Information Retrieval through Various Approximate Matrix Decompositions

Kathryn Linehan

Advisor: Dr. Dianne O'Leary

Information Retrieval

- Extracting information from databases
- We need an efficient way of searching large amounts of data
- Example: web search engine

Querying a Document Database

- The problem: return documents that are relevant to entered search terms
- In real systems such as Google, the problem is formulated in terms of matrices:
  - Term-Document Matrix
  - Query vector

Term-Document Matrix

- Entry (i, j) : weight of term i in document j

Example:

Term	Document 1	2	3	4
Mark	15	0	0	0
Twain	15	0	20	0
Samuel	0	10	5	0
Clemens	0	20	10	0
Purple	0	0	0	20
Fairy	0	0	0	15

Example taken from [3]

Query Vector

- Entry (i) : weight of term i in the query

Example:

search for “Mark Twain”

Term	
Mark	1
Twain	1
Samuel	0
Clemens	0
Purple	0
Fairy	0

Example taken from [3]

Literal Term Matching

- Given:
  - Term-Document matrix, A
  - Query vector, q
- Using A and q, we need to determine which documents are relevant to the query
  - Formulate score vector: $s = q^T A$
  - Return the highest scoring documents

Document Scoring

Term		Document 1	2	3	4		Scores
Mark	$\begin{bmatrix}1\end{bmatrix}^T$	$\begin{bmatrix}15 & 0 & 0 & 0\end{bmatrix}$				$\begin{bmatrix}30\end{bmatrix}^T$	Doc 1
Twain	$\begin{bmatrix}1\end{bmatrix}$	$\begin{bmatrix}15 & 0 & 20 & 0\end{bmatrix}$				$\begin{bmatrix}0\end{bmatrix}$	Doc 2
Samuel	$\begin{bmatrix}0\end{bmatrix}$	$*\begin{bmatrix}0 & 10 & 5 & 0\end{bmatrix}$				$=\begin{bmatrix}20\end{bmatrix}$	Doc 3
Clemens	$\begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}0 & 20 & 10 & 0\end{bmatrix}$				$\begin{bmatrix}0\end{bmatrix}$	Doc 4
Purple	$\begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}0 & 0 & 0 & 20\end{bmatrix}$					
Fairy	$\begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}0 & 0 & 0 & 15\end{bmatrix}$					

- Doc 1 and Doc 3 will be returned as relevant, but Doc 2 will not

Example taken from [3]

Latent Semantic Indexing

- Problem: We need a way to return relevant documents that may not contain the exact query terms
- Solution: Latent Semantic Indexing (LSI)
  - Use an approximation to the term-document matrix

Rank-k SVD

- Standard approximation used in LSI: rank-k SVD
- SVD Review: $A = U\Sigma V^T$
  - U: orthogonal, left singular vectors of A
  - Σ: diagonal, singular values in decreasing order
  - V: orthogonal, right singular vectors of A
- Rank-k SVD: $A_k = U_k\Sigma_k V_k^T$
  - U_k: first k columns of U
  - Σ_k: diagonal, top k singular values in decreasing order
  - V_k: first k columns of V

Rank-k SVD

- Example: rank-2 approximation to original term-document matrix

Original matrix

Term	Document 1	2	3	4
Mark	3.7	3.5	5.5	0
Twain	11.0	10.3	16.1	0
Samuel	4.1	3.9	6.1	0
Clemens	8.3	7.8	12.2	0
Purple	0	0	0	20
Fairy	0	0	0	15

Example taken from [3]

Document Scoring

Original Scores

Term	Document 1	Document 2	Document 3	Document 4	Scores
Mark	3.7	3.5	5.5	0	Doc 1
Twain	11.0	10.3	16.1	0	Doc 2
Samuel	4.1	3.9	6.1	0	Doc 3
Clemens	8.3	7.8	12.2	0	Doc 4
Purple	0	0	0	20	
Fairy	0	0	0	15	

$$\begin{bmatrix}1\\1\\0\\0\\0\\0\end{bmatrix}^T * \begin{bmatrix}3.7 & 3.5 & 5.5 & 0\\11.0 & 10.3 & 16.1 & 0\\4.1 & 3.9 & 6.1 & 0\\8.3 & 7.8 & 12.2 & 0\\0 & 0 & 0 & 20\\0 & 0 & 0 & 15\end{bmatrix} = \begin{bmatrix}14.7\\13.8\\21.6\\0\end{bmatrix}^T$$

- Doc 1, Doc 2, and Doc 3 will all be returned as relevant

Example taken from [3]

Literal Term Matching vs. LSI

- Literal term matching: use sparse (m x n) term-document matrix A
- LSI: use approximation to A. For example, $A_k = U_k \Sigma_k V_k^T$, where U_k: m x k, Σ_k: k x k, V_k: n x k.
  - These factors can be dense
  - Computing A_k is a one-time expense

	Storage	Query Time
Literal term matching	O(nz(A))	O(nz(A))
LSI	O((m+n)k+k)	O((m+n)k+k)

Precision and Recall

- We need a measurement of performance for document retrieval
- Let Retrieved = number of documents retrieved,
  Relevant = total number of relevant documents to the query,
  RetRel = number of documents retrieved that are relevant.

- Precision: $$P(\text{Retrieved}) = \frac{\text{RetRel}}{\text{Retrieved}}$$

- Recall: $$R(\text{Retrieved}) = \frac{\text{RetRel}}{\text{Relevant}}$$

Precision and Recall

- Example:

  Given: set of 25 documents, query, list of documents relevant to the query = {22, 1, 11, 9, 5}

  Task: retrieve four documents in order of relevance score

  Results: 22, 3, 9, 7

$$P(1) = 1,\ P(2) = \frac{1}{2},\ P(3) = \frac{2}{3},\ P(4) = \frac{2}{4}$$

$$R(1) = \frac{1}{5},\ R(2) = \frac{1}{5},\ R(3) = \frac{2}{5},\ R(4) = \frac{2}{5}$$

Validation

- Three common information retrieval data sets found at **www.cs.utk.edu/~lsi/**
  - Listed under CISI, CRAN, MED
  - Each data set includes a term-document matrix, a term list, queries, and a list of relevant documents for each query

Project Goals

- To use matrix approximations other than the rank-k SVD in LSI
  - Test performance using average precision and recall, where the average is taken over all queries in the data set
  - Investigate storage, computation time and relative error of matrix approximations
- To investigate improvement to the CUR matrix approximation algorithm

SDD

- In [3], LSI using the semidiscrete decomposition (SDD) is investigated

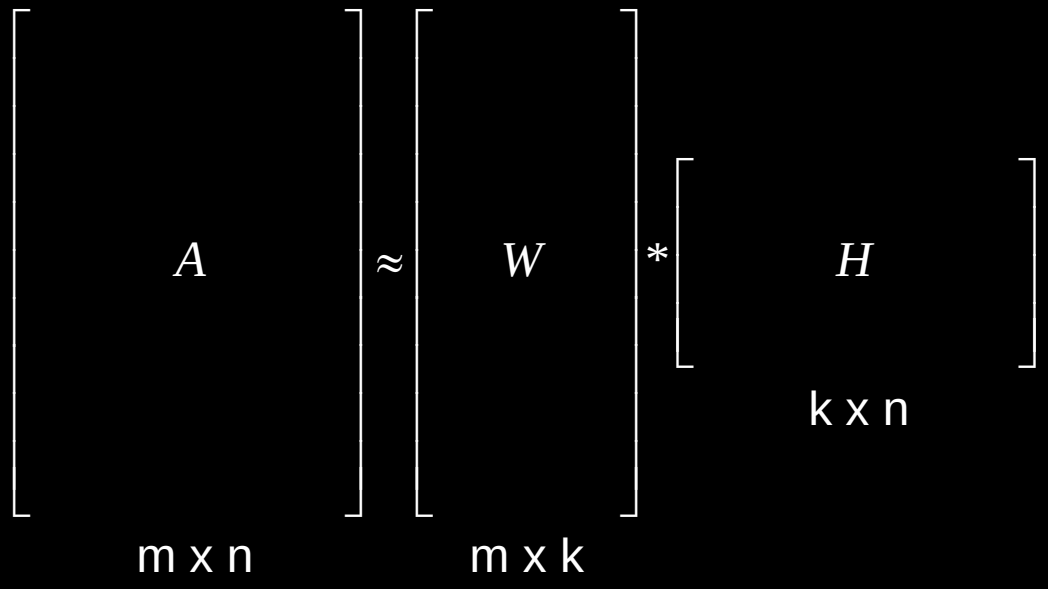
- X_k and Y_k contain entries from the set {-1,0,1}
- D_k is diagonal with positive entries
- $rank(X_k D_k Y_k^T) \leq k$

$$\underset{m \times n}{\begin{bmatrix} A_k \end{bmatrix}} = \underset{m \times k}{\begin{bmatrix} X_k \end{bmatrix}} * \underset{k \times k}{\begin{bmatrix} D_k \end{bmatrix}} * \underset{k \times n}{\begin{bmatrix} Y_k^T \end{bmatrix}}$$

Nonnegative Matrix Factorization

- Term-document matrix is nonnegative

  - W and H are nonnegative
  - $rank(WH) \leq k$



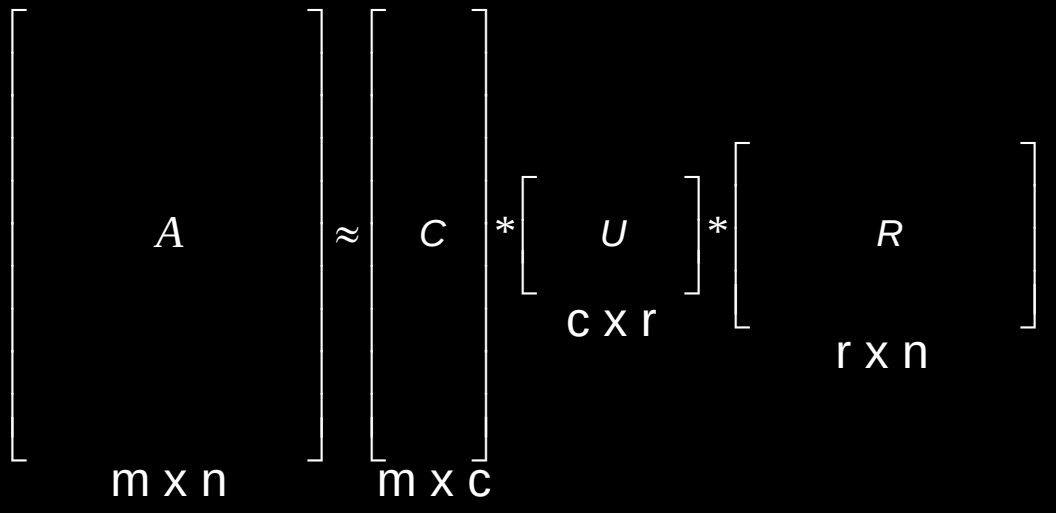
CUR

- Term-document matrix is sparse

• C (R) holds c (r) sampled and rescaled columns (rows) of A

• U is computed using C and R

• $rank(CUR) \leq k$, where k is a rank parameter



Further Topics

- Time permitting investigations
  - Parallel implementations of matrix approximations
  - Testing performance of matrix approximations in forming a multidocument summary

References

[1] Michael W. Berry, Murray Browne, Amy N. Langville, V. Paul Pauca, and Robert J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics and Data Analysis*, 52(1):155-173, September 2007.

[2] Petros Drineas, Ravi Kannan, and Michael W. Mahoney. Fast Monte Carlo algorithms for matrices iii: Computing a compressed approximate matrix decomposition. *SIAM Journal on Computing*, 36(1):184-206, 2006.

[3] Tamara G. Kolda and Dianne P. O'Leary. A semidiscrete matrix decomposition for latent semantic indexing in information retrieval. *ACM Transactions on Information Systems*, 16(4):322-346, October 1998.