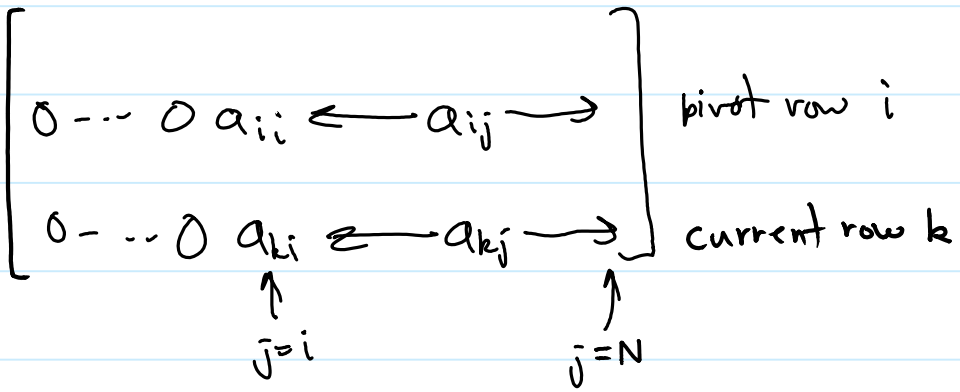


GE Version 2:

for i = 1:N-1

end

To update the entire row k :



for j = i+1:N

end

(Note: $a_{ki} = 0$ by design)

Final GE Algorithm:

for i = 1:N-1

end

Notes: 1) The lower-triangular part is all 0.
2) We may use those elements to store the multipliers.

LU Factorization Algorithm

Consider the first step of GE,

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & & \\ a_{N1} & \cdots & a_{NN} \end{bmatrix} \xrightarrow{\text{row operations}} \begin{bmatrix} a_{11}^{(1)} & & a_{1N}^{(1)} \\ 0 & a_{22}^{(1)} & \cdots \\ \vdots & \vdots & \\ 0 & a_{N2}^{(1)} & \cdots a_{NN}^{(1)} \end{bmatrix} = A^{(1)}$$

Matrix interpretation:

row operations $\Longleftrightarrow$ matrix multiplication

ie. $M^{(1)} A = A^{(1)}$

where

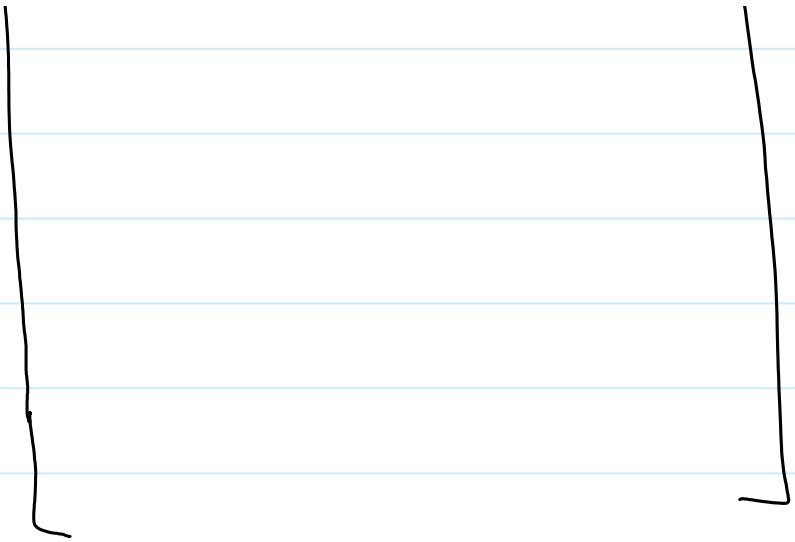
$$M^{(1)} = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$$

In general, at the k-th step…

$$M^{(k)} A^{(k-1)} = A^{(k)}$$

where

$$M^{(k)} = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$$



The effect of left-multiplying $A^{(k-1)}$ by $M^{(k)}$ is to eliminate x_k from rows $k+1$ to N.

At the final step: $A^{(N-1)}$ is

Recall that $M^{(k)} A^{(k-1)} = A^{(k)}$

For $k=1 \Rightarrow$

Amazing Facts

1) If B and C are lower-Δ and unit diagonal, then
2) If B is lower-Δ and unit diagonal, then

By fact (1), $M^{(N-1)} \cdots M^{(1)}$ is

By fact (2), $\left[M^{(N-1)} \cdots M^{(1)}\right]^{-1}$ is

Define

Theorem: $A = LU$

L is lower-Δ & unit diagonal

U is upper-Δ.

Properties of $M^{(k)}$:

1)

$$(M^{(k)})^{-1} = \begin{bmatrix} 1 & & & & \\ & \ddots & & 0 & \\ & & 1 & & \\ & 0 & & \ddots & \\ & & & & 1 \end{bmatrix}$$

2) $L =$

$$= \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & 0 & & 1 \end{bmatrix}$$

i.e. the k^{th} col. of L is the k^{th} col. of $(M^{(k)})^{-1}$.

eg. $\begin{bmatrix} 1 & 0 & 0 \\ \alpha & 1 & 0 \\ \beta & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \gamma & 1 \end{bmatrix} =$

Note that $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \gamma & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ \alpha & 1 & 0 \\ \beta & 0 & 1 \end{bmatrix}$ is not so straight forward.

More precisely, write $L = (l_{jk})$. Then

$= \{$

Example:

$A = \begin{bmatrix} 2 & -1 & 3 \\ -4 & 6 & -5 \\ 6 & 13 & 16 \end{bmatrix} \longrightarrow \begin{bmatrix} 2 & -1 & 3 \\ & & \end{bmatrix}$

$L = \begin{bmatrix} 1 & 0 & 0 \\ & 1 & 0 \\ & & 1 \end{bmatrix}$ $\searrow$ $\begin{bmatrix} 2 & -1 & 3 \\ & & \end{bmatrix} = U$

Check:

LA-07: Pivoting

Goal: To stabilize LU factorization using row pivoting.

Stability of LU Factorization

In LU factorization, a problem arises when we have:

(1) a zero pivot i.e. $a_{kk}^{(k-1)} = 0$

$\Rightarrow$ multipliers $\frac{a_{jk}}{a_{kk}}$ are undefined

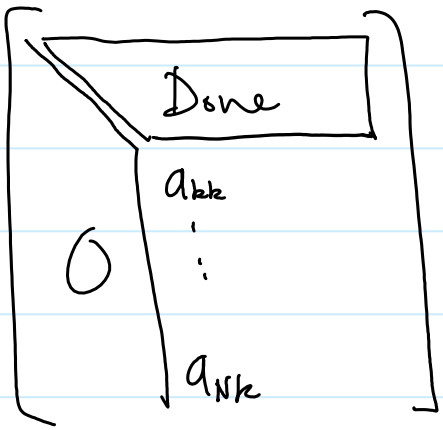
(2) $a_{kk}^{(k-1)} \approx 0$

$\Rightarrow$ multipliers become large

$\Rightarrow$ calculations become unstable

Pivoting

$A^{(k-1)} =$



Find

Swap rows k^* and k, and continue.

Note: $a_{k^*k}^{(k-1)} \neq 0$

Otherwise, $a_{jk}^{(k-1)} = 0 \quad \forall\ k \leq j \leq N$

$\Rightarrow$

As mentioned earlier, these row-swapping operations can be represented by matrix multiplication by a permutation matrix, P.
eg. to swap rows i & j, simply swap rows i & j in the identity matrix.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \xrightarrow{\text{Swap } ② \text{ \& } ③} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = P$$

Let's put it all together.

eg. $A = \begin{bmatrix} 1 & 1 & 1 \\ 4 & 16 & 64 \\ 2 & 2 & 8 \end{bmatrix}$

$\downarrow$

$\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$ $\quad P = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$

$\downarrow$

$\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$ $\quad L = \begin{bmatrix} 1 & 0 & 0 \\ & 1 & 0 \\ & & 1 \end{bmatrix}$

$\downarrow$

$\begin{bmatrix} & & \\ & & \end{bmatrix}$ $\quad P = \begin{bmatrix} & & \\ & & \end{bmatrix}$

$$\begin{bmatrix} & \downarrow & \\ & & \end{bmatrix} \qquad P = \begin{bmatrix} & \\ & \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ & 1 & 0 \\ & & 1 \end{bmatrix}$$

$$\downarrow$$

$$\begin{bmatrix} & \\ & \end{bmatrix}$$

$$\therefore \begin{bmatrix} 1 & 0 & 0 \\ & 1 & 0 \\ & & 1 \end{bmatrix} \begin{bmatrix} 0 & \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} & \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 4 & 16 & 64 \\ 2 & 2 & 8 \end{bmatrix}$$

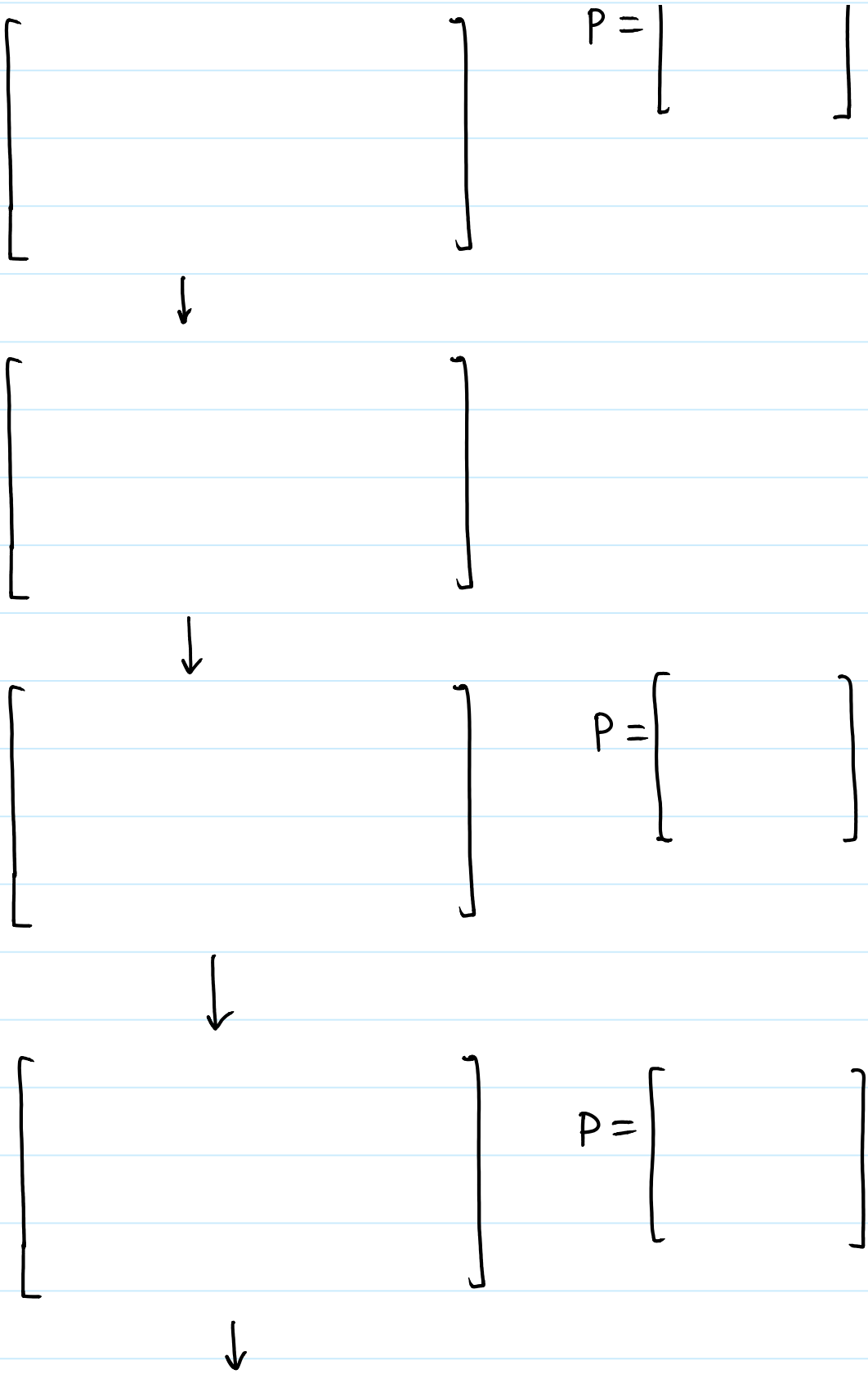
$$L \qquad U \qquad = \qquad P \qquad A$$

Example:

$$A = \begin{bmatrix} -6 & 27 & -42 & -15 \\ -24 & 12 & 24 & -12 \\ -12 & 14 & -10 & 20 \\ -6 & -9 & 42 & 15 \end{bmatrix}$$

$$\downarrow$$

$$\begin{bmatrix} & \end{bmatrix} \qquad P = \begin{bmatrix} & \\ & \end{bmatrix}$$


P =
P =
P =

$$\begin{bmatrix} & \end{bmatrix}$$

$$\therefore \begin{bmatrix} & \end{bmatrix} \begin{bmatrix} -6 & 27 & -42 & -15 \\ -24 & 12 & 24 & -12 \\ -12 & 14 & -10 & 20 \\ -6 & -9 & 42 & 15 \end{bmatrix}$$

$$= \begin{bmatrix} & \end{bmatrix} \begin{bmatrix} & \end{bmatrix}$$

LA-08: SVD

Goal: To learn about the Singular Value Decomposition, and some of its applications.

Singular Value Decomposition (SVD)

A geometric observation:

The image of a unit sphere under any $m \times n$ matrix is a hyperellipse.

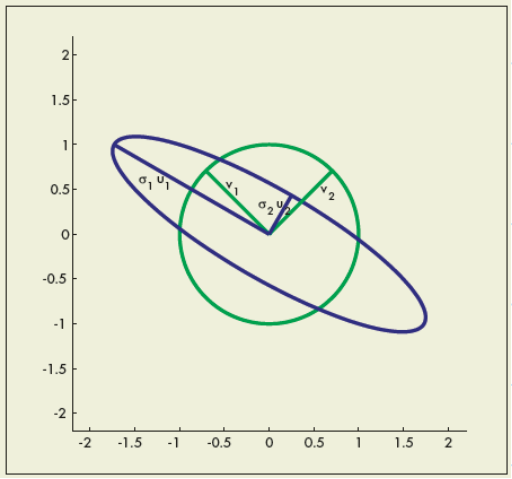
Let S be a unit sphere in $\mathbb{R}^n$

The image, AS, is an ellipse in $\mathbb{R}^m$.

The n singular values of A are the lengths of the n principle semi-axes of AS, $\sigma_1, \sigma_2, \ldots, \sigma_n$.

By convention,

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0.$$



The n left singular vectors of A are the unit vectors $\{u_1, u_2, \ldots, u_n\}$ in the directions of the principle axes.

$\therefore \sigma_i u_i$ is the i^{th} largest principal semiaxis.

The n right singular vectors are the unit vectors

$$\{v_1, v_2, \ldots, v_n\} \in S \quad \text{s.t.} \quad Av_j = \sigma_j u_j$$

Theorem:

Every matrix $A \in \mathbb{R}^{m \times n}$ has an SVD. The singular values are unique. If A is square, and σ_j are distinct, then the left and right singular vectors are uniquely determined (modulo sign).

If A is square, then so are U, Σ, and V.

Note that U and V are unitary (orthogonal) matrices.

i.e.

If A is taller than it is wide

Full SVD

Reduced SVD

Note that the first n columns of U are $\hat{U}$, and that Σ is just $\hat{\Sigma}$ augmented with rows of zeros.

$$\Sigma = \begin{bmatrix} \hat{\Sigma} \\ 0 \end{bmatrix}$$

SVD… What is it good for?

- The rank of A is the
- is the span of the right singular vectors (v_j) corresponding to zero singular values.
- cond(A) =
- computing norms: $\|A\|_2 =$ $\|A\|_F =$
- eigenvalues: if $U\Sigma V^T = A^T A$, then

- low-rank approximation

Low-Rank Approximation using the SVD

Theorem:

For any integer k, $0 \le k \le \text{rank}(A)$, define

$$A_k = \sum_{j=1}^{k} \sigma_j u_j v_j^T$$

Then,

$$\|A - A_k\|_2 =$$

In other words, if you want a rank-k approximation to A, simply set all singular values to zero except for the largest k.

$$A_k = U \begin{bmatrix} & \\ & \end{bmatrix} V^T$$

This result has profound implications in statistics and scientific computing.

<u>Application: Principal Component Analysis</u>

Suppose you have a bunch of points in $\mathbb{R}^n$ but you would like to represent your data as closely as possible in a lower-dimensional space.

eg. a cloud of data points in $\mathbb{R}^3$ might actually be well approximated by a plane.

Place the coordinates of the points in the columns of A, then compute the SVD $\Rightarrow U \Sigma V^T = A$

Let $\Sigma_2 =$

Then $A_2 = U \Sigma_2 V^T$ holds the projection of all the points onto the best plane of approximation. (Matlab demo)

<u>Example: Image Compression</u>

Example: Image Compression

Instead of storing the whole image array as a matrix A, we can store a few singular vectors (and associated singular values) that give us A_k.

ie. if $A = U\Sigma V^T$, and $\Sigma_k = \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_k & \\ & & & 0 \ddots 0 \end{bmatrix}$

then $A_k = U\Sigma_k V^T$ is the best rank-k approximation to A.

(Matlab demo)

Application: Query Matching

Consider the following 7 book titles (documents):

D1: Infants and Toddler First Aid
D2: Babies & Children's Room (for your home)
D3: Child Safety at Home
D4: Your Baby's Health and Safety: From Infant to Toddler
D5: Baby Proofing Basics
D6: Your Guide to Easy Rust Proofing
D7: Beanie Babies Collectors Guide

The following 9 terms appear in many of the titles:

T1: Baby (babies, baby's)
T2: Child (children)
T3: Guide
T4: Health
T5: Home
T6: Infant
T7: Proofing
T8: Safety
T9: Toddler

We can summarize our collection of decuments in a term-document matrix.

$$A = \begin{matrix} \\ \text{term } i \end{matrix} \overset{\text{Doc. } j}{\begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}}$$

T1: Baby (babies, baby's)
T2: Child (children)
T3: Guide
T4: Health
T5: Home
T6: Infant

term i

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

T4: Health
T5: Home
T6: Infant
T7: Proofing
T8: Safety
T9: Toddler

Suppose we have a query, and we want to find related documents.

query = "Child proofing" ⇒

We can compute the correlation (dot product) between our query vector and each document.

Let $\langle d, q \rangle = \dfrac{d \cdot q}{\|d\| \, \|q\|}$

$\langle d, q \rangle$

D1: Infants and Toddler First Aid
D2: Babies & Children's Room (for your home)
D3: Child Safety at Home
D4: Your Baby's Health and Safety: From Infant to Toddler
D5: Baby Proofing Basics
D6: Your Guide to Easy Rust Proofing
D7: Beanie Babies Collectors Guide

Conclusion: Comparison of documents based solely on term frequency is not adequate.

Instead, we automatically extract the most relevant subspace, and compare documents based on their classification in this "concept space". We accomplish this using a low-order approximagion to A.

$A = U\Sigma V^T$, then form A_2 (rank-2).

U_2 = 1st 2 cols of U
Σ_2 = 2x2 submatrix
V_2 = 1st 2 cols of V

$\Big\} \; A_2 = U_2 \Sigma_2 V_2^T$

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 0.1355 & 0.4652 & 0.2460 & 0.3882 & 0.5540 & 0.4072 & 0.5540 \\ 0.2266 & 0.2704 & 0.2698 & 0.3134 & 0.0809 & -0.0577 & 0.0809 \\ -0.1445 & 0.1191 & -0.0908 & -0.0071 & 0.4342 & 0.4612 & 0.4342 \\ 0.1044 & 0.1198 & 0.1230 & 0.1413 & 0.0292 & -0.0337 & 0.0292 \\ 0.2266 & 0.2704 & 0.2698 & 0.3134 & 0.0809 & -0.0577 & 0.0809 \\ 0.2482 & 0.2403 & 0.2799 & 0.3064 & -0.0045 & -0.1445 & -0.0045 \\ -0.1445 & 0.1191 & -0.0908 & -0.0071 & 0.4342 & 0.4612 & 0.4342 \\ 0.2326 & 0.2448 & 0.2678 & 0.3001 & 0.0284 & -0.1070 & 0.0284 \\ 0.2482 & 0.2403 & 0.2799 & 0.3064 & -0.0045 & -0.1445 & -0.0045 \end{bmatrix}$$

T1: Baby (babies, baby's)
T2: Child (children)
T3: Guide
T4: Health
T5: Home
T6: Infant
T7: Proofing
T8: Safety
T9: Toddler

The low-order approximation forms a bottleneck through which the association between term frequency and documents is classified. This is the concept space.

$U_2^T d$ gives the coordinates of a document d in the concept space.

$\Sigma_2^{-1} V_2^T t$ gives the coordinates of a term frequency vector t in concept space.