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)

Part 2: Nonnegative Matrix Factorization (2000)

- 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 $\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_n$
- create term-by-document matrix $\mathbf{A}_{m \times n} = [\, \mathbf{d}_1 | \mathbf{d}_2 | \cdots | \mathbf{d}_n \,]$
- to retrieve info., create query vector $\mathbf{q}$, 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 $\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_n$
- create term-by-document matrix $\mathbf{A}_{m\times n} = [\, \mathbf{d}_1 | \mathbf{d}_2 | \cdots | \mathbf{d}_n \,]$
- to retrieve info., create query vector $\mathbf{q}$, which is a pseudo-doc

GOAL: find doc. $\mathbf{d}_i$ closest to $\mathbf{q}$

— angular cosine measure used: $\delta_i = cos\,\theta_i = \mathbf{q}^T\mathbf{d}_i / (\|\mathbf{q}\|_2 \|\mathbf{d}_i\|_2)$

Latent Semantic Indexing (1990s)

Susan Dumais's improvement to VSM = LSI

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

$\mathbf{A}_{m\times n}$: rank r term-by-document matrix

- SVD: $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$
- LSI: use $\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$ 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 $\mathbf{u}_i$ are orthogonal
- u_{ij}, v_{ij} are mixed in sign

$$\underset{nonneg}{\mathbf{A}_k} = \underset{mixed}{\mathbf{U}_k} \quad \underset{nonneg}{\Sigma_k} \quad \underset{mixed}{\mathbf{V}_k^T}$$

- **U**, **V** are dense
- *uniqueness*—while there are many SVD algorithms, they all create the same (truncated) factorization
- of all rank-k approximations, $\mathbf{A}_k$ is optimal (in Frobenius norm)

$$\|\mathbf{A} - \mathbf{A}_k\|_F = \min_{rank(\mathbf{B}) \leq k} \|\mathbf{A} - \mathbf{B}\|_F$$

Strengths and Weaknesses of LSI

Strengths

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

Weaknesses

- storage—$\mathbf{U}_k$ and $\mathbf{V}_k$ are usually completely dense
- interpretation of basis vectors $\mathbf{u}_i$ is impossible due to mixed signs
- good truncation point k is hard to determine
- orthogonality restriction

Other Low-Rank Approximations

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

  $\mathbf{A}_k = \mathbf{X}_k\mathbf{D}_k\mathbf{Y}_k^T$, where $\mathbf{D}_k$ is diagonal, and elements of $\mathbf{X}_k, \mathbf{Y}_k \in \{-1, 0, 1\}$.

Other Low-Rank Approximations

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

  $\mathbf{A}_k = \mathbf{X}_k\mathbf{D}_k\mathbf{Y}_k^T$, where $\mathbf{D}_k$ is diagonal, and elements of $\mathbf{X}_k, \mathbf{Y}_k \in \{-1, 0, 1\}$.

BUT

All create basis vectors that are mixed in sign. Negative elements make interpretation difficult.

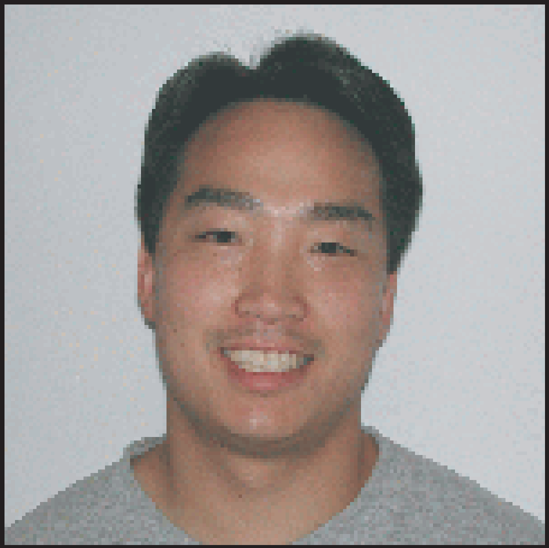
The Power of Positivity

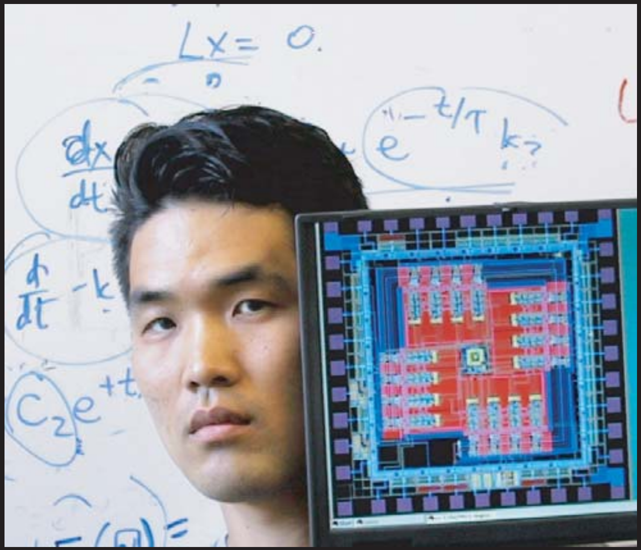
- Positive anything is better than negative nothing.—Elbert Hubbard
- It takes but one positive thought when given a chance to survive and thrive to overpower an entire army of negative thoughts.—Robert H. Schuller
- Learn to think like a winner. Think positive and visualize your strengths.—Vic Braden
- 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

$$\underset{nonneg}{\mathbf{A}_k} = \underset{mixed}{\mathbf{U}_k} \; \underset{nonneg}{\Sigma_k} \; \underset{mixed}{\mathbf{V}_k^T}$$

$$\underset{nonneg}{\mathbf{A}_k} = \underset{nonneg}{\mathbf{W}_k} \; \underset{nonneg}{\mathbf{H}_k}$$

Interpretation with NMF

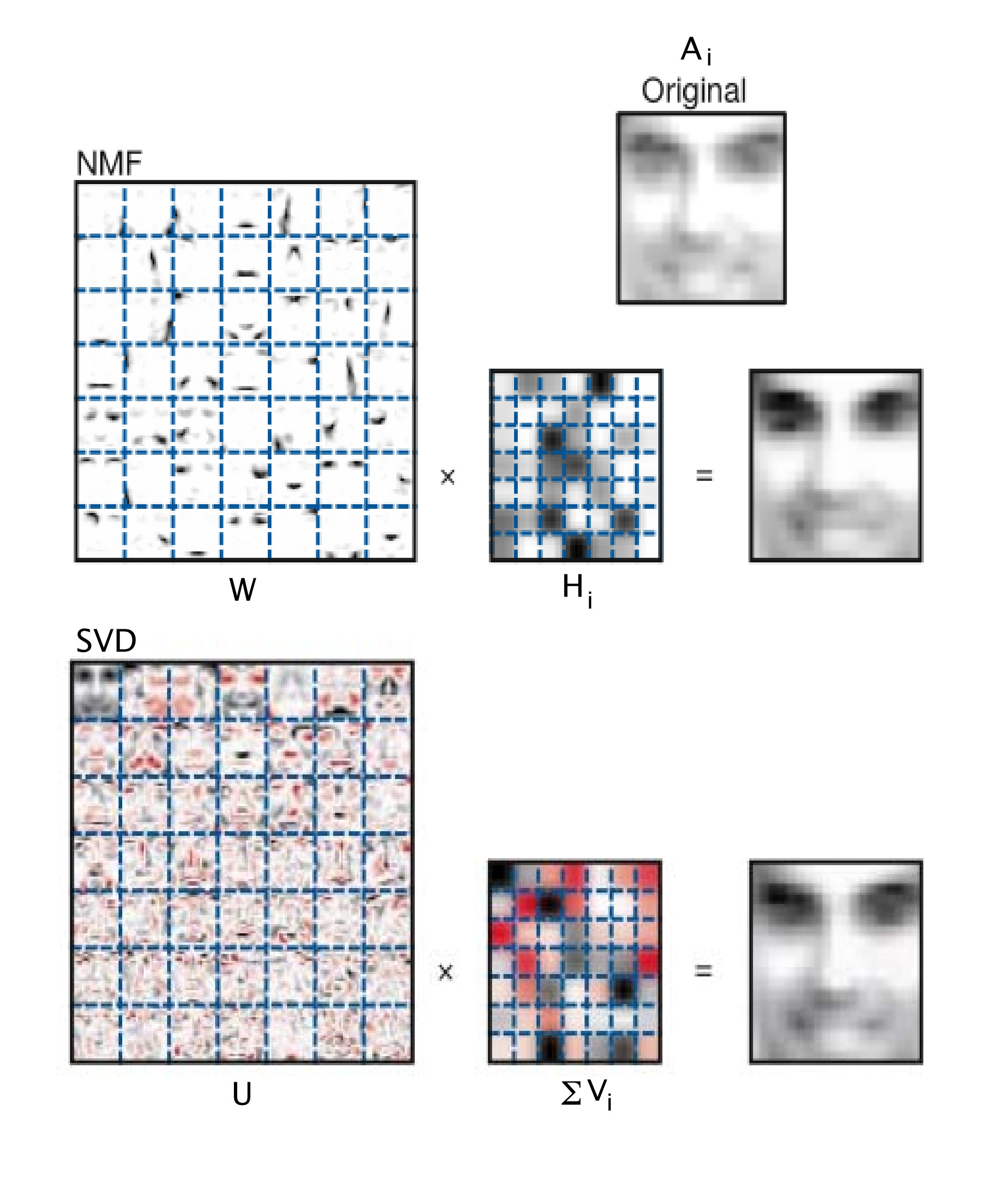
- columns of **W** are the underlying basis vectors, i.e., each of the n columns of **A** can be built from k columns of **W**.

- columns of **H** give the weights associated with each basis vector.

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

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

Image Mining



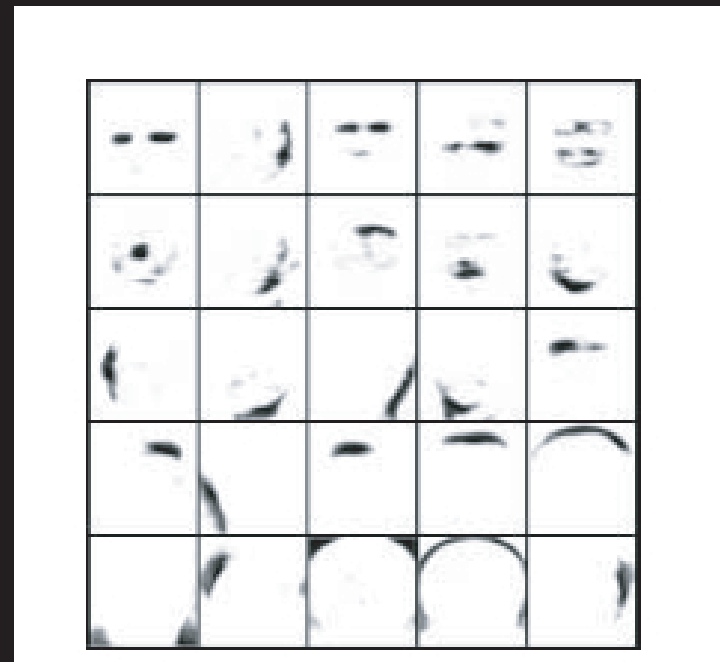


Image Mining Applications

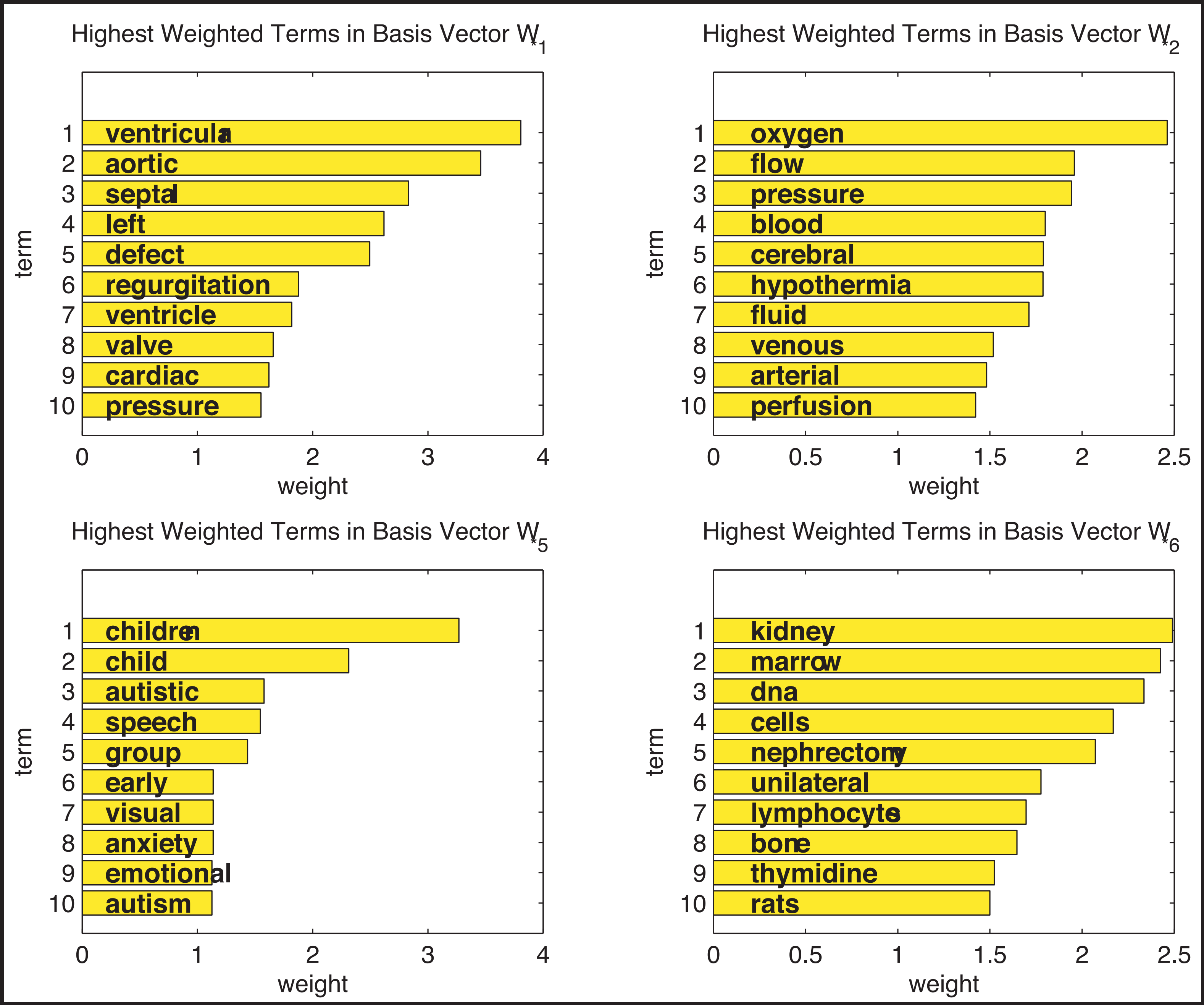
- Data compression
- Find similar images
- Cluster images

Original Image
r = 400

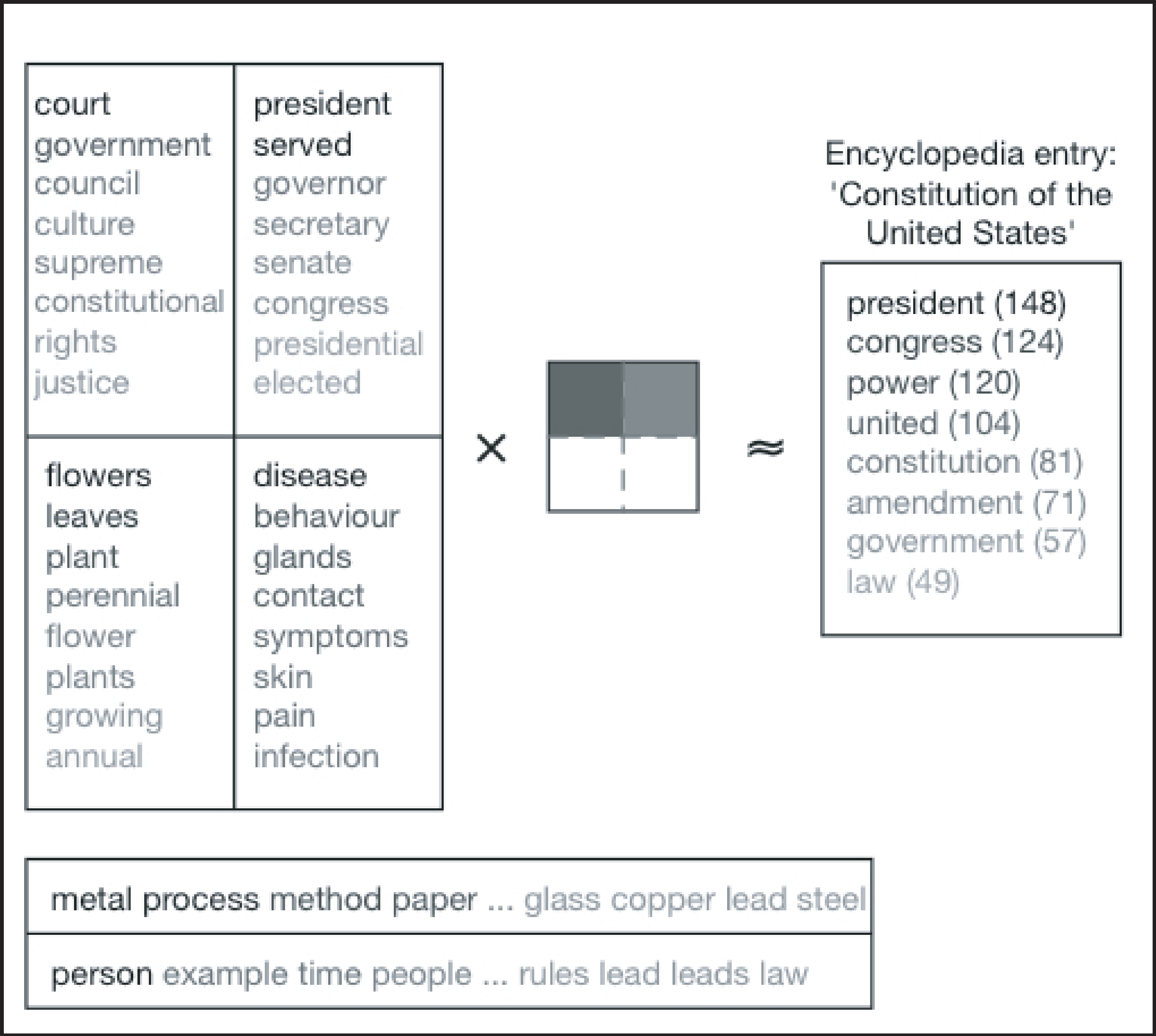
Reconstructed Images
k = 100

Text Mining

MED dataset ($k = 10$)



Text Mining



- polysems broken across several basis vectors $\mathbf{w}_i$

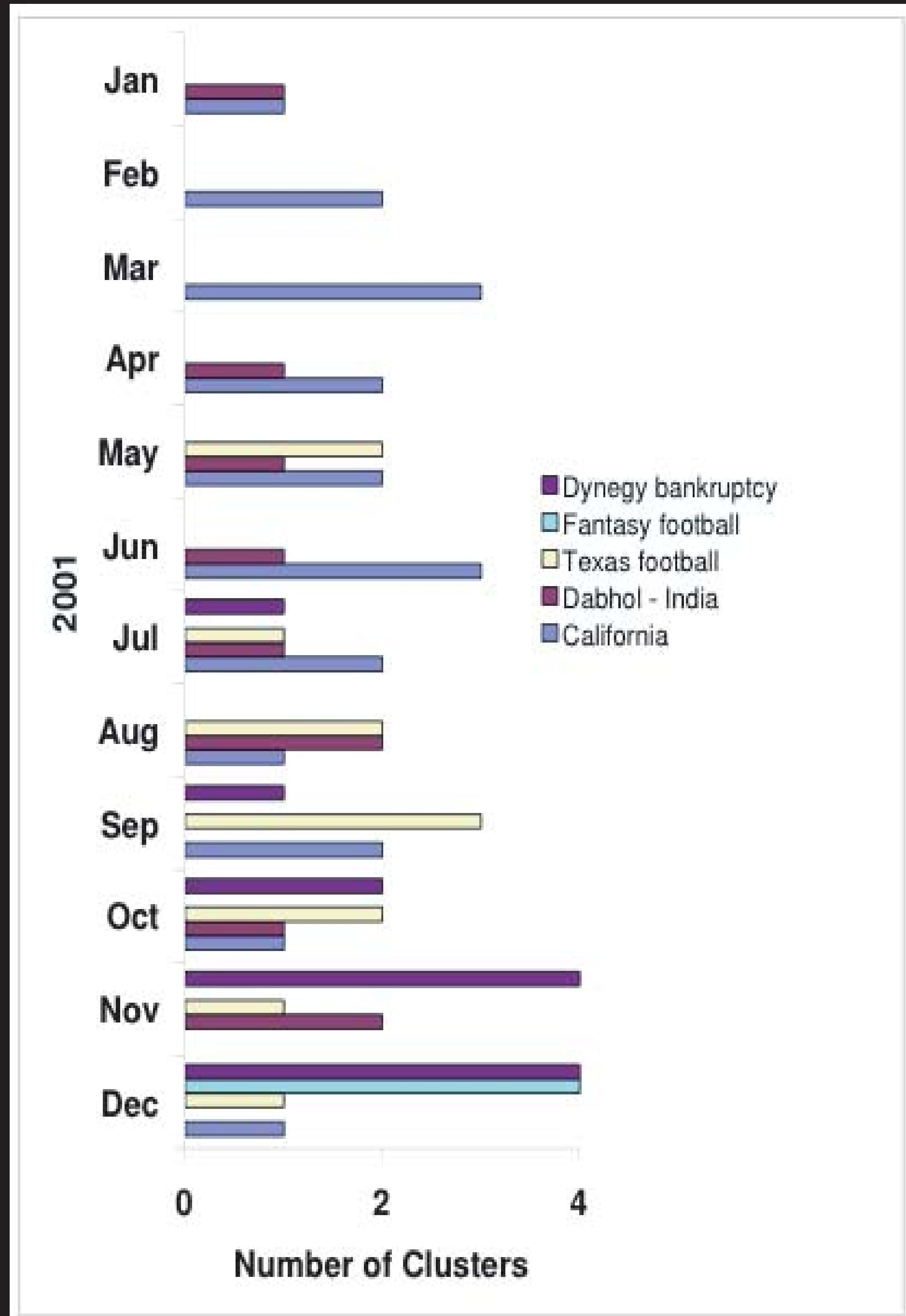
Text Mining Applications

- Data compression
- Find similar terms
- Find similar documents
- Cluster documents
- Topic detection and tracking

Text Mining Applications

Enron email messages 2001

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



Recommendation Systems

purchase history matrix

$$\mathbf{A} = \begin{array}{c} \\ \text{Item 1} \\ \text{Item 2} \\ \vdots \\ \text{Item m} \end{array} \begin{array}{c} \begin{array}{cccc} \text{User 1} & \text{User 2} & \dots & \text{User n} \end{array} \\ \begin{pmatrix} 1 & 5 & \dots & 0 \\ 0 & 0 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \dots & 2 \end{pmatrix} \end{array}$$

- Create profiles for classes of users from basis vectors $\mathbf{w}_i$
- Find similar users
- Find similar items

Properties of NMF

- basis vectors $\mathbf{w}_i$ are not $\perp \Rightarrow$ can have overlap of topics
- can restrict $\mathbf{W}$, $\mathbf{H}$ to be sparse
- $\mathbf{W}_k$, $\mathbf{H}_k \geq 0 \Rightarrow$ immediate interpretation (additive parts-based rep.)

  EX: large w_{ij}'s $\Rightarrow$ basis vector $\mathbf{w}_i$ is mostly about terms j

  EX: h_{i1} how much doc_1 is pointing in the "direction" of topic vector $\mathbf{w}_i$

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

- NMF is algorithm-dependent: $\mathbf{W}$, $\mathbf{H}$ not unique

Computation of NMF

(Lee and Seung 2000)

MEAN SQUARED ERROR OBJECTIVE FUNCTION

$$\min \|\mathbf{A} - \mathbf{W}\mathbf{H}\|_F^2 \quad s.t. \quad \mathbf{W}, \mathbf{H} \geq 0$$

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: $\mathbf{A}_{70K \times 1K}$ and k=10 topics $\Longrightarrow$ 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
  - AHCLS

NMF Algorithm: Lee and Seung 2000

Mean Squared Error objective function

$$\min \|\mathbf{A} - \mathbf{WH}\|_F^2$$

$$s.t. \quad \mathbf{W}, \mathbf{H} \geq 0$$

W = abs(randn(m,k));

H = abs(randn(k,n));

for i = 1 : maxiter

  $\mathbf{H} = \mathbf{H}$.* $(\mathbf{W}^T\mathbf{A})$ ./ $(\mathbf{W}^T\mathbf{WH} + 10^{-9})$;

  $\mathbf{W} = \mathbf{W}$.* $(\mathbf{AH}^T)$ ./ $(\mathbf{WHH}^T + 10^{-9})$;

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

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

$$s.t. \quad \mathbf{W}, \mathbf{H} \geq 0$$

W = abs(randn(m,k));

H = abs(randn(k,n));

for i = 1 : maxiter

$\mathbf{H} = \mathbf{H} \ .* \ (\mathbf{W}^T(\mathbf{A} \ ./ \ (\mathbf{WH} + 10^{-9}))) \ ./ \ \mathbf{W}^T\mathbf{ee}^T;$

$\mathbf{W} = \mathbf{W} \ .* \ ((\mathbf{A} \ ./ \ (\mathbf{WH} + 10^{-9}))\mathbf{H}^T) \ ./ \ \mathbf{ee}^T\mathbf{H}^T;$

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 $\mathbf{W}^{(0)}, \mathbf{H}^{(0)}$ speeds convergence and gets to better fixed point

Cons

– fixed point may be local min or saddle point
– good initialization $\mathbf{W}^{(0)}, \mathbf{H}^{(0)}$ 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. $\|\mathbf{x}\|_2^2$
- Hoyer $spar(\mathbf{x}_{n\times 1}) = \frac{\sqrt{n}-\|\mathbf{x}\|_1/\|\mathbf{x}\|_2}{\sqrt{n}-1}$
- Diversity measure $E^{(p)}(\mathbf{x}) = \sum_{i=1}^{n} |x_i|^p, \ 0 \leq p \leq 1$

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

  Rao and Kreutz-Delgado: algorithms for minimizing $E^{(p)}(\mathbf{x})$ s.t. $\mathbf{Ax} = \mathbf{b}$, but expensive iterative procedure
- Ideal $nnz(\mathbf{x})$ not continuous, NP-hard to use this in optim.

NMF Algorithm: Berry et al. 2004

Gradient Descent–Constrained Least Squares

$\mathbf{W}$ = abs(randn(m,k)); (scale cols of $\mathbf{W}$ to unit norm)

$\mathbf{H}$ = zeros(k,n);

for i = 1 : maxiter

CLS for j = 1 : $\#docs$, solve

$$\min_{\mathbf{H}_{*j}} \|\mathbf{A}_{*j} - \mathbf{W}\mathbf{H}_{*j}\|_2^2 + \lambda\|\mathbf{H}_{*j}\|_2^2$$

$$\text{s.t. } \mathbf{H}_{*j} \geq 0$$

GD $\mathbf{W} = \mathbf{W}$.* $(\mathbf{A}\mathbf{H}^T)$ ./ $(\mathbf{W}\mathbf{H}\mathbf{H}^T + 10^{-9})$; (scale cols of $\mathbf{W}$)

end

NMF Algorithm: Berry et al. 2004

GRADIENT DESCENT–CONSTRAINED LEAST SQUARES

W = abs(randn(m,k)); (scale cols of **W** to unit norm)

H = zeros(k,n);

for i = 1 : maxiter

CLS for j = 1 : $\#docs$, solve

$$\min_{\mathbf{H}_{*j}} \|\mathbf{A}_{*j} - \mathbf{W}\mathbf{H}_{*j}\|_2^2 + \lambda\|\mathbf{H}_{*j}\|_2^2$$

$$\text{s.t. } \mathbf{H}_{*j} \geq 0$$

solve for **H**: $(\mathbf{W}^T\mathbf{W} + \lambda\,\mathbf{I})\,\mathbf{H} = \mathbf{W}^T\mathbf{A}$; (small matrix solve)

GD $\mathbf{W} = \mathbf{W}$.* $(\mathbf{A}\mathbf{H}^T)$ ./ $(\mathbf{W}\mathbf{H}\mathbf{H}^T + 10^{-9})$; (scale cols of **W**)

end

(objective function tails off after 15-30 iterations)

Berry et al. 2004 Summary

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 *locked*
- – no sparsity parameter for **W**
- – ad hoc nonnegativity: negative elements in **H** are set to 0, could run lsqnonneg or snnls instead
- – no convergence theory

PMF Algorithm: Paatero & Tapper 1994

Mean Squared Error—Alternating Least Squares

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

W = abs(randn(m,k));

for i = 1 : maxiter

LS for j = 1 : $\#docs$, solve

$$\min_{\mathbf{H}_{*j}} \|\mathbf{A}_{*j} - \mathbf{WH}_{*j}\|_2^2$$
$$\text{s.t. } \mathbf{H}_{*j} \geq 0$$

LS for j = 1 : $\#terms$, solve

$$\min_{\mathbf{W}_{j*}} \|\mathbf{A}_{j*} - \mathbf{W}_{j*}\mathbf{H}\|_2^2$$
$$\text{s.t. } \mathbf{W}_{j*} \geq 0$$

end

ALS Algorithm

```
W = abs(randn(m,k));
for i = 1 : maxiter
    LS      solve matrix equation W^T W H = W^T A for H
    NONNEG  H = H .* (H >= 0)
    LS      solve matrix equation H H^T W^T = H A^T for W
    NONNEG  W = W .* (W >= 0)
end
```

ALS Summary

Pros

+ fast
+ works well in practice
+ speedy convergence
+ only need to initialize $\mathbf{W}^{(0)}$
+ 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 = abs(randn(m,k));

for i = 1 : maxiter

CLS for j = 1 : $\#docs$, solve

$$\min_{\mathbf{H}_{*j}} \|\mathbf{A}_{*j} - \mathbf{W}\mathbf{H}_{*j}\|_2^2 + \lambda_H \|\mathbf{H}_{*j}\|_2^2$$
$$\text{s.t. } \mathbf{H}_{*j} \geq 0$$

CLS for j = 1 : $\#terms$, solve

$$\min_{\mathbf{W}_{j*}} \|\mathbf{A}_{j*} - \mathbf{W}_{j*}\mathbf{H}\|_2^2 + \lambda_W \|\mathbf{W}_{j*}\|_2^2$$
$$\text{s.t. } \mathbf{W}_{j*} \geq 0$$

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?

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

for i = 1 : maxiter

CLS solve for **H**: $(\mathbf{W}^T\mathbf{W} + \lambda_H \mathbf{I})\ \mathbf{H} = \mathbf{W}^T\mathbf{A}$

NONNEG $\mathbf{H} = \mathbf{H}. * (\mathbf{H} >= 0)$

CLS solve for **W**: $(\mathbf{H}\mathbf{H}^T + \lambda_W \mathbf{I})\ \mathbf{W}^T = \mathbf{H}\mathbf{A}^T$

NONNEG $\mathbf{W} = \mathbf{W}. * (\mathbf{W} >= 0)$

end

ACLS Summary

Pros

+ fast: 6.6 sec vs. 9.8 sec (gd-cls)
+ works well in practice
+ speedy convergence
+ only need to initialize $\mathbf{W}^{(0)}$
+ 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 lsqnonneg or snnls instead (doesn't improve accuracy much)
– no convergence theory

ACLS + spar(x)

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

$$\text{spar}(\mathbf{x}_{n\times 1}) = \frac{\sqrt{n}-\|\mathbf{x}\|_1/\|\mathbf{x}\|_2}{\sqrt{n}-1} \quad \Leftrightarrow \quad ((1-\text{spar}(\mathbf{x}))\sqrt{n}+\text{spar}(\mathbf{x}))\|\mathbf{x}\|_2-\|\mathbf{x}\|_1=0$$

$$(\text{spar}(\mathbf{W}_{j*})=\alpha_W \text{ and } \text{spar}(\mathbf{H}_{*j})=\alpha_H)$$

W = abs(randn(m,k));

for i = 1 : maxiter

CLS for j = 1 : $\#docs$, solve

$$\min_{\mathbf{H}_{*j}} \|\mathbf{A}_{*j} - \mathbf{W}\mathbf{H}_{*j}\|_2^2 + \lambda_H(((1-\alpha_H)\sqrt{k}+\alpha_H)\|\mathbf{H}_{*j}\|_2^2 - \|\mathbf{H}_{*j}\|_1^2)$$

$$\text{s.t. } \mathbf{H}_{*j} \geq 0$$

CLS for j = 1 : $\#terms$, solve

$$\min_{\mathbf{W}_{j*}} \|\mathbf{A}_{j*} - \mathbf{W}_{j*}\mathbf{H}\|_2^2 + \lambda_W(((1-\alpha_W)\sqrt{k}+\alpha_W)\|\mathbf{W}_{j*}\|_2^2 - \|\mathbf{W}_{j*}\|_1^2)$$

$$\text{s.t. } \mathbf{W}_{j*} \geq 0$$

end

AHCLS

(spar($\mathbf{W}_{j*}$)=α_W and spar($\mathbf{H}_{*j}$)=α_H)

$\mathbf{W}$ = 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 : maxiter

CLS solve for **H**: $(\mathbf{W}^T\mathbf{W} + \lambda_H \beta_H \mathbf{I} - \lambda_H \mathbf{E})\ \mathbf{H} = \mathbf{W}^T\mathbf{A}$

NONNEG $\mathbf{H} = \mathbf{H}. * (\mathbf{H} >= 0)$

CLS solve for **W**: $(\mathbf{H}\mathbf{H}^T + \lambda_W \beta_W \mathbf{I} - \lambda_W \mathbf{E})\ \mathbf{W}^T = \mathbf{H}\mathbf{A}^T$

NONNEG $\mathbf{W} = \mathbf{W}. * (\mathbf{W} >= 0)$

end

AHCLS Summary

Pros

+ fast: 6.8 vs. 9.8 sec (gd-cls)
+ works well in practice
+ speedy convergence
+ only need to initialize $\mathbf{W}^{(0)}$
+ 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

$p-$way factorization $\quad \mathbf{A} = \mathbf{A}_1\mathbf{A}_2\ldots\mathbf{A}_p \quad \mathbf{A}, \mathbf{A}_i \geq 0$

Embedded NMF

$$\mathbf{A} = \text{term}\overset{\text{topic}}{\begin{pmatrix} \mathbf{A}_1 \end{pmatrix}} \; \text{topic}\overset{\text{doc}}{\begin{pmatrix} \mathbf{A}_2 \end{pmatrix}}, \quad \text{then } \mathbf{A}_1 = \text{term}\overset{\text{subtopic}}{\begin{pmatrix} \mathbf{B}_1 \end{pmatrix}} \; \text{subtopic}\overset{\text{doc}}{\begin{pmatrix} \mathbf{B}_2 \end{pmatrix}}.$$

NMF on Web's hyperlink matrix

— terms from anchor text create **A**

$$\mathbf{A} = \begin{array}{c} \\ \text{term 1} \\ \text{term 2} \\ \vdots \\ \text{term m} \end{array} \begin{array}{c} \begin{array}{cccc} \text{node 1} & \text{node 2} & \ldots & \text{node n} \end{array} \\ \begin{pmatrix} 1 & 5 & \ldots & 0 \\ 0 & 0 & \ldots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \ldots & 2 \end{pmatrix} \end{array}$$