

Latent Semantic Indexing

Seminar “Theoretical Topics in Data Science”

Vahe Eminyan

vahe.eminyan@rwth-aachen.de

16.12.2023

Overview

Introduction

LSI Background

Original Paper Overview and Emphasized Aspect

LSI by Random Projection

References

Introduction

Motivation

- Large datasets, often organized in tabular form, represented as **matrices**
 - Term-document matrix representing word occurrence in documents
 - Movie-user matrix representing watched movies of users
- Interesting aspects
 - **Find** documents semantically associated with a **query**
 - **Recommend** a new movie to a user

	Doc 1	Doc 2	...	Doc m
Term 1	0	1	...	1
Term 2	1	0	...	1
...
Term n	1	0	...	0



Documents

Terms
$$\begin{pmatrix} 0 & 1 & \dots & 1 \\ 1 & 0 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \dots & 0 \end{pmatrix}$$
$$n \times m$$

Latent Semantic Indexing

- LSI as an information retrieval method
- Finds the latent (hidden) semantic structure of textual data
- Represent term-document matrix as product of three matrices: term-topic, topic-topic and topic-document matrix
- Answer queries with help of these matrices
- Based on singular value decomposition of the matrix

Singular Value Decomposition (SVD) [4]

- Any n by m matrix can be factored into

$$A_{n \times m} = U_{[n \times r]} D_{[r \times r]} (V_{[m \times r]})^T = (\text{orthogonal})(\text{diagonal})(\text{orthogonal}).$$

- U : left singular vectors (n terms and r topics)
- V : right singular vectors (m documents and r topics)
- D : Singular values $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r$ in decreasing order ($r \times r$ diagonal matrix representing the "importance" of each topic, where r rank of matrix A)
- Vector notation

$$A = UDV^T = \sum_{i=1}^r \sigma_i u_i v_i^t$$

Singular Value Decomposition (SVD) Example: Matrix A with rank $r = 3$

$$\begin{array}{c} \text{Terms} \end{array} \begin{array}{c} \text{Documents} \\ \left(\begin{array}{ccc} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{array} \right) \\ A \end{array} = \begin{array}{c} \text{Term-Topic similarity} \\ \left(\begin{array}{ccc} -0.48 & -0.79 & -0.11 \cdot 10^{-14} \\ -0.58 & 0.16 & 0.71 \\ \mathbf{-0.34} & \mathbf{0.56} & 0.42 \cdot 10^{-15} \\ -0.56 & 0.16 & -0.71 \end{array} \right) \\ U \end{array} \times \begin{array}{c} \text{Topic "importance"} \\ \left(\begin{array}{ccc} 2.1 & 0 & 0 \\ 0 & 1.26 & 0 \\ 0 & 0 & 1 \end{array} \right) \\ D \end{array} \\ \times \begin{array}{c} \text{Topic-Document similarity} \\ \left(\begin{array}{ccc} -0.5 & \mathbf{-0.71} & -0.5 \\ -0.5 & \mathbf{0.71} & -0.5 \\ 0.71 & 0.67 \cdot 10^{-15} & -0.711 \end{array} \right) \\ V^T \end{array}$$

Latent Semantic Indexing based on SVD

- LSI considers A_k the rank k approximation of A (i.e. keep only k most relevant topics)
- In the example $k = 2$
- Map a query to k dimensional space with U_k and then apply cosine similarity to find similar documents in $D_k V_k^T$

$$\begin{array}{c} \text{Terms} \end{array} \begin{array}{c} \text{Documents} \\ \begin{pmatrix} 1.0 & 0.01 & 1 \\ 0.51 & 1.01 & 0.51 \\ 0.0 & 1.01 & 0.0 \\ 0.49 & 0.98 & 0.49 \end{pmatrix} \\ A_k \end{array} = \begin{array}{c} \text{Term-Topic similarity} \\ \begin{pmatrix} -0.48 & -0.79 \\ -0.58 & 0.16 \\ \mathbf{-0.34} & \mathbf{0.56} \\ -0.56 & 0.16 \end{pmatrix} \\ U_k \end{array} \times \begin{array}{c} \text{Topic "importance"} \\ \begin{pmatrix} 2.1 & 0 \\ 0 & 1.26 \end{pmatrix} \\ D_k \end{array} \times \begin{array}{c} \text{Topic-Document similarity} \\ \begin{pmatrix} -0.5 & \mathbf{-0.71} & -0.5 \\ -0.5 & \mathbf{0.71} & -0.5 \end{pmatrix} \\ V_k^T \end{array}$$

Latent Semantic Indexing based on SVD

Theorem (Eckart and Young [1])

Among all $n \times m$ matrices C of rank at most k , A_k is the one that minimizes $\|A - C\|_F^2 = \sum_{i,j} (A_{ij} - C_{ij})^2$, where F denotes the Frobenius norm of a matrix.

$$\begin{array}{c} \text{Terms} \end{array} \begin{array}{c} \text{Documents} \\ \begin{pmatrix} 1.0 & 0.01 & 1 \\ 0.51 & 1.01 & 0.51 \\ 0.0 & 1.01 & 0.0 \\ 0.49 & 0.98 & 0.49 \end{pmatrix} \\ A_k \end{array} \approx \begin{array}{c} \text{Terms} \end{array} \begin{array}{c} \text{Documents} \\ \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \\ A \end{array}$$

Original Paper Overview and Emphasized Aspect

- LSI has shown strong empirical results
- Two important aspects
 - Why does LSI find **semantically related** documents?
 - How can we **reduce the computational time** ?
- Papadimitriou et al. [3] investigated both aspects:
 1. Under certain constraints on the term-document matrix, semantically related documents are mapped to **similar vectors**
 2. Instead of LSI use **LSI by random projection**. This reduces the computational time:
 - Map the original term-document matrix into a lower dimensional space
 - Use LSI on the lower dimensional matrix
- In this presentation we focus on the **second** aspect

LSI by Random Projection

- In this section we will investigate the question "How we can speed up the computation": Informal formulation of the main theorem of this section (Theorem 5 original paper)
- Introduction of theorems and lemmas that are necessary for the proof of the main theorem
- Introduction: the main theorem (Theorem 5 original paper)
- Proof of the main theorem (Theorem 5 original paper)
- Computational savings achieved by LSI by random projection

Random Projection for Dimensionality Reduction

Given a matrix $A \in \mathbb{R}^{n \times m}$ and a matrix $R \in \mathbb{R}^{\ell \times n}$. Use matrix R to represent the matrix A in lower dimensional space by preserving pairwise distances between any two points:

$$B = \sqrt{\frac{n}{\ell}} \cdot R^T A \in \mathbb{R}^{\ell \times m}$$

Lemma (Johnson and Lindenstrauss [2])

Let $v \in \mathbb{R}^n$ be a unit vector, let H be a random ℓ -dimensional subspace through the origin, and let the random variable X denote the square of the length of the projection of v onto H . Suppose $0 < \epsilon < 0.5$, and $24 \log n < 1 < \sqrt{n}$. Then, $E[X] = \frac{\ell}{n}$, and

$$Pr(|X - \frac{\ell}{n}| > \epsilon \frac{\ell}{n}) < 2\sqrt{\ell} e^{-(\ell-1)\epsilon^2/4}$$

LSI by Random Projection

Two Step LSI

1. Apply a random projection onto ℓ dimensions, where ℓ is a small value greater than k , on A .

$$B = \sqrt{\frac{n}{\ell}} \cdot \begin{pmatrix} | & | & \dots & | \\ r_1 & r_2 & \dots & r_\ell \\ | & | & \dots & | \end{pmatrix}^T \cdot A$$

2. Apply rank $O(k)$ LSI (because of the random projection, the number of singular values kept may have to be slightly increased).

This leads to an improved running time while preventing the expressiveness of the original matrix.

LSI by Random Projection

Background

References

-  [C. Reinsch J. H. Wilkinson.](#)
Handbook for Automatic Computation.
Springer Berlin, Heidelberg, volume ii: linear algebra edition, 1971.
-  [William B Johnson.](#)
Extensions of lipshitz mapping into hilbert space.
In *Conference modern analysis and probability, 1984*, pages 189–206, 1984.
-  [Christos H. Papadimitriou, Prabhakar Raghavan, Hisao Tamaki, and Santosh Vempala.](#)
Latent semantic indexing: A probabilistic analysis.
Journal of Computer and System Sciences, 61(2):217–235, 2000.
URL: <https://www.sciencedirect.com/science/article/pii/S0022000000917112>, doi:10.1006/jcss.2000.1711.
-  [Gilbert Strang.](#)
Linear Algebra and Its Applications.
Cengage Learning, 4th edition edition, 2005.

References

The End