Latent Semantic Indexing

Chapter 18

LSI

- Applications in Search Engine Optimization (SEO) video http://www.youtube.com/watch?v=LOPY1hPcZEM

Latent Semantic Indexing (LSI)
(AKA "Latent Semantic Analysis" (LSA))

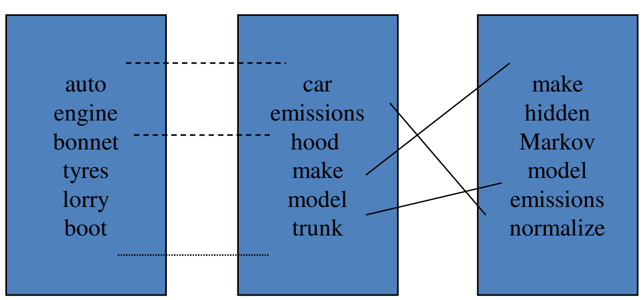
- Problem: How to capture semantic similarity between documents in a natural corpus (e.g., problems of homonymy, polysemy, synonymy, etc.)

- "LSA assumes that there exists a LATENT structure in word usage – obscured by variability in word choice" (http://ir.dcs.gla.ac.uk/oldseminars/Girolami.ppt)

From *www.cs.nmsu.edu/~mmartin/LSA_Intro_AI_Seminar.ppt*

The Problem

- Example: Vector Space Model
  - (from Lillian Lee)



Synonymy

Will have small cosine

but are related

Polysemy

Will have large cosine

but not truly related

Principal Components Analysis

- PCA used to reduce dimensions of data without much loss of information.
- Used in machine learning and in signal processing and image compression (among other things).

PCA is "an orthogonal linear transformation that transfers the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (*first principal component*), the second greatest variance lies on the second coordinate (*second principal component*), and so on."

Background for PCA

- Suppose attributes are A_1 and A_2, and we have n training examples. x's denote values of A_1 and y's denote values of A_2 over the training examples.

- Variance of an attribute:

$$\mathrm{var}(A_1) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{(n-1)}$$

- Covariance of two attributes:

$$\mathrm{cov}(A_1, A_2) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)}$$

- If covariance is positive, both dimensions increase together. If negative, as one increases, the other decreases. Zero: independent of each other.

- Covariance matrix
  - Suppose we have n attributes, A_1, ..., A_n.
  - Covariance matrix:

$$C^{n \times n} = (c_{i,j}), \text{ where } c_{i,j} = \mathrm{cov}(A_i, A_j)$$

	Hours(H)	Mark(M)
Data	9	39
	15	56
	25	93
	14	61
	10	50
	18	75
	0	32
	16	85
	5	42
	19	70
	16	66
	20	80
Totals	167	749
Averages	13.92	62.42

Covariance:

H	M	$(H_i - \bar{H})$	$(M_i - \bar{M})$	$(H_i - \bar{H})(M_i - \bar{M})$
9	39	-4.92	-23.42	115.23
15	56	1.08	-6.42	-6.93
25	93	11.08	30.58	338.83
14	61	0.08	-1.42	-0.11
10	50	-3.92	-12.42	48.69
18	75	4.08	12.58	51.33
0	32	-13.92	-30.42	423.45
16	85	2.08	22.58	46.97
5	42	-8.92	-20.42	182.15
19	70	5.08	7.58	38.51
16	66	2.08	3.58	7.45
20	80	6.08	17.58	106.89
Total				1149.89
Average				104.54

Table 2.2: 2-dimensional data set and covariance calculation

$$\begin{pmatrix} \mathrm{cov}(H,H) & \mathrm{cov}(H,M) \\ \mathrm{cov}(M,H) & \mathrm{cov}(M,M) \end{pmatrix}$$

$$= \begin{pmatrix} \mathrm{var}(H) & 104.5 \\ 104.5 & \mathrm{var}(M) \end{pmatrix}$$

$$= \begin{pmatrix} 47.7 & 104.5 \\ 104.5 & 370 \end{pmatrix}$$

Covariance matrix

- Eigenvectors:
  - Let **M be** an $n \times n$ matrix.
    - **v** is an *eigenvector* of M if M × **v = λv**
    - λ is called the *eigenvalue* associated with **v**
  - For any eigenvector **v** of **M** and scalar *a*,

$$\mathbf{M} \times a\mathbf{v} = \lambda a\mathbf{v}$$

  - Thus you can always choose eigenvectors of length 1:

$$\sqrt{v_1{}^2 + \ldots + v_n{}^2} = 1$$

  - If **M** has any eigenvectors, it has *n* of them, and they are orthogonal to one another.
  - Thus eigenvectors can be used as a new basis for a n-dimensional vector space.

PCA

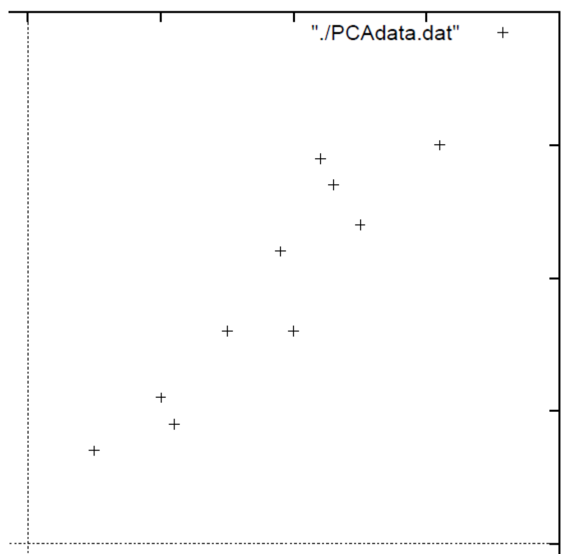
1. Given original data set S = {$\mathbf{x}^1$, ..., $\mathbf{x}^k$}, produce new set by subtracting the mean of attribute A_i from each x_i.

Data =

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9
Mean: 1.81	1.91

DataAdjust =

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01
Mean: 0	0



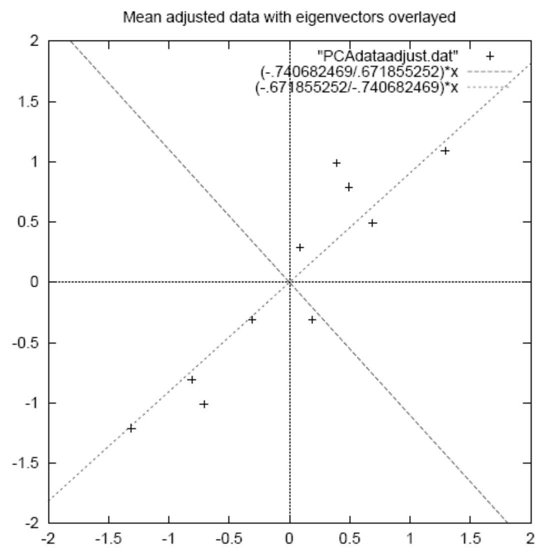
2. Calculate the covariance matrix:

$$cov = \begin{array}{c c} & \begin{array}{cc} x & y \end{array} \\ \begin{array}{c} x \\ y \end{array} & \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix} \end{array}$$

3. Calculate the (unit) eigenvectors and eigenvalues of the covariance matrix:

$$eigenvalues = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$



Eigenvector with largest eigenvalue traces linear pattern in data

Figure 3.2: A plot of the normalised data (mean subtracted) with the eigenvectors of the covariance matrix overlayed on top.

4. Order eigenvectors by eigenvalue, highest to lowest.

$$\mathbf{v}_1 = \begin{pmatrix} -.677873399 \\ -.735178956 \end{pmatrix} \quad \lambda = 1.28402771$$

$$\mathbf{v}_2 = \begin{pmatrix} -.735178956 \\ .677873399 \end{pmatrix} \quad \lambda = .0490833989$$

In general, you get n components. To reduce dimensionality to p, ignore $n-p$ components at the bottom of the list.

Construct new feature vector.

Feature vector = ($\mathbf{v}_1$, $\mathbf{v}_2$, ...$\mathbf{v}_p$)

$$FeatureVector1 = \begin{pmatrix} -.677873399 & -.735178956 \\ -.735178956 & .677873399 \end{pmatrix}$$

or reduced dimension feature vector :

$$FeatureVector2 = \begin{pmatrix} -.677873399 \\ -.735178956 \end{pmatrix}$$

5. Derive the new data set.

$$TransformedData = RowFeatureVector \times RowDataAdjust$$

$$RowFeatureVector1 = \begin{pmatrix} -.677873399 & -.735178956 \\ -.735178956 & .677873399 \end{pmatrix}$$

$$RowFeatureVector2 = \begin{pmatrix} -.677873399 & -.735178956 \end{pmatrix}$$

$$RowDataAdjust = \begin{pmatrix} .69 & -1.31 & .39 & .09 & 1.29 & .49 & .19 & -.81 & -.31 & -.71 \\ .49 & -1.21 & .99 & .29 & 1.09 & .79 & -.31 & -.81 & -.31 & -1.01 \end{pmatrix}$$

This gives original data in terms of chosen components (eigenvectors)—that is, along these axes.

Transformed Data=

x	y
-.827970186	-.175115307
1.77758033	.142857227
-.992197494	.384374989
-.274210416	.130417207
-1.67580142	-.209498461
-.912949103	.175282444
.0991094375	-.349824698
1.14457216	.0464172582
.438046137	.0177646297
1.22382056	-.162675287

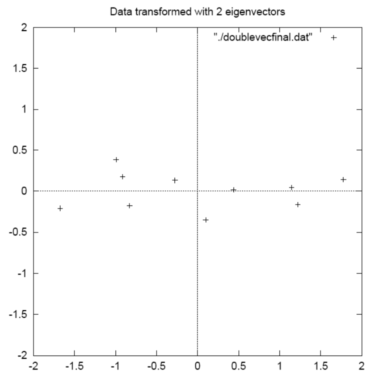
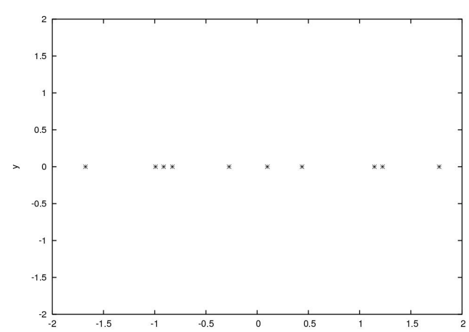


Figure 3.3: The table of data by applying the PCA analysis using both eigenvectors, and a plot of the new data points.

Transformed Data (Single eigenvector)

x
-.827970186
1.77758033
-.992197494
-.274210416
-1.67580142
-.912949103
.0991094375
1.14457216
.438046137
1.22382056



Reconstructing the original data

We did:

$$TransformedData = RowFeatureVector \times RowDataAdjust$$

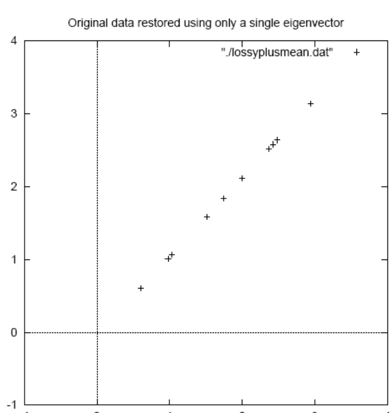
so we can do

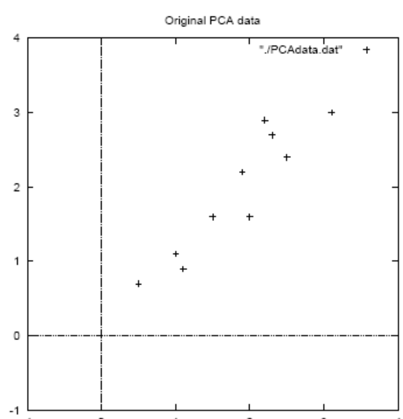
$$RowDataAdjust = RowFeatureVector^{-1} \times TransformedData$$

$$= RowFeatureVector^{T} \times TransformedData$$

and

$$RowDataOriginal = RowDataAdjust + OriginalMean$$





Example: Linear discrimination using PCA for face recognition

1. Preprocessing: “Normalize” faces

   - Make images the same size
   - Line up with respect to eyes
   - Normalize intensities

2. Raw features are pixel intensity values (2061 features)

3. Each image is encoded as a vector Γ_i of these features

4. Compute "mean" face in training set:

$$\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i$$

The average face and first four eigenfaces

- Subtract the mean face from each face vector

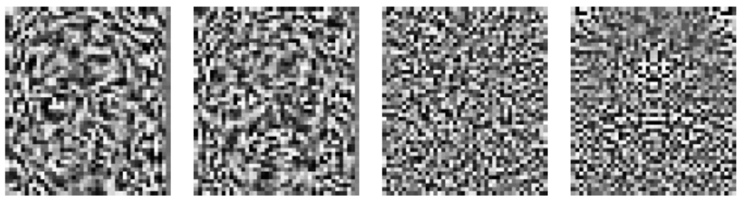
$$\Phi_i = \Gamma_i - \Psi$$

- Compute the covariance matrix **C**

- Compute the (unit) eigenvectors $\mathbf{v}_i$ of **C**

- Keep only the first *K* principal components (eigenvectors)

Eigenfaces 15, 100, 200, 250, 300



Eigenfaces 400, 450, 1000, 2000

The eigenfaces encode the principal sources of variation in the dataset (e.g., absence/presence of facial hair, skin tone, glasses, etc.).

We can represent any face as a linear combination of these "basis" faces.

Use this representation for:

- Face recognition
  (e.g., Euclidean distance from known faces)

- Linear discrimination
  (e.g., "glasses" versus "no glasses",
  or "male" versus "female")

Latent Semantic Analysis
(Landauer et al.)

- From training data (large sample of documents), create word-by-document matrix.

Technical Memo Example

Titles:

c1: *Human* machine *interface* for Lab ABC *computer* applications
c2: A *survey* of *user* opinion of *computer system response time*
c3: The *EPS user interface* management *system*
c4: *System* and *human system* engineering testing of *EPS*
c5: Relation of *user*-perceived *response time* to error measurement

m1: The generation of random, binary, unordered *trees*
m2: The intersection *graph* of paths in *trees*
m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4: *Graph minors*: A *survey*

A sample dataset consisting of the titles of 9 technical memoranda. Terms occurring in more than one title are italicized. There are two classes of documents - five about human-computer interaction (c1-c5) and four about graphs (m1-m4). This dataset can be described by means of a term by document matrix where each cell entry indicates the frequency with which a term occurs in a document.

From Deerwester et al., Indexing by latent semantic analysis

Terms	**Documents**								
	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

- Now apply "singular value decomposition" to this matrix
- SVD is similar to principal components analysis
- Basically, reduce dimensionality of the matrix by re-representing matrix in terms of "features" (derived from eigenvalues and eigenvectors), and using only the ones with highest value.
- Result: Each document is represented by a vector of *features* obtained by SVD.
- Given a new document (or query), compute its representation vector in this *feature space*, compute its similarity with other documents using cosine between vector angles. Retrieve documents with highest similarities.

Example 18.3: We now illustrate the singular-value decomposition of a 4×2 matrix of rank 2; the singular values are $\Sigma_{11} = 2.236$ and $\Sigma_{22} = 1$.

$$C = \begin{pmatrix} 1 & -1 \\ 0 & 1 \\ 1 & 0 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} -0.632 & 0.000 \\ 0.316 & -0.707 \\ -0.316 & -0.707 \\ 0.632 & 0.000 \end{pmatrix} \begin{pmatrix} 2.236 & 0.000 \\ 0.000 & 1.000 \end{pmatrix} \begin{pmatrix} -0.707 & 0.707 \\ -0.707 & -0.707 \end{pmatrix}.$$

www.cs.nmsu.edu/~mmartin/LSA_Intro_AI_Seminar.ppt

- SVD
  - can be viewed as a method for rotating the axes in n-dimensional space, so that the first axis runs along the direction of the largest variation among the documents
    - the second dimension runs along the direction with the second largest variation
    - and so on

www.cs.nmsu.edu/~mmartin/LSA_Intro_AI _Seminar.ppt

LSI

- Four basic steps
  - Rank-reduced Singular Value Decomposition (SVD) performed on matrix
    - all but the k highest singular values are set to 0
    - produces k-dimensional approximation of the original matrix
    - this is the "semantic space"
  - Compute similarities between entities in semantic space (usually with cosine)

C

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

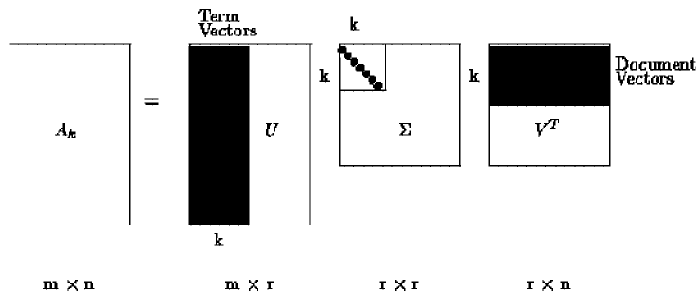
r (human.user) = -.38   r (human.minors) = -.29

- Singular Value Decomposition

$$C=U\Sigma V^T$$

- Dimension Reduction

$$\sim C=\sim U\sim\Sigma\sim V$$



- **U** =

0.22	-0.11	0.29	-0.41	-0.11	-0.34	0.52	-0.06	-0.41
0.20	-0.07	0.14	-0.55	0.28	0.50	-0.07	-0.01	-0.11
0.24	0.04	-0.16	-0.59	-0.11	-0.25	-0.30	0.06	0.49
0.40	0.06	-0.34	0.10	0.33	0.38	0.00	0.00	0.01
0.64	-0.17	0.36	0.33	-0.16	-0.21	-0.17	0.03	0.27
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.30	-0.14	0.33	0.19	0.11	0.27	0.03	-0.02	-0.17
0.21	0.27	-0.18	-0.03	-0.54	0.08	-0.47	-0.04	-0.58
0.01	0.49	0.23	0.03	0.59	-0.39	-0.29	0.25	-0.23
0.04	0.62	0.22	0.00	-0.07	0.11	0.16	-0.68	0.23
0.03	0.45	0.14	-0.01	-0.30	0.28	0.34	0.68	0.18

- Σ =

3.34								
	2.54							
		2.35						
			1.64					
				1.50				
					1.31			
						0.85		
							0.56	
								0.36

- **V**=

0.20	0.61	0.46	0.54	0.28	0.00	0.01	0.02	0.08
-0.06	0.17	-0.13	-0.23	0.11	0.19	0.44	0.62	0.53
0.11	-0.50	0.21	0.57	-0.51	0.10	0.19	0.25	0.08
-0.95	-0.03	0.04	0.27	0.15	0.02	0.02	0.01	-0.03
0.05	-0.21	0.38	-0.21	0.33	0.39	0.35	0.15	-0.60
-0.08	-0.26	0.72	-0.37	0.03	-0.30	-0.21	0.00	0.36
0.18	-0.43	-0.24	0.26	0.67	-0.34	-0.15	0.25	0.04
-0.01	0.05	0.01	-0.02	-0.06	0.45	-0.76	0.45	-0.07
-0.06	0.24	0.02	-0.08	-0.26	-0.62	0.02	0.52	-0.45

~C

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
user	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
system	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
graph	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

r (human.user) = .94 r (human.minors) = -.83

Correlation

Raw data

	c1	c2	c3	c4	c5	m 1	m 2	m 3
c2	- 019							
c3	0.00	0.00						
c4	0.00	0.00	0.47					
c5	- 033	0.58	0.00	- 031				
m 1	- 017	- 030	- 021	- 016	- 017			
m 2	- 026	- 045	- 032	- 024	- 026	0.67		
m 3	- 033	- 058	- 041	- 031	- 033	0.52	0.77	
m 4	- 033	- 019	- 041	- 031	- 033	- 017	0.26	0.56

0.02	
- 030	**0.44**

Correlations in first-two dimension space

c2	0.91							
c3	1.00	0.91						
c4	1.00	0.88	1.00					
c5	0.85	0.99	0.85	0.81				
m 1	- 085	- 056	- 085	- 088	- 045			
m 2	- 085	- 056	- 085	- 088	- 044	1.00		
m 3	- 085	- 056	- 085	- 088	- 044	1.00	1.00	
m 4	- 081	- 050	- 081	- 084	- 037	1.00	1.00	1.00

0.92	
-0.72	1.00

Semantic Similarity Measure

- To find similarity between two documents, project them in LS space
- Then calculate the cosine measure between their projection

Summary

- Some Issues
  - SVD Algorithm complexity O(n^2k^3)
    - n = number of terms
    - k = number of dimensions in semantic space (typically small ~50 to 350)
    - for stable document collection, only have to run once
    - dynamic document collections: might need to rerun SVD, but can also “fold in” new documents

Summary

- Some issues
  - Finding optimal dimension for semantic space
    - precision-recall improve as dimension is increased until hits optimal, then slowly decreases until it hits standard vector model
    - run SVD once with big dimension, say k = 1000
      - then can test dimensions <= k
    - in many tasks 150-350 works well, still room for research

Summary

- Some issues
  - SVD assumes normally distributed data
    - term occurrence is not normally distributed
    - matrix entries are weights, not counts, which may be normally distributed even when counts are not

Some General LSA Based Applications

From http://lsa.colorado.edu/~quesadaj/pdf/LSATutorial.pdf

Information Retrieval

Text Assessment

Compare document to documents of known quality / content

Automatic summarization of text

Determine best subset of text to portray same meaning

Categorization / Classification

Place text into appropriate categories or taxonomies

Application: Automatic Essay Scoring (in collaboration with Educational Testing Service)

Create domain semantic space

Compute vectors for essays, add to vector database

To predict grade on a new essay, compare it to ones previously scored by humans

From http://lsa.colorado.edu/~quesadaj/pdf/LSATutorial.pdf

Mutual information between two sets of grades:

human – human .90
LSA – human .81

From http://lsa.colorado.edu/~quesadaj/pdf/LSATutorial.pdf

Demo

http://www.pearsonkt.com/
http://www.pearsonkt.com/prodIEA.shtml
http://www.pearson.com/investors/our-news/?i=772
http://www.youtube.com/results?search_query=Karen+Lochbaum&aq=f