Matrix Approximations

Sanjiv Kumar, Google Research, NY

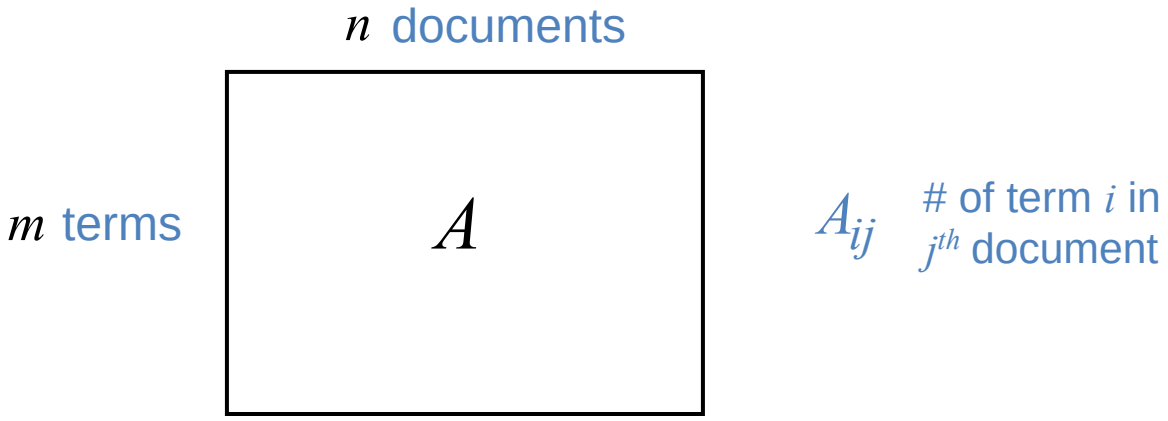
EECS-6898, Columbia University - Fall, 2010

Latent Semantic Indexing (LSI)

Given n documents with words from a vocabulary of size m, discover hidden topics and represent documents semantically.

Process

- form a term-document matrix



- top k left singular vectors represent topics
- transform input query in the k-dim semantic space
- match against the semantically transformed database items

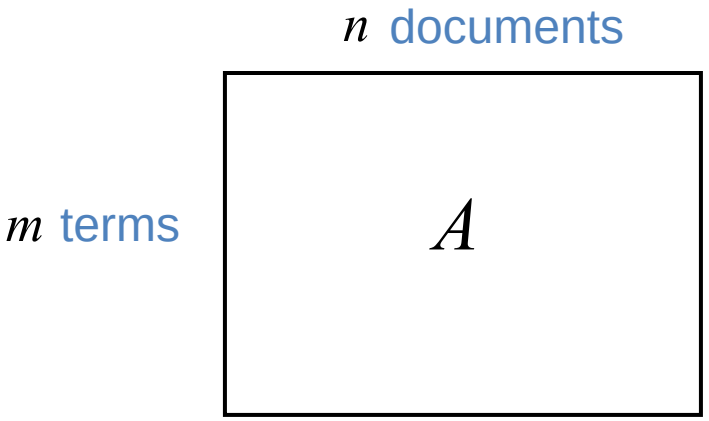
$n \sim O(B),\ m \sim O(100K)$ Large sparse/dense matrix

Latent Semantic Indexing (LSI)

Given n documents with words from a vocabulary of size m, discover hidden topics and represent documents semantically.

Document representation and retrieval

$$A \approx U_k \Sigma_k V_k^T$$



Latent Semantic Indexing (LSI)

Given n documents with words from a vocabulary of size m, discover hidden topics and represent documents semantically.

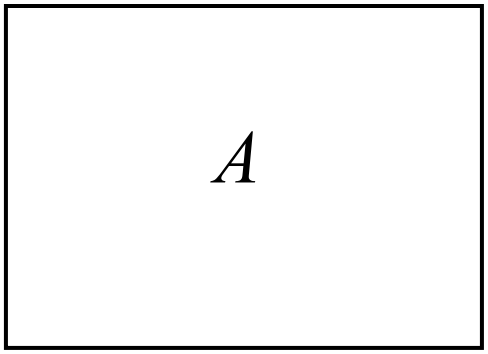
Document representation and retrieval

n documents

$$A \approx U_k \Sigma_k V_k^T$$

d_j j^{th} column $\hat{d}_j$

m terms



$$d_j \approx U_k \Sigma_k \hat{d}_j \implies \hat{d}_j \approx \Sigma_k^{-1} U_k^T d_j$$

Given a query q $\quad \hat{q} \approx \Sigma_k^{-1} U_k^T q$

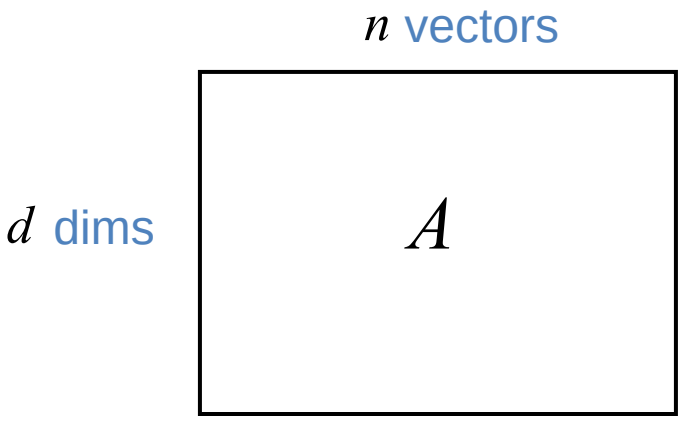
Find the nearest neighbors from database using $sim(\hat{q}, \hat{d}_j)$

Principal Component Analysis (PCA)

Given n vectors, find the linear reconstruction with minimum mean squared error.

Process

- form the centered data matrix (i.e. subtract mean from each vector)



- get the top k left singular vectors (equivalent to eigenvectors of covariance matrix)
- project the input data on the singular vectors

$n \sim O(B),\ d \sim O(100K)$ Large dense matrix

Kernel Methods

Commonly used for nonlinear extensions in classification, regression, dimensionality reduction etc.

Kernel Matrix

- Most kernel methods boil down to manipulation of kernel matrix

$$K = \begin{array}{c} \overset{n}{\boxed{k(x_i, x_j)}} \\ n \end{array}$$

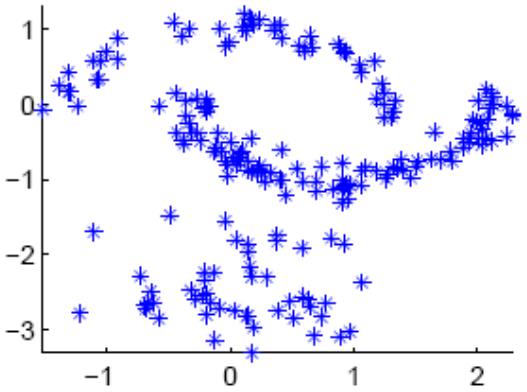
- Symmetric Positive Semi-Definite (SPSD) matrix of size $n \times n$
- Get top/bottom eigenvectors (eg., kernel PCA) or low-rank approximation (SVM)

$n \sim O(B)$ Very large dense matrix

Spectral Clustering

Graph-based clustering method

- Less assumptions on data distribution than parametric model e.g.,GMM

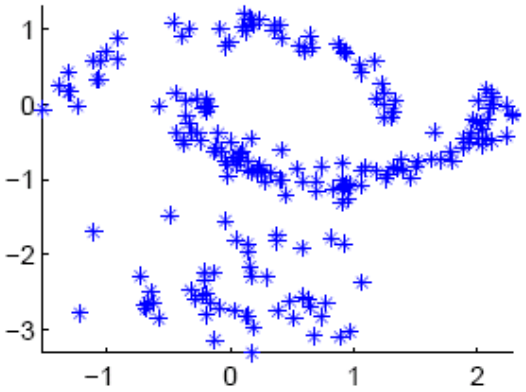


Luxburg [2]

Spectral Clustering

Graph-based clustering method

- Less assumptions on data distribution than parametric model e.g.,GMM



Key Idea: Find a (low) k-dim embedding such that neighbors remain close

$$\hat{Y} = Arg\min_{Y} \sum_{i,j} W_{ij} \left\| y_i - y_j \right\|_2^2 = Arg\min_{Y} Y^T L Y$$

$L = D - W$

Solution: Find bottom k eigenvectors of L (ignoring last)

assuming graph is connected

Spectral Clustering

Graph-based clustering method

- Less assumptions on data distribution than parametric model e.g.,GMM

Procedure:

- Compute weight matrix W:

$$W_{ij} = \exp(-\|x_i - x_j\|^2 / \sigma^2) \quad \text{or} \quad W_{ij} = \begin{cases} \exp(-\|x_i - x_j\|^2 / \sigma^2) & \text{if } i \sim j \\ 0 & \text{otherwise} \end{cases}$$

Dense — Sparse

- Compute normalized Laplacian

$$A = I - D^{-1/2} W D^{-1/2} \quad \text{or} \quad A = I - D^{-1} W$$

Symmetric — Asymmetric

$$D_{ii} = \sum_j W_{ij}$$

- Do a k-means clustering in optimal k reduced dims of A: $D^{1/2}U_k$ *or* U_k

Bottom eigenvectors of A, ignoring last

$n \sim O(B),\ d \sim O(100K)$ Large dense or sparse matrix

Two scenarios

Dense Matrices

1. Number of elements $O(n^2)$
2. Matrix-vector products $O(n^2)$
3. Spectrum usually falls exponentially for most real-world data
   - Most energy contained in top few eigenvalues
   - Low-rank approximation gives reasonable results
4. Commonly arise in kernel methods, regression, manifold learning, second order optimization (hessian)

Sparse Matrices

1. Number of elements $O(n)$
2. Very fast matrix-vector products $O(n)$
3. Spectrum is usually flatter (energy falls more slowly)
4. Commonly arise in spectral clustering, manifold learning, semi-supervised learning

Low-rank Approximation

Given a matrix A of rank $k < \min(m, n)$

$$\underset{m \times n}{A} = \underset{m \times k}{B} \; \underset{k \times n}{C}$$

Low-rank Approximation

Given a matrix A of rank $k < \min(m, n)$

$$\underset{m \times n}{A} = \underset{m \times k}{B} \; \underset{k \times n}{C}$$

Advantages

- Commonly used to discover structure in data i.e., Latent Semantic Indexing (LSI) e.g., discovering topics in documents or Recommendation system e.g., predicting user preferences
- Storage reduces to only $O((m+n)k)$ instead of $O(mn)$
- Matrix-vector product requires only $O((m+n)k)$ instead of $O(mn)$

Singular Value Decomposition (SVD)

$$A = U_A \sum_A V_A^T$$

$$[m \times n] \quad [m \times n] \quad [n \times n] \quad [n \times n]$$

Singular Value Decomposition (SVD)

$$A = U_A \Sigma_A V_A^T$$

$$[m \times n] \quad [m \times n] \quad [n \times n] \quad [n \times n]$$

wlog suppose $m > n$

Left singular vectors

$$U_A^T U_A = I$$

Eigenvectors of AA^T

Singular Value Decomposition (SVD)

$$A = U_A \Sigma_A V_A^T$$

$$[m \times n] \quad [m \times n] \quad [n \times n] \quad [n \times n]$$

wlog suppose $m > n$

Left singular vectors

$$U_A^T U_A = I$$

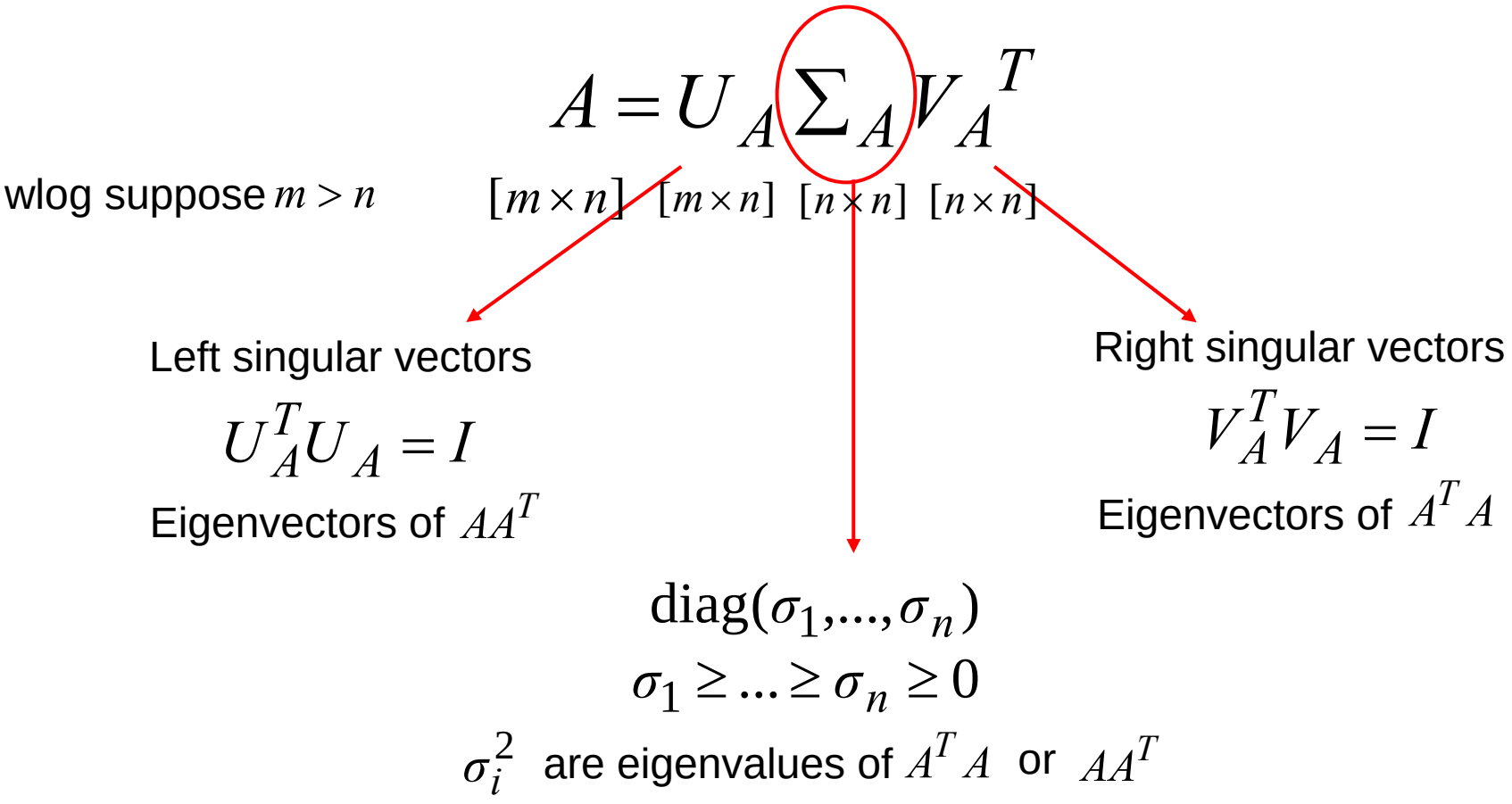
Eigenvectors of AA^T

Right singular vectors

$$V_A^T V_A = I$$

Eigenvectors of $A^T A$

Singular Value Decomposition (SVD)



Low-rank Approximation

Frobenius Norm $\|A\|_F^2 = \sum_{i,j} A_{ij}^2 = Tr(A^T A) = Tr(AA^T) = \sum_i \sigma_i^2$

Spectral Norm $\|A\|_2 = \max_{x \neq 0} \|Ax\|_2 / \|x\|_2 = \sigma_1$ $\qquad \|A\|_2 \leq \|A\|_F \leq \sqrt{n}\|A\|_2$

Low-rank Approximation

Frobenius Norm $\|A\|_F^2 = \sum_{i,j} A_{ij}^2 = Tr(A^T A) = Tr(AA^T) = \sum_i \sigma_i^2$

Spectral Norm $\|A\|_2 = \max_{x \neq 0} \|Ax\|_2 / \|x\|_2 = \sigma_1$ $\qquad \|A\|_2 \leq \|A\|_F \leq \sqrt{n}\|A\|_2$

$$A_k = \underset{D}{Arg\min} \|A - D\|_{F,2} \quad rank(D) = k$$

$$\underset{m \times n}{A} \approx \underset{m \times k}{A_k} = \underset{m \times k}{U_k} \; \underset{k \times k}{\Sigma_k} \; \underset{k \times n}{V_k^T}$$

Low-rank Approximation

Frobenius Norm $\|A\|_F^2 = \sum_{i,j} A_{ij}^2 = Tr(A^T A) = Tr(AA^T) = \sum_i \sigma_i^2$

Spectral Norm $\|A\|_2 = \max_{x \neq 0} \|Ax\|_2 / \|x\|_2 = \sigma_1$ $\qquad \|A\|_2 \leq \|A\|_F \leq \sqrt{n}\|A\|_2$

$$A_k = \underset{D}{Arg\min} \|A - D\|_{F,2} \quad rank(D) = k$$

$$\underset{m \times n}{A} \approx \underset{m \times k}{A_k} = \underset{m \times k}{U_k} \underset{k \times k}{\Sigma_k} \underset{k \times n}{V_k^T}$$

Removing small entries in Σ has noise-removal interpretation e.g., PCA

Computational Complexity

full SVD $\qquad O(mn \min\{m,n\})$

rank-k SVD $\qquad O(mnk)$

Matrix Projection

$$A_k = U_k \sum_k V_k^T = U_k U_k^T A = A V_k V_k^T$$

Matrix Projection

$$A_k = U_k \sum_k V_k^T = \underbrace{U_k U_k^T A}_{\text{Projection of A on space spanned by columns of } U_k} = A V_k V_k^T$$

Matrix Projection

$$A_k = U_k \Sigma_k V_k^T = \underbrace{U_k U_k^T A} = A V_k V_k^T$$

Projection of A on space spanned by columns of U_k

$$\tilde{A}_k = \tilde{U}_k \tilde{\Sigma}_k \tilde{V}_k^T \neq \tilde{U}_k \tilde{U}_k^T A \neq A \tilde{V}_k \tilde{V}_k^T$$

For approximate decomposition, low-rank approximation using SVD and matrix projection give different results !

Power Method

To compute the largest eigenvalue/eigenvector of a matrix

Assume A is a symmetric matrix, i.e. it is diagonalizable with,

$$U^{-1}AU = diag(\lambda_1, \lambda_2, ..., \lambda_n)$$

$$|\lambda_1| \geq |\lambda_2| \geq ... \geq |\lambda_n|$$

Power Method

To compute the largest eigenvalue/eigenvector of a matrix

Assume A is a symmetric matrix, i.e. it is diagonalizable with,

$$U^{-1}AU = diag(\lambda_1, \lambda_2, ..., \lambda_n)$$

$$|\lambda_1| \geq |\lambda_2| \geq ... \geq |\lambda_n|$$

Power iteration

1. Start with random vector $q_0 \in \Re^n$
2. for k = 1, 2, …, K

$$q_k = Aq_{k-1}$$ repeated matrix-vector product

3. Normalize and compute eigenvalue

$$z_K = q_K / \|q_K\|_2$$

$$\lambda_K = z_K^T A z_K$$

Convergence properties

Suppose, $q_0 = a_1 u_1 + a_2 u_2 + \ldots + a_n u_n$ Since columns of U make a basis in R^n

$$A^k q_0 = a_1 \lambda_1^k u_1 + a_2 \lambda_2^k u_2 + \ldots + a_n \lambda_n^k u_n$$

Since $Au_i = \lambda u_i \Rightarrow A^k u_i = \lambda^k u_i$

Convergence properties

Suppose, $q_0 = a_1 u_1 + a_2 u_2 + \ldots + a_n u_n$ Since columns of U make a basis in R^n

$$A^k q_0 = a_1 \lambda_1^k u_1 + a_2 \lambda_2^k u_2 + \ldots + a_n \lambda_n^k u_n$$

Since $Au_i = \lambda u_i \Rightarrow A^k u_i = \lambda^k u_i$

if $a_1 \neq 0$

$$A^k q_0 = a_1 \lambda_1^k \left(u_1 + \sum_{j=2}^{n} \frac{a_j}{a_1} \left(\frac{\lambda_j}{\lambda_1} \right)^k u_j \right)$$

$$dist(q_k, u_1) = O(|\lambda_2 / \lambda_1|^k)$$

Convergence properties

Suppose, $q_0 = a_1 u_1 + a_2 u_2 + \ldots + a_n u_n$ Since columns of U make a basis in R^n

$$A^k q_0 = a_1 \lambda_1^k u_1 + a_2 \lambda_2^k u_2 + \ldots + a_n \lambda_n^k u_n$$

Since $Au_i = \lambda u_i \Rightarrow A^k u_i = \lambda^k u_i$

if $a_1 \neq 0$

$$A^k q_0 = a_1 \lambda_1^k \left(u_1 + \sum_{j=2}^{n} \frac{a_j}{a_1} \left(\frac{\lambda_j}{\lambda_1} \right)^k u_j \right)$$

$$dist(q_k, u_1) = O(|\lambda_2 / \lambda_1|^k)$$

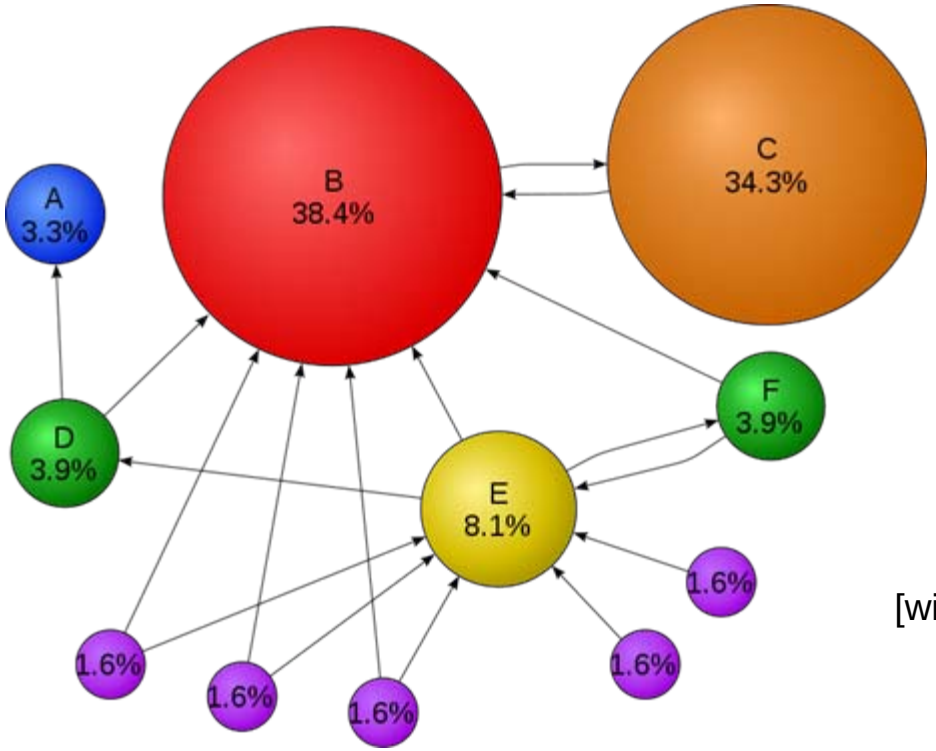
easy to satisfy by random initialization

bigger gap leads to faster convergence

Can be generalized to multiple eigenvectors simultaneously by Orthogonal Iteration via QR-decomposition !

Page Rank

Which document or link is more “popular” independent of the query?



[wikipedia]

Random walk over pages

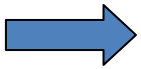
If one randomly clicks the links then what is the probability that one will be at page B?

Page Rank

Stationary probability distribution over pages

Adjacency matrix $[A]_{n\times n}$

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



Transition matrix $[T]_{n\times n}$

$$T_{ij} = \begin{cases} p(j \mid i) & \text{if } i \to j \\ 0 & \text{otherwise} \end{cases}$$

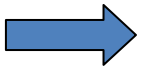
Transition matrix is stochastic, i.e., rows sum to 1

Page Rank

Stationary probability distribution over pages

Adjacency matrix $[A]_{n\times n}$

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



Transition matrix $[T]_{n\times n}$

$$T_{ij} = \begin{cases} p(j \mid i) & \text{if } i \to j \\ 0 & \text{otherwise} \end{cases}$$

Transition matrix is stochastic, i.e., rows sum to 1

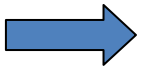
Stationary Probability distribution given by the eigenvector corresponding to largest eigenvalue (i.e. 1) of T $\boxed{u = T^T u}$

Page Rank

Stationary probability distribution over pages

Adjacency matrix $[A]_{n\times n}$

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



Transition matrix $[T]_{n\times n}$

$$T_{ij} = \begin{cases} p(j \mid i) & \text{if } i \to j \\ 0 & \text{otherwise} \end{cases}$$

Transition matrix is stochastic, i.e., rows sum to 1

Stationary Probability distribution given by the eigenvector corresponding to largest eigenvalue (i.e. 1) of T $u = T^T u$

Power Iteration $u_k = \left(T^T\right)^k u_0$

Damping is used to make sure graph connected

$$u_{k+1} = \alpha T^T u_k + (1-\alpha)(1/n)\,1$$

1 → n-vector of 1's

Lanczos Method

A technique to solve large sparse symmetric eigenproblems

- Useful when only top or bottom few eigenvalues/vectors are needed
- Generates a sequence of tridiagonal matrices whose extremal eigenvalues progressively better estimates of desired values

Lanczos Method

A technique to solve large sparse symmetric eigenproblems

- Useful when only top or bottom few eigenvalues/vectors are needed
- Generates a sequence of tridiagonal matrices whose extremal eigenvalues progressively better estimates of desired values

Krylov Subspaces

$$\mathrm{K}(A, q_1, k) = span\{q_1, Aq_1, ..., A^{k-1}q_1\}$$

Lanczos Method

A technique to solve large sparse symmetric eigenproblems

- Useful when only top or bottom few eigenvalues/vectors are needed
- Generates a sequence of tridiagonal matrices whose extremal eigenvalues progressively better estimates of desired values

Krylov Subspaces

$$\mathrm{K}(A, q_1, k) = span\{q_1, Aq_1, ..., A^{k-1}q_1\}$$

Key Idea

- Successively generate a new orthonormal vector such that

$$span\{q_1, q_2, ..., q_{k+1}\} = span\{q_1, Aq_1, ..., A^{k}q_1\}$$

How to find $\{q_1, q_2, ..., q_{k+1}\}$

Lanczos Method

Why Tridiagonalization

For any symmetric matrix A: $Q^T A Q = T$

Where Q is n x n orthogonal matrix and T is a tridiagonal matrix

$$T = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \beta_3 & \\ \vdots & & & & \\ \vdots & & & & \\ & & & \beta_{n-1} & \alpha_n \end{bmatrix}$$

Lanczos Method

Why Tridiagonalization

For any symmetric matrix A: $Q^T A Q = T$

Where Q is n x n orthogonal matrix and T is a tridiagonal matrix

$$AQ = QT$$

let $Q = [q_1\ q_2\ \dots\ q_n]$

$$q_1 = Qe_1$$

$$e_k = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \leftarrow k^{th}$$

$$T = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \beta_3 & \\ \vdots & & & & \\ \vdots & & & & \\ & & & \beta_{n-1} & \alpha_n \end{bmatrix}$$

Lanczos Method

Why Tridiagonalization

For any symmetric matrix A: $Q^T A Q = T$

Where Q is n x n orthogonal matrix and T is a tridiagonal matrix

$$AQ = QT$$

let $Q = [q_1\ q_2\ \dots\ q_n]$

$$q_1 = Qe_1$$

$$e_k = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \leftarrow k^{th}$$

$$T = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \beta_3 & \\ \vdots & & & & \\ \vdots & & & & \\ & & & \beta_{n-1} & \alpha_n \end{bmatrix}$$

Krylov Matrix $K_m(A, q_1, n) = [q_1\ Aq_1\ \dots\ A^{n-1}q_1]$

$$= Q\underbrace{[e_1\ Te_1\ T^2e_1 \dots T^{n-1}e_1]}_{R}$$

Columns of Q form orthogonal bases for Krylov Matrix !

Lanczos Method

Construct partial Q and partial T iteratively

$$AQ = QT$$

$$T = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \beta_3 & \\ \vdots & & & & \\ \vdots & & & & \\ & & & \beta_{n-1} & \alpha_n \end{bmatrix}$$

Let's focus on k^{th} column of Q

$$Aq_k = \beta_{k-1}q_{k-1} + \alpha_k q_k + \beta_k q_{k+1}$$

$$\beta_k q_{k+1} = (A - \alpha_k I)q_k - \beta_{k-1}q_{k-1} \equiv r_k$$

$$\text{if } r_k \neq 0 \quad q_{k+1} = r_k / \beta_k \quad \text{where} \quad \beta_k = \|r_k\|_2$$

Lanczos Method

Construct partial Q and partial T iteratively

$$AQ = QT$$

$$T = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \beta_3 & \\ \vdots & & & & \\ \vdots & & & & \\ & & & \beta_{n-1} & \alpha_n \end{bmatrix}$$

Let's focus on k^{th} column of Q

$$Aq_k = \beta_{k-1}q_{k-1} + \alpha_k q_k + \beta_k q_{k+1}$$

$$\beta_k q_{k+1} = (A - \alpha_k I)q_k - \beta_{k-1}q_{k-1} \equiv r_k$$

$$\text{if } r_k \neq 0 \quad q_{k+1} = r_k / \beta_k \quad \text{where} \quad \beta_k = \|r_k\|_2$$

premultiply by q_{k+1}^T

using orthormality of q's

$r_0 = q_1, \beta_0 = 1, k = 1, \ldots$

while $(\beta_k \neq 0)$

$q_k = r_{k-1} / \beta_{k-1}$

$\alpha_k = q_k^T A q_k$

$r_k = (A - \alpha_k I)q_k - \beta_{k-1}q_{k-1}$

$\beta_k = \|r_k\|_2$

Need matrix-vector product Aq_k

Lanczos Method

When to stop?

- Ideally until k = rank of Krylov matrix
- Usually too long, so a threshold is used on residual

$$AQ_k = Q_k T_k + r_k e_k^T$$

- Decompose T_k $\quad S_k^T T_k S_k = diag(\theta_1,...,\theta_k)$

Lanczos Method

When to stop?

- Ideally until k = rank of Krylov matrix
- Usually too long, so a threshold is used on residual

$$AQ_k = Q_k T_k + r_k e_k^T$$

- Decompose T_k $\quad S_k^T T_k S_k = diag(\underbrace{\theta_1,...,\theta_k}_{O(k)})$

estimated eigenvectors of A: $Q_k S_k$ $\qquad$ estimated eigenvalues of A

Lanczos Method

When to stop?

- Ideally until k = rank of Krylov matrix
- Usually too long, so a threshold is used on residual

$$AQ_k = Q_k T_k + r_k e_k^T$$

- Decompose T_k $\quad S_k^T T_k S_k = diag(\underbrace{\theta_1,...,\theta_k}_{\text{estimated eigenvalues of } A})$

estimated eigenvectors of A: $Q_k S_k$

Accuracy $\quad \min_{\mu \in \lambda(A)} |\theta_i - \mu| \le |\beta_k \| s_{ki}|$

Convergence

Kaniel-Page Theory

$$\lambda_1 \ge \theta_1 \ge \lambda_1 - (\lambda_1 - \lambda_n)\tan(\varphi_1)^2 / (c_{k-1}(1+2\rho_1))^2$$

$$\cos(\varphi_1) = |q_1^T u_1|$$

u_1: top eigenvector of A

Chebyshev polynomial

$$\rho = (\lambda_1 - \lambda_2)/(\lambda_1 - \lambda_n)$$

Lanczos Method

When to stop?

- Ideally until k = rank of Krylov matrix
- Usually too long, so a threshold is used on residual

$$AQ_k = Q_k T_k + r_k e_k^T$$

- Decompose T_k $\quad S_k^T T_k S_k = diag(\underbrace{\theta_1,...,\theta_k}_{\text{estimated eigenvalues of } A})$

estimated eigenvectors of A: $Q_k S_k$

Accuracy $\quad \min_{\mu \in \lambda(A)} |\theta_i - \mu| \le |\beta_k \| s_{ki}|$

Convergence

Kaniel-Page Theory

$$\lambda_1 \ge \theta_1 \ge \lambda_1 - (\lambda_1 - \lambda_n)\tan(\varphi_1)^2 / (c_{k-1}(1+2\rho_1))^2$$

Chebyshev polynomial

$$\cos(\varphi_1) = |q_1^T u_1|$$

top eigenvector of A

Lanczos iterations converge faster than power iteration!

Practical Implementation: Avoid rounding errors

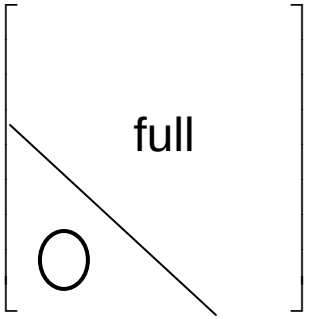
Arnoldi's Method

A technique to solve large sparse asymmetric eigenproblems

- Generates a sequence of Hessenberg matrices with extremal eigenvalues progressively better estimates

$$Q^T A Q = H \Rightarrow AQ = QH$$

$$Aq_k = \sum_{i=1}^{k+1} h_{ik} q_i \Rightarrow r_k = Aq_k - \sum_{i=1}^{k} h_{ik} q_i$$

$H =$ 

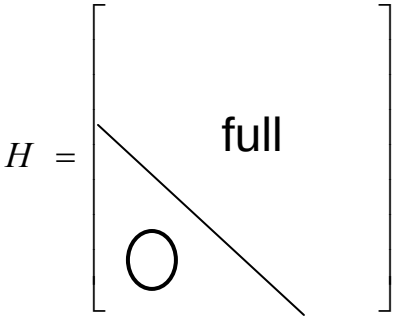
Arnoldi's Method

A technique to solve large sparse asymmetric eigenproblems

- Generates a sequence of Hessenberg matrices with extremal eigenvalues progressively better estimates

$$Q^T A Q = H \quad \Rightarrow \quad AQ = QH$$

$$Aq_k = \sum_{i=1}^{k+1} h_{ik} q_i \quad \Rightarrow \quad r_k = Aq_k - \sum_{i=1}^{k} h_{ik} q_i$$



$$\text{if } h_{k+1,k} \neq 0 \qquad q_{k+1} = r_k / h_{k+1,k} \qquad h_{k+1,k} = \|r_k\|_2$$

After k steps

$$AQ_k = Q_k H_k + r_k e_k^T$$

$$H_k y = \lambda y \longrightarrow O(k^2)$$

$$(A - \lambda I) Q_k y = (e_k^T y) r_k$$

ritz value

ritz vector

In practice, applied with multiple restarts!

Randomized SVD and low-rank approx

Basic Problem

oversampling

Given an $m \times n$ matrix A, and an integer $l = k + p$, find an $m \times l$ orthonormal matrix Q such that

$$A \approx Q\, Q^T A$$

$m \times n \quad m \times l$

$l << m, n$

The columns of Q form orthonormal basis for the range of A

Randomized SVD and low-rank approx

Basic Problem

oversampling

Given an $m \times n$ matrix A, and an integer $l = k + p$, find an $m \times l$ orthonormal matrix Q such that

$$A \approx Q\,Q^T A \qquad l << m, n$$

$m \times n \quad m \times l$

The columns of Q form orthonormal basis for the range of A

How to do SVD?

1. Form a small $(l \times n)$ matrix: $B = Q^T A$ $\quad O(mnl)$
2. Compute SVD of B: $B = \hat{U}\,\Sigma\,V^T$ $\quad O(nl^2)$
3. Set $U = Q\hat{U}$ $\quad O(ml^2)$

Of course, the best Q is: $Q = U_l$ Expensive!

How to build approximate Q ?

Randomized method

Basic Problem

Given an $m \times n$ matrix A, and an integer l, find an $m \times l$ orthonormal matrix Q such that $A \approx Q\,Q^T A$

Procedure

1. Draw random vectors from $P(w)$ $\quad w_1, w_2, ..., w_l \in \Re^n$

2. Form projected sample vectors $\quad y_1 = Aw_1, ..., y_l = Aw_l \in \Re^m$

3. Form orthonormal vectors $\quad q_1, q_2, ...q_l \in \Re^m$

$$\text{Span}(q_1, q_2, ..., q_l) = \text{Span}(y_1, y_2, ..., y_l)$$ Gram-Schmidt or QR

Randomized method

Basic Problem

Given an $m \times n$ matrix A, and an integer l, find an $m \times l$ orthonormal matrix Q such that $A \approx Q\,Q^T A$

Matrix-version

1. Construct a random matrix Ω_l of size $n \times l$ $O(nl)$
2. Form $m \times l$ sample matrix $Y_l = A\,\Omega_l$ $O(mnl)$ most expensive! Use randomized FFT for $O(mn\log(l))$
3. Form an $m \times l$ orthonormal matrix Q_l such that $Y_l = Q_l Q_l^T Y_l$ $O(ml^2)$

Randomized method

Basic Problem

Given an $m \times n$ matrix A, and an integer l, find an $m \times l$ orthonormal matrix Q such that $A \approx Q\,Q^T A$

Matrix-version

1. Construct a random matrix Ω_l of size $n \times l$ $O(nl)$
2. Form $m \times l$ sample matrix $Y_l = A\,\Omega_l$ $O(mnl)$ most expensive! Use randomized FFT for $O(mn\log(l))$
3. Form an $m \times l$ orthonormal matrix Q_l such that $Y_l = Q_l Q_l^T Y_l$ $O(ml^2)$

Error in Approximation $\left\| A - Q\,Q^T A \right\|_2 \geq \sigma_{l+1}$

$$\left\| A - Q_l\,Q_l^T A \right\|_2 \leq [1 + 11\sqrt{k+p}.\sqrt{\min(m,n)}]\sigma_{k+1}$$ with probability $1 - 6p^{-p}$

Matrices with slowly decaying spectrum

Basic Idea

- Incorporate Krylov-space type power iteration

$$B = (AA^T)^q A \Longrightarrow$$

same singular vectors as of A but fast decaying singular values

Matrices with slowly decaying spectrum

Basic Idea

- Incorporate Krylov-space type power iteration

$$B = (AA^T)^q A \Rightarrow$$ same singular vectors as of A but fast decaying singular values

Algorithm

1. Construct a random matrix Ω_l of size $n \times l$
2. Form $m \times l$ sample matrix $Z = (A\ A^T)^q A\ \Omega_l$
3. Form an $m \times l$ orthonormal matrix Q_l from Z using partial QR

Matrices with slowly decaying spectrum

Basic Idea

- Incorporate Krylov-space type power iteration

$$B = (AA^T)^q A \Rightarrow$$ same singular vectors as of A but fast decaying singular values

Algorithm

1. Construct a random matrix Ω_l of size $n \times l$
2. Form $m \times l$ sample matrix $Z = (A\ A^T)^q A\ \Omega_l$
3. Form an $m \times l$ orthonormal matrix Q_l from Z using partial QR

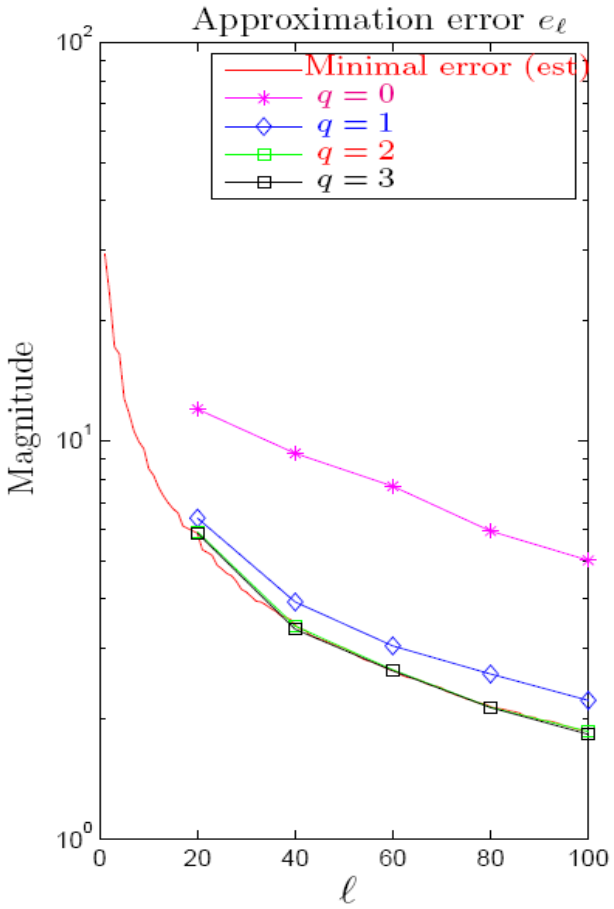
Error in Approximation

$$\left\| A - Q_l\ Q_l^T A \right\|_2 \leq [1 + 11\sqrt{k+p}.\sqrt{\min(m,n)}]^{1/(2q+1)} \sigma_{k+1}$$ with probability $1 - 6p^{-p}$

Randomized method: Example

Eigenfaces

$m = 7254 \;\; faces \quad n = 384 \times 256 \;\; pixels$



Randomized method

Advantages

1. Much faster than standard methods: $O(mn \log k)$ instead of $O(mnk)$
2. Need to look at data only very few times (sometimes single pass!)
3. Easy parallel implementation
4. The error in approximation can be made arbitrarily small with very high probability (even 10^{-10}) inexpensively
5. Simple to code

Randomized Vs Sparse Methods

Randomized methods

Build the approximate range of A by multiplying A with a new random vector progressively

$$y_1 = Aw_1, \ldots, y_l = Aw_l$$

Sparse Methods

Build the approximate range of A by multiplying A recursively with the outcome of the previous stage starting with a random vector

$$y_1 = Aw_1, \ldots, y_l = Ay_{l-1}$$ if A is square

Advantages of randomized method

1. Similar computational complexity
2. Can be parallelized
3. Provide better approximations even for small spectrum gaps
4. Can learn from very few (sometimes just one) passes over the data

References

1. Deerwester, S., et al, Improving Information Retrieval with Latent Semantic Indexing, Proceedings of the 51st Annual Meeting of the American Society for Information Science 25, 1988 .
2. Ulrike Von Luxburg, A tutorial on spectral clustering , Tech Report, TR-149, Max-plank Institute.
3. Golub and Van Loan, Matrix Computations, Johns Hopkins Press, 1996.
4. Sergey Brin, Larry Page (1998). "The Anatomy of a Large-Scale Hypertextual Web Search Engine". *Proceedings of the 7th international conference on World Wide Web (WWW)*. Brisbane, Australia. pp. 107–117.
5. A. Frieze, R. Kannan and S. Vempala, Fast Monte-Carlo Algorithms for finding low-rank approximations, Proceedings of the Foundations of Computer Science, 1998.
6. P. Drineas and R. Kannan, M. Mahoney, Fast Monte Carlo Algorithms for Matrices I: Approximating Matrix Multiplication, Tech-Report TR-1269.
7. Halko, Martinsson, & Tropp, "Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions“, ACM Report, 2009.