

L19: 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.

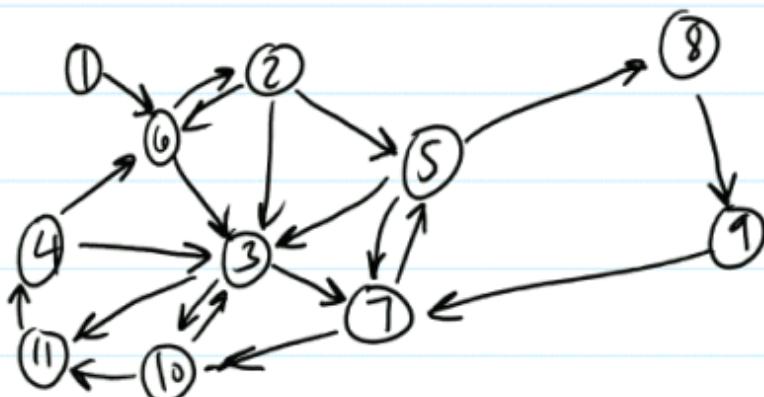


Each time he clicks to a new web page, he looks at the links on the page, and chooses one randomly (hence his name).

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 Waterloo 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 ③ more often than ⑨.

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-chosen 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

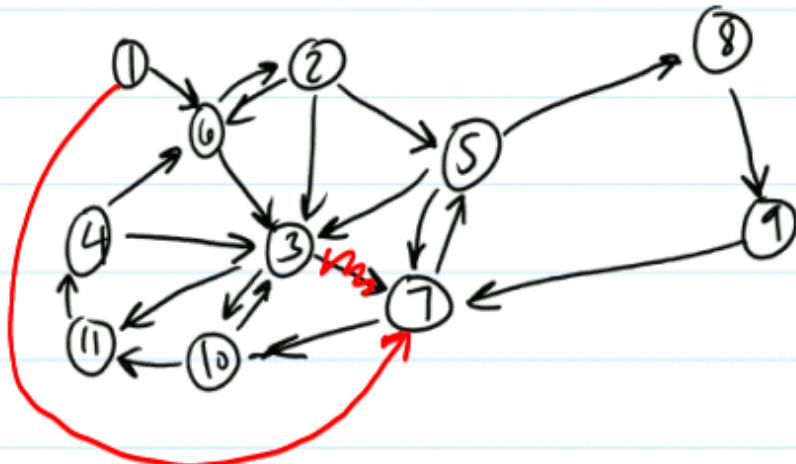
plot(1:N, visits);

```

One way to measure the value or importance of a web page is
what fraction of time a random surfer will spend there.

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.

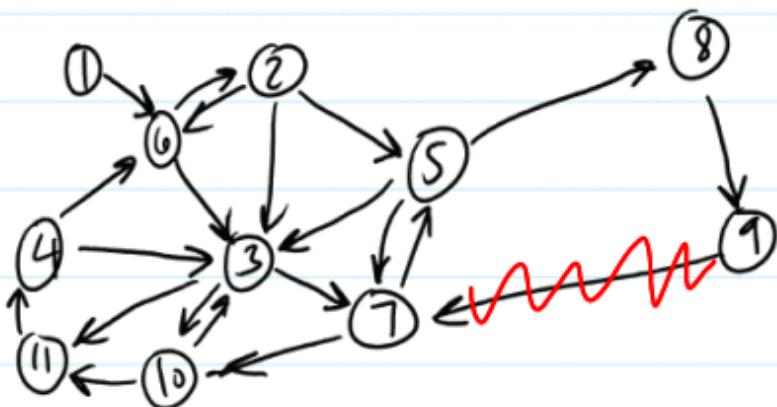
Try removing the link from $3 \rightarrow 7$, and replace it with $1 \rightarrow 7$



The importance of
 $\textcircled{7}$ decreases
Note: it matters what
is linking to $\textcircled{7}$
 $(\textcircled{3} \text{ vs. } \textcircled{1})$

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, jump to another webpage chosen randomly

Problem: The above issue is actually more insidious.

Consider...



This is a terminal branch, and can be very difficult to detect.

Solution: Random teleportation

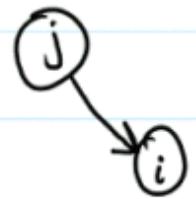
At each page, we jump to a random webpage with probability $(1-\alpha)$

\therefore we follow an outlink with prob α

Mathematical Formulation of Random Surfer Ranking

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

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



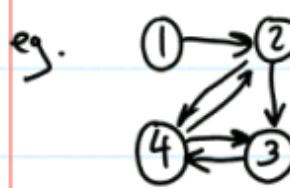
Consider, instead, matrix $P \in [0,1]^{R \times R}$

$$P_{ij} = \begin{cases} \frac{1}{\deg(j)} & \text{if node } j \text{ links to node } i \\ 0 & \text{otherwise} \end{cases}$$

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 .

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



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

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...

$$\text{i.e. } x = [0 \ 1 \ 0 \ 0]^T$$

$$Px = [0 \ 0 \ \frac{1}{2} \ \frac{1}{2}]^T$$

$$P^2x = P(Px) = \begin{bmatrix} 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \end{bmatrix}^T$$

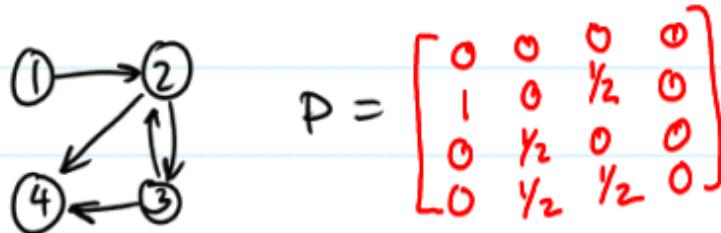
$$P^3x = \begin{bmatrix} 0 & \frac{1}{4} & \frac{3}{8} & \frac{3}{8} \end{bmatrix}^T$$

\therefore after 3 clicks, $\frac{3}{8}$ of the surfers 37.5% 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



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

$$Px = P[0 \ 1 \ 0 \ 0]^T = [0 \ 0 \ \frac{1}{2} \ \frac{1}{2}]^T$$

$$P^2x = [0 \ \frac{1}{4} \ 0 \ \frac{1}{4}]^T \rightarrow \text{Only } 50\% \text{ of the surfers left,}$$

$$P^3x = [0 \ 0 \ \frac{1}{8} \ \frac{1}{8}]^T \rightarrow \text{Only } 33\% \text{ left}$$

$$P^{\infty}x = \dots \rightarrow \text{Only } 0.0001\% \text{ left!}$$

The matrix equivalent of the teleportation effect can be included...

$$P' = P + \frac{1}{R} e d^T$$

$d_j = \begin{cases} 1 & \text{if node } j \text{ has no outlinks} \\ 0 & \text{otherwise} \end{cases}$
 $e = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_R$

In our example,

$$\frac{1}{4} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} [0001]$$

$$P' = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & \frac{1}{4} \\ 0 & 0 & 0 & \frac{1}{4} \\ 0 & 0 & 0 & \frac{1}{4} \\ 0 & 0 & 0 & \frac{1}{4} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{4} \\ 1 & 0 & \frac{1}{2} & \frac{1}{4} \\ 0 & \frac{1}{2} & 0 & \frac{1}{4} \\ 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}$$

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

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

$\xrightarrow{(1-\alpha)} \frac{1}{R} \begin{bmatrix} \frac{1}{4} & \cdots & \frac{1}{4} \\ \vdots & \ddots & \vdots \\ \frac{1}{4} & \cdots & \frac{1}{4} \end{bmatrix}$

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

M is called a Markov Transition Matrix.

Markov Transition Matrices

Def'n A matrix Q is a Markov matrix if

$$0 \leq Q_{ij} \leq 1 \text{ and } \sum_i Q_{ij} = 1 \quad (\text{sum down cols})$$

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

Def'n A vector q is a probability vector if

$$0 \leq q_i \leq 1 \quad \text{and} \quad \sum_i q_i = 1$$

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

Consider $p = Mx$

$$\begin{aligned} \sum_i p_i &= \sum_i [Mx]_i \\ &= \sum_i \sum_j M_{ij} x_j \\ &= \sum_j x_j \sum_i M_{ij} \\ &= \sum_j x_j \\ &= 1 \end{aligned}$$

$$\sum_i p_i = 1$$

$$\sum_i p_i = 1$$

$\therefore p$ is a prob vector.

Page Rank

Let " v " 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

$$p^{n+1} = Mp^n \quad (1)$$

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

i.e. $p = Mp$

This is actually an eigenvalue problem,

$$p = Mp \Leftrightarrow (I - M)p = 0$$

Eigenvalues & Eigenvectors

The eigenvectors, \mathbf{x} , of a square matrix Q satisfy

$$Q\mathbf{x} = \lambda\mathbf{x}$$

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

To find eigenvalues, we can form the characteristic polynomial.

$$\begin{aligned} Q\mathbf{x} &= \lambda\mathbf{x} \\ \Rightarrow (\lambda I - Q)\mathbf{x} &= 0 \quad (2) \end{aligned}$$

Any nontrivial (ie. not entirely zero) \mathbf{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.

$$\Rightarrow \det(\lambda I - Q) = 0$$

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 eigenvector corresponding to an eigenvalue of 1,

$$(I - P)x = 0$$

It is not feasible to use standard algebraic methods to solve

Instead, watch this... 

Result,

```
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;
```

Quickly converges to a steady state solution of
 $x^{n+1} = Px^n$

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

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

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

$$\Rightarrow M p^\infty = p^\infty \quad (p^\infty \text{ is a fixed point of the mapping } M)$$

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 $e = [1 \ 1 \ \dots \ 1]^T$.

Since Markov matrix
columns add to 1

$$Q^T e = \begin{bmatrix} \sum Q_{j1} e_j \\ \sum Q_{j2} e_j \\ \vdots \\ \sum Q_{jn} e_j \end{bmatrix} = \begin{bmatrix} \sum Q_{j1} \\ \sum Q_{j2} \\ \vdots \\ \sum Q_{jn} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = e$$

so $Q^T e = \lambda e$
for $\lambda = 1$

so 1 is an eigenvalue
of Q

Comment: OK, so now we know that there is a solution to $M_p = p$.

But how do we explain convergence?

Note:
 $\det(\pi I - Q) = 0$
 $\lambda e((\pi I - Q)^{-1}) = 0$
 $\det(\pi I - Q)^{-1}$

Thm: Every (possibly complex) ϵ -value λ of a Markov matrix M satisfies $|\lambda| \leq 1$.

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

Thm: Gershgorin Circle Thm.

Let B be an arbitrary matrix. Then the ϵ -vals.

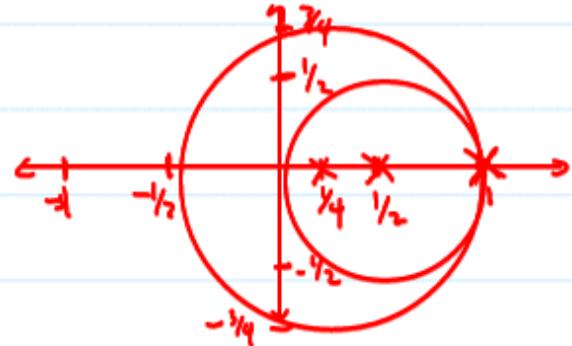
λ of B are located in the union of the n disks (one disk for each row in B)

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

↑ centre ↑ radius = sum of absolute value of all non-diagonal entries in column k

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

	center	radius
col 1 \Rightarrow	1	0
col 2 \Rightarrow	$\frac{1}{2}$	$\frac{1}{2}$
col 3 \Rightarrow	$\frac{1}{4}$	$\frac{3}{4}$

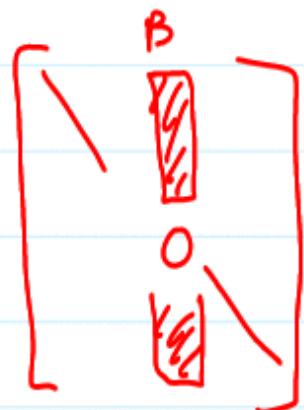


The Gershgorin Thm. implies

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

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

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



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

■

Def'n: A Markov matrix Q is positive if

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

Ihm: 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 $M_p = p$.

Ihm: If M is a positive Markov matrix, then

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

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

$$p^0 = \sum c_j x_j = c_1 x_1 + \sum_{j=2}^R c_j x_j$$

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

Notice that $Mp^0 = c_1 Mx_1 + \sum_{j=2}^R c_j Mx_j = c_1 \lambda_1 x_1 + \sum_{j=2}^R c_j \lambda_j x_j$

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

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

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

and that $\lambda_1^n = 1$, but $\lambda_l^n \rightarrow 0$ as $n \rightarrow \infty$

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

Thus,

$$\lim_{n \rightarrow \infty} (M)^n p^0 = c_1 x_1 = p^\infty.$$

■

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 $\{\lambda_2, \dots, \lambda_R\}$ to decay down close to zero.

Implementation

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

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

Recall:

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

$$= \alpha \left(P + \frac{1}{R} e d^T \right) + (1-\alpha) \frac{1}{R} e e^T$$

$$P' = P + \frac{1}{R} e d^T$$

$d_j = \begin{cases} 1 & \text{if } m \text{ out links from } j \\ 0 & \text{otherwise} \end{cases}$

Recall:

$$\begin{aligned} M &= \alpha P' + (1-\alpha) \frac{1}{R} e e^T \\ &= \alpha \left(P + \frac{1}{R} e d^T \right) + (1-\alpha) \frac{1}{R} e e^T \end{aligned}$$

So, $M_p = \alpha P_p + \frac{\alpha}{R} e d^T p + (1-\alpha) \frac{1}{R} e e^T p \rightarrow = \sum p_i = 1$

$$= \alpha P_p + \left(\frac{\alpha d^T p}{R} \right) e + \left(\frac{1-\alpha}{R} \right) e$$

Since P is sparse, it can be stored and applied in $\mathcal{O}(R)$.

Thus, M_p takes $\mathcal{O}(R)$ flops.

Sparse Matrix e.g.

$$\left[\begin{array}{c} [[a_{13}, 3], [a_{197}, 97]], \\ [[a_{24}, 4]], \\ \dots \end{array} \right]$$

Solving Triangular Systems

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

e.g. $\begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{bmatrix}$ is ... *upper triangular matrix* $\begin{bmatrix} a & 0 & 0 & 0 \\ b & c & 0 & 0 \\ d & e & f & 0 \\ g & h & i & j \end{bmatrix}$ is ... *lower triangular matrix*

$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$ is ... *upper + lower triangular matrix*
 \Rightarrow *diagonal*

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

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

Start with the last row

$$U_{NN}x_N = z_N \Rightarrow x_N = \frac{z_N}{U_{NN}}$$

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

$$U_{N-1,N-1}x_{N-1} + U_{N-1,N}x_N = z_{N-1}$$

$$U_{N-1,N-1}x_{N-1} = z_{N-1} - U_{N-1,N}x_N$$

$$x_{N-1} = \frac{z_{N-1} - U_{N-1,N}x_N}{U_{N-1,N-1}}$$

The i-th row...

$$U_{i,i}x_i + U_{i,i+1}x_{i+1} + U_{i,i+2}x_{i+2} + \dots + U_{i,N}x_N = z_i$$

$$U_{i,i}x_i = z_i - (U_{i,i+1}x_{i+1} + \dots + U_{i,N}x_N)$$

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

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 $\cancel{x_i \cancel{\frac{z_i}{U_{ii}}}}$ dividing by U_{ii}

$$\text{Sum over } i = \sum_{i=1}^N \left(\sum_{j=i+1}^N z_j \right) + 1 \text{ and } \alpha \text{ operations}$$

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

$$= \sum_{i=1}^N (2N - 2i + 1)$$

$$= \sum_{i=1}^N 2N - \sum_{i=1}^N 2i + \sum_{i=1}^N 1$$

We know

$$\sum_{i=1}^n i = \frac{n(n+1)}{2}$$

$$= 2N^2 - \frac{2(N)(N+1)}{2} + N$$
$$= N^2$$

Forward Substitution

$$Lx = z \quad L \text{ is } N \times N \text{ lower-}\Delta$$

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

From 1st row,
 $l_{11}x_1 = z_1$
 $x_1 = \frac{z_1}{l_{11}}$

ith 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

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

one can use Gaussian elimination.

1) For the augmented matrix.

$$\begin{array}{c} \textcircled{1} \\ \textcircled{2} \\ \textcircled{3} \end{array} \begin{array}{c} \left[\begin{array}{ccc|c} 1 & 1 & 2 & 8 \\ -1 & -2 & 3 & 1 \\ 3 & -7 & 4 & 10 \end{array} \right] \xrightarrow{\textcircled{1} + \textcircled{1}} \left[\begin{array}{ccc|c} 1 & 1 & 2 & 8 \\ 0 & -1 & 5 & 9 \\ 3 & -7 & 4 & 10 \end{array} \right] \xrightarrow{\textcircled{3} - 10\textcircled{1}} \left[\begin{array}{ccc|c} 1 & 1 & 2 & 8 \\ 0 & -1 & 5 & 9 \\ 0 & 0 & -52 & -104 \end{array} \right] \\ A \quad \mid \quad b \end{array}$$

2) Perform linear row operations to get an upper- Δ form

$$\xrightarrow{-\frac{1}{5}\textcircled{2}} \left[\begin{array}{ccc|c} 1 & 1 & 2 & 8 \\ 0 & -1 & 5 & 9 \\ 0 & 0 & 1 & 2 \end{array} \right] \xrightarrow{-\textcircled{2}} \left[\begin{array}{ccc|c} 1 & 1 & 2 & 8 \\ 0 & 1 & -5 & -9 \\ 0 & 0 & 1 & 2 \end{array} \right]$$

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

$$\Rightarrow \dots \Rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 \end{array} \right] \therefore \text{solution is } \begin{array}{l} x_1 = 3 \\ x_2 = 1 \\ x_3 = 2 \end{array}$$

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

Solve

$$\begin{bmatrix} 1 & 1 & 2 \\ 0 & 1 & -5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ -9 \\ 2 \end{bmatrix} \quad \begin{array}{l} x_1 + x_2 + 2x_3 = 8 \Rightarrow x_1 = 3 \\ x_2 - 5x_3 = -9 \Rightarrow x_2 = 1 \\ x_3 = 2 \end{array}$$

L23: 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 matrix such that

$$LU = PA$$

P is a permutation matrix used to swap rows.

LU factorization is essentially the same as Gaussian elimination.

eg.

$$\begin{bmatrix} 2 & 3 & -1 \\ 4 & 6 & -1 \\ -2 & 2 & -3 \end{bmatrix} \xrightarrow{\substack{(2) - 2(1) \\ (3) + (1)}} \begin{bmatrix} 2 & 3 & -1 \\ 0 & 0 & 1 \\ 0 & 5 & -4 \end{bmatrix}$$

pivot element

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

$$\begin{bmatrix} 2 & 3 & -1 \\ 0 & 5 & -4 \\ 0 & 0 & 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}}_{\text{Permutation Matrix}} \begin{bmatrix} 2 & 3 & -1 \\ 0 & 0 & 1 \\ 0 & 5 & 4 \end{bmatrix}$$

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 $Ax = b$

Given $LU = PA$,

$$Ax = b \Rightarrow PAx = Pb \Rightarrow LUx = Pb$$

Let $z = Ux$ and $\beta = Pb$

Two steps to compute x

1) Solve $Lz = Pb$ for some $z \in \mathbb{R}^N$

2) Solve $Ux = z$ for $x \in \mathbb{R}^N$

Gaussian elimination = LU factorization + forward sub
 $O(N^3)$ + back sub
 $O(N^4)$

b) Solving $AX = B$

Suppose X and B each have M columns.

$$A \begin{bmatrix} X_1 & | & \dots & | & X_M \end{bmatrix} = \begin{bmatrix} B_1 & | & \dots & | & B_M \end{bmatrix}$$

1) Factor: $LU = PA \leftarrow O(N^3)$

2) Solve: $LUx_i = Pb_i$

⋮

$$LUx_M = Pb_M$$

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

Gaussian Elimination (GE)

2x2 example

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

$$(2) - \frac{a_{21}}{a_{11}}(1) \Rightarrow \left(a_{21} - \frac{a_{21}}{a_{11}}a_{11} \right)x_1 + \left(a_{22} - \frac{a_{21}}{a_{11}}a_{12} \right)x_2 = b_2 - \frac{a_{21}}{a_{11}}b_1$$

$a_{22}^{(1)}$

In matrix form:

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

$$(2) - \frac{a_{21}}{a_{11}}(1) \Rightarrow \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} \\ 0 & a_{22}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \end{bmatrix}$$

For general N , the big picture of GE...

$$\begin{array}{c} \begin{bmatrix} X & X & X & X \\ X & X & X & X \\ X & X & X & X \\ X & X & X & X \end{bmatrix} \rightarrow \begin{bmatrix} X & X & X & X \\ 0 & X & X & X \\ 0 & X & X & X \\ 0 & X & X & X \end{bmatrix} \rightarrow \begin{bmatrix} X & X & X & X \\ 0 & X & X & X \\ 0 & 0 & X & X \\ 0 & 0 & X & X \end{bmatrix} \rightarrow \begin{bmatrix} X & X & X & X \\ 0 & X & X & X \\ 0 & 0 & X & X \\ 0 & 0 & 0 & X \end{bmatrix} \\ A \qquad \qquad A^{(1)} \qquad \qquad A^{(2)} \qquad \qquad A^{(3)} \end{array}$$

GE Version 1:

for i = 1:N-1

eliminate x_i from rows $i+1$ to N
end

At the i -th stage of GE...

$$\begin{array}{l} \text{pivot row} \\ \text{current row } k \end{array} \rightarrow \left[\begin{array}{cccc} x & x & x & x \\ 0 & a_{ii} & x & x \\ 0 & 0 & x & x \\ 0 & a_{ki} & x & x \end{array} \right] \xrightarrow{\textcircled{k}} -\frac{a_{k,i}}{a_{i,i}} \textcircled{1} \rightarrow \left[\begin{array}{cccc} x & x & x & x \\ 0 & a_{ii} & x & x \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{array} \right]$$

$k = i+1, \dots, N$

GE Version 2:

for $i = 1:N-1$

 for $k = i+1$ to N

 mult = a_{ki} / a_{ii}

 row(k) = row(k) - mult * row(i)

 end

end

To update the entire row k :

$$\left[\begin{array}{cccc} 0 & \dots & 0 & a_{ii} \xleftarrow{a_{ij}} \\ 0 & \dots & 0 & a_{ki} \xleftarrow{a_{kj}} \end{array} \right] \begin{array}{l} \text{pivot row } i \\ \text{current row } k \end{array}$$

$\uparrow \quad \uparrow$
 $j=i \quad j=N$

for $j = i+1:N$

$$a_{kj} = a_{kj} - \text{mult} * a_{ij}$$

end

end

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

Final GE Algorithm:

for $i = 1:N-1$

 for $k = i+1$ to N

$$\text{mult} = a_{ki}/a_{ii}$$

 for $j = i+1$ to N

$$2 \quad a_{kj} = a_{kj} - \text{mult} \times a_{ij}$$

 end

$$a_{ki} = 0$$

end

end

$$\begin{aligned} & \sum_{i=1}^{N-1} \sum_{k=i+1}^N \left[1 + \sum_{j=i+1}^N 2 \right] \\ &= \sum_{i=1}^{N-1} \sum_{k=i+1}^N 1 + \sum_{i=1}^{N-1} \sum_{k=i+1}^N \sum_{j=i+1}^N 2 \\ &= \sum_{i=1}^{N-1} (N-i) + \sum_{i=1}^{N-1} \sum_{k=i+1}^N (N-i) 2 \\ &= \sum_{i=1}^{N-1} (N-i) + 2 \sum_{i=1}^{N-1} (N-i)^2 \\ &= \sum_{i=1}^{N-1} i + 2 \sum_{i=1}^{N-1} i^2 \\ &\therefore \Rightarrow O(n^3) = \frac{(N-1)N}{2} + 2 \frac{(N-1)N(2N-1)}{6} \\ &\quad \cdots = \frac{2}{3}N^3 - \frac{1}{2}N^2 - \frac{1}{6}N \end{aligned}$$

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

L24: LU Factorization

Goal: To find out how to compute LU factorization.

LU Factorization Algorithm

Consider the first step of GE,

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

Matrix interpretation:

row operations \Leftrightarrow matrix multiplication

$$\text{i.e. } M^{(1)} A = A^{(1)}$$

where

$$M^{(1)} = \begin{bmatrix} 1 & & & \\ -\frac{a_{21}}{a_{11}} & 1 & & \\ & \ddots & \ddots & \\ -\frac{a_{N1}}{a_{11}} & & 1 & \end{bmatrix}$$

In general, at the k-th step...

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

where

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

$$M^{(k)} = \begin{bmatrix} 1 & & & & \\ -\frac{a_{(k+1),k}}{a_{kk}} & 1 & & & \\ & \vdots & 1 & & \\ & & -\frac{a_{(N-k),k}}{a_{kk}} & 1 & \\ 0 & 0 & \ddots & 0 & \end{bmatrix}$$

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 *upper-triangular* 

Recall that

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

For $k=1 \Rightarrow M^{(1)} A = A^{(1)}$

$$M^{(2)} M^{(1)} = M^{(2)} A^{(1)} = A^{(2)}$$

$$\vdots$$

$$M^{(N-1)} \dots M^{(1)} A = A^{(N-1)} = V$$

$$A = [M^{(N-1)} \dots M^{(1)}]^{-1} V$$

Amazing Facts

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

By fact (1), $M^{(N-1)} \cdots M^{(1)}$ is lower- Δ unit diag.

By fact (2), $[M^{(N-1)} \cdots M^{(1)}]^{-1}$ is lower- Δ unit diag.

Define $L = [M^{(N-1)} \cdots M^{(1)}]^{-1}$

Theorem: $A = LU$

L is lower- Δ & unit diagonal

U is upper- Δ .

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

1)

$$(M^{(k)})^{-1} = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 0 & & \\ & & & \frac{\alpha_{1k}}{\alpha_{kk}} & \\ & & & ; & \\ & & & \frac{\alpha_{nk}}{\alpha_{kk}} & \end{bmatrix}$$

2) $L = [M^{(N-1)} \cdots M^{(1)}]^{-1} = (M^{(1)})^{-1} \cdots (M^{(N-1)})^{-1}$

$$= \begin{bmatrix} 1 & & & \\ & 1 & & 0 \\ & & \ddots & \\ & & & 1 \\ \frac{a_{ik}}{a_{kk}} & & & \\ \frac{a_{jk}}{a_{kk}} & & & \\ = \text{multipliers} & & & \end{bmatrix}$$

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

$$\text{e.g. } \begin{bmatrix} 1 & 0 & 0 \\ \alpha & 1 & 0 \\ \beta & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \alpha & 1 & 0 \\ \beta & 0 & 1 \end{bmatrix}$$

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

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

$$L_{jk} = \begin{cases} 1 & \text{if } j=k \\ 0 & \text{if } j < k \\ \frac{a_{jk}}{a_{kk}} & \text{if } j > k \end{cases}$$

Example:

$$A = \begin{bmatrix} 2 & -1 & 3 \\ -4 & 6 & -5 \\ 6 & 13 & 16 \end{bmatrix} \xrightarrow{\begin{array}{l} \textcircled{2} - \frac{-4}{2} \textcircled{1} \\ \textcircled{3} - \frac{6}{2} \textcircled{1} \end{array}} \begin{bmatrix} 2 & -1 & 3 \\ 0 & 4 & 1 \\ 0 & 16 & 7 \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 3 & 4 & 1 \end{bmatrix} \xrightarrow{\textcircled{3} - \frac{16}{4} \textcircled{2}} \begin{bmatrix} 2 & -1 & 3 \\ 0 & 4 & 1 \\ 0 & 0 & 3 \end{bmatrix} = U$$

Check:

check:

$$\begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 3 & 4 & 1 \end{bmatrix} \begin{bmatrix} 2 & -1 & 3 \\ 0 & 4 & 1 \\ 0 & 0 & 3 \end{bmatrix} = \begin{bmatrix} & & \\ & -5 & \\ & 16 & \end{bmatrix}$$

Example:

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \text{Solution: } \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 10^{-4} & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

GE:

$$② \xrightarrow{\frac{1}{10} \rightarrow 0} \left[\begin{array}{cc|c} 10^{-4} & 1 & 1 \\ 0 & 1-10^{-4} & 2-10^{-4} \\ -10^{-4} & -10^{-4} & -10^{-4} \end{array} \right] \rightarrow -9999x^2 = -9998$$
$$x^2 = \frac{9998}{9999}$$

$$x_1 = \left(1 - \frac{9998}{9999}\right)10^4 = \frac{10000}{9999}$$

$$(\beta, t, L, U) = (10, 3, -9, 9)$$

$$\Rightarrow x_2 = 1$$

$$x_1 = (1-1)10^4 = 0$$

Stability of LU Factorization

In LU factorization, a problem arises when we have:

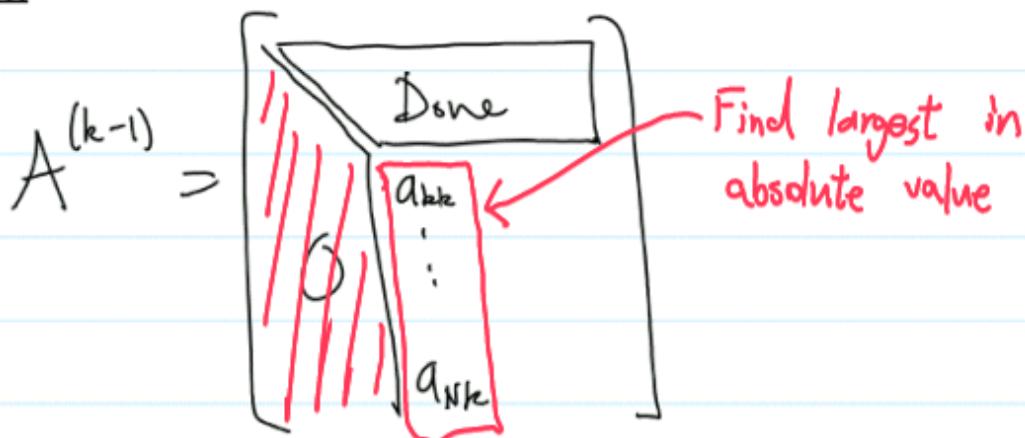
(1) a zero pivot i.e. $a_{kk}^{(k-1)} = 0$

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

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

\Rightarrow multipliers become large

\Rightarrow calculations become unstable

Pivoting

$$\text{Find } \max_{k \leq j \leq N} |a_{jk}^{(k-1)}| = |a_{k^*k}^{(k-1)}|$$

Swap rows k^* and k , and continue.

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

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

$\Rightarrow A$ is singular

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.

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

Let's put it all together.

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

\downarrow swap $\textcircled{1}$ & $\textcircled{2}$

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

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

\downarrow

$$\begin{bmatrix} 4 & 16 & 64 \\ 0 & -3 & -15 \\ 0 & -6 & -24 \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{4} & 1 & 0 \\ \frac{1}{2} & 0 & 1 \end{bmatrix}$$

$$\begin{array}{l} \textcircled{2} - \frac{1}{4}\textcircled{1} \\ \textcircled{3} - \frac{2}{4}\textcircled{1} \end{array}$$

Swap
② & ③

$$\begin{bmatrix} 4 & 16 & 64 \\ 0 & -6 & -24 \\ 0 & -3 & -15 \end{bmatrix}$$

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

$$L = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{4} & \frac{1}{2} & 1 \end{bmatrix} \quad \leftarrow$$

$$\begin{bmatrix} 4 & 16 & 64 \\ 0 & -6 & -24 \\ 0 & 0 & -3 \end{bmatrix} = U$$

$$\textcircled{3} - \frac{-3}{-6} \textcircled{2}$$

$$\therefore \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 16 & 64 \\ 0 & -6 & -24 \\ 0 & 0 & -3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 4 & 16 & 64 \\ 2 & 2 & 8 \end{bmatrix}$$

$$L \quad U \quad = \quad P \quad A$$

Example:

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



, []

swap
① & ②

$$\left[\begin{array}{cccc} -24 & 12 & 24 & -12 \\ -6 & 27 & -42 & -15 \\ -12 & 14 & 10 & 20 \\ -6 & -9 & 42 & 15 \end{array} \right]$$

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

$$\left[\begin{array}{cc|cc} -24 & 12 & 24 & -12 \\ \hline 1/4 & 24 & -48 & -12 \\ 1/2 & 8 & -22 & 26 \\ 1/4 & -12 & 36 & 18 \end{array} \right]$$

$\textcircled{3} - \frac{-12}{-24} \textcircled{1}$
 $\textcircled{4} - \frac{-6}{-24} \textcircled{1}$

$$\left[\begin{array}{cc|cc} -24 & 12 & 24 & -12 \\ \hline 1/4 & 24 & -48 & -12 \\ 1/2 & 1/3 & -6 & 30 \\ 1/4 & -1/2 & 12 & 12 \end{array} \right]$$

$\textcircled{3} - \frac{1}{24} \textcircled{2}$
 $\textcircled{4} - \frac{-12}{24} \textcircled{2}$

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

swap
③ & ④

$$\left[\begin{array}{cc|cc} -24 & 12 & 24 & -12 \\ \hline 1/4 & 24 & -48 & -12 \\ 1/4 & -1/2 & 12 & 12 \\ 1/2 & 1/3 & -6 & 30 \end{array} \right]$$

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

$$\textcircled{4} - \frac{-6}{12} \textcircled{3} \left[\begin{array}{ccccc} -24 & 12 & 24 & -12 \\ 1/4 & 24 & -48 & -12 \\ 1/4 & -1/2 & 12 & 12 \\ 1/2 & 1/3 & -1/2 & 36 \end{array} \right]$$

$$\therefore \left[\begin{array}{cccc} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array} \right] \left[\begin{array}{cccc} -6 & 27 & -42 & -15 \\ -24 & 12 & 24 & -12 \\ -12 & 14 & -10 & 20 \\ -6 & -9 & 42 & 15 \end{array} \right]$$

$$= \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 1/4 & 1 & 0 & 0 \\ 1/4 & -1/2 & 1 & 0 \\ 1/2 & 1/3 & -1/2 & 1 \end{array} \right] \left[\begin{array}{cccc} -24 & 12 & 24 & -12 \\ 0 & 24 & -48 & -12 \\ 0 & 0 & 12 & 12 \\ 0 & 0 & 0 & 36 \end{array} \right]$$

Singular Value Decomposition (SVD)

A geometric observation:

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

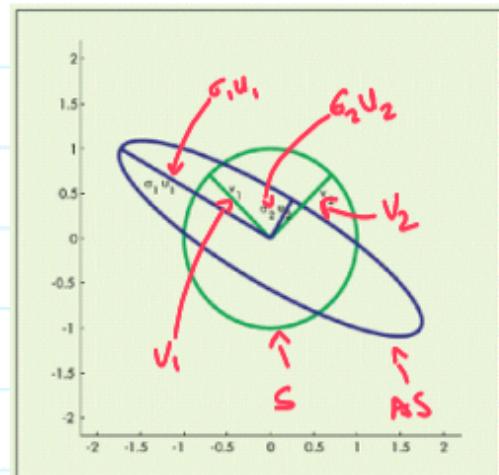
Let S be a unit sphere in \mathbb{R}^n .

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 , $\sigma_1, \sigma_2, \dots, \sigma_n$.

By convention,

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



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

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

The n right singular vectors are the unit vectors, orthogonal,

$$\{v_1, v_2, \dots, v_n\} \in S \text{ st. } Av_i = \sigma_i u_i$$

$$AV = U\Sigma \Rightarrow A = U\Sigma V^T$$

Σ is a diagonal matrix

Theorem:

Every matrix $A \in \mathbb{R}^{m \times n}$ has an SVD. The singular values are unique.

If A is square, and σ_i are distinct, then the left and right singular vectors are uniquely determined (modulo sign). $A = U\Sigma V^T$

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

$$\boxed{A} = \boxed{U} \boxed{\Sigma} \boxed{V^T}$$

$$\boxed{U} = \boxed{U}$$

$$\boxed{\Sigma} = \boxed{\Sigma}$$

$$\boxed{V^T} = \boxed{V^T}$$

$$Ax = b$$

$$U\Sigma V^T x = b$$

$$\Sigma V^T x = U^T b$$

$$V^T x = \Sigma^{-1} U^T b$$

$$x = V \Sigma^{-1} U^T b$$

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

$$\text{i.e. } U^T U = V^T V = I$$

If A is taller than it is wide

Full SVD

$$\begin{matrix} \boxed{} \\ A \\ m \times n \end{matrix} = \begin{matrix} \boxed{} \\ U \\ m \times m \end{matrix} \begin{matrix} \boxed{} \\ \Sigma \\ m \times n \end{matrix} \begin{matrix} \boxed{} \\ V^T \\ n \times n \end{matrix}$$

Reduced SVD

$$\begin{matrix} \boxed{} \\ A \\ m \times n \end{matrix} = \begin{matrix} \boxed{\phantom{\hat{U}}} \\ \hat{U} \\ n \times n \end{matrix} \begin{matrix} \boxed{\phantom{\hat{\Sigma}}} \\ \hat{\Sigma} \\ n \times n \end{matrix} \begin{matrix} \boxed{} \\ V^T \\ n \times n \end{matrix}$$

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 # of non-zero singular values
- $\text{null}(A)$ is the span of the right singular vectors (v_j) corresponding to zero singular values. $\rightarrow Ax=0$
 $x=A^{-1}0$
 x is non-zero solution
- $\text{cond}(A) = \sigma_1 / \sigma_n$
- computing norms: $\|A\|_2 = \sigma_1$ $\|A\|_F = \sqrt{\sigma_1^2 + \sigma_2^2 + \dots + \sigma_n^2}$
- eigenvalues: if $U\Sigma V^T = A^T A$, then

$$AV_j = \sqrt{\sigma_j} v_j$$
- low-rank approximation

Low-Rank Approximation using the SVD

Theorem:

For any integer k , $1 \leq k \leq \text{rank}(A)$, define

$$A_k = \sum_{j=1}^k \sigma_j u_j v_j^T = \boxed{\quad} \quad \boxed{\begin{matrix} * & * & * \\ * & * & * \\ * & * & * \end{matrix}} \quad \boxed{\quad}$$

only keep k singular values

Then,

$$\|A - A_k\|_2 = \inf_{\text{rank}(B) \leq k} \|A - B\|_2 = \sigma_{k+1}$$

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

$$A_k = U \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k & 0 & \dots & 0 \end{bmatrix} V^T$$

This result has profound implications in statistics and scientific computing.

Application: Principal Component Analysis

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.

e.g. 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 $\Sigma_2 = \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & 0 \end{bmatrix}$

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

Example: Image Compression

Instead of storing the whole image array as a matrix A , we can store

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 .

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

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 documents in a term-document matrix.

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

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

$$\text{term}^i \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{array}{l} T4: \text{Health} \\ T5: \text{Home} \\ T6: \text{Infant} \\ T7: \text{Proofing} \\ T8: \text{Safety} \\ T9: \text{Toddler} \end{array}$$

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

$$\text{query} = \text{"Child proofing"} \Rightarrow q = [0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0]^T$$

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

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

$$\underline{\langle d, q \rangle}$$

- 0 D1: Infants and Toddler First Aid ✗
- 0.41 D2: Babies & Children's Room (for your home) ✓
- 0.41 D3: Child Safety at Home ✓
- 0 D4: Your Baby's Health and Safety: From Infant to Toddler ✗
- 0.5 D5: Baby Proofing Basics ✓
- 0.5 D6: Your Guide to Easy Rust Proofing ✓
- 0 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 approximation to A .

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

$$\left. \begin{array}{l} U_2 = 1^{\text{st}} \text{ 2 cols of } U \\ \Sigma_2 = 2 \times 2 \text{ submatrix} \\ V_2 = 1^{\text{st}} \text{ 2 cols of } V \end{array} \right\} A_2 = U_2 \Sigma_2 V_2^T$$

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

$$A_2 = \begin{bmatrix} 0.1355 & 0.4652 & 0.2460 & 0.3882 & 0.5540 & 0.4072 & 0.5540 \\ 0.2266 & 0.2704 & 0.2698 & 0.3134 & 0.0809 & -0.0577 & 0.0809 \\ -0.1445 & 0.1191 & -0.0908 & -0.0071 & 0.4342 & 0.4612 & 0.4342 \\ 0.1044 & 0.1198 & 0.1230 & 0.1413 & 0.0292 & -0.0337 & 0.0292 \\ 0.2266 & 0.2704 & 0.2698 & 0.3134 & 0.0809 & -0.0577 & 0.0809 \\ 0.2482 & 0.2403 & 0.2799 & 0.3064 & -0.0045 & -0.1445 & -0.0045 \\ -0.1445 & 0.1191 & -0.0908 & -0.0071 & 0.4342 & 0.4612 & 0.4342 \\ 0.2326 & 0.2448 & 0.2678 & 0.3001 & 0.0284 & -0.1070 & 0.0284 \\ 0.2482 & 0.2403 & 0.2799 & 0.3064 & -0.0045 & -0.1445 & -0.0045 \end{bmatrix} \quad \begin{array}{l} T1: \text{Baby (babies, baby's)} \\ T2: \text{Child (children)} \\ T3: \text{Guide} \\ T4: \text{Health} \\ T5: \text{Home} \\ T6: \text{Infant} \\ T7: \text{Proofing} \\ T8: \text{Safety} \\ T9: \text{Toddler} \end{array}$$

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

$$A_2 = U_2 | Z_2 V_2^T \quad \boxed{\quad} = \boxed{\quad} \oplus \boxed{\quad}$$

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

$Z_2 V_2^T t$ gives the coordinates of a term frequency vector t in concept space.

