CS 2750

Applications of PCA and SVD

Presentation by Daniel Steinberg

Some Slides are New but Most are From or Have been Adapted From:
Fall 2011 Slides by Sherry Sahebi,
Fall 2007 Slides by Cem Akkaya,
And Slides by Iyad Batal.
Also Based on Bishop PRML, Assigned Readings, Assorted Websites, and Wikipedia.

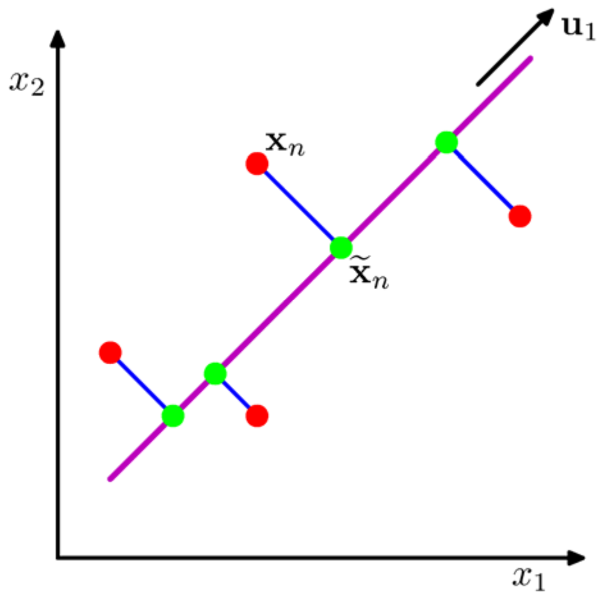
Outline

- PCA Tips, Tricks, Other
- PCA for Dimensionality Reduction
- PCA for Standardizing/Whitening/Sphering
- SVD for PCA
- Latent Semantic Indexing (LSI)
- PCA for Image Compression
- PCA for Facial Recognition
- Kleinberg's Algorithm (HITS)
- PageRank Algorithm
- Additional Applications

PCA Tips, Tricks, Other

- Center each variable in the data around zero (required for some of the formulas)
- Other Unsupervised Methods of Dimensionality Reduction
  - Independent Component Analysis (ICA)
  - Canonical Correlation Analysis (CCA)
- Last class formulated the PCA solution as that which maximized the variance of the projected data.
  - An alternative formulation is to find a solution which minimizes the sum of the squared projection errors.
    - Under the assumptions we've been making, both formulations return the same solution.

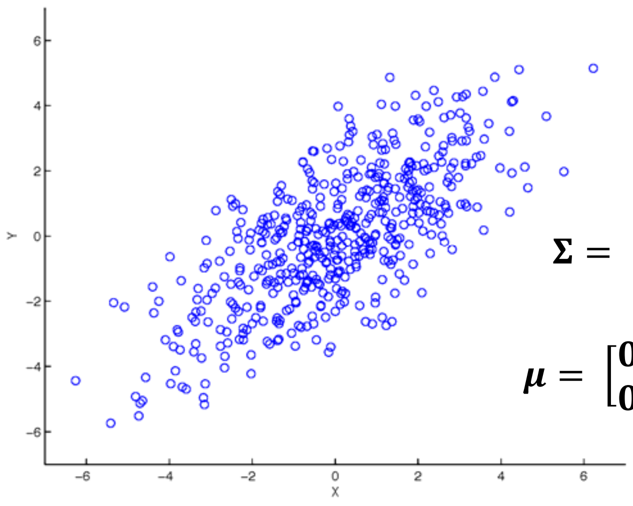
Minimize The Sum-of-Squared Projection Errors



PCA for Dimensionality Reduction

- $\mathbf{X} \in \mathbb{R}^{n \times d}$
- Get eigenvalues and eigenvectors of $\mathbf{X}$'s covariance matrix $\mathbf{\Sigma}$
- Create a change-of-basis matrix $\mathbf{M}$, where the columns are the eigenvectors of $\mathbf{\Sigma}$, sorted in descending order by their corresponding eigenvalues.

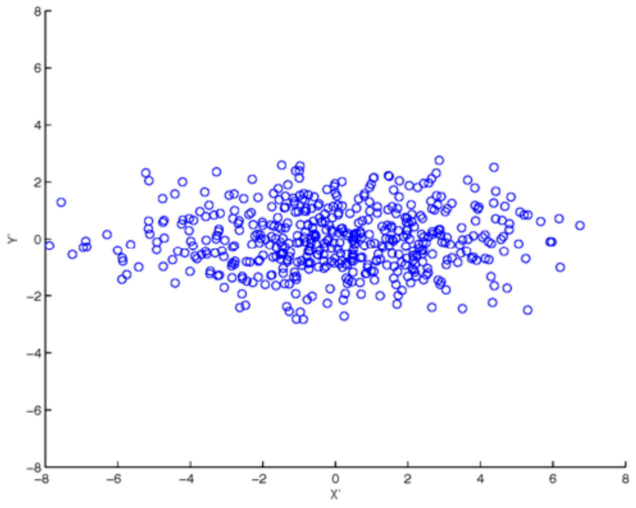
Multivariate Normal $\mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$



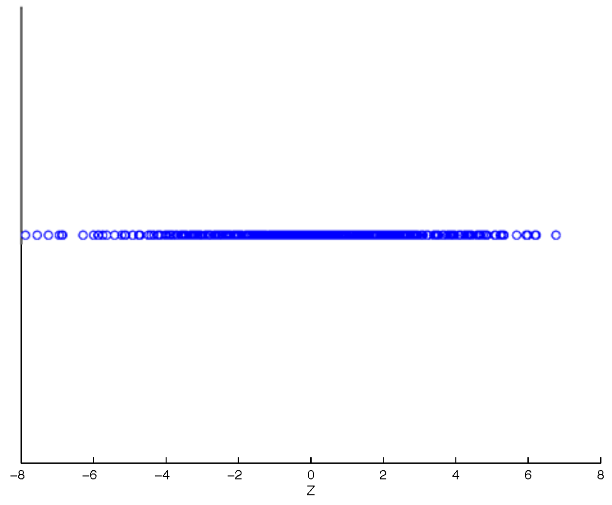
$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{4} & \mathbf{3} \\ \mathbf{3} & \mathbf{4} \end{bmatrix}$$

$$\boldsymbol{\mu} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

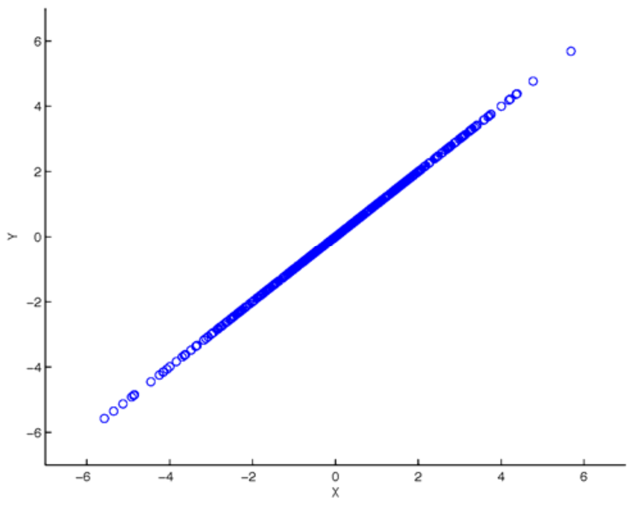
Projected Onto 2 Principal Components



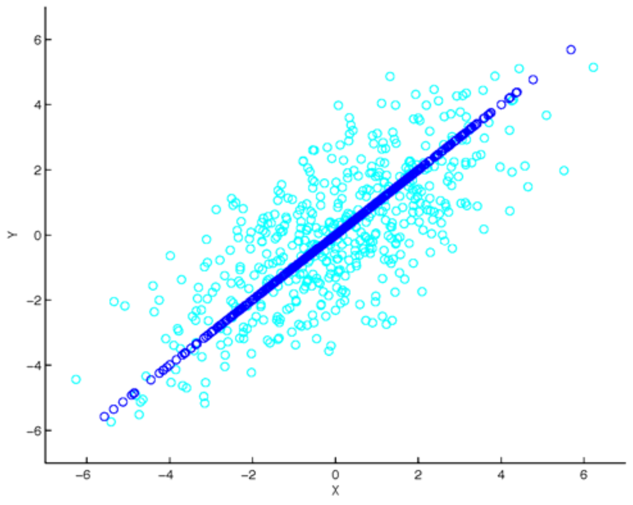
Projected Onto a Single Dimension Z



Projected Data *Restored* to 2 Dimensions



Original Versus Restored



PCA for Standardizing/Whitening/Sphering

- Whitening (aka Sphering) is a way of standardizing data after it has been transformed with PCA.
- Suppose a d-dimensional dataset is projected onto d principal components.
  - Even if the original data before PCA was standardized (unit variance), the variance of the projected data will be the eigenvalues of the original covariance matrix.
- Whitening standardizes the projected data by dividing feature i by $\sqrt{\lambda_i}$
- Now the projected data are uncorrelated and have unit variance (standardized)
- Beware of numerical instability from possibly very small eigenvalues (will result in dividing by a number very close to zero).

Singular Value Decomposition (SVD)

- SVD decomposes $\mathbf{A}$ into the product of three matrices
- $\mathbf{U}$ and $\mathbf{V}$ have orthonormal columns. (they are unit vectors and orthogonal to each other)
- $\mathbf{\Sigma}$ is a diagonal matrix containing singular values of $\mathbf{A}$ in descending order. The number of non-zero singular values gives the rank of $\mathbf{A}$.
- Columns of $\mathbf{U}$ are the orthogonal eigenvectors of $\mathbf{A}\mathbf{A}^\top$
- Columns of $\mathbf{V}$ are the orthogonal eigenvectors of $\mathbf{A}^\top\mathbf{A}$

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$$

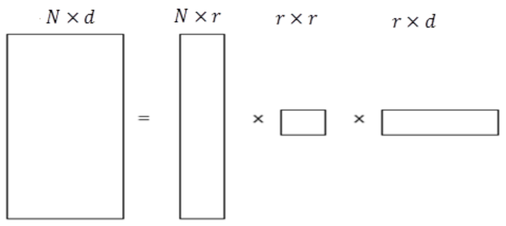
$\mathbf{A} \in \mathbb{R}^{n\times d}$ $\quad$ $\mathbf{\Sigma} \in \mathbb{R}^{r\times r}$

$\mathbf{U} \in \mathbb{R}^{n\times r}$ $\quad$ $\mathbf{V} \in \mathbb{R}^{d\times r}$

SVD

- Any $N \times d$ matrix $\mathbf{A}$ can be uniquely expressed as:

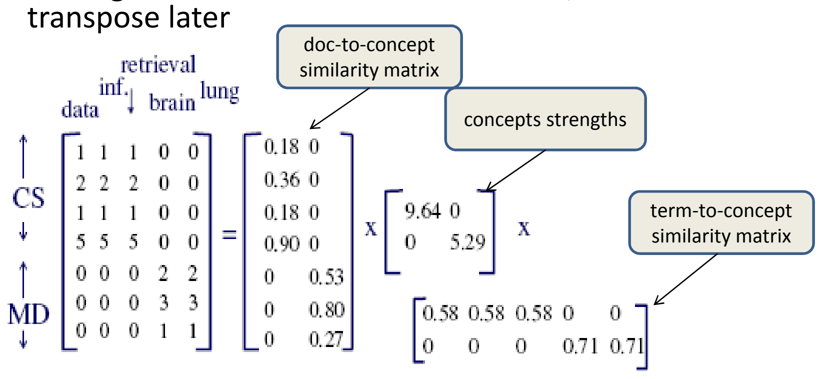
$$\mathbf{A} = \mathbf{U\Sigma V}^{\mathsf{T}}$$



- r is the rank of the matrix $\mathbf{A}$ (# of linearly independent columns/rows)
- $\mathbf{U}$ is a column-orthonormal $N \times r$ matrix.
- $\mathbf{\Sigma}$ is a diagonal $r \times r$ matrix where the singular values σ_i are sorted in descending order.
- $\mathbf{V}$ is a column-orthonormal $d \times r$ matrix.

SVD Example

- Covered in Detail Soon
- Don't get too used to matrix orientation, since we transpose later


doc-to-concept similarity matrix
concepts strengths
term-to-concept similarity matrix

$$\begin{array}{c|ccccc} & \text{data} & \text{inf.} & \text{retrieval} & \text{brain} & \text{lung} \\ \hline \text{CS} & 1 & 1 & 1 & 0 & 0 \\ & 2 & 2 & 2 & 0 & 0 \\ & 1 & 1 & 1 & 0 & 0 \\ & 5 & 5 & 5 & 0 & 0 \\ \text{MD} & 0 & 0 & 0 & 2 & 2 \\ & 0 & 0 & 0 & 3 & 3 \\ & 0 & 0 & 0 & 1 & 1 \end{array} = \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$$

- The rank of this matrix is 2, because we have two types of documents (CS and Medical documents), i.e., two concepts.

SVD Example (cont.)

$$\begin{array}{c|ccccc} & \text{data} & \text{inf.} & \text{retrieval} & \text{brain} & \text{lung} \\ \hline \text{CS} & 1 & 1 & 1 & 0 & 0 \\ & 2 & 2 & 2 & 0 & 0 \\ & 1 & 1 & 1 & 0 & 0 \\ & 5 & 5 & 5 & 0 & 0 \\ \text{MD} & 0 & 0 & 0 & 2 & 2 \\ & 0 & 0 & 0 & 3 & 3 \\ & 0 & 0 & 0 & 1 & 1 \end{array} = \overset{\text{doc-to-concept similarity matrix}}{\begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix}} \times \overset{\text{concepts strengths}}{\begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix}} \times \overset{\text{term-to-concept similarity matrix}}{\begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}}$$

$\mathbf{U}$: document-to-concept similarity matrix
$\mathbf{V}$: term-to-concept similarity matrix.
Example: $\mathbf{U}_{1,1}$ is the weight of CS concept in document d_1, σ_1 is the strength of the CS concept, $\mathbf{V}_{1,1}$ is the weight of 'data' in the CS concept.
$\mathbf{V}_{1,2} = 0$ means data has zero similarity with the 2nd concept (Medical).

SVD for PCA

- $\mathbf{X} \in \mathbb{R}^{n \times d}$, mean centered
- $\mathbf{\Sigma}$ is the covariance matrix of $\mathbf{X}$
- PCA eigenvectors can be obtained using SVD of either $\mathbf{X}$ or $\mathbf{\Sigma}$.

$$\mathbf{\Sigma} = \mathbf{U_\Sigma S_\Sigma V_\Sigma}^{\mathsf{T}} \qquad \mathbf{X} = \mathbf{U_X S_X V_X}^{\mathsf{T}}$$

$\mathbf{U_\Sigma}$ and $\mathbf{V_X}$ both have the eigenvectors for applying PCA

Computing principal components with SVD is nice because it sorts eigenvectors, and could avoid some numerical precision issues that may arise when calculating covariance matrix directly.

SVD for PCA

- Theorem 1: if square $d \times d$ matrix $\mathbf{S}$ is a real and symmetric matrix ($\mathbf{S} = \mathbf{S}^\mathsf{T}$) then

$$\mathbf{S} = \mathbf{V\Lambda V}^\mathsf{T}$$

where $\mathbf{V} = [v_1 \quad \cdots \quad v_d]$ are the eigenvectors of $\mathbf{S}$ and $\mathbf{\Lambda} = diag(\lambda_1, \dots, \lambda_d)$ are the eigenvalues.

- Theorem 2: Let $\mathbf{A} = \mathbf{U\Sigma V}^\mathsf{T}$ be the SVD of an $N \times d$ matrix $\mathbf{A}$ and $\mathbf{C} = \frac{1}{N-1}\mathbf{A}^\mathsf{T}\mathbf{A}$ be the $d \times d$ covariance matrix. The eigenvectors of C are the same as the right singular vectors of $\mathbf{A}$.

Vector Space Model

- Documents and queries are represented as vectors
- Each dimension corresponds to a separate term
- If a term occurs in a document or query, its value is non-zero

$$d_j = [w_{1,j} \quad w_{2,j} \quad \cdots \quad w_{t,j}]^\mathsf{T}$$

$$q = [w_1 \quad w_2 \quad \cdots \quad w_t]^\mathsf{T}$$

Vector Space Model (cont.)

Documents

D1:How to bake bread without recipes
D2:The classic art of Viennese pastry
D3:Numerical recipes: The art of scientific computing
D4:Breads, pastries, pies and cakes: quantity baking recipes
D5:Pastry: A book of best french recipes

Terms

T1:bak(e,ing)	**T2:recipes**	**T3:bread**	**T4:cake**	**T5:pastr(y,ies)**	**T6:pie**

stemming

$$d_1 = [1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0]^{\mathrm{T}}$$

Vector Space Model (cont.)

Whole database: *d* documents described by *t* terms

t x d term-by-document matrix

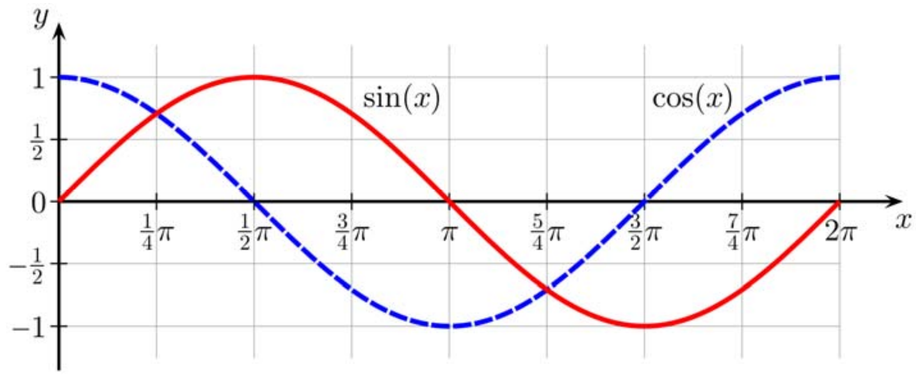
$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad \hat{\mathbf{A}} = \begin{bmatrix} .5774 & 0 & 0 & .4082 & 0 \\ .5774 & 0 & 1 & .4082 & .7071 \\ .5774 & 0 & 0 & .4082 & 0 \\ 0 & 0 & 0 & .4082 & 0 \\ 0 & 1 & 0 & .4082 & .7071 \\ 0 & 0 & 0 & .4082 & 0 \end{bmatrix}$$

Similarity Measure

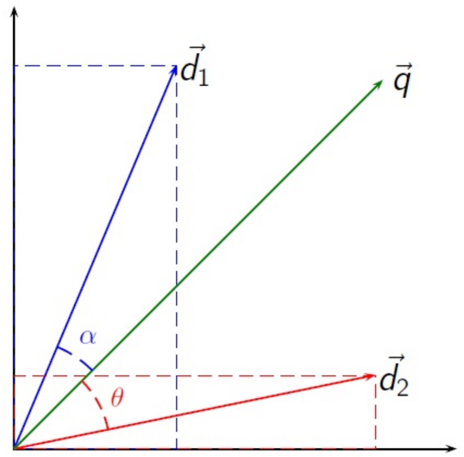
- Relevant documents are identified by simple vector operations
- Using spatial proximity for semantic proximity
- Cosine similarity is a widespread similarity measure. It gives the cosine of the angle between two vectors.

$$\cos(\theta_{\vec{x},\vec{y}}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \, \|\vec{y}\|}$$

Denominator is one when dealing with unit vectors



Two-Dimensional Example (two terms)



Term Weighting

- Simplest term (vector component) weightings are:
  - Count of number of times word occurs in document
  - Binary: word does or doesn't occur in document
- A document may be considered a better match if a word occurs three times, as opposed to once, but not a three times better match.
  - Weighting Functions
    - $1 + \log(x)$ if $x > 0$ else $\sqrt{x}$
- But… Occurrence of a term in a document may be more important if that term does not occur in many other documents
  - Solution: weight = global weight × local weight

Vector Space Shortcomings

- Inability to address *synonymy* and *polysemy*.
  - synonymy refers to a case where two different words (say car and automobile) have the same meaning.
    - synonymy $\rightarrow$ underestimate true similarity
  - polysemy on the other hand refers to the case where a term such as charge has multiple meanings
    - polysemy $\rightarrow$ overestimate true similarity

Could we use the co-occurrences of terms (e.g., “car” and “automobile”) to capture the latent semantic associations of terms?

Latent Semantic Indexing (LSI)

Latent Semantic Indexing (LSI)

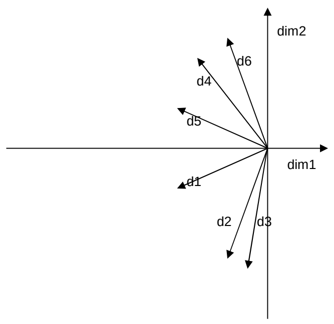
- Possible solution to synonymy/polysemy issue.
- Goal: Cluster similar documents which may share no terms in the latent semantic space, which is a low-dimensional subspace.
- LSI projects queries and documents into a space with latent semantic dimensions.
  - co-occurring words are projected on the same dimensions
  - non-co-occurring words are projected onto different dimensions
- LSI can seen as a method for dimensionality reduction
  - for example, we want to project “car” and automobile onto the same dimension

Latent Semantic Indexing (cont.)

- Dimensions of the reduced semantic space correspond to the axes of greatest variation in the original space
- LSI is accomplished by applying SVD to term-by-document matrix
- Steps:
  - Preprocessing: Compute optimal low-rank approximation (latent semantic space) to the original term-by-document matrix with help of SVD
  - Evaluation: Rank similarity of terms and docs to query in the latent semantic space via a usual similarity measure

LSI (cont.)

$$A = \left(\begin{array}{lcccccc} & \boldsymbol{d1} & \boldsymbol{d2} & \boldsymbol{d3} & \boldsymbol{d4} & \boldsymbol{d5} & \boldsymbol{d6} \\ \mathbf{cos}\, \boldsymbol{monaut} & 1 & 0 & 1 & 0 & 0 & 0 \\ \boldsymbol{astronaut} & 0 & 1 & 0 & 0 & 0 & 0 \\ \boldsymbol{moon} & 1 & 1 & 0 & 0 & 0 & 0 \\ \boldsymbol{car} & 1 & 0 & 0 & 1 & 1 & 0 \\ \boldsymbol{truck} & 0 & 0 & 0 & 1 & 0 & 1 \end{array} \right)$$



- A is a term-by-document matrix
- We want to reduce to less dimensions
  - Maybe two?
    - Space
    - Automobiles

SVD for LSI

$$\mathbf{A} = \mathbf{TSD}^{\mathsf{T}}$$

- $\mathbf{A}$ is the term-by-document matrix
- $\mathbf{T}$ is interpreted as a term-to-concept similarity matrix
- $\mathbf{S}$ is interpreted as the concept strengths
- $\mathbf{D}$ is interpreted as concept-to-doc similarity matrix
- If rank of $\mathbf{A}$ is smaller than term count, we can directly project into a reduced dimensionality space. We may also want to reduce the dimensionality of $\mathbf{A}$ further by setting small singular values of $\mathbf{S}$ to zero.

LSI Steps (visual example follows)

- Compute SVD of $\mathbf{A} = \mathbf{TSD}^{\mathsf{T}}$
- Form $\widehat{\mathbf{A}}$ by replacing the $r - k$ smallest singular values on the diagonal by zeros, which is the optimal reduced *rank-k* approximation of $\mathbf{A}$.
- Projection of documents from the original space to the reduced rank-k approximation
  - in the original space, n dimensions correspond to terms
  - in the new reduced space, k dimensions correspond to concepts
- Project the query from the original space to the reduced rank-k approximation
- Then we can rank similarity of documents to query in the reduced latent semantic space via a usual similarity measure

Example

$$A = \begin{pmatrix} & d1 & d2 & d3 & d4 & d5 & d6 \\ \cos monaut & 1 & 0 & 1 & 0 & 0 & 0 \\ astronaut & 0 & 1 & 0 & 0 & 0 & 0 \\ moon & 1 & 1 & 0 & 0 & 0 & 0 \\ car & 1 & 0 & 0 & 1 & 1 & 0 \\ truck & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

$$T = \begin{pmatrix} & \dim 1 & \dim 2 & \dim 3 & \dim 4 & \dim 5 \\ \cos monaut & -0.44 & -0.30 & 0.57 & 0.58 & 0.25 \\ astronaut & -0.13 & -0.33 & -0.59 & 0.00 & 0.73 \\ moon & -0.48 & -0.51 & -0.37 & 0.00 & -0.61 \\ car & -0.70 & 0.35 & 0.15 & -0.58 & 0.16 \\ truck & -0.26 & 0.65 & -0.41 & 0.58 & -0.09 \end{pmatrix}$$

$$S = \begin{pmatrix} 2.16 & 0 & 0 & 0 & 0 \\ 0 & 1.59 & 0 & 0 & 0 \\ 0 & 0 & 1.28 & 0 & 0 \\ 0 & 0 & 0 & 1.00 & 0 \\ 0 & 0 & 0 & 0 & 0.39 \end{pmatrix}$$

$$D^T = \begin{pmatrix} & d1 & d2 & d3 & d4 & d5 & d6 \\ \dim 1 & -0.75 & -0.28 & -0.20 & -0.45 & -0.33 & -0.12 \\ \dim 2 & -0.29 & -0.53 & -0.19 & 0.63 & 0.22 & 0.41 \\ \dim 3 & 0.28 & -0.75 & 0.45 & -0.20 & 0.12 & -0.33 \\ \dim 4 & 0 & 0 & 0.58 & 0 & -0.58 & 0.58 \\ \dim 5 & -0.53 & 0.29 & -0.63 & 0.19 & 0.41 & -0.22 \end{pmatrix}$$

Example (cont.)

- Rank-2 Approximation

$$S^r = \begin{pmatrix} 2.16 & 0 & 0 & 0 & 0 \\ 0 & 1.59 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

We can get rid of zero valued columns and rows
And have a 2 x 2 concept strength matrix

$$T^r = \begin{pmatrix} & \dim 1 & \dim 2 & \dim 3 & \dim 4 & \dim 5 \\ \cos monaut & -0.44 & -0.30 & 0 & 0 & 0 \\ astronaut & -0.13 & -0.33 & 0 & 0 & 0 \\ moon & -0.48 & -0.51 & 0 & 0 & 0 \\ car & -0.70 & 0.35 & 0 & 0 & 0 \\ truck & -0.26 & 0.65 & 0 & 0 & 0 \end{pmatrix}$$

We can get rid of zero valued columns
And have a 5 x 2 *term-to-concept similarity* matrix

$$D^{rT} = \begin{pmatrix} & d1 & d2 & d3 & d4 & d5 & d6 \\ \dim 1 & -0.75 & -0.28 & -0.20 & -0.44 & -0.33 & -0.12 \\ \dim 2 & -0.29 & -0.53 & -0.19 & 0.65 & 0.22 & 0.41 \\ \dim 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ \dim 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ \dim 5 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

We can get rid of zero valued columns
And have a 2 x 6 *concept-to-doc similarity matrix*

dim1 and dim2 are the new concepts

Query

Original space

Reduced latent semantic space

$$Q = \begin{pmatrix} \cos monaut & 1 \\ astronaut & 0 \\ moon & 0 \\ car & 0 \\ truck & 0 \end{pmatrix} \longrightarrow Q^r = \begin{pmatrix} \text{dim}\,1 & -0.44 \\ \text{dim}\,2 & -0.30 \end{pmatrix}$$

$$A = \begin{pmatrix} & d1 & d2 & d3 & d4 & d5 & d6 \\ \cos monaut & 1 & 0 & 1 & 0 & 0 & 0 \\ astronaut & 0 & 1 & 0 & 0 & 0 & 0 \\ moon & 1 & 1 & 0 & 0 & 0 & 0 \\ car & 1 & 0 & 0 & 1 & 1 & 0 \\ truck & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \longrightarrow B = \begin{pmatrix} & d1 & d2 & d3 & d4 & d5 & d6 \\ \text{dim}\,1 & -1.62 & -0.60 & -0.44 & -0.97 & -0.70 & -0.26 \\ \text{dim}\,2 & -0.46 & -0.84 & -0.30 & 1.00 & 0.35 & 0.65 \end{pmatrix}$$

$$\cos(Q, d2) = 0 \longrightarrow \cos(Q^r, d2) = 0.88$$

We see that query is not related to document 2 in the original space but in the latent semantic space they become highly related

PCA for Facial Recognition

- We have n images of different faces.

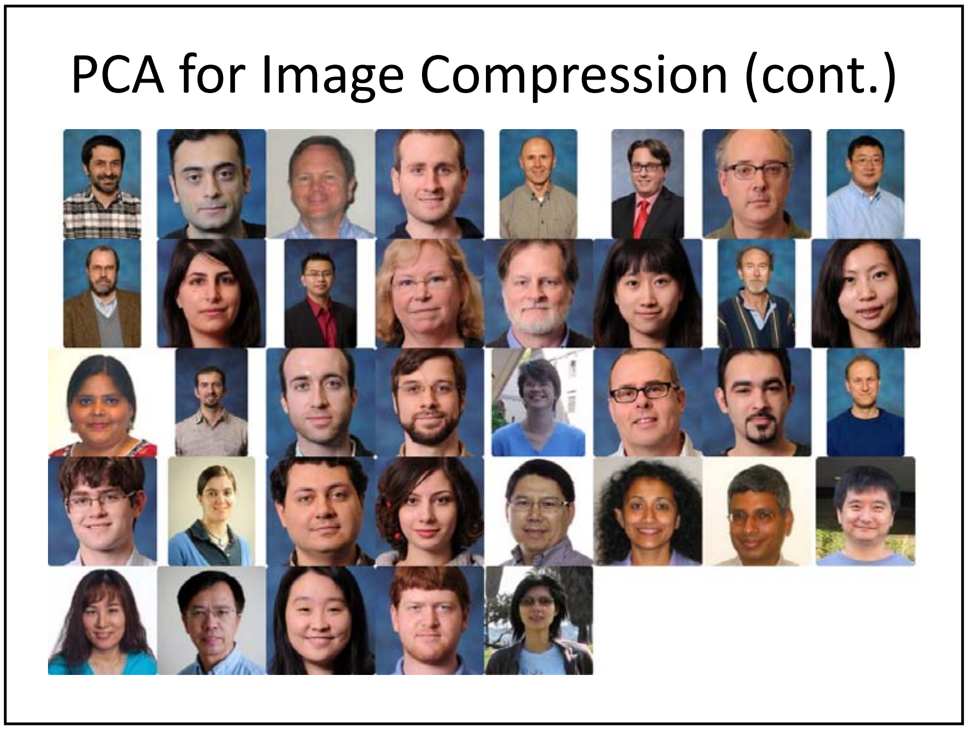
- We receive a *new* image of a face and want to match the face in the new image to a face in our existing dataset.

PCA for Facial Recognition (cont.)

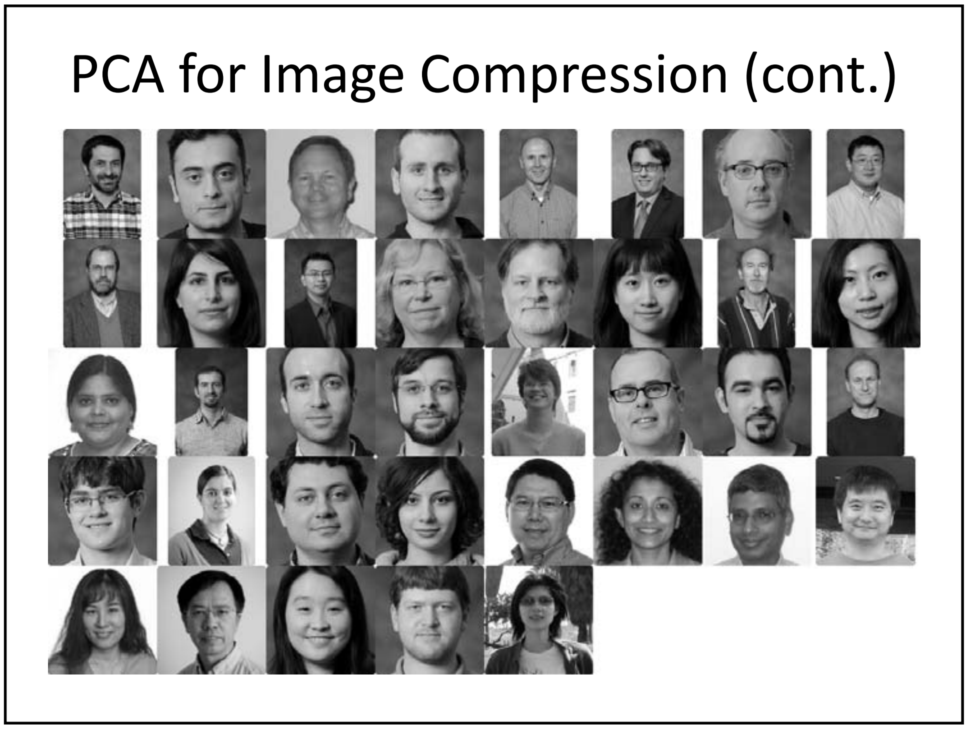
- A vector of pixel intensities (e.g., a single greyscale value) is used to represent each image.
- Match the new face image to our existing dataset by minimizing the distance to the images in our dataset.
- It turns out that this method works better if we measure distance not along the original axes, but the principal component axes.
  - There will be $w \cdot h$ (width times height of image) components

PCA for Image Compression

- aka Hotelling, or Karhunen and Leove (KL) Transform
- For facial recognition, we calculated *wh* eigenvectors for our PCA transformation matrix.
- For image compression, we transpose our data before calculating the eigenvectors.
  - So the observations (a vector of pixel intensities) in our original dataset, are now treated like the features, and the original features (intensity of a particular pixel), are now treated like observations.
  - Covariance matrix will be $n \times n$, where n is the number of images, whereas it is typically $d \times d$, where d is the number of dimensions.
- For compressing the data, we keep apply PCA using k dimensions, where $k \leq n$. A lower k will result in more compression.



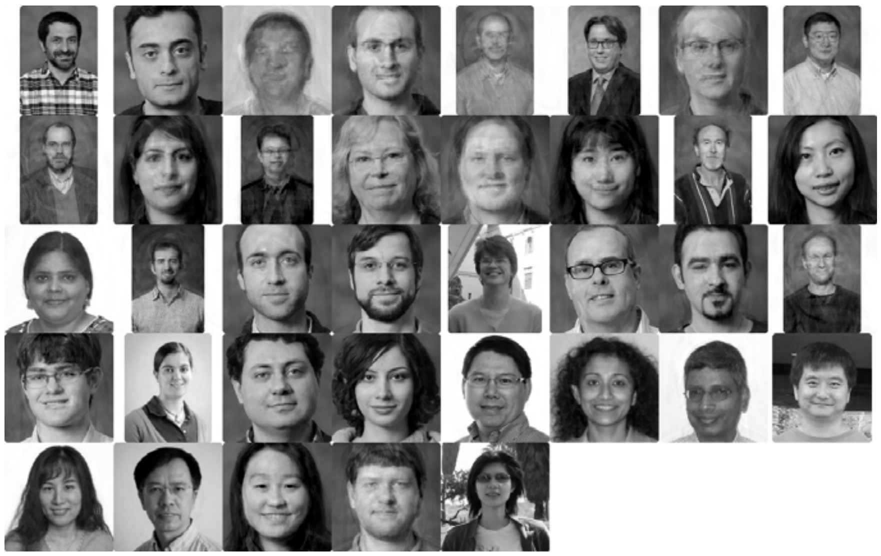
PCA for Image Compression (cont.)



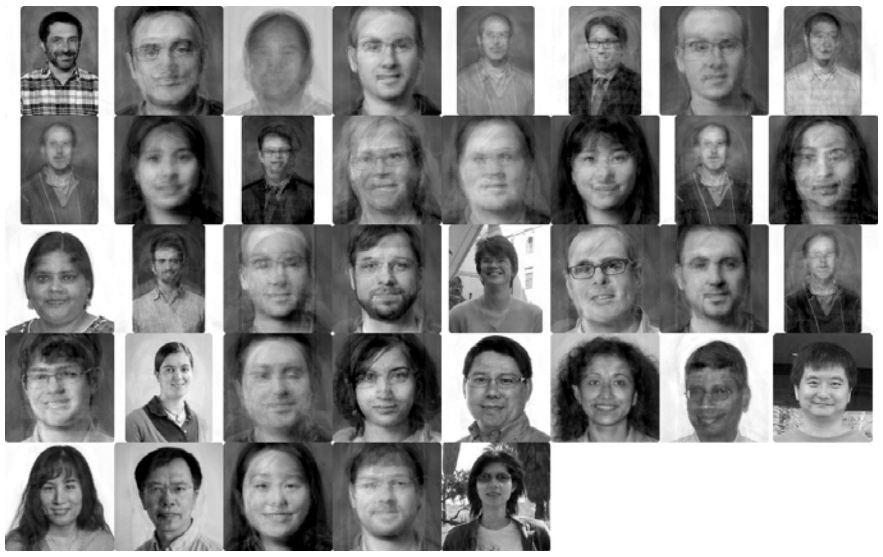
PCA for Image Compression (cont.)

k=37

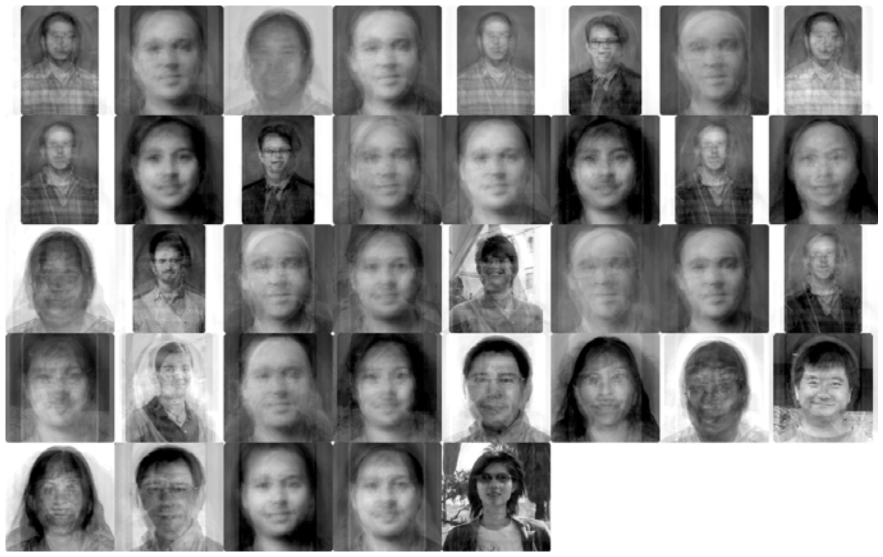
k=30



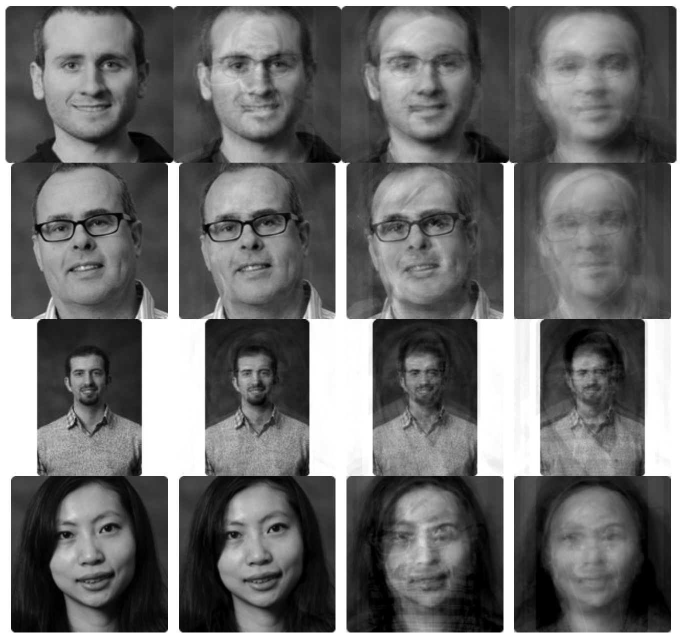
k=20



k=10



k = 37,30,20,10



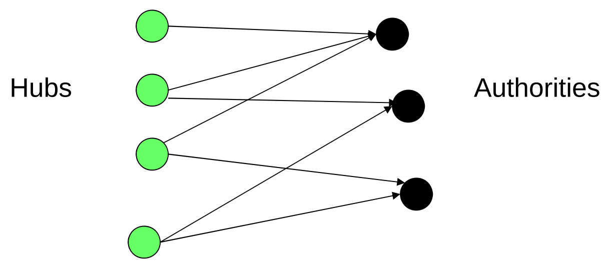
PCA for Image Compression (cont.)

- The algorithm requires storing an $n \times k$ transformation matrix of eigenvectors for compressing and decompressing (n is number of images in dataset, and k is the number of principal components).
- Compression Ratio is n: k
- I did not try compressing and uncompressing images that were outside the dataset used to generate the eigenvectors.

Kleinberg's Algorithm Hyperlink-Induced Topic Search (HITS) aka 'hubs and authorities'

- Extracting information from link structures of a hyperlinked environment
- Basic essentials
  - Authorities
  - Hubs
- For a topic, authorities are relevant nodes which are referred by many hubs
- For a topic, hubs are nodes which connect many related authorities for that topic
- Authorities are defined in terms of hubs and hubs defined in terms of authorities
  - Mutually enforcing relationship (global nature)
- Pages each receive two scores, the authority score and hub score
  - Authority score estimates value of content on page
  - Hub score estimates value of links on page

Authorities and Hubs



- The algorithm can be applied to arbitrary hyperlinked environments
  - World Wide Web (nodes correspond to web pages with links)
  - Publications Database (nodes correspond to publications and links to co-citation relationship)

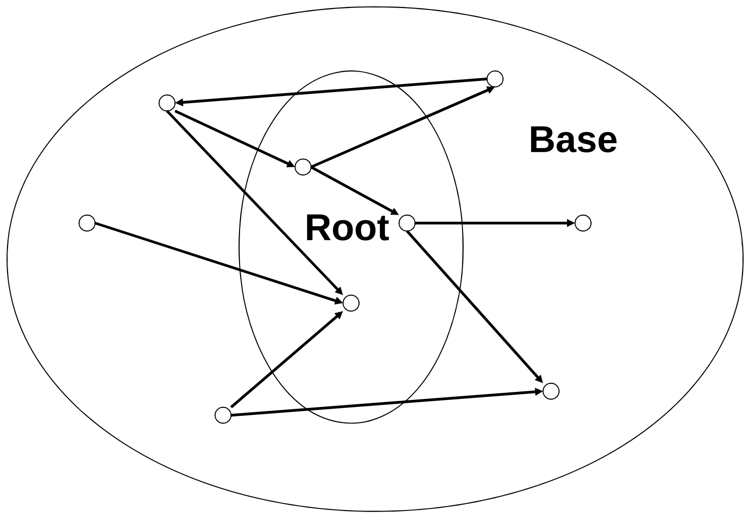
HITS (cont.)

- Three Steps
  - Create a focused sub-graph of the Web
  - Iteratively compute hub and authority scores
  - Filter out the top hubs and authorities

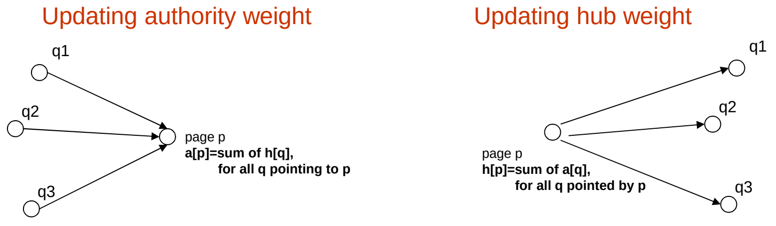
HITS (cont.)

- For the success of the algorithm base set (sub-graph) should be
  - relatively small
  - rich in relevant pages
  - contains most of the strongest authorities
- Start first with a root set
  - obtained from a text-based search engine
  - does not satisfy third condition of a useful subgraph
- Solution: extending root set
  - add any page pointed by a page in the root set to it
  - add any page that points to a page in the root set to it (at most d)
  - the extended root set becomes our base set

Root and Base Set

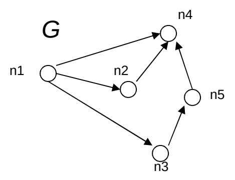


Two Operations



- a[p] … authority weight for page p
- h[p] … hub weight for page p
- Iterative algorithm
  1. set all weights for each page to 1
  2. apply both operations on each page from the base set and normalize authority and hub weights separately (sum of squares=1)
  3. repeat step 2 until weights converge

Matrix Notation



$$A = \begin{array}{c|ccccc} & n1 & n2 & n3 & n4 & n5 \\ \hline n1 & 0 & 1 & 1 & 1 & 0 \\ n2 & 0 & 0 & 0 & 1 & 0 \\ n3 & 0 & 0 & 0 & 0 & 1 \\ n4 & 0 & 0 & 0 & 0 & 0 \\ n5 & 0 & 0 & 0 & 1 & 0 \end{array}$$

- G (root set) is a directed graph with web pages as nodes and their links
- G can be presented as a connectivity matrix A
  1. A(i,j)=1 only if i-th page points to j-th page
- Authority weights can be represented as a unit vector a
  1. a(i) is the authority weight of the i-th page
- Hub weights can be represented as a unit vector h
  1. h(i) is the hub weight of the i-th page

Convergence

- Two mentioned basic operations can be written as matrix operations (all values are updated simultaneously)
  1. Updating authority weights: $\mathbf{a} = \mathbf{A}^{\mathsf{T}}\mathbf{h}$
  2. Updating hub weights: $\mathbf{h} = \mathbf{A}\mathbf{a}$
- After k iterations:

$$\begin{array}{l} \boldsymbol{a}_1 = \boldsymbol{A}^T \boldsymbol{h}_0 \\ \boldsymbol{h}_1 = \boldsymbol{A}\boldsymbol{a}_1 \end{array} \xrightarrow{\qquad} \boldsymbol{h}_1 = \boldsymbol{A}\boldsymbol{A}^T \boldsymbol{h}_0 \rightarrow \boldsymbol{h}_k = (\boldsymbol{A}\boldsymbol{A}^T)^k \boldsymbol{h}_0$$

- Thus
  1. h_k is a unit vector in the direction of $(AA^T)^k h_0$
  2. a_k is a unit vector in the direction of $(A^TA)^{k-1} h_0$
- Theorem
  1. converges to the principal eigenvector of A^TA
  2. h_k converges to the principal eigenvector of AA^T

Note: This slide was not included when presenting.

Convergence

- $(A^TA)^k$ x v\` ≈ (const) $v_1$ where k>>1, v\` is a random vector, v_1 is the eigenvector of A^TA
- Proof:

$(A^TA)^k = (A^TA) \times (A^TA) \times \ldots = (V\Lambda^2V^T) \times (V\Lambda^2V^t) \times \ldots$

$= (V\Lambda^2V^T) \times \ldots = (V\Lambda^4V^T) \times \ldots. = (V\Lambda^{2k}V^T)$

Using spectral decomposition:

$(A^TA)^k = (V\Lambda^{2k}V^T) = \lambda_1^{2k} v_1 v_1^T + \lambda_2^{2k} v_2 v_2^T + \ldots + \lambda_n^{2k} v_n v_n^T$

because $\lambda_1 > \lambda_{i\neq1} \rightarrow \lambda_1^{2k} >> \lambda_{i\neq1}^{2k}$

thus $(A^TA)^k \approx \lambda_1^{2k} v_1 v_1^T$

now $(A^TA)^k$ x v\` = $\lambda_1^{2k} v_1 v_1^T$ x v\` = (const) v_1

because v_1^Tx v\` is a scalar.

Note: This slide was not included when presenting.

Sign of Eigenvector

- We know that $(A^TA)^k \approx \lambda_1^{2k} v_1 v_1^T$
- Since A is the adjacency matrix, elements of $(A^TA)^k$ are all positive
- $\rightarrow \lambda_1^{2k} v_1 v_1^T$ should be positive
- λ_1^{2k} is positive $\rightarrow$ $v_1 v_1^T$ is positive $\rightarrow$ all elements of v_1 should have the same sign (either all elements are positive or all are negative)

Note: This slide was not included when presenting.

Sub-Communities

- Authority vector converges to the principal eigenvector of A^TA, which lets us choose strong authorities
- Hub vector converges to the principal eigenvector of AA^T which lets us choose strong hubs
- These chosen authorities and hubs build a cluster in our network
- However there can exist different clusters of authorities and hubs for a given topic, which correspond to:
  - different meanings of a term (e.g. jaguar → animal,car,team)
  - different communities for a term (e.g. randomized algorithms)
  - polarized thoughts for a term (e.g. abortion)
- Extension:
  - each eigenvector of A^TA and AA^T represents distinct authority and hub vectors for a sub-community in Graph G, respectively.

Note: This slide was not included when presenting.

PageRank Algorithm

- PageRank is a link analysis algorithm that assigns weights to nodes of a hyperlinked environment
- It assigns importance scores to every node in the set which is similar to the authority scores in Kleinberg algorithm
- It is an iterative algorithm like Kleinberg algorithm
- Main assumptions:
  - in-degree of nodes are indicators of their importance
  - links from different nodes are not counted equally. They are normalized by the out-degree of its source.
- Not performed at query time, like Kleinberg's algorithm.

Simplified PageRank (WWW)

$$\mathbf{Pr}(u) = \sum_{v \in B(u)} \frac{\mathbf{Pr}(v)}{L(v)}$$

B(u) is the set of nodes which have a link to u

- PageRank Algorithm simulates a random walk over web pages.
- Pr value are interpreted as probabilities
- In each iteration we update Pr values of each page simultaneously
- After several passes, Pr value converges to a probability distribution used to represent the probability that a person randomly clicking on links will arrive at any particular page

Note: This slide was not included when presenting.

Matrix Notation

$$\overbrace{\mathbf{Pr}_{kx1} = M_{kxk} \times \mathbf{Pr}_{kx1}}^{\text{Update step}} \qquad M_{ij} = \begin{cases} \frac{1}{|B_j|} & \text{, if i} \in B_j \\ 0 & \text{, else} \end{cases}$$

k is the number of total pages
B_i is the set of pages which have a link to i-th page

- M(i,j) is the transition matrix and defines the fragment of the j^{th} page's Pr value which contributes to the Pr value of the i^{th} page

Note: This slide was not included when presenting.

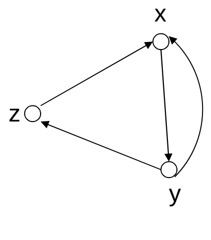
PageRank and Markov Chain

- PageRank defines a Markov Chain on the pages
  - with transition matrix M and stationary distribution Pr
    - states are pages
    - transitions are the links between pages (all equally probable)

- Pr value of a page is the probability of being at that page after lots of clicks.

Note: This slide was not included when presenting.

Matrix Notation



Update step

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 & 0.5 & 1 \\ 1 & 0 & 0 \\ 0 & 0.5 & 0 \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$x = 0 \cdot x + 1/2 \cdot y + 1 \cdot z$$
$$y = 1 \cdot x + 0 \cdot y + 0 \cdot z$$
$$z = 0 \cdot x + 1/2 \cdot y + 0 \cdot z$$

Note: This slide was not included when presenting.

Non-Simplified PageRank (WWW)

$$\Pr(u) = \frac{1-d}{k} + d \cdot \sum_{v \in B(u)} \frac{\Pr(v)}{L(v)}$$

Matrix Notation

$$\overbrace{\mathbf{Pr}_{kx1} = \boldsymbol{M}_{kxk} \times \mathbf{Pr}_{kx1}}$$

$$\boldsymbol{M}_{ij} = \begin{cases} \frac{\mathbf{1}-\boldsymbol{d}}{\boldsymbol{k}} + \frac{\boldsymbol{d}}{|\boldsymbol{B}_j|} & \text{, if i} \in \boldsymbol{B}_j \\ \frac{\mathbf{1}-\boldsymbol{d}}{\boldsymbol{k}} & \text{, else} \end{cases}$$

k is the number of total pages
B_i is the set of pages which have a link to i-th page

- (1-d) defines the probability to jump to a page, for which there is no link from the current page
  - Pr converges to the principal eigenvector of the transition matrix M

Note: This slide was not included when presenting.

Randomized HITS

- Random walk on HITS
- Odd time steps: update authority
- Even time steps: update hubs

$$a^{(t+1)} = \epsilon\vec{1} + (1-\epsilon)\mathbf{A}_{\text{row}}{}^{\mathsf{T}} h^{(t)}$$

$$h^{(t+1)} = \epsilon\vec{1} + (1-\epsilon)\mathbf{A}_{\text{col}}{}^{\mathsf{T}} a^{(t)}$$

- t: a very large odd number, large enough that the random walk converged → The authority weight of a page = the chance that the page is visited on time step t

Note: This slide was not included when presenting.

Stability of Algorithms

- Being stable to perturbations of the link structure.
- HITS: if the eigengap is big, insensitive to small perturbations; if it's small there may be a small perturbation that can dramatically change its results.
- PageRank: if the perturbed/modified web pages did not have high overall PageRank, then the perturbed PageRank scores will not be far from the original.
- Randomized HITS: insensitive to small perturbations

Note: This slide was not included when presenting.

Additional PCA and SVD Applications

- Pattern Detection and Visualization in Gene Expression Data
- Recommendation Systems and Collaborative Filtering (winning entry for Netflix Prize included SVD models)

Thanks to Sherry, Cem, and Iyad!