Matrix Decomposition and Latent Semantic Indexing (LSI)

Introduction to Information Retrieval
INF 141
Donald J. Patterson
Efficient Cosine Ranking

- Find the k docs in the corpus "nearest" to the query
- the k largest query-doc cosines

**CosineScore**($q$)

1. **Initialize**($Scores[d \in D]$)
2. **Initialize**($Magnitude[d \in D]$)
3. for each term($t \in q$)
   4. do $p \leftarrow$ **FetchPostingsList**($t$)
   5. $df_t \leftarrow$ **GetCorpusWideStats**($p$)
   6. $\alpha_{t,q} \leftarrow$ **WeightInQuery**($t, q, df_t$)
   7. for each $(d, t f_{t,d}) \in p$
      8. do $Scores[d] += \alpha_{t,q} \cdot$ **WeightInDocument**($t, q, df_t$)
4. for $d \in Scores$
   5. do **Normalize**($Scores[d], Magnitude[d]$)
6. return $top K \in Scores$
Outline

• Introduction
• Linear Algebra Refresher
Star Cluster NGC 290 - ESA & NASA
Latent Semantic Indexing - Introduction

Star Cluster NGC 290 - ESA & NASA

- A picture of the sky is two dimensional
- The stars are not in two dimensions
- When we take a photo of stars we are projecting them into 2-D
  - projecting can be defined mathematically
- When we see two stars that are close...
  - They may not be close in space
- When we see two stars that appear far...
  - They may not be far in 3-D space
Latent Semantic Indexing - Introduction

Star Cluster NGC 290 - ESA & NASA

- A picture of the sky is two dimensional
- The stars are not in two dimensions
- When we take a photo of stars we are projecting them into 2-D
  - projecting can be defined mathematically
- When we see two stars that are close...
  - They may not be close in space
- When we see two stars that appear far...
  - They may not be far in 3-D space
Star Cluster NGC 290 - ESA & NASA

- When we see two stars that are close in a photo
  - They really are close for some applications
  - For example pointing a big telescope at them
  - Large shared telescopes order their views according to how “close” they are.
Overhead projector example
Overhead projector example

- Depending on where we put the light (and the wall) we can make things in three dimensions appear close or far away in two dimensions.
- Even though the “real” position of the 3-d objects never moved.
Mathematically speaking

• This is taking a 3-D point and projecting it into 2-D

\[
\begin{pmatrix}
10 \\
10 \\
10
\end{pmatrix}
\rightarrow
\begin{pmatrix}
10 \\
10
\end{pmatrix}
\]

• The arrow in this picture acts like the overhead projector
Mathematically speaking

• We can project from any number of dimensions into any other number of dimensions.

• **Increasing** dimensions adds redundant information
  • But sometimes useful
  • Support Vector Machines (kernel methods) do this effectively

• Latent Semantic Indexing always **reduces** the number of dimensions
• Latent Semantic Indexing always reduces the number of dimensions.
Mathematically speaking

- Latent Semantic Indexing always reduces the number of dimensions

\[
\begin{pmatrix}
10 \\
10 \\
\end{pmatrix} \rightarrow \begin{pmatrix}
10 \\
\end{pmatrix}
\]
Mathematically speaking

- Latent Semantic Indexing always reduces the number of dimensions.

\[
\begin{pmatrix}
10 \\
10
\end{pmatrix} \rightarrow \begin{pmatrix}
10
\end{pmatrix}
\]
Mathematically speaking

- Latent Semantic Indexing always reduces the number of dimensions
Mathematically speaking

- Latent Semantic Indexing can project on an arbitrary axis, not just a principal axis.
Mathematically speaking

• Our documents were just points in an N-dimensional term space
• We can project them also
Mathematically speaking

- Latent Semantic Indexing makes the claim that these new axes represent *semantics* - deeper meaning than just a term.
Mathematically speaking
- A term vector that is projected on new vectors may uncover deeper meanings
- For example
  - Transforming the 3 axes of a term matrix from “ball” “bat” and “cave” to
    - An axis that merges “ball” and “bat”
    - An axis that merges “bat” and “cave”
  - Should be able to separate differences in meaning of the term “bat”
- Bonus: less dimensions is faster
Let C be an M by N matrix with real-valued entries. For example, our term document matrix. A matrix with the same number of rows and columns is called a square matrix. An M by M matrix with elements only on the diagonal is called a diagonal matrix. The identity matrix is a diagonal matrix with ones on the main diagonal.
Linear Algebra Refresher

• Let C be an M by N matrix with real-valued entries, for example our term document matrix.

• A matrix with the same number of rows and columns is called a **square matrix**.

• An M by M matrix with elements only on the diagonal is called a **diagonal matrix**.

• The **identity matrix** is a diagonal matrix with ones on the main diagonal.
Linear Algebra Refresher

• Let \( C \) be an \( M \) by \( N \) matrix with real-valued entries
  
  • for example our term document matrix

• A matrix with the same number of rows and columns is called a square matrix

• An \( M \) by \( M \) matrix with elements only on the diagonal is called a diagonal matrix

• The identity matrix is a diagonal matrix with ones on the main diagonal

\[
C = \begin{bmatrix}
1 & 2 & 3 & 4 & 5 \\
0 & 2 & 3 & 2 & 1 \\
1 & 0 & 0 & 1 & 1
\end{bmatrix}
\]
Linear Algebra Refresher

• Let $C$ be an $M$ by $N$ matrix with real-valued entries, for example our term document matrix.

• A matrix with the same number of rows and columns is called a square matrix.

• An $M$ by $M$ matrix with elements only on the diagonal is called a diagonal matrix.

• The identity matrix is a diagonal matrix with ones on the main diagonal.

\[
\begin{bmatrix}
1 & 2 & 3 & 4 & 5 \\
0 & 2 & 3 & 2 & 1 \\
1 & 0 & 0 & 1 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 2 & 3 \\
0 & 2 & 3 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
5 & 0 & 0 & 0 & 0 \\
0 & 2 & 0 & 0 & 0 \\
0 & 0 & 5 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 7 & 0 \\
0 & 0 & 0 & 0 & 2 \\
\end{bmatrix}
\]
Let C be an M by N matrix with real-valued entries, for example our term document matrix. A matrix with the same number of rows and columns is called a square matrix. An M by M matrix with elements only on the diagonal is called a diagonal matrix. The identity matrix is a diagonal matrix with ones on the main diagonal.
Matrix Decomposition

• Singular Value Decomposition
  • Splits a matrix into three matrices
  • Such that
    • If
    • then
    • and
    • and
    • also Sigma is almost a diagonal matrix
Matrix Decomposition

• Singular Value Decomposition
  • Splits a matrix into three matrices
  • Such that
    • If
    • then
    • and
    • and
    • also Sigma is almost a diagonal matrix
Matrix Decomposition

• Singular Value Decomposition
  • Splits a matrix into three matrices
  • Such that
    • If
    • then
    • and
    • and
    • also Sigma is almost a diagonal matrix
Matrix Decomposition

- Singular Value Decomposition
  - Splits a matrix into three matrices
  - Such that
    - If
    - then
    - and
    - and
    - also Sigma is almost a diagonal matrix

\[ C = U \Sigma V^T \]

\[ C \text{ is } (M \text{ by } N) \]
Matrix Decomposition

- Singular Value Decomposition

- Splits a matrix into three matrices

- Such that

- If

- then

- and

- and

- also Sigma is almost a diagonal matrix

\[ C = U \Sigma V^T \]

\[ C \text{ is } (M \text{ by } N) \]

\[ U \text{ is } (M \text{ by } M) \]
Matrix Decomposition

• Singular Value Decomposition
  • Splits a matrix into three matrices
  • Such that
    • If
    • then
    • and
    • and
  • also Sigma is almost a diagonal matrix

\[ C = U\Sigma V^T \]
\[ U \text{ is (M by M)} \]
\[ \Sigma \text{ is (M by N)} \]
Matrix Decomposition

• Singular Value Decomposition
  • Splits a matrix into three matrices
  • Such that
    • If
    • then
    • and
    • and
  • also Sigma is almost a diagonal matrix

\[ C = U \Sigma V^T \]

\[ U \text{ is (} M \text{ by } M \text{)} \]

\[ \Sigma \text{ is (} M \text{ by } N \text{)} \]

\[ V^T \text{ is (} N \text{ by } N \text{)} \]
Matrix Decomposition

\[ C = U \Sigma V^T \]
Matrix Decomposition

- Singular Value Decomposition
  - Is a technique that splits a matrix into three components with these properties.
  - They also have some other properties which are relevant to latent semantic indexing
Matrix Decomposition

- Singular Value Decomposition

• Is a technique that splits a matrix into three components with these properties.

\[ C = U \Sigma V^T \]
Matrix Decomposition

• Singular Value Decomposition
  • SVD enables lossy compression of your term-document matrix
  • reduces the **dimensionality** or the **rank**
  • you can arbitrarily reduce the dimensionality by putting zeros in the bottom right of sigma
  • this is a mathematically optimal way of reducing dimensions

\[ U \quad \Sigma \quad V^T \]
Matrix Decomposition

- Singular Value Decomposition

- If the old dimensions were based on terms

- after reducing the rank of the matrix the dimensionality is based on concepts or semantics

- a concept is a linear combination of terms

\[
SV D_{dimension_1} = a \cdot td_{dim_1} + b \cdot td_{dim_2} + c \cdot td_{dim_3} + d \cdot td_{dim_4}
\]

\[
SV D_{dimension_2} = a' \cdot td_{dim_1} + b' \cdot td_{dim_2} + c' \cdot td_{dim_3} + d' \cdot td_{dim_4}
\]

\[
SV D_{dimension_3} = a'' \cdot td_{dim_1} + b'' \cdot td_{dim_2} + c'' \cdot td_{dim_3} + d'' \cdot td_{dim_4}
\]
Matrix Decomposition

• Singular Value Decomposition

\[ SV D_{\text{dimension}_1} = a \times td_{\text{dim}_1} + b \times td_{\text{dim}_2} + c \times td_{\text{dim}_3} + d \times td_{\text{dim}_4} \]

\[ SV D_{\text{dimension}_2} = a' \times td_{\text{dim}_1} + b' \times td_{\text{dim}_2} + c' \times td_{\text{dim}_3} + d' \times td_{\text{dim}_4} \]

\[ SV D_{\text{dimension}_3} = a'' \times td_{\text{dim}_1} + b'' \times td_{\text{dim}_2} + c'' \times td_{\text{dim}_3} + d'' \times td_{\text{dim}_4} \]

• 4 dimensions to 3 dimensions

\[
\begin{pmatrix}
  a & b & c & d \\
  a' & b' & c' & d' \\
  a'' & b'' & c'' & d'' \\
\end{pmatrix}
\]
Matrix Decomposition

• Singular Value Decomposition

\[ SVD_{dimension_1} = a \times td_{dim_1} + b \times td_{dim_2} + c \times td_{dim_3} + d \times td_{dim_4} \]

\[ SVD_{dimension_2} = a' \times td_{dim_1} + b' \times td_{dim_2} + c' \times td_{dim_3} + d' \times td_{dim_4} \]

\[ SVD_{dimension_3} = a'' \times td_{dim_1} + b'' \times td_{dim_2} + c'' \times td_{dim_3} + d'' \times td_{dim_4} \]

• 4 dimensions to 3 dimensions

\[
\begin{bmatrix}
  a & b & c & d \\
  a' & b' & c' & d' \\
  a'' & b'' & c'' & d''
\end{bmatrix}
\times
\begin{bmatrix}
  td_{dim_1} \\
  td_{dim_2} \\
  td_{dim_3} \\
  td_{dim_4}
\end{bmatrix}
\]
Matrix Decomposition

- Singular Value Decomposition

\[ SV D_{\text{dimension}_1} = a \times td_{\text{dim}_1} + b \times td_{\text{dim}_2} + c \times td_{\text{dim}_3} + d \times td_{\text{dim}_4} \]

\[ SV D_{\text{dimension}_2} = a' \times td_{\text{dim}_1} + b' \times td_{\text{dim}_2} + c' \times td_{\text{dim}_3} + d' \times td_{\text{dim}_4} \]

\[ SV D_{\text{dimension}_3} = a'' \times td_{\text{dim}_1} + b'' \times td_{\text{dim}_2} + c'' \times td_{\text{dim}_3} + d'' \times td_{\text{dim}_4} \]

\[
\begin{array}{cccc}
SV D_{\text{dim}_1} & = & a & b & c & d \\
SV D_{\text{dim}_2} & = & a' & b' & c' & d' \\
SV D_{\text{dim}_3} & = & a'' & b'' & c'' & d'' \\
\end{array}
\times
\begin{array}{c}
TD_{\text{dim}_1} \\
TD_{\text{dim}_2} \\
TD_{\text{dim}_3} \\
TD_{\text{dim}_4} \\
\end{array}\]
Matrix Decomposition

- Singular Value Decomposition

\[ SV \text{D}_{\text{dimension}_1} = a \ast t_{d_{\text{dim}_1}} + b \ast t_{d_{\text{dim}_2}} + c \ast t_{d_{\text{dim}_3}} + d \ast t_{d_{\text{dim}_4}} \]

\[ SV \text{D}_{\text{dimension}_2} = a' \ast t_{d_{\text{dim}_1}} + b' \ast t_{d_{\text{dim}_2}} + c' \ast t_{d_{\text{dim}_3}} + d' \ast t_{d_{\text{dim}_4}} \]

\[ SV \text{D}_{\text{dimension}_3} = a'' \ast t_{d_{\text{dim}_1}} + b'' \ast t_{d_{\text{dim}_2}} + c'' \ast t_{d_{\text{dim}_3}} + d'' \ast t_{d_{\text{dim}_4}} \]

\[
\begin{array}{c|cccc}
SV \text{D}_{\text{dim}_1} & | & a & b & c & d \\
SV \text{D}_{\text{dim}_2} & | & a' & b' & c' & d' \\
SV \text{D}_{\text{dim}_3} & | & a'' & b'' & c'' & d'' \\
\end{array}
\ast
\begin{array}{c|cccc}
t_{d_{\text{dim}_1}} \\
t_{d_{\text{dim}_2}} \\
t_{d_{\text{dim}_3}} \\
t_{d_{\text{dim}_4}} \\
\end{array}
\]

\[ SV \text{D}_{\text{ConceptSpace}} = M \ast \text{query} \ast \text{TermSpace} \]
Matrix Decomposition

- **Singular Value Decomposition**

\[
\begin{align*}
SV D_{dim_1} & = \begin{pmatrix} a & b & c & d \\ a' & b' & c' & d' \\ a'' & b'' & c'' & d'' \end{pmatrix} \\
SV D_{dim_2} & = \begin{pmatrix} t_{d_{dim_1}} \\ t_{d_{dim_2}} \\ t_{d_{dim_3}} \\ t_{d_{dim_4}} \end{pmatrix}
\end{align*}
\]

\[
SV D_{ConceptSpace} = M \ast query_{TermSpace}
\]

\[
\begin{align*}
c & = u \Sigma v^T
\end{align*}
\]
Matrix Decomposition

- Singular Value Decomposition

\[
\begin{bmatrix}
SV D_{dim_1} \\
SV D_{dim_2} \\
SV D_{dim_3}
\end{bmatrix}
= 
\begin{bmatrix}
a & b & c & d \\
a' & b' & c' & d' \\
a'' & b'' & c'' & d''
\end{bmatrix}
\times
\begin{bmatrix}
td_{dim_1} \\
td_{dim_2} \\
td_{dim_3} \\
td_{dim_4}
\end{bmatrix}
\]

\[
SVD_{ConceptSpace} = M \times query_{TermSpace}
\]

\[
M = \Sigma_k^{-1} U_k^T
\]

\[
c = u \Sigma v^T
\]
Matrix Decomposition

• Singular Value Decomposition

\[
\begin{align*}
SV D_{\text{dim}_1} & = \begin{bmatrix} a & b & c & d \\ a' & b' & c' & d' \\ a'' & b'' & c'' & d'' \end{bmatrix} & \begin{bmatrix} td_{\text{dim}_1} \\ td_{\text{dim}_2} \\ td_{\text{dim}_3} \\ td_{\text{dim}_4} \end{bmatrix} \\
SV D_{\text{dim}_2} & = \begin{bmatrix} a & b & c & d \\ a' & b' & c' & d' \\ a'' & b'' & c'' & d'' \end{bmatrix} \\
SV D_{\text{dim}_3} & = \begin{bmatrix} a & b & c & d \\ a' & b' & c' & d' \\ a'' & b'' & c'' & d'' \end{bmatrix}
\end{align*}
\]

\[
SV D_{\text{ConceptSpace}} = M \ast \text{query}_{\text{TermSpace}}
\]

\[
M = \sum_{k}^{-1} U_{k}^{T}
\]

\[
\text{query}_{\text{ConceptSpace}} = \sum_{k}^{-1} U_{k}^{T} \text{query}_{\text{TermSpace}}
\]
Matrix Decomposition

- Singular Value Decomposition
- SVD is an algorithm that gives us $\Sigma U V^T$
- With these quantities we can reduce dimensionality
- With reduced dimensionality
  - synonyms are mapped onto the same location
    - “bat” “chiroptera”
  - polysemyies are mapped onto different locations
    - “bat” (baseball) vs. “bat” (small furry mammal)
Latent Semantic Indexing - Linear Algebra Refresher

• Computing SVD takes a significant amount of CPU
• It is possible to add documents to a corpus without recalculating SVD
  • The result becomes an approximation
• To get mathematical guarantees the whole SVD needs to be computed from scratch
• LSI doesn’t support negation queries
• LSI doesn’t support boolean queries
Matrix Decomposition

- “I am not crazy”
- Netflix
Matrix Decomposition

- “I am not crazy”
- Netflix
Matrix Decomposition

• “I am not crazy”

• Netflix

• Machine translations
  • Just like “bat” and “chiroptera” map the same
  • “bat” and “murciélago” can map to the same thing
Matrix Decomposition

• “I am not crazy”
• Netflix
• Machine translations
  • Just like “bat” and “chiroptera” map the same
  • “bat” and “murciélago” can map to the same thing

The math is hard but it's beautiful and powerful
Matrix Decomposition

- “I am not crazy”
- Netflix
- Machine translations
  - Just like “bat” and “chiroptera” map the same
  - “bat” and “murciélago” can map to the same thing

The math is hard but it's beautiful and powerful

La matemáticas es dura pero es hermosa y de gran alcance
Matrix Decomposition

- “I am not crazy”
- Netflix
- Machine translations
  - Just like “bat” and “chiroptera” map the same
  - “bat” and “murciélago” can map to the same thing

The math is hard but it's beautiful and powerful
La matemáticas es dura pero es hermosa y de gran alcance
Jene mathematisch ist hart, aber ist und an langer Reichweite schön
Matrix Decomposition

• “I am not crazy”

• Netflix

• Machine translations
  
  • Just like “bat” and “chiroptera” map the same
  
  • “bat” and “murciélago” can map to the same thing

The math is hard but it's beautiful and powerful

La matemáticas es dura pero es hermosa y de gran alcance

Jene mathematisch ist hart, aber ist und an langer Reichweite schön

That one mathematically is hard, but is beautiful and at long range