The Nonnegative Matrix Factorization in Data Mining

Amy Langville
langvillea@cofc.edu

Mathematics Department
College of Charleston
Charleston, SC

Yahoo! Research
10/18/2005
Outline

Part 1: Historical Developments in Data Mining
- Vector Space Model (1960s-1970s)
- Latent Semantic Indexing (1990s)
- Other VSM decompositions (1990s)

- Applications in Image and Text Mining
- Algorithms
- Current and Future Work
Vector Space Model (1960s and 1970s)

Gerard Salton’s Information Retrieval System
SMART: System for the Mechanical Analysis and Retrieval of Text
(Salton’s Magical Automatic Retriever of Text)

- turn $n$ textual documents into $n$ document vectors $d_1, d_2, \ldots, d_n$
- create term-by-document matrix $A_{m \times n} = [d_1 | d_2 | \cdots | d_n ]$
- to retrieve info., create query vector $q$, which is a pseudo-doc
Vector Space Model (1960s and 1970s)

Gerard Salton’s Information Retrieval System
SMART: System for the Mechanical Analysis and Retrieval of Text
(Salton’s Magical Automatic Retriever of Text)

- turn $n$ textual documents into $n$ document vectors $d_1, d_2, \ldots, d_n$
- create term-by-document matrix $A_{m \times n} = [d_1 | d_2 | \cdots | d_n]$
- to retrieve info., create query vector $q$, which is a pseudo-doc

**GOAL:** find doc. $d_i$ closest to $q$

- *angular cosine* measure used: $\delta_i = \cos \theta_i = q^T d_i / (\|q\|_2 \|d_i\|_2)$
Latent Semantic Indexing (1990s)

Susan Dumais’s improvement to VSM = LSI

Idea: use low-rank approximation to $A$ to filter out noise

$A_{m \times n}$: rank $r$ term-by-document matrix

- SVD: $A = U \Sigma V^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T$

- LSI: use $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ in place of $A$

- Why?
  
  — reduce storage when $k << r$
  
  — filter out uncertainty, so that performance on text mining tasks (e.g., query processing and clustering) improves
Properties of SVD

- Basis vectors $u_i$ are orthogonal

- $u_{ij}, v_{ij}$ are mixed in sign

\[
A_k = U_k \Sigma_k V_k^T
\]

- $U, V$ are dense

- Uniqueness—while there are many SVD algorithms, they all create the same (truncated) factorization

- Of all rank-$k$ approximations, $A_k$ is optimal (in Frobenius norm)

\[
\|A - A_k\|_F = \min_{\text{rank}(B) \leq k} \|A - B\|_F
\]
Strengths and Weaknesses of LSI

Strengths

- using $A_k$ in place of $A$ gives improved performance
- dimension reduction considers only essential components of term-by-document matrix, filters out noise
- best rank-$k$ approximation

Weaknesses

- storage—$U_k$ and $V_k$ are usually completely dense
- interpretation of basis vectors $u_i$ is impossible due to mixed signs
- good truncation point $k$ is hard to determine
- orthogonality restriction
Other Low-Rank Approximations

- QR decomposition
- any $\text{URV}^T$ factorization
- Semidiscrete decomposition (SDD)

$$A_k = X_k D_k Y_k^T,$$

where $D_k$ is diagonal, and elements of $X_k, Y_k \in \{-1, 0, 1\}$. 
Other Low-Rank Approximations

- QR decomposition
- any $\text{URV}^T$ factorization
- Semidiscrete decomposition (SDD)

\[ A_k = X_k D_k Y_k^T, \] where $D_k$ is diagonal, and elements of $X_k, Y_k \in \{-1, 0, 1\}$.

**BUT**

All create basis vectors that are mixed in sign. **Negative** elements make interpretation difficult.
The Power of Positivity

• Positive anything is better than negative nothing.—Elbert Hubbard

• It takes but one positive thought when given a chance to survive and thrive to overpower an entire army of negative thoughts.—Robert H. Schuller

• Learn to think like a winner. Think positive and visualize your strengths.—Vic Braden

• Positive thinking will let you do everything better than negative thinking will.—Zig Ziglar
The Power of Nonnegativity

- **Nonnegative** anything is better than negative nothing.—Elbert Hubbard

- It takes but one **nonnegative** thought when given a chance to survive and thrive to overpower an entire army of negative thoughts.—Robert H. Schuller

- Learn to think like a winner. Think **nonnegative** and visualize your strengths.—Vic Braden

- **Nonnegative** thinking will let you do everything better than negative thinking will.—Zig Ziglar
Nonnegative Matrix Factorization (2000)

Daniel Lee and Sebastian Seung's Nonnegative Matrix Factorization

Idea: use low-rank approximation with nonnegative factors to improve LSI

\[ \mathbf{A}_k = \mathbf{U}_k \Sigma_k \mathbf{V}^T_k \]

\[ \mathbf{A}_k = \mathbf{W}_k \mathbf{H}_k \]

\( \text{nonneg} \quad \text{mixed} \quad \text{nonneg} \quad \text{mixed} \)

\( \text{nonneg} \quad \text{nonneg} \quad \text{nonneg} \)
Interpretation with NMF

- columns of $\mathbf{W}$ are the underlying basis vectors, i.e., each of the $n$ columns of $\mathbf{A}$ can be built from $k$ columns of $\mathbf{W}$.

- columns of $\mathbf{H}$ give the weights associated with each basis vector.

$$\mathbf{A}_k \mathbf{e}_1 = \mathbf{W}_k \mathbf{H} \ast 1 = \begin{bmatrix} \vdots \\ \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_k \end{bmatrix} \mathbf{h}_{11} + \begin{bmatrix} \vdots \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_k \end{bmatrix} \mathbf{h}_{21} + \cdots + \begin{bmatrix} \vdots \\ \mathbf{w}_k \end{bmatrix} \mathbf{h}_{k1}$$

- $\mathbf{W}_k, \mathbf{H}_k \geq 0 \Rightarrow$ immediate interpretation (additive parts-based rep.)
Image Mining

NMF

\[
\begin{align*}
W & \times \quad \text{Original}
\end{align*}
\]

SVD

\[
\begin{align*}
U & \times \quad \Sigma V_i \\
& = \quad \text{Original}
\end{align*}
\]
Image Mining Applications

- Data compression
- Find similar images
- Cluster images
Text Mining
MED dataset ($k = 10$)

Highest Weighted Terms in Basis Vector $W_1$

1. ventricle
2. aortic
3. septal
4. left
5. defect
6. regurgitation
7. ventricle
8. valve
9. cardiac
10. pressure

Highest Weighted Terms in Basis Vector $W_2$

1. oxygen
2. flow
3. pressure
4. blood
5. cerebral
6. hypothermia
7. fluid
8. venous
9. arterial
10. perfusion

Highest Weighted Terms in Basis Vector $W_5$

1. children
2. child
3. autistic
4. speech
5. group
6. early
7. visual
8. anxiety
9. emotional
10. autism

Highest Weighted Terms in Basis Vector $W_6$

1. kidney
2. marrow
3. dna
4. cells
5. nephrectomy
6. unilateral
7. lymphocyte
8. bone
9. thymidine
10. rats
**Text Mining**

<table>
<thead>
<tr>
<th>court</th>
<th>president</th>
</tr>
</thead>
<tbody>
<tr>
<td>government</td>
<td>served</td>
</tr>
<tr>
<td>council</td>
<td>governor</td>
</tr>
<tr>
<td>supreme</td>
<td>secretary</td>
</tr>
<tr>
<td>constitutional</td>
<td>congress</td>
</tr>
<tr>
<td>rights</td>
<td>presidential</td>
</tr>
<tr>
<td>justice</td>
<td>elected</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>flowers</th>
<th>disease</th>
</tr>
</thead>
<tbody>
<tr>
<td>leaves</td>
<td>behaviour</td>
</tr>
<tr>
<td>plant</td>
<td>glands</td>
</tr>
<tr>
<td>perennial</td>
<td>contact</td>
</tr>
<tr>
<td>flower</td>
<td>symptoms</td>
</tr>
<tr>
<td>plants</td>
<td>skin</td>
</tr>
<tr>
<td>growing</td>
<td>pain</td>
</tr>
<tr>
<td>annual</td>
<td>infection</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>metal</th>
<th>process</th>
<th>method</th>
<th>paper</th>
<th>glass</th>
<th>copper</th>
<th>lead</th>
<th>steel</th>
</tr>
</thead>
</table>

| person | example | time | people | rules | lead | leads | law |

- polysems broken across several basis vectors $w_i$
Text Mining Applications

- Data compression
- Find similar terms
- Find similar documents
- Cluster documents
- Topic detection and tracking
Text Mining Applications
Enron email messages 2001

<table>
<thead>
<tr>
<th>Feature Index (k)</th>
<th>Cluster Size</th>
<th>Topic Description</th>
<th>Dominant Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>497</td>
<td>California</td>
<td>ca, cpuc, gov, socalgas, sempra, org, sce, gmssr, aedaw, ci</td>
</tr>
<tr>
<td>23</td>
<td>43</td>
<td>Louise Kitchen named top woman by Fortune</td>
<td>evp, fortune, britain, woman, ceo, avon, fiorina, cfo, hewlett, packard</td>
</tr>
<tr>
<td>26</td>
<td>231</td>
<td>Fantasy football</td>
<td>game, wr, qb, play, rb, season, injury, updated, fantasy, image</td>
</tr>
<tr>
<td>33</td>
<td>233</td>
<td>Texas longhorn football newsletter</td>
<td>UT, orange, longhorn[s], texas, true, truorange, recruiting, oklahoma, defensive</td>
</tr>
<tr>
<td>34</td>
<td>65</td>
<td>Enron collapse</td>
<td>partnership[s], fastow, shares, sec, stock, shareholder, investors, equity, lay</td>
</tr>
<tr>
<td>39</td>
<td>235</td>
<td>Emails about India</td>
<td>dahhol, dpc, india, mseb, maharashtra, indian, lenders, delhi, foreign, minister</td>
</tr>
<tr>
<td>46</td>
<td>127</td>
<td>Enron collapse</td>
<td>dow, debt, reserved, wall, copyright jones, cents, analysts, reuters, spokesman</td>
</tr>
</tbody>
</table>
Recommendation Systems

A purchase history matrix $A = \begin{pmatrix}
\text{User 1} & \text{User 2} & \ldots & \text{User n} \\
\text{Item 1} & 1 & 5 & \ldots & 0 \\
\text{Item 2} & 0 & 0 & \ldots & 1 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\text{Item m} & 0 & 1 & \ldots & 2
\end{pmatrix}

- Create profiles for classes of users from basis vectors $w_i$
- Find similar users
- Find similar items
Properties of NMF

- basis vectors $w_i$ are not $\perp \Rightarrow$ can have overlap of topics
- can restrict $W$, $H$ to be sparse
- $W_k$, $H_k \geq 0 \Rightarrow$ immediate interpretation (additive parts-based rep.)

**EX:** large $w_{ij}$’s $\Rightarrow$ basis vector $w_i$ is mostly about terms $j$

**EX:** $h_{i1}$ how much $doc_1$ is pointing in the “direction” of topic vector $w_i$

$$A_k e_1 = W_k H_1 = \begin{bmatrix} \vdots \\ w_1 \\ \vdots \end{bmatrix} h_{11} + \begin{bmatrix} \vdots \\ w_2 \\ \vdots \end{bmatrix} h_{21} + \cdots + \begin{bmatrix} \vdots \\ w_k \\ \vdots \end{bmatrix} h_{k1}$$

- NMF is algorithm-dependent: $W$, $H$ not unique
Computation of NMF  
(Lee and Seung 2000)

Mean squared error objective function

\[
\min \|A - WH\|_F^2 \quad s.t. \quad W, H \geq 0
\]

Nonlinear Optimization Problem

— convex in \( W \) or \( H \), but not both \( \Rightarrow \) tough to get global min

— huge # unknowns: \( mk \) for \( W \) and \( kn \) for \( H \)
  
  (EX: \( A_{70K \times 1K} \) and \( k=10 \) topics \( \Rightarrow \) 800K unknowns)

— above objective is one of many possible

— convergence to local min NOT guaranteed for any algorithm
NMF Algorithms

- Multiplicative update rules
  - Lee-Seung 2000
  - Hoyer 2002

- Gradient Descent
  - Hoyer 2004
  - Berry-Plemmons 2004

- Alternating Least Squares
  - Paatero 1994
  - ACLS
  - AHCLLS
NMF Algorithm: Lee and Seung 2000

Mean Squared Error objective function

\[
\min_{W,H} \|A - WH\|_F^2 \\
\text{s.t. } W, H \geq 0
\]

\[
W = \text{abs}(\text{randn}(m,k)); \\
H = \text{abs}(\text{randn}(k,n)); \\
\text{for } i = 1 : \text{maxiter} \\
\quad H = H .* \left( \frac{W^T A}{W^T WH + 10^{-9}} \right); \\
\quad W = W .* \left( \frac{A H^T}{WHH^T + 10^{-9}} \right); \\
\text{end}
\]

Many parameters affect performance (k, obj. function, sparsity constraints, algorithm, etc.). — NMF is not unique!

(proof of convergence to fixed point based on E-M convergence proof)
NMF Algorithm: Lee and Seung 2000

**Divergence objective function**

\[
\min \sum_{i,j} (A_{ij} \log \frac{A_{ij}}{[WH]_{ij}} - A_{ij} + [WH]_{ij})
\]

\[s.t. \quad W, H \geq 0\]

\[
W = \text{abs} \left( \text{randn}(m,k) \right);
\]

\[
H = \text{abs} \left( \text{randn}(k,n) \right);
\]

for \( i = 1 : \text{maxiter} \)

\[
H = H \times \left( W^T (A ./ (WH + 10^{-9})) \right) ./ W^T ee^T;
\]

\[
W = W \times \left( ((A ./ (WH + 10^{-9}))H^T) \right) ./ ee^T H^T;
\]

end

(proof of convergence to fixed point based on E-M convergence proof)

(objective function tails off after 50-100 iterations)
Multiplicative Update Summary

Pros

+ convergence theory: guaranteed to converge to fixed point
+ good initialization $W^{(0)}, H^{(0)}$ speeds convergence and gets to better fixed point

Cons

– fixed point may be local min or saddle point
– good initialization $W^{(0)}, H^{(0)}$ speeds convergence and gets to better fixed point
– slow: many M-M multiplications at each iteration
– hundreds/thousands of iterations until convergence
– no sparsity of $W$ and $H$ incorporated into mathematical setup
– 0 elements locked
Multiplicative Update and Locking

During iterations of mult. update algorithms, once an element in $W$ or $H$ becomes 0, it can never become positive.

- Implications for $W$: In order to improve objective function, algorithm can only take terms out, not add terms, to topic vectors.

- Very inflexible: once algorithm starts down a path for a topic vector, it must continue in that vein.

- ALS-type algorithms do not lock elements, greater flexibility allows them to escape from path heading towards poor local min
Sparsity Measures

- Berry et al. \( \|x\|_2^2 \)

- Hoyer \( spar(x_{n \times 1}) = \frac{\sqrt{n} - \|x\|_1/\|x\|_2}{\sqrt{n-1}} \)

- Diversity measure \( E^{(p)}(x) = \sum_{i=1}^{n} |x_i|^p, \ 0 \leq p \leq 1 \)
  
  \( E^{(p)}(x) = -\sum_{i=1}^{n} |x_i|^p, \ p < 0 \)

  Rao and Kreutz-Delgado: algorithms for minimizing \( E^{(p)}(x) \)
  s.t. \( Ax = b \), but expensive iterative procedure

- Ideal \( nnz(x) \) not continuous, NP-hard to use this in optim.
NMF Algorithm: Berry et al. 2004

Gradient Descent–Constrained Least Squares

\[ W = \text{abs} \left( \text{randn}(m,k) \right); \quad \text{(scale cols of } W \text{ to unit norm)} \]

\[ H = \text{zeros}(k,n); \]

for \( i = 1 : \text{maxiter} \)

\[ \text{CLS} \quad \text{for } j = 1 : \# \text{docs}, \text{ solve} \]

\[ \min_{H_{*j}} \| A_{*j} - WH_{*j} \|_2^2 + \lambda \| H_{*j} \|_2^2 \]

\[ \text{s.t. } H_{*j} \geq 0 \]

\[ \text{GD} \quad W = W .* (A H^T) ./ (W H H^T + 10^{-9}); \quad \text{(scale cols of } W) \]

end
NMF Algorithm: Berry et al. 2004

**Gradient Descent–Constrained Least Squares**

\[
W = \text{abs} \left( \text{randn}(m,k) \right); \quad \text{(scale cols of } W \text{ to unit norm)}
\]

\[
H = \text{zeros}(k,n);
\]

for i = 1 : maxiter

\[\text{CLS} \quad \text{for } j = 1 : \text{#docs}, \text{ solve}
\]

\[
\min_{H_{*j}} \|A_{*j} - WH_{*j}\|_2^2 + \lambda \|H_{*j}\|_2^2
\]

s.t. \(H_{*j} \geq 0\)

solve for \(H\):

\[
(W^T W + \lambda I) H = W^T A; \quad \text{(small matrix solve)}
\]

\[\text{GD} \quad W = W \cdot (A H^T) ./ (W H H^T + 10^{-9}); \quad \text{(scale cols of } W)\]

end

(objective function tails off after 15-30 iterations)
Pros
  + fast: less work per iteration than most other NMF algorithms
  + fast: small # of iterations until convergence
  + sparsity parameter for $H$

Cons
  – 0 elements in $W$ are \textit{locked}
  – no sparsity parameter for $W$
  – ad hoc nonnegativity: negative elements in $H$ are set to 0, could run \texttt{lsqnonneg} or \texttt{snnls} instead
  – no convergence theory
PMF Algorithm: Paatero & Tapper 1994

**Mean Squared Error—Alternating Least Squares**

\[
\text{min} \| A - WH \|_F^2 \\
\text{s.t.} \quad W, H \geq 0
\]

\[
W = \text{abs}(\text{randn}(m,k));
\]

\text{for } i = 1 : \text{maxiter}

\hspace{1cm} \text{LS for } j = 1 : \#\text{docs}, \text{ solve}

\[
\text{min}_{H_{*j}} \| A_{*j} - WH_{*j} \|_2^2 \\
\text{s.t.} \quad H_{*j} \geq 0
\]

\hspace{1cm} \text{LS for } j = 1 : \#\text{terms}, \text{ solve}

\[
\text{min}_{W_{j*}} \| A_{j*} - W_{j*}H \|_2^2 \\
\text{s.t.} \quad W_{j*} \geq 0
\]

end
ALS Algorithm

\[ W = \text{abs(randn}(m,k)); \]

for \( i = 1 : \text{maxiter} \)

\[ \text{LS} \quad \text{solve matrix equation} \quad W^T WH = W^T A \text{ for } H \]

\[ \text{NONNEG} \quad H = H \cdot (H \geq 0) \]

\[ \text{LS} \quad \text{solve matrix equation} \quad HH^T W^T = HA^T \text{ for } W \]

\[ \text{NONNEG} \quad W = W \cdot (W \geq 0) \]

end
ALS Summary

Pros

+ fast
+ works well in practice
+ speedy convergence
+ only need to initialize $W^{(0)}$
+ 0 elements not locked

Cons

– no sparsity of $W$ and $H$ incorporated into mathematical setup
– ad hoc nonnegativity: negative elements are set to 0
– ad hoc sparsity: negative elements are set to 0
– no convergence theory
Alternating Constrained Least Squares

If the very fast ALS works well in practice and no NMF algorithms guarantee convergence to local min, why not use ALS?

\( W = \text{abs} (\text{randn}(m,k)) \);

for i = 1 : maxiter

CLS  for j = 1 : \#docs, solve

\[
\begin{align*}
\min_{H_{*j}} & \| A_{*j} - WH_{*j} \|_2^2 + \lambda_H \| H_{*j} \|_2^2 \\
\text{s.t.} & \quad H_{*j} \geq 0
\end{align*}
\]

CLS  for j = 1 : \#terms, solve

\[
\begin{align*}
\min_{W_{j*}} & \| A_{j*} - W_{j*}H \|_2^2 + \lambda_W \| W_{j*} \|_2^2 \\
\text{s.t.} & \quad W_{j*} \geq 0
\end{align*}
\]

end
Alternating Constrained Least Squares

If the very fast ALS works well in practice and no NMF algorithms guarantee convergence to local min, why not use ALS?

\[ \mathbf{W} = \text{abs}(\text{randn}(m,k)); \]

\text{for } i = 1 : \text{maxiter} \\
\quad \text{CLS} \quad \text{solve for } \mathbf{H}: \quad (\mathbf{W}^T \mathbf{W} + \lambda \mathbf{H} I) \mathbf{H} = \mathbf{W}^T \mathbf{A} \\
\quad \text{NONNEG} \quad \mathbf{H} = \mathbf{H} .* (\mathbf{H} \geq 0) \\
\quad \text{CLS} \quad \text{solve for } \mathbf{W}: \quad (\mathbf{H}^T \mathbf{H} + \lambda \mathbf{W} I) \mathbf{W} = \mathbf{H} \mathbf{A}^T \\
\quad \text{NONNEG} \quad \mathbf{W} = \mathbf{W} .* (\mathbf{W} \geq 0) \\
\text{end}
ACLS Summary

Pros

+ fast: 6.6 sec vs. 9.8 sec (gd-clsl)
+ works well in practice
+ speedy convergence
+ only need to initialize $W^{(0)}$
+ 0 elements not locked
+ allows for sparsity in both $W$ and $H$

Cons

– ad hoc nonnegativity: after LS, negative elements set to 0, could run \texttt{lsqnonneg} or \texttt{snnls} instead (doesn’t improve accuracy much)
– no convergence theory
ACLS + \text{spar}(x)

Is there a better way to measure sparsity and still maintain speed of ACLS?

\[ \text{spar}(x_{n \times 1}) = \frac{\sqrt{n} - \|x\|_1/\|x\|_2}{\sqrt{n} - 1} \quad \Leftrightarrow \quad ((1 - \text{spar}(x))\sqrt{n} + \text{spar}(x))\|x\|_2 - \|x\|_1 = 0 \]

\( (\text{spar}(W_{j*}) = \alpha_W \text{ and } \text{spar}(H_{*j}) = \alpha_H) \)

\[ W = \text{abs} \text{(randn}(m,k)); \]

for \( i = 1 : \text{maxiter} \)

\text{CLS} for \( j = 1 : \#\text{docs} \), solve

\[ \min_{H_{*j}} \|A_{*j} - WH_{*j}\|_2^2 + \lambda_H (((1 - \alpha_H)\sqrt{k} + \alpha_H)\|H_{*j}\|_2^2 - \|H_{*j}\|_1^2) \]

s.t. \( H_{*j} \geq 0 \)

\text{CLS} for \( j = 1 : \#\text{terms} \), solve

\[ \min_{W_{j*}} \|A_{j*} - W_{j*}H\|_2^2 + \lambda_W (((1 - \alpha_W)\sqrt{k} + \alpha_W)\|W_{j*}\|_2^2 - \|W_{j*}\|_1^2) \]

s.t. \( W_{j*} \geq 0 \)

end
AHCLS

\[(\text{spar}(W_{j*}) = \alpha_W \text{ and } \text{spar}(H_{*j}) = \alpha_H)\]

\[W = \text{abs(randn}(m,k)\];
\[\beta_H = ((1 - \alpha_H)\sqrt{k} + \alpha_H)^2\]
\[\beta_W = ((1 - \alpha_W)\sqrt{k} + \alpha_W)^2\]

for \(i = 1 : \text{maxiter}\)

**CLS** solve for \(H\):
\[(W^T W + \lambda_H \beta_H I - \lambda_H E) H = W^T A\]

**NONNEG** \(H = H \ast (H \geq 0)\)

**CLS** solve for \(W\):
\[(H H^T + \lambda_W \beta_W I - \lambda_W E) W^T = HA^T\]

**NONNEG** \(W = W \ast (W \geq 0)\)

end
AHCLS Summary

Pros

+ fast: 6.8 vs. 9.8 sec (gd-clls)
+ works well in practice
+ speedy convergence
+ only need to initialize $W^{(0)}$
+ 0 elements not locked
+ allows for more explicit sparsity in both $W$ and $H$

Cons

– ad hoc nonnegativity: after LS, negative elements set to 0, could run lsqnonneg or snnls instead (doesn’t improve accuracy much)
– no convergence theory
Strengths and Weaknesses of NMF

Strengths

- Great Interpretability
- Performance for data mining tasks comparable to LSI
- Sparsity of factorization allows for significant storage savings
- Scalability good as $k, m, n$ increase
- Possibly faster computation time than SVD

Weaknesses

- Factorization is not unique $\Rightarrow$ dependency on algorithm and parameters
- Unable to reduce the size of the basis without recomputing the NMF
Current NMF Research

- Algorithms
- Alternative Objective Functions
- Convergence Criterion
- Updating NMF
- Initializing NMF
- Choosing $k$
Extensions for NMF

Tensor NMF

\[ p \text{-way factorization} \quad A = A_1 A_2 \ldots A_p \quad A, A_i \geq 0 \]

Embedded NMF

\[ A = \begin{pmatrix} \text{topic} \\ \text{doc} \end{pmatrix} A_1 \begin{pmatrix} \text{topic} \\ \text{doc} \end{pmatrix} \], \quad \text{then} \quad A_1 = \begin{pmatrix} \text{subtopic} \\ \text{doc} \end{pmatrix} B_1 \begin{pmatrix} \text{subtopic} \\ \text{doc} \end{pmatrix} . \]

NMF on Web’s hyperlink matrix — terms from anchor text create \( A \)

\[ A = \begin{pmatrix} \text{node 1} & \text{node 2} & \ldots & \text{node n} \\ \text{term 1} & 1 & 5 & \ldots & 0 \\ \text{term 2} & 0 & 0 & \ldots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \text{term m} & 0 & 1 & \ldots & 2 \end{pmatrix} \]