# Matrix Factorization

Hamburg Coding School

*lasse.schmidt@bonprix.net*

February 9, 2019

# Overview

# Introduction

Matrix Factorization is a technique to reduce dimensionality of the data. The basic idea is that we can rebuild a basis matrix $D$ by the dot product of two (or more) matrices $W$ and $H$ with lower dimensionality.

These matrices $W$ and $H$ are also called "embeddings". These embeddings are supposed to represent the information from matrix $D$ in a more compact way.

# Non-Negative Matrix Factorization (NMF)

NMF is a dimensionality reduction technique tailored to clustering:

- suitable for non-negative and sparse data
- hence often used for text mining
- matrices $W$ and $H$ do not contain negative values
- this makes the approach highly interpretable

# Non-Negative Matrix Factorization (NMF)

$$Formula : D \approx W \cdot H^T \tag{1}$$

where $W$ is the embedding for the rows

and $H$ is the embedding for the columns

# Non-Negative Matrix Factorization (NMF)

Matrix $D$

| | red dress | green dress | red socks | green socks | black skirt | blue jeans |
|---|---|---|---|---|---|---|
| customer 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| customer 2 | 1 | 0 | 1 | 0 | 0 | 0 |
| customer 3 | 0 | 1 | 0 | 1 | 1 | 0 |
| customer 4 | 0 | 1 | 0 | 1 | 0 | 0 |
| customer 5 | 0 | 0 | 1 | 0 | 0 | 1 |
| customer 6 | 0 | 0 | 0 | 1 | 0 | 1 |

# Non-Negative Matrix Factorization (NMF)

Matrix $W$

| | latent factor 1 | latent factor 2 |
|---|---|---|
| customer 1 | 0.06163281 | 1.00645414 |
| customer 2 | 0 | 0.82291355 |
| customer 3 | 0.92051861 | 0.06733389 |
| customer 4 | 0.75260906 | 0 |
| customer 5 | 0.03490373 | 0.57124949 |
| customer 6 | 0.52232672 | 0.03827449 |

# Non-Negative Matrix Factorization (NMF)

Matrix $H$

| | red dress | green dress | red socks | green socks | black skirt | blue jeans |
|---|---|---|---|---|---|---|
| latent factor 1 | 0 | 0.98907004 | 0 | 1.29784367 | 0.53333592 | 0.30257195 |
| latent factor 2 | 0.90452538 | 0 | 1.18697799 | 0 | 0.48769841 | 0.27685045 |

# Non-Negative Matrix Factorization (NMF)

Matrix $D \approx W \cdot H^T$

| | red dress | green dress | red socks | green socks | black skirt | blue jeans |
|---|---|---|---|---|---|---|
| customer 1 | 0.910 | 0.060 | 1.194 | 0.079 | 0.523 | 0.297 |
| customer 2 | 0.744 | 0 | 0.976 | 0 | 0.401 | 0.227 |
| customer 3 | 0.060 | 0.910 | 0.079 | 1.194 | 0.523 | 0.297 |
| customer 4 | 0 | 0.744 | 0 | 0.976 | 0.401 | 0.227 |
| customer 5 | 0.516 | 0.034 | 0.678 | 0.045 | 0.297 | 0.168 |
| customer 6 | 0.034 | 0.516 | 0.045 | 0.677 | 0.297 | 0.168 |

# Non-Negative Matrix Factorization (NMF)

For comparison: Matrix $D$

| | red dress | green dress | red socks | green socks | black skirt | blue jeans |
|---|---|---|---|---|---|---|
| customer 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| customer 2 | 1 | 0 | 1 | 0 | 0 | 0 |
| customer 3 | 0 | 1 | 0 | 1 | 1 | 0 |
| customer 4 | 0 | 1 | 0 | 1 | 0 | 0 |
| customer 5 | 0 | 0 | 1 | 0 | 0 | 1 |
| customer 6 | 0 | 0 | 0 | 1 | 0 | 1 |

# Non-Negative Matrix Factorization (NMF)

Interpretation of $W$

| | latent factor 1 "green" | latent factor 2 "red" |
|---|---|---|
| customer 1 | 0.06163281 | 1.00645414 |
| customer 2 | 0 | 0.82291355 |
| customer 3 | 0.92051861 | 0.06733389 |
| customer 4 | 0.75260906 | 0 |
| customer 5 | 0.03490373 | 0.57124949 |
| customer 6 | 0.52232672 | 0.03827449 |

each column of the costumer embedding represents how much that customer appreciates a "concept" (here: red or green clothes)

# Non-Negative Matrix Factorization (NMF)

How does the NMF compute $W$ and $H$?

Optimization function:

$$J = \frac{1}{2}||D - UV^T||^2 \tag{2}$$

which is the squared Frobenius Norm (like the Euclidean Distance for matrices)

$\Rightarrow$ solving is done via Kuhn-Tucker

# Singular Value Decomposition (SVD)

- closely related to PCA, but more general because it provides two sets of basis vectors instead of one $\Rightarrow$ for the rows and the columns of the data matrix (PCA does it only for the rows)
- SVD and PCA provide the same basis vectors for the rows if the mean of each attribute is 0
- SVD often applied without mean centering to sparse, non-negative data

# Singular Value Decomposition (SVD)

$$Formula : D = Q\Sigma P^T \tag{3}$$

where Q are the left singular vectors (n$x$n for the rows)

$\Sigma$ is the diagonal matrix containing the singular values (always non-negative)

and P are the right singular values (d$x$d for the columns)

# Singular Value Decomposition (SVD)

With this formula, we still have three (possibly) large matrices. To reduce dimensionality, we can order $\Sigma$ in decreasing order (P and Q are arranged accordingly) and only use truncated d*x*k and n*x*k matrices for $Q_k$, $\Sigma_k$ and $P_k$

$$Formula : D \approx Q_k \Sigma_k P_k^T \tag{4}$$

where $Q_k \Sigma_k$ correspond to $W$ and $P$ corresponds to $H$

# Singular Value Decomposition (SVD)

Matrix $D$

| | red dress | green dress | red socks | green socks | black skirt | blue jeans |
|---|---|---|---|---|---|---|
| customer 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| customer 2 | 1 | 0 | 1 | 0 | 0 | 0 |
| customer 3 | 0 | 1 | 0 | 1 | 1 | 0 |
| customer 4 | 0 | 1 | 0 | 1 | 0 | 0 |
| customer 5 | 0 | 0 | 1 | 0 | 0 | 1 |
| customer 6 | 0 | 0 | 0 | 1 | 0 | 1 |

# Singular Value Decomposition (SVD)

Matrix $Q_k \Sigma_k$

| | latent factor 1 | latent factor 2 |
|---|---|---|
| customer 1 | -0.99250764 | 1.29205525 |
| customer 2 | -0.99250746 | 0.85370666 |
| customer 3 | 0.9925074 | 1.29205537 |
| customer 4 | 0.99250787 | 0.8537069 |
| customer 5 | -0.55734551 | 0.74116856 |
| customer 6 | 0.55734533 | 0.74116874 |

# Singular Value Decomposition (SVD)

Matrix $P$

| | red dress | green dress | red socks | green socks | black skirt | blue jeans |
|---|---|---|---|---|---|---|
| latent factor 1 | -0.43 | -0.435 | -0.57 | 0.57 | 0 | 0 |
| latent factor 2 | 0.34 | 0.34 | 0.49 | 0.49 | 0.44 | 0.25 |

# Singular Value Decomposition (SVD)

Matrix $D \approx Q_k \Sigma_k P_k^T$

| | red dress | green dress | red socks | green socks | black skirt | blue jeans |
|---|---|---|---|---|---|---|
| customer 1 | 0.902 | 0.038 | 1.186 | 0.08 | 0.566 | 0.325 |
| customer 2 | 0.743 | -0.121 | 0.971 | -0.135 | 0.374 | 0.215 |
| customer 3 | 0.039 | 0.902 | 0.080 | 1.186 | 0.566 | 0.325 |
| customer 4 | -0.121 | 0.743 | -0.135 | 0.971 | 0.374 | 0.215 |
| customer 5 | 0.512 | 0.027 | 0.674 | 0.052 | 0.325 | 0.186 |
| customer 6 | 0.027 | 0.512 | 0.052 | 0.674 | 0.324 | 0.186 |

# Singular Value Decomposition (SVD)

For comparison: Matrix $D$

| | red dress | green dress | red socks | green socks | black skirt | blue jeans |
|---|---|---|---|---|---|---|
| customer 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| customer 2 | 1 | 0 | 1 | 0 | 0 | 0 |
| customer 3 | 0 | 1 | 0 | 1 | 1 | 0 |
| customer 4 | 0 | 1 | 0 | 1 | 0 | 0 |
| customer 5 | 0 | 0 | 1 | 0 | 0 | 1 |
| customer 6 | 0 | 0 | 0 | 1 | 0 | 1 |

# Comparison between NMF and SVD

Approximation of Matrix D

Matrix Decomposition:

$$NMF : D \approx W \cdot H^T \tag{5}$$

$$SVD : D \approx Q_k \cdot \Sigma_k \cdot P_k^T \tag{6}$$

# Comparison between NMF and SVD

- both minimize the squared error of the data representation
- main difference: different constraints in the corresponding optimization formula
- SVD imposes orthogonality; NMF imposes non-negativity

# Comparison between NMF and SVD

1. latent factors of NMF easily interpretable for clustering $\rightarrow$ in text domain each factor of W stands for a document cluster and how closely it is related to that document
2. unlike SVD, the $k$ latent factors of NMF are not orthogonal to one another. This is a disadvantage because orthogonality of the axis-system allows intuitive interpretation of data transformation as axis-rotation $\rightarrow$ distance computations between transformed data points more meaningful in SVD
3. non-negativity constraint reduces quality of optimization problem $\rightarrow$ SVD provides better rank-k approximations than NMF
4. SVD harder to implement for incomplete data compared to NMF

# Other Applications of NMF and SVD

1. Noise reduction: while removal of smaller eigenvectors/singular vectors can lead to information loss, it can also lead to an improvement in the quality of data representation $\rightarrow$ can lead to quality benefits
2. Data imputation: collaborative filtering $\rightarrow$ we can estimate values for sparse data

# The End