$A \approx BC$

Matrix Approximation Problems

Suvrit Sra
EU Regional School, RWTH Aachen
April 28, 2010

(MPI für biologische Kybernetik, Tübingen)

MAX-PLANCK-GESELLSCHAFT

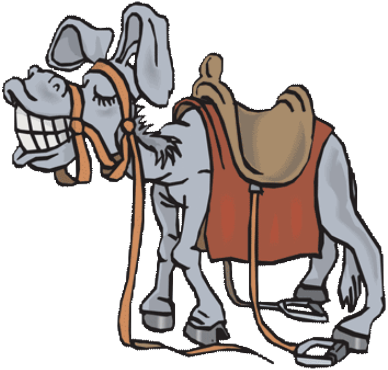
MPI FOR BIOLOGICAL CYBERNETICS

What's the course about?

$$\boxed{\boldsymbol{A} \approx \hat{\boldsymbol{A}}}$$

What's the course about?

$$\boldsymbol{A} \approx \widehat{\boldsymbol{A}}$$



What's the course about?

$$\boldsymbol{A} \approx \hat{\boldsymbol{A}}$$

What's the course about?

$$\boxed{\boldsymbol{A} \approx \widehat{\boldsymbol{A}}}$$

Not quite!

What's the course about?

$$\boxed{\boldsymbol{A} \approx \widehat{\boldsymbol{A}}}$$

Given an input matrix $\boldsymbol{A}$ compute a matrix $\widehat{\boldsymbol{A}}$ that satisfies certain desired properties, e.g.,

What's the course about?

$$\boldsymbol{A} \approx \hat{\boldsymbol{A}}$$

Given an input matrix $\boldsymbol{A}$ compute a matrix $\hat{\boldsymbol{A}}$ that satisfies certain desired properties, e.g.,

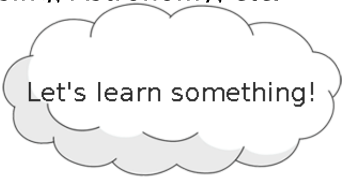
- symmetry, $\hat{\boldsymbol{A}}^T = \hat{\boldsymbol{A}}$
- sparsity, # nnz($\hat{\boldsymbol{A}}$) is small
- positive definiteness, $\hat{\boldsymbol{A}} \succeq 0$
- low-rank, $\hat{\boldsymbol{A}} = \boldsymbol{BC}$
- constraints, $\hat{\boldsymbol{A}} \in \mathcal{A}$
- ...

Today's lecture touches

1. Matrix Analysis
2. Numerical linear algebra
3. Computer Science
4. High-performance computing
5. Numerical optimization
6. Statistics
7. Data mining & machine learning
8. Image Processing, Astronomy, etc.

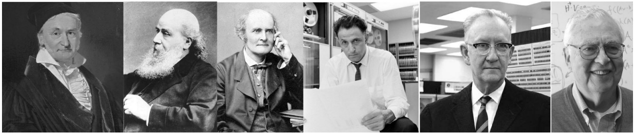
Today's lecture touches

1. Matrix Analysis
2. Numerical linear algebra
3. Computer Science
4. High-performance computing
5. Numerical optimization
6. Statistics
7. Data mining & machine learning
8. Image Processing, Astronomy, etc.



Introduction – matrices all over

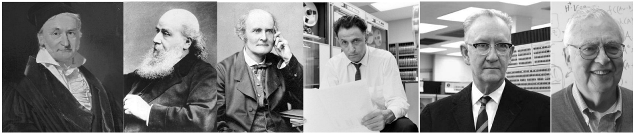
- Images



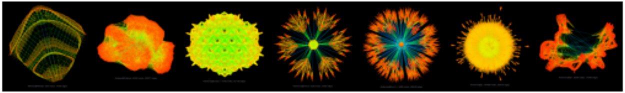
Matrix Collage made from images on Wikipedia; Sci. Comp. images take from Tim Davis' website; Internet graph from Wikipedia;[1]

Introduction – matrices all over

- Images



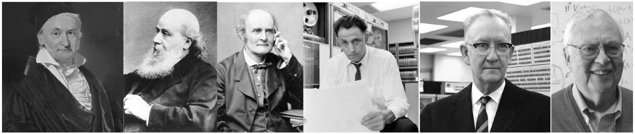
- Scientific Computing



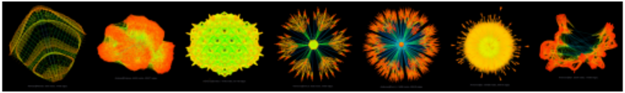
[1] Matrix Collage made from images on Wikipedia; Sci. Comp. images take from Tim Davis' website; Internet graph from Wikipedia;

Introduction – matrices all over

- Images



- Scientific Computing



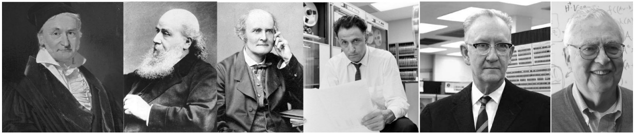
- Statistics

IV:	Ziel: >90	97,3 %	100 %	65,8 %	93,8 %	90,5 %	100 %	87,5 %	98,5 %
Darstellung der hirnversorg. Arterien	Auffälligkeit: <80	n=185	n=79	n=76	n=128	n=74	n=190	n=56	n=130
V:	Ziel: n. b.	21,3 %	21,0 %	23,4 %	32,8 %	30,2 %	25,2 %	17,9 %	16,1 %
Schluck-störungen	Auffälligkeit: <20	n=183	n=62	n=64	n=119	n=53	n=147	n=39	n=87
VI:	Ziel: >80	38,6 %	94,5 %	82,8 %	97,1 %	81,0 %	87,3 %	68,0 %	30,6 %
Logopädie	Auffälligkeit: <60	n=83	n=39	n=29	n=68	n=21	n=79	n=25	n=36
VII:	Ziel: >90	97,7 %	98,0 %	89,8 %	100 %	97,4 %	98,4 %	100 %	95,5%
Physio-/Ergotherapie	Auffälligkeit: <70	n=143	n=51	n=49	n=93	n=39	n=122	n=33	n=67

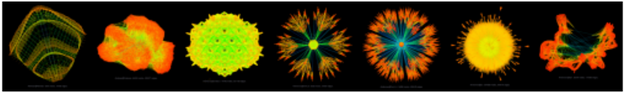
[1] Matrix Collage made from images on Wikipedia; Sci. Comp. images take from Tim Davis' website; Internet graph from Wikipedia;

Introduction – matrices all over

- Images



- Scientific Computing



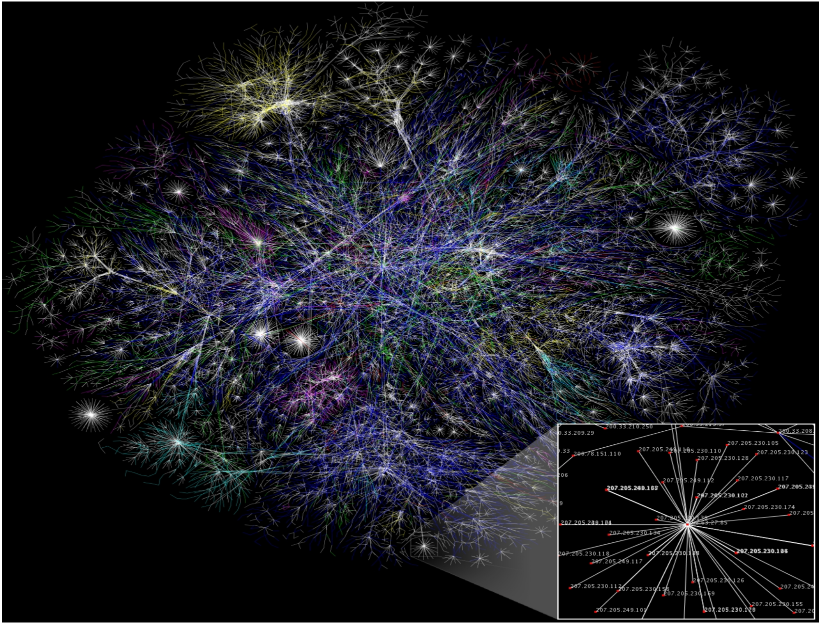
- Statistics

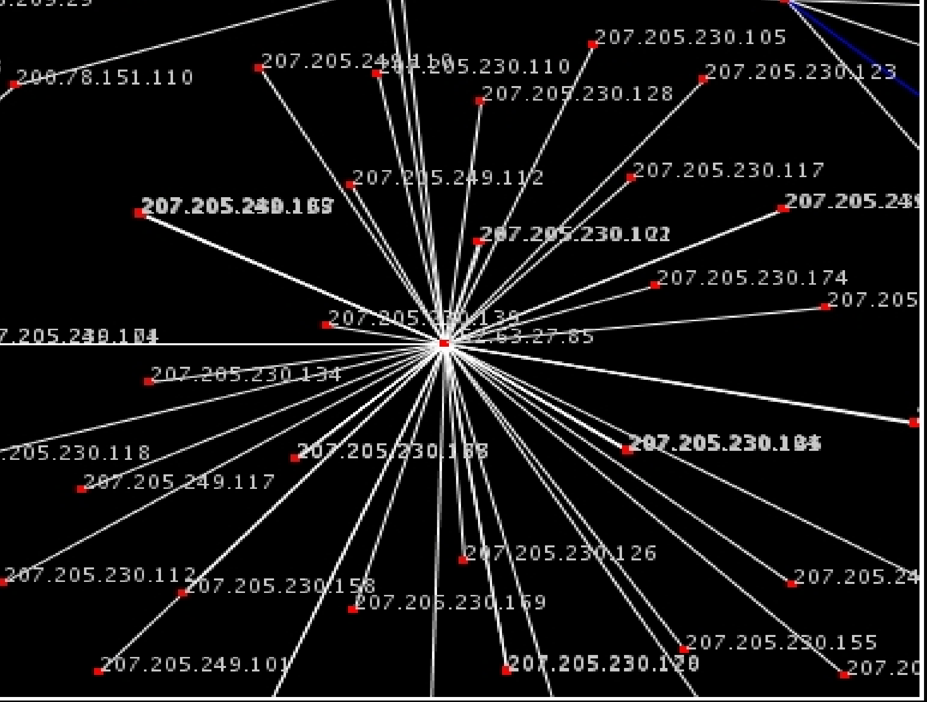
IV: Darstellung der hirnversorg. Arterien	Ziel: >90 Auffälligkeit: <80	97,3 % n=185	100 % n=79	65,8 % n=76	93,8 % n=128	90,5 % n=74	100 % n=190	87,5 % n=56	98,5 % n=130
V: Schluck-störungen	Ziel: n. b. Auffälligkeit: <20	21,3 % n=183	21,0 % n=62	23,4 % n=64	32,8 % n=119	30,2 % n=53	25,2 % n=147	17,9 % n=39	16,1 % n=87
VI: Logopädie	Ziel: >80 Auffälligkeit: <60	38,6 % n=83	94,5 % n=39	82,8 % n=29	97,1 % n=68	81,0 % n=21	87,3 % n=79	68,0 % n=25	30,6 % n=36
VII: Physio-/Ergotherapie	Ziel: >90 Auffälligkeit: <70	97,7 % n=143	98,0 % n=51	89,8 % n=49	100 % n=93	97,4 % n=39	98,4 % n=122	100 % n=33	95,5% n=67

- Computer Science

The Internet Graph[1]

[1] Matrix Collage made from images on Wikipedia; Sci. Comp. images take from Tim Davis' website; Internet graph from Wikipedia;


0.33.209.29
200.33.210.250
200.78.151.110
207.205.230.105
207.205.230.128
207.205.230.123
207.205.230.117
207.205.230.174
207.205.230.134
207.205.230.118
207.205.249.117
207.205.230.112
207.205.249.101
207.205.230.155



207.205.230.105
207.205.230.110
207.205.230.123
207.205.230.128
207.205.230.117
207.205.230.174
207.205.230.134
207.205.249.117
207.205.230.112
207.205.230.126
207.205.230.155
207.205.249.101

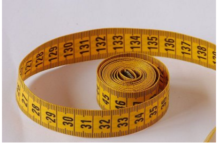
Introduction – Why approximate?

Introduction – Why approximate?

Measurements fail to satisfy expectation:

Introduction – Why approximate?

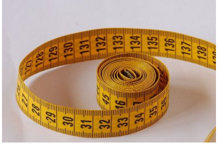
Measurements fail to satisfy expectation:



	A	B	C
A	0	3	8
B	2.8	0	4
C	7.9	4.1	0

Introduction – Why approximate?

Measurements fail to satisfy expectation:



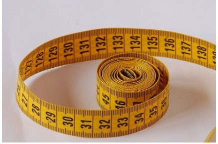
	A	B	C
A	0	3	8
B	2.8	0	4
C	7.9	4.1	0

	A	B	C
A	0	3	7.5
B	3	0	4.5
C	7.5	4.5	0

AC $\neq$ CA and AC $>$ AB + BC!

Introduction – Why approximate?

Measurements fail to satisfy expectation:



	A	B	C
A	0	3	8
B	2.8	0	4
C	7.9	4.1	0

	A	B	C
A	0	3	7.5
B	3	0	4.5
C	7.5	4.5	0

$AC \neq CA$ and $AC > AB + BC$!

Rounding errors, noise confound:

Expected symmetric, orthogonal, real, posdef, etc., but obtained something else!

Introduction – Why approximate?

Algorithm requires input to satisfy a property

Introduction – Why approximate?

Algorithm requires input to satisfy a property

Dimensionality reduction:

- Reduce storage
- Numerical benefits
- Expose structure
- Enable visualization
- Easier analysis
- E.g., for face recognition

Introduction – Why approximate?

Algorithm requires input to satisfy a property

Dimensionality reduction:

Hires (3MB)

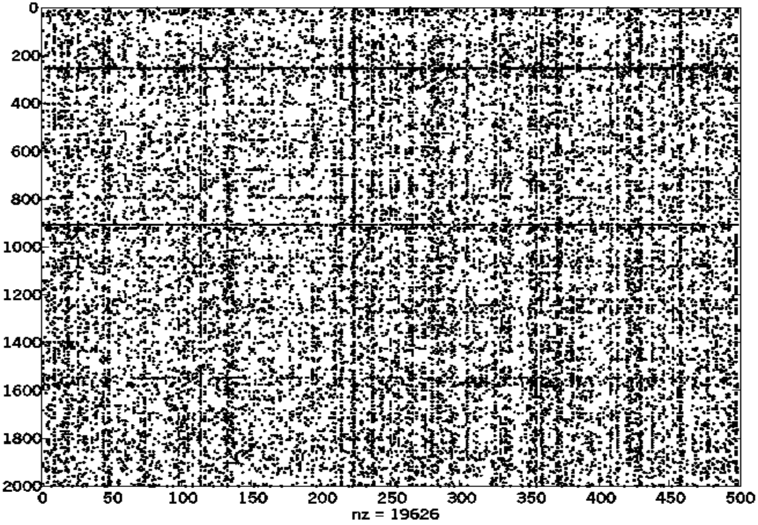
Lores (3KB!)

Introduction – Why approximate?

Discover structure:

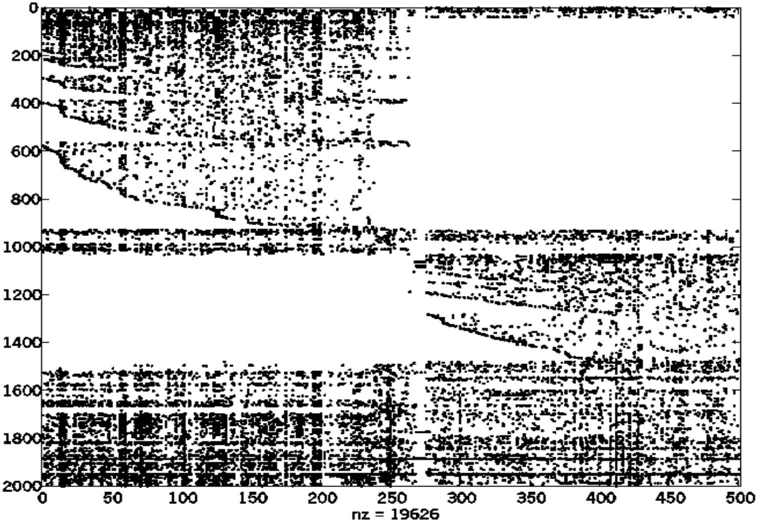
Introduction – Why approximate?

Discover structure:



Introduction – Why approximate?

Discover structure:



Introduction – Why approximate?

For €€ reasons!

Introduction – Why approximate?

For €€ reasons!

- Netflix million-$ prize problem!
- Typical *matrix completion* problem

Introduction – Why approximate?

For €€ reasons!

- Netflix million-\$ prize problem!
- Typical *matrix completion* problem
- Input: matrix $\boldsymbol{A}$ with several missing entries
- "Predict" missing entries to "complete" the matrix

Introduction – Why approximate?

For €€ reasons!

- Netflix million-$ prize problem!
- Typical *matrix completion* problem
- Input: matrix $\boldsymbol{A}$ with several missing entries
- “Predict” missing entries to “complete” the matrix
- Netflix: movies x users matrix; available entries were ratings given to movies by users
- Task was to predict missing entries, 10% better than Netflix’s inhouse system

Introduction – Why approximate?

For €€ reasons!

- Netflix million-$ prize problem!
- Typical *matrix completion* problem
- Input: matrix $\boldsymbol{A}$ with several missing entries
- "Predict" missing entries to "complete" the matrix
- Netflix: movies x users matrix; available entries were ratings given to movies by users
- Task was to predict missing entries, 10% better than Netflix's inhouse system
- Winners, and most top-performing methods: ultimately based on *matrix approximation* ideas!

Preliminaries

Introduction – preliminary concepts

Suppose we wish to approx. matrix $\boldsymbol{A}$ by $\widehat{\boldsymbol{A}}$. Ideally, $\widehat{\boldsymbol{A}}$ is the "nearest" matrix satisfying a desired property (eg. $\widehat{\boldsymbol{A}} \in \Omega$)?

Introduction – preliminary concepts

Suppose we wish to approx. matrix $\boldsymbol{A}$ by $\hat{\boldsymbol{A}}$. Ideally, $\hat{\boldsymbol{A}}$ is the "nearest" matrix satisfying a desired property (eg. $\hat{\boldsymbol{A}} \in \Omega$)?

$$\boxed{\text{First define } \textit{nearest}!}$$

Introduction – preliminary concepts

Suppose we wish to approx. matrix $\boldsymbol{A}$ by $\hat{\boldsymbol{A}}$. Ideally, $\hat{\boldsymbol{A}}$ is the "nearest" matrix satisfying a desired property (eg. $\hat{\boldsymbol{A}} \in \Omega$)?

$$\boxed{\text{First define } \textit{nearest}!}$$

We measure "distance" between two matrices using Δ

$$\boxed{\Delta(\boldsymbol{A}, \hat{\boldsymbol{A}})}$$

Introduction – preliminary concepts

Suppose we wish to approx. matrix $\boldsymbol{A}$ by $\hat{\boldsymbol{A}}$. Ideally, $\hat{\boldsymbol{A}}$ is the "nearest" matrix satisfying a desired property (eg. $\hat{\boldsymbol{A}} \in \Omega$)?

First define *nearest*!

We measure "distance" between two matrices using Δ

$$\Delta(\boldsymbol{A}, \hat{\boldsymbol{A}})$$

"Nearest" means: $\hat{\boldsymbol{A}} \in \Omega$ having smallest Δ value

Introduction – preliminary concepts

Suppose we wish to approx. matrix $\boldsymbol{A}$ by $\widehat{\boldsymbol{A}}$. Ideally, $\widehat{\boldsymbol{A}}$ is the "nearest" matrix satisfying a desired property (eg. $\widehat{\boldsymbol{A}} \in \Omega$)?

First define *nearest*!

We measure "distance" between two matrices using Δ

$$\Delta(\boldsymbol{A}, \widehat{\boldsymbol{A}})$$

"Nearest" means: $\widehat{\boldsymbol{A}} \in \Omega$ having smallest Δ value

Commonly used: $\Delta(\boldsymbol{A}, \widehat{\boldsymbol{A}}) = \|\boldsymbol{A} - \widehat{\boldsymbol{A}}\|$

Digression: Matrix Norms

An (operator) *norm* of a matrix $\boldsymbol{A}$ is defined as

$$\boxed{\|\boldsymbol{A}\| = \max_{\|\boldsymbol{x}\|=1} \|\boldsymbol{A}\boldsymbol{x}\|}$$

Example: Maximum singular value, $\sigma_1(\boldsymbol{A}) = \|\boldsymbol{A}\|_2$

Digression: Matrix Norms

An (operator) *norm* of a matrix $\boldsymbol{A}$ is defined as

$$\boxed{\|\boldsymbol{A}\| = \max_{\|\boldsymbol{x}\|=1} \|\boldsymbol{A}\boldsymbol{x}\|}$$

Example: Maximum singular value, $\sigma_1(\boldsymbol{A}) = \|\boldsymbol{A}\|_2$

The *Frobenius norm* $\|\boldsymbol{A}\|_{\mathrm{F}}$ is defined as

$$\boxed{\|\boldsymbol{X}\|_{\mathrm{F}} = \sqrt{\sum_{ij} x_{ij}^2}}$$

Digression: Matrix Norms

An (operator) *norm* of a matrix $\boldsymbol{A}$ is defined as

$$\boxed{\|\boldsymbol{A}\| = \max_{\|\boldsymbol{x}\|=1} \|\boldsymbol{A}\boldsymbol{x}\|}$$

Example: Maximum singular value, $\sigma_1(\boldsymbol{A}) = \|\boldsymbol{A}\|_2$

The *Frobenius norm* $\|\boldsymbol{A}\|_\mathrm{F}$ is defined as

$$\boxed{\|\boldsymbol{X}\|_\mathrm{F} = \sqrt{\sum_{ij} x_{ij}^2}}$$

I. Exercise: prove $\|\boldsymbol{X}\|_\mathrm{F}^2 = \mathrm{Tr}(\boldsymbol{X}^T\boldsymbol{X})$ where $\mathrm{Tr}(\blacksquare) \triangleq \sum_i \blacksquare_{ii}$ II.
Bonus: verify that $\sigma_1(\boldsymbol{A}) = \|\boldsymbol{A}\|_2$

Digression: Matrix Norms

An (operator) *norm* of a matrix $\boldsymbol{A}$ is defined as

$$\boxed{\|\boldsymbol{A}\| = \max_{\|\boldsymbol{x}\|=1} \|\boldsymbol{A}\boldsymbol{x}\|}$$

Example: Maximum singular value, $\sigma_1(\boldsymbol{A}) = \|\boldsymbol{A}\|_2$

The *Frobenius norm* $\|\boldsymbol{A}\|_F$ is defined as

$$\boxed{\|\boldsymbol{X}\|_F = \sqrt{\sum_{ij} x_{ij}^2}}$$

I. Exercise: prove $\|\boldsymbol{X}\|_F^2 = \mathrm{Tr}(\boldsymbol{X}^T\boldsymbol{X})$ where $\mathrm{Tr}(\blacksquare) \triangleq \sum_i \blacksquare_{ii}$ II.
Bonus: verify that $\sigma_1(\boldsymbol{A}) = \|\boldsymbol{A}\|_2$

We will mostly use the Frobenius norm for convenience

Warmup example

Suppose $\boldsymbol{A} \in \mathbb{R}^{n \times n}$. What is the nearest symmetric matrix?

$$\boxed{\min \quad \|\boldsymbol{A} - \widehat{\boldsymbol{A}}\|_F \quad \text{s.t.} \quad \widehat{\boldsymbol{A}}^T = \widehat{\boldsymbol{A}}}$$

Warmup example

Suppose $\boldsymbol{A} \in \mathbb{R}^{n \times n}$. What is the nearest symmetric matrix?

$$\boxed{\min \quad \|\boldsymbol{A} - \widehat{\boldsymbol{A}}\|_F \quad \text{s.t.} \quad \widehat{\boldsymbol{A}}^T = \widehat{\boldsymbol{A}}}$$

Solution: FaHo55

$\widehat{\boldsymbol{A}} = (\boldsymbol{A} + \boldsymbol{A}^T)/2$. To verify, do the following:

1. Let $\boldsymbol{X}$ be any $n \times n$ symmetric matrix
2. Prove that $\|\boldsymbol{A} - \widehat{\boldsymbol{A}}\|_F \leq \|\boldsymbol{A} - \boldsymbol{X}\|_F$

Warmup example

Suppose $\boldsymbol{A} \in \mathbb{R}^{n\times n}$. What is the nearest symmetric matrix?

$$\boxed{\min \quad \|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_{\mathrm{F}} \quad \text{s.t.} \quad \hat{\boldsymbol{A}}^T = \hat{\boldsymbol{A}}}$$

Solution: FaHo55

$\hat{\boldsymbol{A}} = (\boldsymbol{A} + \boldsymbol{A}^T)/2$. To verify, do the following:

1. Let $\boldsymbol{X}$ be any $n \times n$ symmetric matrix
2. Prove that $\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_{\mathrm{F}} \leq \|\boldsymbol{A} - \boldsymbol{X}\|_{\mathrm{F}}$

   $\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_{\mathrm{F}} = \frac{1}{2}\|\boldsymbol{A} - \boldsymbol{X} + \boldsymbol{X}^T - \boldsymbol{A}^T\|_{\mathrm{F}}$

Warmup example

Suppose $\boldsymbol{A} \in \mathbb{R}^{n \times n}$. What is the nearest symmetric matrix?

$$\boxed{\min \quad \|\boldsymbol{A} - \widehat{\boldsymbol{A}}\|_F \quad \text{s.t.} \quad \widehat{\boldsymbol{A}}^T = \widehat{\boldsymbol{A}}}$$

Solution: FaHo55

$\widehat{\boldsymbol{A}} = (\boldsymbol{A} + \boldsymbol{A}^T)/2$. To verify, do the following:

1. Let $\boldsymbol{X}$ be any $n \times n$ symmetric matrix
2. Prove that $\|\boldsymbol{A} - \widehat{\boldsymbol{A}}\|_F \leq \|\boldsymbol{A} - \boldsymbol{X}\|_F$

$$\begin{aligned}\|\boldsymbol{A} - \widehat{\boldsymbol{A}}\|_F &= \tfrac{1}{2}\|\boldsymbol{A} - \boldsymbol{X} + \boldsymbol{X}^T - \boldsymbol{A}^T\|_F \\ &\leq \tfrac{1}{2}\|\boldsymbol{A} - \boldsymbol{X}\|_F + \tfrac{1}{2}\|(\boldsymbol{X} - \boldsymbol{A})^T\|_F = \|\boldsymbol{A} - \boldsymbol{X}\|_F,\end{aligned}$$

Warmup example

Suppose $\boldsymbol{A} \in \mathbb{R}^{n \times n}$. What is the nearest symmetric matrix?

$$\boxed{\min \quad \|\boldsymbol{A} - \widehat{\boldsymbol{A}}\|_{\mathrm{F}} \quad \text{s.t.} \quad \widehat{\boldsymbol{A}}^T = \widehat{\boldsymbol{A}}}$$

Solution: FaHo55

$\widehat{\boldsymbol{A}} = (\boldsymbol{A} + \boldsymbol{A}^T)/2$. To verify, do the following:

1. Let $\boldsymbol{X}$ be any $n \times n$ symmetric matrix
2. Prove that $\|\boldsymbol{A} - \widehat{\boldsymbol{A}}\|_{\mathrm{F}} \leq \|\boldsymbol{A} - \boldsymbol{X}\|_{\mathrm{F}}$

$$\begin{aligned}\|\boldsymbol{A} - \widehat{\boldsymbol{A}}\|_{\mathrm{F}} &= \tfrac{1}{2}\|\boldsymbol{A} - \boldsymbol{X} + \boldsymbol{X}^T - \boldsymbol{A}^T\|_{\mathrm{F}} \\ &\leq \tfrac{1}{2}\|\boldsymbol{A} - \boldsymbol{X}\|_{\mathrm{F}} + \tfrac{1}{2}\|(\boldsymbol{X} - \boldsymbol{A})^T\|_{\mathrm{F}} = \|\boldsymbol{A} - \boldsymbol{X}\|_{\mathrm{F}},\end{aligned}$$

since $\|\boldsymbol{X}\|_{\mathrm{F}} = \|\boldsymbol{X}^T\|_{\mathrm{F}}$.

More challenging example

Suppose $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ (we assume throughout $m \geq n$). What is the nearest rank-k matrix, where $k < r = \operatorname{rank}(\boldsymbol{A})$?

More challenging example

Suppose $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ (we assume throughout $m \geq n$). What is the nearest rank-k matrix, where $k < r = \operatorname{rank}(\boldsymbol{A})$?

Let $\boldsymbol{B} \in \mathbb{R}^{m \times k}$ and $\boldsymbol{C} \in \mathbb{R}^{k \times n}$. Then, $\operatorname{rank}(\boldsymbol{BC}) \leq k$. And we have the formula from the title slide:

$$\boxed{\boldsymbol{A} \approx \boldsymbol{BC}}$$

More challenging example

Suppose $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ (we assume throughout $m \geq n$). What is the nearest rank-k matrix, where $k < r = \operatorname{rank}(\boldsymbol{A})$?

Let $\boldsymbol{B} \in \mathbb{R}^{m \times k}$ and $\boldsymbol{C} \in \mathbb{R}^{k \times n}$. Then, $\operatorname{rank}(\boldsymbol{BC}) \leq k$. And we have the formula from the title slide:

$$\boxed{\boldsymbol{A} \approx \boldsymbol{BC}}$$

"Factors" $\boldsymbol{B}$, $\boldsymbol{C}$ can be computed by solving

$$\boxed{\min \tfrac{1}{2} \|\boldsymbol{A} - \boldsymbol{BC}\|_F^2}$$

But How??

The SVD

Recall fundamental matrix *factorization*:

$$\boxed{\text{Singular Value Decomposition}}$$

The SVD

Recall fundamental matrix *factorization*:

$$\boxed{\text{Singular Value Decomposition}}$$

SVD (Thm. 2.5.2 [GoLo96])

Let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. There exist *orthogonal* matrices $\boldsymbol{U}$ and $\boldsymbol{V}$

$$\boldsymbol{U}^T \boldsymbol{A} \boldsymbol{V} = \mathrm{Diag}(\sigma_1, \ldots, \sigma_p), \quad p = \min(m, n),$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$.

The SVD

Recall fundamental matrix *factorization*:

Singular Value Decomposition

SVD (Thm. 2.5.2 [GoLo96])

Let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. There exist *orthogonal* matrices $\boldsymbol{U}$ and $\boldsymbol{V}$

$$\boldsymbol{U}^T \boldsymbol{A} \boldsymbol{V} = \mathrm{Diag}(\sigma_1, \ldots, \sigma_p), \quad p = \min(m, n),$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$.

$$\boldsymbol{A}_{m \times n} = \boldsymbol{U}_{m \times m} \begin{bmatrix} \Sigma_{n \times n} \\ 0 \end{bmatrix} \boldsymbol{V}_{n \times n}^T$$

Exercise: $\boldsymbol{A} = \sum_i \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$ $\quad (\boldsymbol{U} = [\boldsymbol{u}_i]$ and $\boldsymbol{V} = [\boldsymbol{v}_i])$

Approximation example: truncated SVD

- Reveals a lot about the structure of matrix

Approximation example: truncated SVD

- Reveals a lot about the structure of matrix
- Makes explicit (algebraically, and numerically) the notions of *rank*, *range space*, *null space* of $\boldsymbol{A}$.

Approximation example: truncated SVD

- Reveals a lot about the structure of matrix
- Makes explicit (algebraically, and numerically) the notions of *rank*, *range space*, *null space* of $\boldsymbol{A}$.
- Has numerous applications; for us, interesting because

Approximation example: truncated SVD

- Reveals a lot about the structure of matrix
- Makes explicit (algebraically, and numerically) the notions of *rank*, *range space*, *null space* of $\boldsymbol{A}$.
- Has numerous applications; for us, interesting because

Theorem (Optimality of SVD)

Let $\boldsymbol{A}$ *have the SVD* $\boldsymbol{U\Sigma V}^T$. *If* $k < \operatorname{rank}(\boldsymbol{A})$ *and*

$$\boldsymbol{A}_k = \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T, \qquad \textit{then,}$$

$$\|\boldsymbol{A} - \boldsymbol{A}_k\|_2 \le \|\boldsymbol{A} - \boldsymbol{B}\|_2, \quad \textit{s.t.} \quad \operatorname{rank}(\boldsymbol{B}) \le k$$
$$\|\boldsymbol{A} - \boldsymbol{A}_k\|_F \le \|\boldsymbol{A} - \boldsymbol{B}\|_F, \quad \textit{s.t.} \quad \operatorname{rank}(\boldsymbol{B}) \le k.$$

Truncated SVD (TSVD) – Proof Sketch

Prove: TSVD yields "best" Rank-k approximation to matrix $\boldsymbol{A}$

Proof: (2-norm).

1. First verify that $\|\boldsymbol{A} - \boldsymbol{A}_k\|_2 = \sigma_{k+1}$

Truncated SVD (TSVD) – Proof Sketch

Prove: TSVD yields "best" Rank-k approximation to matrix $\boldsymbol{A}$

Proof: (2-norm).

1. First verify that $\|\boldsymbol{A} - \boldsymbol{A}_k\|_2 = \sigma_{k+1}$
2. Let $\boldsymbol{B}$ be any rank-k matrix

Truncated SVD (TSVD) – Proof Sketch

Prove: TSVD yields "best" Rank-k approximation to matrix $\boldsymbol{A}$

Proof: (2-norm).

1. First verify that $\|\boldsymbol{A} - \boldsymbol{A}_k\|_2 = \sigma_{k+1}$
2. Let $\boldsymbol{B}$ be any rank-k matrix
3. Prove that $\|\boldsymbol{A} - \boldsymbol{B}\|_2 \geq \sigma_{k+1}$

Truncated SVD (TSVD) – Proof Sketch

Prove: TSVD yields "best" Rank-k approximation to matrix $\boldsymbol{A}$

Proof: (2-norm).

1. First verify that $\|\boldsymbol{A} - \boldsymbol{A}_k\|_2 = \sigma_{k+1}$
2. Let $\boldsymbol{B}$ be any rank-k matrix
3. Prove that $\|\boldsymbol{A} - \boldsymbol{B}\|_2 \geq \sigma_{k+1}$

Since $\text{rank}(\boldsymbol{B}) = k$, there are $n - k$ vectors that span the null-space $\mathcal{N}(\boldsymbol{B})$. But $\mathcal{N}(\boldsymbol{B}) \cap \boldsymbol{V}_{k+1} \neq \{0\}$ (??), so we can pick a unit-norm vector $\boldsymbol{z} \in \mathcal{N}(\boldsymbol{B}) \cap \boldsymbol{V}_{k+1}$. Now $\boldsymbol{Bz} = 0$, so

Truncated SVD (TSVD) – Proof Sketch

Prove: TSVD yields "best" Rank-k approximation to matrix $\boldsymbol{A}$

Proof: (2-norm).

1. First verify that $\|\boldsymbol{A} - \boldsymbol{A}_k\|_2 = \sigma_{k+1}$
2. Let $\boldsymbol{B}$ be any rank-k matrix
3. Prove that $\|\boldsymbol{A} - \boldsymbol{B}\|_2 \geq \sigma_{k+1}$

Since $\operatorname{rank}(\boldsymbol{B}) = k$, there are $n - k$ vectors that span the null-space $\mathcal{N}(\boldsymbol{B})$. But $\mathcal{N}(\boldsymbol{B}) \cap \boldsymbol{V}_{k+1} \neq \{0\}$ (??), so we can pick a unit-norm vector $\boldsymbol{z} \in \mathcal{N}(\boldsymbol{B}) \cap \boldsymbol{V}_{k+1}$. Now $\boldsymbol{Bz} = 0$, so

$$\|\boldsymbol{A} - \boldsymbol{B}\|_2^2 \geq \|(\boldsymbol{A} - \boldsymbol{B})\boldsymbol{z}\|_2^2 = \|\boldsymbol{Az}\|_2^2 = \sum_i^{k+1} \sigma_i^2 (\boldsymbol{v}_i^T \boldsymbol{z})^2 \geq \sigma_{k+1}^2$$

Truncated SVD (TSVD) – Proof Sketch

Prove: TSVD yields "best" Rank-k approximation to matrix $\boldsymbol{A}$

Proof: (2-norm).

1. First verify that $\|\boldsymbol{A} - \boldsymbol{A}_k\|_2 = \sigma_{k+1}$
2. Let $\boldsymbol{B}$ be any rank-k matrix
3. Prove that $\|\boldsymbol{A} - \boldsymbol{B}\|_2 \geq \sigma_{k+1}$

Since $\operatorname{rank}(\boldsymbol{B}) = k$, there are $n-k$ vectors that span the null-space $\mathcal{N}(\boldsymbol{B})$. But $\mathcal{N}(\boldsymbol{B}) \cap \boldsymbol{V}_{k+1} \neq \{0\}$ (??), so we can pick a unit-norm vector $\boldsymbol{z} \in \mathcal{N}(\boldsymbol{B}) \cap \boldsymbol{V}_{k+1}$. Now $\boldsymbol{Bz} = 0$, so

$$\|\boldsymbol{A} - \boldsymbol{B}\|_2^2 \geq \|(\boldsymbol{A} - \boldsymbol{B})\boldsymbol{z}\|_2^2 = \|\boldsymbol{Az}\|_2^2 = \sum_i^{k+1} \sigma_i^2 (\boldsymbol{v}_i^T \boldsymbol{z})^2 \geq \sigma_{k+1}^2$$

We used: $\|\boldsymbol{Az}\|_2 \leq \|\boldsymbol{A}\|_2 \|\boldsymbol{z}\|_2$ $\square$

TSVD – Message

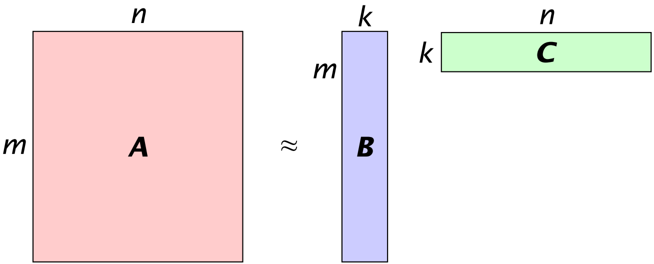
If we are seeking a rank-k approximation to $\boldsymbol{A}$

$$\boxed{\boldsymbol{A} \approx \boldsymbol{B}\boldsymbol{C}}$$

TSVD – Message

If we are seeking a rank-k approximation to $\boldsymbol{A}$

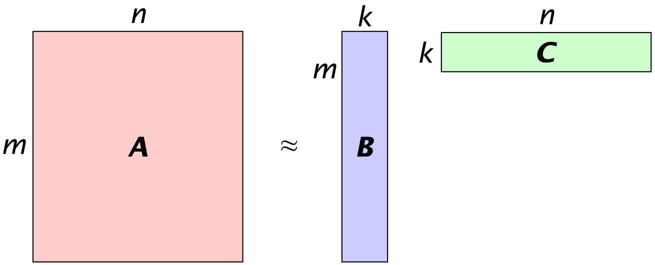
$$\boxed{\boldsymbol{A} \approx \boldsymbol{BC}}$$



TSVD – Message

If we are seeking a rank-k approximation to $\boldsymbol{A}$

$$\boxed{\boldsymbol{A} \approx \boldsymbol{B}\boldsymbol{C}}$$



$$\boxed{\text{TSVD yields: } \boldsymbol{B} = \boldsymbol{U}_k\Sigma_k, \text{ and } \boldsymbol{C} = \boldsymbol{V}_k^T}$$

Example Problems

1. Truncated SVD, PCA

1. Truncated SVD, PCA
2. Nonnegative matrix approximation (aka NMF)

1. Truncated SVD, PCA
2. Nonnegative matrix approximation (aka NMF)
3. Sparsity constrained versions of PCA, NMF

1. Truncated SVD, PCA
2. Nonnegative matrix approximation (aka NMF)
3. Sparsity constrained versions of PCA, NMF
4. Clustering, Co-clustering

1. Truncated SVD, PCA
2. Nonnegative matrix approximation (aka NMF)
3. Sparsity constrained versions of PCA, NMF
4. Clustering, Co-clustering
5. Matrix Completion

1. Truncated SVD, PCA
2. Nonnegative matrix approximation (aka NMF)
3. Sparsity constrained versions of PCA, NMF
4. Clustering, Co-clustering
5. Matrix Completion
6. Probabilistic matrix factorization

1. Truncated SVD, PCA
2. Nonnegative matrix approximation (aka NMF)
3. Sparsity constrained versions of PCA, NMF
4. Clustering, Co-clustering
5. Matrix Completion
6. Probabilistic matrix factorization
7. Nearest positive-definite matrix

1. Truncated SVD, PCA
2. Nonnegative matrix approximation (aka NMF)
3. Sparsity constrained versions of PCA, NMF
4. Clustering, Co-clustering
5. Matrix Completion
6. Probabilistic matrix factorization
7. Nearest positive-definite matrix
8. Parallel variants of all of these

1. Truncated SVD, PCA
2. Nonnegative matrix approximation (aka NMF)
3. Sparsity constrained versions of PCA, NMF
4. Clustering, Co-clustering
5. Matrix Completion
6. Probabilistic matrix factorization
7. Nearest positive-definite matrix
8. Parallel variants of all of these
9. Approximate variants

1. Truncated SVD, PCA
2. Nonnegative matrix approximation (aka NMF)
3. Sparsity constrained versions of PCA, NMF
4. Clustering, Co-clustering
5. Matrix Completion
6. Probabilistic matrix factorization
7. Nearest positive-definite matrix
8. Parallel variants of all of these
9. Approximate variants
10. and so on....

TSVD, PCA

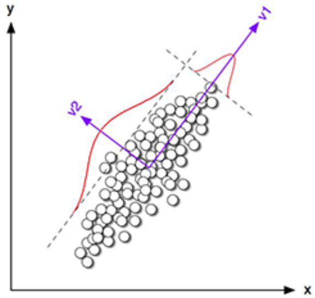
Principal component analysis, aka PCA based on TSVD

PCA computes top-k eigenvectors (*principal components*)

TSVD, PCA

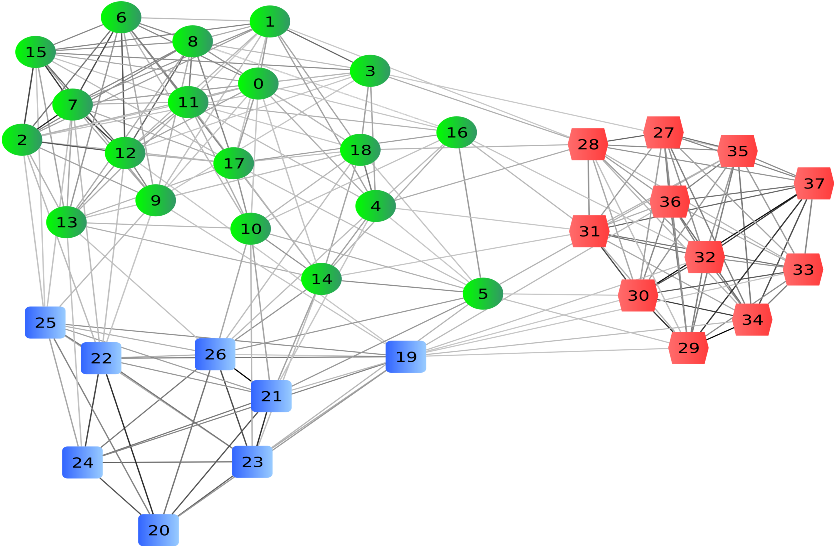
Principal component analysis, aka PCA based on TSVD

PCA computes top-k eigenvectors (*principal components*)
Dimensionality reduction; exploratory data analysis;



Principal components account for variance (spread)

Clustering, Co-clustering



Clustering, Co-clustering

Original matrix

a	+	a	+	+
z	∘	z	∘	∘
a	+	a	+	+
–	*	–	*	*
–	*	–	*	*
z	∘	z	∘	∘

Clustering, Co-clustering

Clustered matrix

a	a	+	+	+
z	z	∘	∘	∘
a	a	+	+	+
–	–	*	*	*
–	–	*	*	*
z	z	∘	∘	∘

After clustering and permutation

Clustering, Co-clustering

Co-clustered matrix

a	a	+	+	+
a	a	+	+	+
z	z	∘	∘	∘
z	z	∘	∘	∘
–	–	*	*	*
–	–	*	*	*

After co-clustering and permutation

Clustering, Co-clustering

Let $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ be the input matrix.

We cluster *columns* of $\boldsymbol{X}$

Well-known *k-means* clustering problem can be written as

$$\min_{\boldsymbol{B},\boldsymbol{C}} \quad \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{B}\boldsymbol{C}\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad \boldsymbol{C}^T\boldsymbol{C} = \mathrm{Diag}(\text{sizes})$$

where $\boldsymbol{B} \in \mathbb{R}^{m \times k}$, and $\boldsymbol{C} \in \{0,1\}^{k \times n}$.

Clustering, Co-clustering

Let $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ be the input matrix.

We cluster *columns* of $\boldsymbol{X}$

Well-known *k-means* clustering problem can be written as

$$\min_{\boldsymbol{B},\boldsymbol{C}} \quad \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{B}\boldsymbol{C}\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad \boldsymbol{C}^T\boldsymbol{C} = \mathrm{Diag}(\text{sizes})$$

where $\boldsymbol{B} \in \mathbb{R}^{m \times k}$, and $\boldsymbol{C} \in \{0,1\}^{k \times n}$.

Teaser: How would you write a co-clustering problem?

Matrix Completion

Recall the Netflix example.

The general *matrix completion* task is:

Recover a matrix from a sampling of its entries!

Matrix Completion

Recall the Netflix example.

The general *matrix completion* task is:

> Recover a matrix from a sampling of its entries!

A very nice topic in itself – no time to cover today.

Matrix Completion

Recall the Netflix example.

The general *matrix completion* task is:

> Recover a matrix from a sampling of its entries!

A very nice topic in itself – no time to cover today.

One recent result:

> Can perfectly recover most low-rank matrices!

Nearest positive definite

Sometimes one needs to find for a *symmetric* $\boldsymbol{A}$

$$\boxed{\min \quad \|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_F \quad \text{s.t.} \quad \hat{\boldsymbol{A}} \succeq 0}$$

Nearest positive definite

Sometimes one needs to find for a *symmetric* $\boldsymbol{A}$

$$\boxed{\min \quad \|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_{\mathrm{F}} \quad \text{s.t.} \quad \hat{\boldsymbol{A}} \succeq 0}$$

Solution: BoXi06

$\boldsymbol{A} = \boldsymbol{A}_+ - \boldsymbol{A}_-$, $\boldsymbol{A}_+ = \boldsymbol{A}_+^T \succeq 0$, $\boldsymbol{A}_- = \boldsymbol{A}_-^T \succeq 0$, $\boldsymbol{A}_+\boldsymbol{A}_- = 0$. Moreover

$$\|\boldsymbol{A} - \boldsymbol{A}_+\|_{\mathrm{F}} = \|\boldsymbol{A}_-\|_{\mathrm{F}} \leq \|\boldsymbol{A} - \boldsymbol{X}\|_{\mathrm{F}}$$

for *any* $\boldsymbol{X} \succeq 0$. (Observe, computing $\boldsymbol{A}_-$ enough)

Nearest positive definite

Sometimes one needs to find for a *symmetric* $\boldsymbol{A}$

$$\boxed{\min \quad \|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_F \quad \text{s.t.} \quad \hat{\boldsymbol{A}} \succeq 0}$$

Solution: BoXi06

$\boldsymbol{A} = \boldsymbol{A}_+ - \boldsymbol{A}_-$, $\boldsymbol{A}_+ = \boldsymbol{A}_+^T \succeq 0$, $\boldsymbol{A}_- = \boldsymbol{A}_-^T \succeq 0$, $\boldsymbol{A}_+\boldsymbol{A}_- = 0$. Moreover

$$\|\boldsymbol{A} - \boldsymbol{A}_+\|_F = \|\boldsymbol{A}_-\|_F \leq \|\boldsymbol{A} - \boldsymbol{X}\|_F$$

for *any* $\boldsymbol{X} \succeq 0$. (Observe, computing $\boldsymbol{A}_-$ enough)

$$\boxed{\text{Modified Cholesky: } \boldsymbol{A} + \boldsymbol{E} \text{ with } \|\boldsymbol{E}\|_2 = O(n)}$$

Nonnegative matrix approximation (aka NMF)

Say we are seeking a *low-rank approx* $\boldsymbol{A} \approx \boldsymbol{BC}$

We could invoke SVD – but sometimes not desirable:

Nonnegative matrix approximation (aka NMF)

Say we are seeking a *low-rank approx* $\boldsymbol{A} \approx \boldsymbol{BC}$

We could invoke SVD – but sometimes not desirable:

- SVD yields dense $\boldsymbol{B}$ and $\boldsymbol{C}$
- $\boldsymbol{B}$ and $\boldsymbol{C}$ full of negative numbers, even if $\boldsymbol{A} \geq 0$
- SVD decomposition might not be that easy to interpret

Nonnegative matrix approximation (aka NMF)

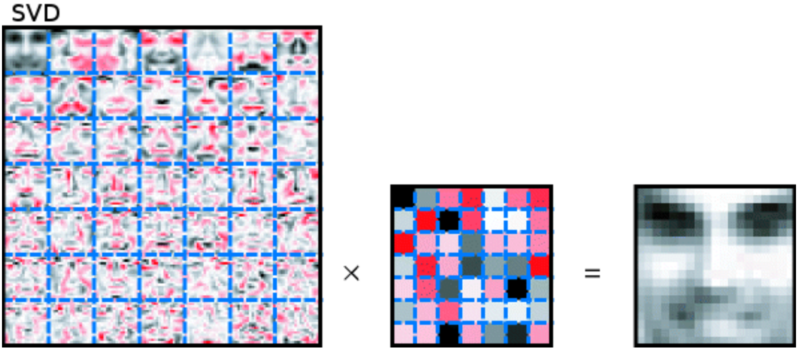
Say we are seeking a *low-rank approx* $\boldsymbol{A} \approx \boldsymbol{BC}$

We could invoke SVD – but sometimes not desirable:

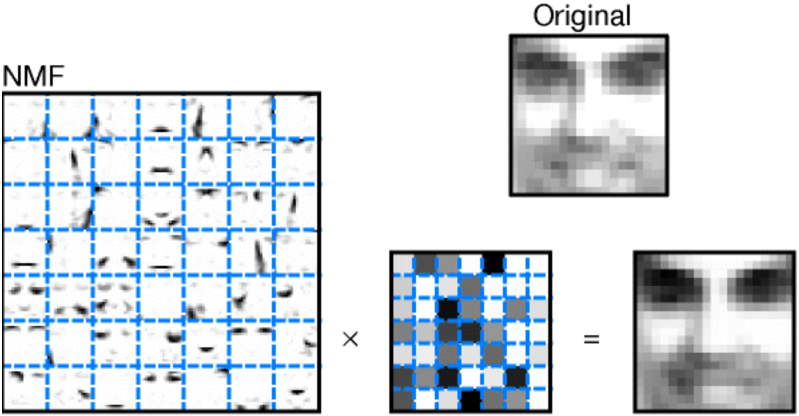
- SVD yields dense $\boldsymbol{B}$ and $\boldsymbol{C}$
- $\boldsymbol{B}$ and $\boldsymbol{C}$ full of negative numbers, even if $\boldsymbol{A} \geq 0$
- SVD decomposition might not be that easy to interpret

So why not impose $\boldsymbol{B} \geq 0$, $\boldsymbol{C} \geq 0$?

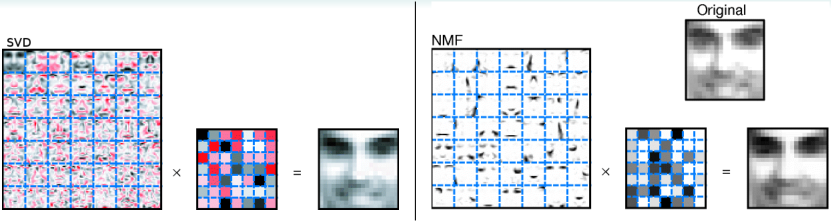
Nonnegative matrix approximation (aka NMF)



Nonnegative matrix approximation (aka NMF)



Nonnegative matrix approximation (aka NMF)



Examples from original Lee/Seung paper on NMA

Other Variants of NMA

- KL-NMA – very interesting variant – more popular for modeling "co-occurrence" data
- Bregman NMA – examples from literature – spam filtering
- Sparsity constrained NMA (Hoyer, etc.)
- Local NMA
- Numerous other variations

Sparsity Constrained Versions

- Sparse PCA
- Semi-discrete decomposition
- Discrete basis problem
- Lasso for variable selection
- Sparse generalized eigenvalue problem
- Other variants

Algorithms & Theory

Algorithms: NMA

We consider the *NMA* problem:

$$\boxed{\boldsymbol{A} \approx \boldsymbol{B}\boldsymbol{C} \quad \text{s.t.} \quad \boldsymbol{B}, \boldsymbol{C} \geq 0.}$$

Algorithms: NMA

Measure quality of approximation using Δ:

$$\boxed{\text{minimize} \quad \Delta(\boldsymbol{A}, \boldsymbol{BC}) \quad \text{s.t.} \quad \boldsymbol{B}, \boldsymbol{C} \geq 0}$$

Algorithms: NMA

Measure quality of approximation using Δ:

$$\boxed{\text{minimize} \quad \Delta(\boldsymbol{A}, \boldsymbol{BC}) \quad \text{s.t.} \quad \boldsymbol{B}, \boldsymbol{C} \geq 0}$$

Instantiations: where Δ is

- $\|\boldsymbol{A} - \boldsymbol{BC}\|_{\mathrm{F}}^2$ – least-squares NMA
- $\|\boldsymbol{A} - \boldsymbol{BC}\|_1$ – robust NMA
- $\mathrm{KL}(\boldsymbol{A}, \boldsymbol{BC})$ – relative entropy (KL) NMA
- $D(\boldsymbol{A}, \boldsymbol{BC})$ – Bregman divergence NMA

Algorithms: NMA

Measure quality of approximation using Δ:

$$\boxed{\text{minimize} \quad \Delta(\boldsymbol{A}, \boldsymbol{BC}) \quad \text{s.t.} \quad \boldsymbol{B}, \boldsymbol{C} \geq 0}$$

Instantiations: where Δ is

- $\|\boldsymbol{A} - \boldsymbol{BC}\|_F^2$ – least-squares NMA
- $\|\boldsymbol{A} - \boldsymbol{BC}\|_1$ – robust NMA
- $\mathrm{KL}(\boldsymbol{A}, \boldsymbol{BC})$ – relative entropy (KL) NMA
- $D(\boldsymbol{A}, \boldsymbol{BC})$ – Bregman divergence NMA

Least-squares NMA

$$\boxed{\text{minimize} \quad \frac{1}{2}\|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{B}, \boldsymbol{C} \geq 0.}$$

- Is this problem solvable?

Least-squares NMA

$$\boxed{\text{minimize} \quad \tfrac{1}{2}\|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{B}, \boldsymbol{C} \geq 0.}$$

- Is this problem solvable? Yes!

Least-squares NMA

$$\boxed{\text{minimize} \quad \frac{1}{2}\|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{B}, \boldsymbol{C} \geq 0.}$$

- Is this problem solvable? Yes!
- Can we find the solution?

Least-squares NMA

$$\boxed{\text{minimize} \quad \tfrac{1}{2}\|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{B}, \boldsymbol{C} \geq 0.}$$

- Is this problem solvable? Yes!
- Can we find the solution? Hmmm

Least-squares NMA

$$\boxed{\text{minimize} \quad \tfrac{1}{2}\|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{B}, \boldsymbol{C} \geq 0.}$$

- Is this problem solvable? Yes!
- Can we find the solution? Hmmm
- In general, NMF is NP-Hard (Vavasis 2007)

Least-squares NMA

$$\boxed{\text{minimize} \quad \tfrac{1}{2}\|\boldsymbol{A} - \boldsymbol{BC}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{B}, \boldsymbol{C} \geq 0.}$$

- Is this problem solvable? Yes!
- Can we find the solution? Hmmm
- In general, NMF is NP-Hard (Vavasis 2007)
- How about merely a locally optimal solution?

Least-squares NMA

$$\boxed{\text{minimize} \quad \frac{1}{2}\|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{B}, \boldsymbol{C} \geq 0.}$$

- Is this problem solvable? Yes!
- Can we find the solution? Hmmm
- In general, NMF is NP-Hard (Vavasis 2007)
- How about merely a locally optimal solution?
- Even that cannot be found easily!

NMA Algorithms

- Hack: “Zero-out” TSVD
- Alternating methods
- Directly optimizing (won’t cover)
- Online algorithms (won’t cover)

NMA Algorithm: Zero-out SVD

Input: $\boldsymbol{A}$, k

1. $[\boldsymbol{U}, \Sigma, \boldsymbol{V}] = \mathrm{SVD}(\boldsymbol{A}, k)$
2. $\boldsymbol{B} \leftarrow \boldsymbol{U}_k \Sigma_k$, $\boldsymbol{C} \leftarrow \boldsymbol{V}_k^T$
3. $\boldsymbol{B} \leftarrow \max(0, \boldsymbol{B})$, $\boldsymbol{C} \leftarrow \max(0, \boldsymbol{C})$

Advantages: Simple, deterministic

Disadvantages: could be slow, no theoretical guarantees, solution can be really bad!

NMA Algorithm: Alternating Methods

Generic Iterative Alternating Descent

1. Initialize $\boldsymbol{B}^0$, $t \leftarrow 0$

NMA Algorithm: Alternating Methods

Generic Iterative Alternating Descent

1. Initialize $\boldsymbol{B}^0$, $t \leftarrow 0$
2. Compute $\boldsymbol{C}^{t+1} \quad \text{s.t.} \quad \Delta(\boldsymbol{A}, \boldsymbol{B}^t\boldsymbol{C}^{t+1}) \leq \Delta(\boldsymbol{A}, \boldsymbol{B}^t\boldsymbol{C}^t)$

NMA Algorithm: Alternating Methods

Generic Iterative Alternating Descent

1. Initialize $\boldsymbol{B}^0$, $t \leftarrow 0$
2. Compute $\boldsymbol{C}^{t+1}$ s.t. $\Delta(\boldsymbol{A}, \boldsymbol{B}^t \boldsymbol{C}^{t+1}) \leq \Delta(\boldsymbol{A}, \boldsymbol{B}^t \boldsymbol{C}^t)$
3. Compute $\boldsymbol{B}^{t+1}$ s.t. $\Delta(\boldsymbol{A}, \boldsymbol{B}^{t+1} \boldsymbol{C}^{t+1}) \leq \Delta(\boldsymbol{A}, \boldsymbol{B}^t \boldsymbol{C}^{t+1})$

NMA Algorithm: Alternating Methods

Generic Iterative Alternating Descent

1. Initialize $\boldsymbol{B}^0$, $t \leftarrow 0$
2. Compute $\boldsymbol{C}^{t+1}$ s.t. $\Delta(\boldsymbol{A}, \boldsymbol{B}^t \boldsymbol{C}^{t+1}) \leq \Delta(\boldsymbol{A}, \boldsymbol{B}^t \boldsymbol{C}^t)$
3. Compute $\boldsymbol{B}^{t+1}$ s.t. $\Delta(\boldsymbol{A}, \boldsymbol{B}^{t+1} \boldsymbol{C}^{t+1}) \leq \Delta(\boldsymbol{A}, \boldsymbol{B}^t \boldsymbol{C}^{t+1})$
4. $t \leftarrow t + 1$, and repeat until stopping criteria met.

For least-squares NMA

$$\|\boldsymbol{A} - \boldsymbol{B}^{t+1}\boldsymbol{C}^{t+1}\|_F^2 \leq \|\boldsymbol{A} - \boldsymbol{B}^{t}\boldsymbol{C}^{t+1}\|_F^2 \leq \|\boldsymbol{A} - \boldsymbol{B}^{t}\boldsymbol{C}^{t}\|_F^2$$

Alternating least-squares

Alternating Least Squares computes

$$\boldsymbol{C} = \operatorname*{argmin}_{\boldsymbol{C}} \quad \|\boldsymbol{A} - \boldsymbol{B}^t \boldsymbol{C}\|_F^2;$$

Alternating least-squares

Alternating Least Squares computes

$$\boldsymbol{C} = \underset{\boldsymbol{C}}{\operatorname{argmin}} \quad \|\boldsymbol{A} - \boldsymbol{B}^t \boldsymbol{C}\|_F^2; \qquad\qquad \boldsymbol{C}^{t+1} \leftarrow \max(0, \boldsymbol{C})$$

Alternating least-squares

Alternating Least Squares computes

$$\boldsymbol{C} = \underset{\boldsymbol{C}}{\operatorname{argmin}} \quad \|\boldsymbol{A} - \boldsymbol{B}^t\boldsymbol{C}\|_F^2; \qquad\qquad \boldsymbol{C}^{t+1} \leftarrow \max(0, \boldsymbol{C})$$

$$\boldsymbol{B} = \underset{\boldsymbol{B}}{\operatorname{argmin}} \quad \|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}^{t+1}\|_F^2;$$

Alternating least-squares

Alternating Least Squares computes

$$\boldsymbol{C} = \operatorname*{argmin}_{\boldsymbol{C}} \|\boldsymbol{A} - \boldsymbol{B}^t\boldsymbol{C}\|_F^2; \qquad \boldsymbol{C}^{t+1} \leftarrow \max(0, \boldsymbol{C})$$

$$\boldsymbol{B} = \operatorname*{argmin}_{\boldsymbol{B}} \|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}^{t+1}\|_F^2; \qquad \boldsymbol{B}^{t+1} \leftarrow \max(0, \boldsymbol{B})$$

Alternating least-squares

Alternating Least Squares computes

$$\boldsymbol{C} = \operatorname*{argmin}_{\boldsymbol{C}} \quad \|\boldsymbol{A} - \boldsymbol{B}^t\boldsymbol{C}\|_F^2; \qquad\qquad \boldsymbol{C}^{t+1} \leftarrow \max(0, \boldsymbol{C})$$

$$\boldsymbol{B} = \operatorname*{argmin}_{\boldsymbol{B}} \quad \|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}^{t+1}\|_F^2; \qquad\qquad \boldsymbol{B}^{t+1} \leftarrow \max(0, \boldsymbol{B})$$

ALS is fast, simple, often effective, but ...

Alternating least-squares

Alternating Least Squares computes

$$\boldsymbol{C} = \underset{\boldsymbol{C}}{\operatorname{argmin}} \quad \|\boldsymbol{A} - \boldsymbol{B}^t\boldsymbol{C}\|_F^2; \qquad \boldsymbol{C}^{t+1} \leftarrow \max(0, \boldsymbol{C})$$

$$\boldsymbol{B} = \underset{\boldsymbol{B}}{\operatorname{argmin}} \quad \|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}^{t+1}\|_F^2; \qquad \boldsymbol{B}^{t+1} \leftarrow \max(0, \boldsymbol{B})$$

ALS is fast, simple, often effective, but ...

Bad News!

Alternating least-squares

Alternating Least Squares computes

$$C = \underset{C}{\operatorname{argmin}} \; \|A - B^t C\|_F^2; \qquad C^{t+1} \leftarrow \max(0, C)$$

$$B = \underset{B}{\operatorname{argmin}} \; \|A - BC^{t+1}\|_F^2; \qquad B^{t+1} \leftarrow \max(0, B)$$

ALS is fast, simple, often effective, but ...

Bad News!

$$\|A - B^{t+1}C^{t+1}\|_F^2 \le \|A - B^t C^{t+1}\|_F^2 \le \|A - B^t C^t\|_F^2$$

is NOT guaranteed!

Alternating NNLS

"Simple" fix is to instead compute

$$C^{t+1} = \operatorname*{argmin}_{C} \quad \|A - B^t C\|_F^2 \quad \text{s.t.} \quad C \geq 0$$

Alternating NNLS

"Simple" fix is to instead compute

$$\boldsymbol{C}^{t+1} = \underset{\boldsymbol{C}}{\operatorname{argmin}} \quad \|\boldsymbol{A} - \boldsymbol{B}^t \boldsymbol{C}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{C} \geq 0$$

$$\boldsymbol{B}^{t+1} = \underset{\boldsymbol{B}}{\operatorname{argmin}} \quad \|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}^{t+1}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{B} \geq 0$$

Alternating NNLS

"Simple" fix is to instead compute

$$\boldsymbol{C}^{t+1} = \operatorname*{argmin}_{\boldsymbol{C}} \quad \|\boldsymbol{A} - \boldsymbol{B}^t\boldsymbol{C}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{C} \geq 0$$

$$\boldsymbol{B}^{t+1} = \operatorname*{argmin}_{\boldsymbol{B}} \quad \|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}^{t+1}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{B} \geq 0$$

Advantages: Descent is guaranteed; even convergence to local-min!

Alternating NNLS

"Simple" fix is to instead compute

$$\boldsymbol{C}^{t+1} = \underset{\boldsymbol{C}}{\operatorname{argmin}} \quad \|\boldsymbol{A} - \boldsymbol{B}^t\boldsymbol{C}\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad \boldsymbol{C} \geq 0$$

$$\boldsymbol{B}^{t+1} = \underset{\boldsymbol{B}}{\operatorname{argmin}} \quad \|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}^{t+1}\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad \boldsymbol{B} \geq 0$$

Advantages: Descent is guaranteed; even convergence to local-min!
Disadvantages: More complicated optimization problem, slower than ALS

Alternating NNLS

"Simple" fix is to instead compute

$$\boldsymbol{C}^{t+1} = \operatorname*{argmin}_{\boldsymbol{C}} \quad \|\boldsymbol{A} - \boldsymbol{B}^t \boldsymbol{C}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{C} \geq 0$$

$$\boldsymbol{B}^{t+1} = \operatorname*{argmin}_{\boldsymbol{B}} \quad \|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}^{t+1}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{B} \geq 0$$

Advantages: Descent is guaranteed; even convergence to local-min!
Disadvantages: More complicated optimization problem, slower than ALS

How to solve the "argmin"??

Alternating NNLS – subproblem

The *nonnegative least squares* (NNLS) subproblem is

$$\boxed{\min_{\boldsymbol{C}\geq 0} \quad \frac{1}{2}\|\boldsymbol{A}-\boldsymbol{BC}\|_F^2}$$

Essentially the same as solving

$$\boxed{\min_{\boldsymbol{c}\geq 0} \quad f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a}-\boldsymbol{Bc}\|_2^2}$$

Alternating NNLS – subproblem

The *nonnegative least squares* (NNLS) subproblem is

$$\boxed{\min_{\boldsymbol{C}\geq 0} \quad \frac{1}{2}\|\boldsymbol{A}-\boldsymbol{B}\boldsymbol{C}\|_F^2}$$

Essentially the same as solving

$$\boxed{\min_{\boldsymbol{c}\geq 0} \quad f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a}-\boldsymbol{B}\boldsymbol{c}\|_2^2}$$

- Nice, convex optimization problem
- Numerous algorithms for solving
- Let us look at the simplest

Background – Gradient Methods

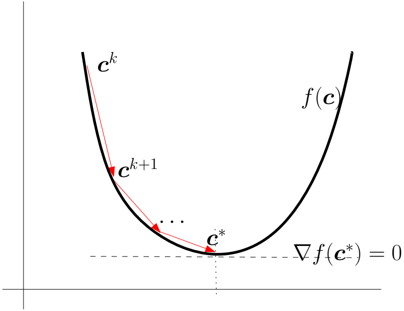
Consider first the *unconstrained* problem

$$\boxed{\min \quad f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_2^2}$$

Background – Gradient Methods

Consider first the *unconstrained* problem

$$\boxed{\min \quad f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_2^2}$$



Familiar gradient descent

Background – Gradient Methods

Gradient descent: Vector $\boldsymbol{c}^{k+1}$ is chosen as

$$\boxed{\boldsymbol{c}^{k+1} = \boldsymbol{c}^k - \alpha_k \nabla f(\boldsymbol{c}^k), \quad k = 0, 1, \ldots}$$

- **Step-size** $\alpha_k \geq 0$
- **Descent direction** $-\nabla f(\boldsymbol{c}^k)$

Background – Gradient Methods

Gradient descent: Vector $\boldsymbol{c}^{k+1}$ is chosen as

$$\boxed{\boldsymbol{c}^{k+1} = \boldsymbol{c}^k - \alpha_k \nabla f(\boldsymbol{c}^k), \quad k = 0, 1, \ldots}$$

- **Step-size** $\alpha_k \geq 0$
- **Descent direction** $-\nabla f(\boldsymbol{c}^k)$

More generally, *Gradient methods* iterate as

$$\boxed{\boldsymbol{c}^{k+1} = \boldsymbol{c}^k + \alpha_k \boldsymbol{d}^k, \quad k = 0, 1, \ldots}$$

where the descent direction is

$$\boxed{\boldsymbol{d}^k \text{ such that } \langle \boldsymbol{d}^k, \nabla f(\boldsymbol{c}^k) \rangle < 0}$$

Gradient Methods

Gradient methods

$$\boxed{\boldsymbol{c}^{k+1} = \boldsymbol{c}^k + \alpha_k \boldsymbol{d}^k, \quad k = 0, 1, \dots}$$

- Different choices of $\boldsymbol{d}^k$
  - Scaled gradient $\boldsymbol{d}^k = -\boldsymbol{D}^k \nabla f(\boldsymbol{c}^k)$, $\boldsymbol{D}^k \succ 0$
  - Note: $\boldsymbol{D}^k = \boldsymbol{I}$ gives *steepest descent*
  - Newton's method, conjugate gradients, etc.

Gradient Methods

Gradient methods

$$\boxed{\boldsymbol{c}^{k+1} = \boldsymbol{c}^k + \alpha_k \boldsymbol{d}^k, \quad k = 0, 1, \ldots}$$

- Different choices of $\boldsymbol{d}^k$
  - Scaled gradient $\boldsymbol{d}^k = -\boldsymbol{D}^k \nabla f(\boldsymbol{c}^k)$, $\boldsymbol{D}^k \succ 0$
  - Note: $\boldsymbol{D}^k = \boldsymbol{I}$ gives *steepest descent*
  - Newton's method, conjugate gradients, etc.
- Different choices of α_k
  - Limited minimization $\alpha_k = \mathrm{argmin}_{0 \leq \alpha \leq s} f(\boldsymbol{c}^k + \alpha \boldsymbol{d}^k)$
  - Armijo-line-search, backtracking, etc.

Gradient Methods

Gradient methods

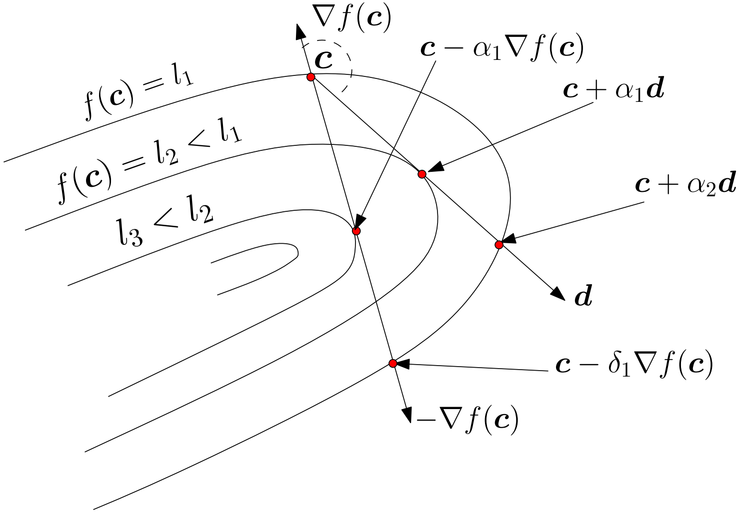
$$\boxed{\boldsymbol{c}^{k+1} = \boldsymbol{c}^k + \alpha_k \boldsymbol{d}^k, \quad k = 0, 1, \ldots}$$

- Different choices of $\boldsymbol{d}^k$
  - Scaled gradient $\boldsymbol{d}^k = -\boldsymbol{D}^k \nabla f(\boldsymbol{c}^k)$, $\boldsymbol{D}^k \succ 0$
  - Note: $\boldsymbol{D}^k = \boldsymbol{I}$ gives *steepest descent*
  - Newton's method, conjugate gradients, etc.
- Different choices of α_k
  - Limited minimization $\alpha_k = \mathrm{argmin}_{0 \le \alpha \le s} f(\boldsymbol{c}^k + \alpha \boldsymbol{d}^k)$
  - Armijo-line-search, backtracking, etc.

Step-sizes α_k chosen to ensure *descent*

$$f(\boldsymbol{c}^{k+1}) < f(\boldsymbol{c}^k)$$

Gradient Methods – Illustration



(adapted from Bertsekas, Nonlinear Programming)

Gradient Methods – Handling constraints

Our problem is **constrained**

$$\boxed{\min_{\mathbf{c}\geq 0} \quad f(\mathbf{c}) = \frac{1}{2}\|\mathbf{a} - \mathbf{B}\mathbf{c}\|_F^2}$$

Recall gradient-descent iteration

$$\boxed{\mathbf{c}^{k+1} = \quad \mathbf{c}^k - \alpha_k \nabla f(\mathbf{c}^k) \ , \quad k = 0, 1, \ldots}$$

Gradient Methods – Handling constraints

Our problem is **constrained**

$$\boxed{\min_{\boldsymbol{c}\geq 0} \quad f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_{\mathrm{F}}^2}$$

Replace it with *Gradient-Projection*!

$$\boxed{\boldsymbol{c}^{k+1} = P_+(\boldsymbol{c}^k - \alpha_k \nabla f(\boldsymbol{c}^k)), \quad k = 0, 1, \ldots}$$

$P_+\boldsymbol{x} = \max(0, \boldsymbol{x})$: projection to ensure *non-negativity*

Gradient Methods – Handling constraints

Our problem is **constrained**

$$\boxed{\min_{\boldsymbol{c}\geq 0} \quad f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_{\mathrm{F}}^{2}}$$

Replace it with *Gradient-Projection*!

$$\boxed{\boldsymbol{c}^{k+1} = P_{+}(\boldsymbol{c}^{k} - \alpha_k \nabla f(\boldsymbol{c}^{k})), \quad k = 0, 1, \ldots}$$

$P_{+}\boldsymbol{x} = \max(0, \boldsymbol{x})$: projection to ensure *non-negativity*

Note: Step-size α_k selected to ensure descent

$$f(\boldsymbol{c}^{k+1}) < f(\boldsymbol{c}^{k})$$

Alternating NNLS – summary

$$\boxed{\text{minimize} \quad \tfrac{1}{2}\|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{B}, \boldsymbol{C} \geq 0.}$$

Alternating NNLS – summary

$$\boxed{\text{minimize} \quad \tfrac{1}{2}\|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{B}, \boldsymbol{C} \geq 0.}$$

by alternating

$$\begin{aligned}
\boldsymbol{C}^{t+1} &= \operatorname*{argmin}_{\boldsymbol{C}\geq 0} \quad F(\boldsymbol{C}) = \|\boldsymbol{A} - \boldsymbol{B}^t\boldsymbol{C}\|_F^2 \\
\boldsymbol{B}^{t+1} &= \operatorname*{argmin}_{\boldsymbol{B}\geq 0} \quad F(\boldsymbol{B}) = \|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}^{t+1}\|_F^2,
\end{aligned}$$

Alternating NNLS – summary

$$\boxed{\text{minimize} \quad \frac{1}{2}\|\boldsymbol{A} - \boldsymbol{BC}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{B}, \boldsymbol{C} \geq 0.}$$

by alternating

$$\begin{aligned} \boldsymbol{C}^{t+1} &= \underset{\boldsymbol{C} \geq 0}{\operatorname{argmin}} \quad F(\boldsymbol{C}) = \|\boldsymbol{A} - \boldsymbol{B}^t \boldsymbol{C}\|_F^2 \\ \boldsymbol{B}^{t+1} &= \underset{\boldsymbol{B} \geq 0}{\operatorname{argmin}} \quad F(\boldsymbol{B}) = \|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}^{t+1}\|_F^2, \end{aligned}$$

where each of the subproblems is solved (for fixed t) via

$$\boxed{\boldsymbol{C}^{k+1} = P_+(\boldsymbol{C}^k - \alpha_k \nabla F(\boldsymbol{C}^k)), \quad k = 0, 1, \ldots}$$

Alternating NNLS – summary

$$\boxed{\text{minimize} \quad \frac{1}{2}\|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad \boldsymbol{B}, \boldsymbol{C} \geq 0.}$$

by alternating

$$\begin{aligned} \boldsymbol{C}^{t+1} &= \underset{\boldsymbol{C} \geq 0}{\operatorname{argmin}} \quad F(\boldsymbol{C}) = \|\boldsymbol{A} - \boldsymbol{B}^t \boldsymbol{C}\|_{\mathrm{F}}^2 \\ \boldsymbol{B}^{t+1} &= \underset{\boldsymbol{B} \geq 0}{\operatorname{argmin}} \quad F(\boldsymbol{B}) = \|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}^{t+1}\|_{\mathrm{F}}^2, \end{aligned}$$

where each of the subproblems is solved (for fixed t) via

$$\boxed{\boldsymbol{C}^{k+1} = P_+(\boldsymbol{C}^k - \alpha_k \nabla F(\boldsymbol{C}^k)), \quad k = 0, 1, \ldots}$$

So are we ready to implement this?

Alternating NNLS – summary

$$\boxed{\text{minimize} \quad \frac{1}{2}\|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{B}, \boldsymbol{C} \geq 0.}$$

by alternating

$$\begin{aligned} \boldsymbol{C}^{t+1} &= \operatorname*{argmin}_{\boldsymbol{C}\geq 0} \quad F(\boldsymbol{C}) = \|\boldsymbol{A} - \boldsymbol{B}^t\boldsymbol{C}\|_F^2 \\ \boldsymbol{B}^{t+1} &= \operatorname*{argmin}_{\boldsymbol{B}\geq 0} \quad F(\boldsymbol{B}) = \|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}^{t+1}\|_F^2, \end{aligned}$$

where each of the subproblems is solved (for fixed t) via

$$\boxed{\boldsymbol{C}^{k+1} = P_+(\boldsymbol{C}^k - \alpha_k \nabla F(\boldsymbol{C}^k)), \quad k = 0, 1, \ldots}$$

So are we ready to implement this?
How to compute $\nabla F(\boldsymbol{C}^k)$?

Background – Matrix Derivatives

Derivative of $f : \mathbb{R}^{m \times n} \to \mathbb{R}$ is defined as

$$\frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}} \triangleq \left[\frac{\partial f(\boldsymbol{X})}{\partial x_{pq}} \right]$$

I. Compute $\partial \mathrm{Tr}(\boldsymbol{X}\boldsymbol{Y}) / \partial \boldsymbol{X}$

Background – Matrix Derivatives

Derivative of $f : \mathbb{R}^{m \times n} \to \mathbb{R}$ is defined as

$$\frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}} \triangleq \left[\frac{\partial f(\boldsymbol{X})}{\partial x_{pq}} \right]$$

I. Compute $\partial \mathrm{Tr}(\boldsymbol{X}\boldsymbol{Y})/\partial \boldsymbol{X}$
Recall $\mathrm{Tr}(\boldsymbol{X}\boldsymbol{Y}) = \sum_{ij} x_{ij} y_{ji}$. Hence, $\partial \mathrm{Tr}(\boldsymbol{X}\boldsymbol{Y})/\partial \boldsymbol{X} = \boldsymbol{Y}^T$.

Background – Matrix Derivatives

Derivative of $f : \mathbb{R}^{m \times n} \to \mathbb{R}$ is defined as

$$\frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}} \triangleq \left[\frac{\partial f(\boldsymbol{X})}{\partial x_{pq}} \right]$$

II. Verify that: $\partial \|\boldsymbol{X}\|_F^2 / \partial \boldsymbol{X} = 2\boldsymbol{X}$

Background – Matrix Derivatives

Derivative of $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ is defined as

$$\frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}} \triangleq \left[\frac{\partial f(\boldsymbol{X})}{\partial x_{pq}}\right]$$

II. Verify that: $\partial \|\boldsymbol{X}\|_F^2 / \partial \boldsymbol{X} = 2\boldsymbol{X}$

Solution:

Recall that $\|\boldsymbol{X}\|_F^2 = \mathrm{Tr}(\boldsymbol{X}^T\boldsymbol{X})$. So,

$$\frac{\partial \|\boldsymbol{X}\|_F^2}{\partial \boldsymbol{X}} = \frac{\partial \mathrm{Tr}(\boldsymbol{X}^T\boldsymbol{X})}{\partial x_{pq}} = \frac{\partial (\sum_{ij} x_{ij}^2)}{\partial x_{pq}} = 2x_{pq}.$$

Background – Matrix Derivatives

Derivative of $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ is defined as

$$\frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}} \triangleq \left[\frac{\partial f(\boldsymbol{X})}{\partial x_{pq}} \right]$$

III. Verify that: $\partial \mathrm{Tr}(\boldsymbol{X}^T \boldsymbol{A}\boldsymbol{X}) / \partial \boldsymbol{X} = (\boldsymbol{A} + \boldsymbol{A}^T)\boldsymbol{X}$

Background – Matrix Derivatives

Derivative of $f : \mathbb{R}^{m\times n} \rightarrow \mathbb{R}$ is defined as

$$\frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}} \triangleq \left[\frac{\partial f(\boldsymbol{X})}{\partial x_{pq}} \right]$$

III. Verify that: $\partial \mathrm{Tr}(\boldsymbol{X}^T \boldsymbol{A}\boldsymbol{X})/\partial \boldsymbol{X} = (\boldsymbol{A} + \boldsymbol{A}^T)\boldsymbol{X}$

Solution: Brute force

$$\mathrm{Tr}(\boldsymbol{X}^T \boldsymbol{A}\boldsymbol{X}) = \sum_{ij} x_{ij} (\boldsymbol{A}\boldsymbol{X})_{ji} = \sum_{ijk} x_{ij} a_{jk} x_{ki}$$

Background – Matrix Derivatives

Derivative of $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ is defined as

$$\frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}} \triangleq \left[\frac{\partial f(\boldsymbol{X})}{\partial x_{pq}} \right]$$

Exercise: IV.

Let $F(\boldsymbol{C}) = \frac{1}{2}\|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}\|_{\mathrm{F}}^2$; compute $\partial F / \partial \boldsymbol{C}$

Background – Matrix Derivatives

Derivative of $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ is defined as

$$\frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}} \triangleq \left[\frac{\partial f(\boldsymbol{X})}{\partial x_{pq}}\right]$$

Exercise: IV.

Let $F(\boldsymbol{C}) = \frac{1}{2}\|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}\|_{\mathrm{F}}^2$; compute $\partial F/\partial \boldsymbol{C}$

Solution:

$F(\boldsymbol{C}) = \|\boldsymbol{A}\|_{\mathrm{F}}^2 - 2\,\mathrm{Tr}(\boldsymbol{C}\boldsymbol{A}^T\boldsymbol{B}) + \mathrm{Tr}(\boldsymbol{C}^T\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{C})$

$$\frac{\partial F(\boldsymbol{C})}{\partial \boldsymbol{C}} = -2\boldsymbol{B}^T\boldsymbol{A} + 2\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{C}.$$

In passing: The Fréchet derivative

Given $f : V \to W$, the *Fréchet differential* at point $\boldsymbol{X}$ is the linear-mapping L that satisfies for all $\boldsymbol{E} \in V$ the relation

$$\boxed{f(\boldsymbol{X} + \boldsymbol{E}) - f(\boldsymbol{X}) - L(\boldsymbol{X}, \boldsymbol{E}) = o(\|\boldsymbol{E}\|)}$$

The *Fréchet derivative* $D_f(\boldsymbol{X})$ (of f at point $\boldsymbol{X}$) identified via:

$$\boxed{L(\boldsymbol{X}, \boldsymbol{E}) = D_f(\boldsymbol{X})(\boldsymbol{E})}$$

Can be used to develop matrix calculus formally.

Implementation

Exercise: LSNMA

Implement the gradient-projection NMA algorithm

Exercise: Complexity

What is the computational complexity per (major) iteration?

Implementation

Exercise: LSNMA

Implement the gradient-projection NMA algorithm

Exercise: Complexity

What is the computational complexity per (major) iteration?

Solution:

A lot! Especially since there might be many (inner) gradient projection iterations for each major iteration.

What to do?

Alternating descent

Idea! Do not insist on minimization

Alternating descent

Idea! Do not insist on minimization

Recall that we originally wanted *descent*

$$\|\boldsymbol{A} - \boldsymbol{B}^{t+1}\boldsymbol{C}^{t+1}\|_F^2 \le \|\boldsymbol{A} - \boldsymbol{B}^{t}\boldsymbol{C}^{t+1}\|_F^2 \le \|\boldsymbol{A} - \boldsymbol{B}^{t}\boldsymbol{C}^{t}\|_F^2$$

Alternating descent

Idea! Do not insist on minimization

Recall that we originally wanted *descent*

$$\|\boldsymbol{A} - \boldsymbol{B}^{t+1}\boldsymbol{C}^{t+1}\|_F^2 \leq \|\boldsymbol{A} - \boldsymbol{B}^{t}\boldsymbol{C}^{t+1}\|_F^2 \leq \|\boldsymbol{A} - \boldsymbol{B}^{t}\boldsymbol{C}^{t}\|_F^2$$

- For each major (t) iteration, run few inner iterations
- Each inner iteration descends, so overall descent ensured

Alternating descent

Idea! Do not insist on minimization

Recall that we originally wanted *descent*

$$\|\boldsymbol{A} - \boldsymbol{B}^{t+1}\boldsymbol{C}^{t+1}\|_F^2 \le \|\boldsymbol{A} - \boldsymbol{B}^{t}\boldsymbol{C}^{t+1}\|_F^2 \le \|\boldsymbol{A} - \boldsymbol{B}^{t}\boldsymbol{C}^{t}\|_F^2$$

- For each major (t) iteration, run few inner iterations
- Each inner iteration descends, so overall descent ensured
- Instead: approximate gradient-projection algorithm

Alternating descent

Idea! Do not insist on minimization

Recall that we originally wanted *descent*

$$\|\mathbf{A} - \mathbf{B}^{t+1}\mathbf{C}^{t+1}\|_F^2 \le \|\mathbf{A} - \mathbf{B}^{t}\mathbf{C}^{t+1}\|_F^2 \le \|\mathbf{A} - \mathbf{B}^{t}\mathbf{C}^{t}\|_F^2$$

- For each major (t) iteration, run few inner iterations
- Each inner iteration descends, so overall descent ensured
- Instead: approximate gradient-projection algorithm

There exists a more popular alternating-descent algorithm!

Multiplicative Updates

The Lee & Seung Algorithm

Lee & Seung (2000) proposed the following "algorithm"

$$\boldsymbol{C}' \leftarrow \boldsymbol{C} \odot \frac{\boldsymbol{B}^T \boldsymbol{A}}{\boldsymbol{B}^T \boldsymbol{B} \boldsymbol{C}}$$
$$\boldsymbol{B}' \leftarrow \boldsymbol{B} \odot \frac{\boldsymbol{A} \boldsymbol{C}'^T}{\boldsymbol{B} \boldsymbol{C}' \boldsymbol{C}'^T}.$$

This algorithm's simplicity made NMA popular.

Note: $\boldsymbol{A} \odot \boldsymbol{B} = [a_{ij} b_{ij}]$ – *elementwise multiplication*

The Lee & Seung Algorithm

Lee & Seung (2000) proposed the following "algorithm"

$$\boldsymbol{C}' \leftarrow \boldsymbol{C} \odot \frac{\boldsymbol{B}^T \boldsymbol{A}}{\boldsymbol{B}^T \boldsymbol{B} \boldsymbol{C}}$$

$$\boldsymbol{B}' \leftarrow \boldsymbol{B} \odot \frac{\boldsymbol{A} \boldsymbol{C}'^T}{\boldsymbol{B} \boldsymbol{C}' \boldsymbol{C}'^T}.$$

This algorithm's simplicity made NMA popular.

Note: $\boldsymbol{A} \odot \boldsymbol{B} = [a_{ij} b_{ij}]$ – *elementwise multiplication*

- Easy to see that nonnegativity respected

The Lee & Seung Algorithm

Lee & Seung (2000) proposed the following "algorithm"

$$\boldsymbol{C}' \leftarrow \boldsymbol{C} \odot \frac{\boldsymbol{B}^T\boldsymbol{A}}{\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{C}}$$

$$\boldsymbol{B}' \leftarrow \boldsymbol{B} \odot \frac{\boldsymbol{A}\boldsymbol{C}'^T}{\boldsymbol{B}\boldsymbol{C}'\boldsymbol{C}'^T}.$$

This algorithm's simplicity made NMA popular.

Note: $\boldsymbol{A} \odot \boldsymbol{B} = [a_{ij}b_{ij}]$ – *elementwise multiplication*

- Easy to see that nonnegativity respected
- Somewhat harder to prove descent

$$\|\boldsymbol{A} - \boldsymbol{B}'\boldsymbol{C}'\|_F^2 \le \|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}'\|_F^2 \le \|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}\|_F^2$$

Multiplicative updates – preliminaries

Let $\boldsymbol{c}$ be an arbitrary column of $\boldsymbol{C}$. Consider the subproblem:

$$\boxed{\min_{\boldsymbol{c}\geq 0} \quad f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_F^2}$$

A general technique for deriving “descent” methods:

Multiplicative updates – preliminaries

Let $\boldsymbol{c}$ be an arbitrary column of $\boldsymbol{C}$. Consider the subproblem:

$$\boxed{\min_{\boldsymbol{c}\geq 0} \quad f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_F^2}$$

A general technique for deriving "descent" methods:

1. Find a function $g(\boldsymbol{c}, \tilde{\boldsymbol{c}})$ that satisfies:

$$\begin{aligned} g(\boldsymbol{c}, \boldsymbol{c}) &= f(\boldsymbol{c}), \quad \text{for all} \quad \boldsymbol{c}, \\ g(\boldsymbol{c}, \tilde{\boldsymbol{c}}) &\geq f(\boldsymbol{c}), \quad \text{for all} \quad \boldsymbol{c}, \tilde{\boldsymbol{c}}. \end{aligned}$$

Multiplicative updates – preliminaries

Let $\boldsymbol{c}$ be an arbitrary column of $\boldsymbol{C}$. Consider the subproblem:

$$\boxed{\min_{\boldsymbol{c}\geq 0} \quad f(\boldsymbol{c}) = \tfrac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_F^2}$$

A general technique for deriving "descent" methods:

1. Find a function $g(\boldsymbol{c}, \tilde{\boldsymbol{c}})$ that satisfies:

$$\begin{aligned} g(\boldsymbol{c}, \boldsymbol{c}) &= f(\boldsymbol{c}), \quad \text{for all} \quad \boldsymbol{c}, \\ g(\boldsymbol{c}, \tilde{\boldsymbol{c}}) &\geq f(\boldsymbol{c}), \quad \text{for all} \quad \boldsymbol{c}, \tilde{\boldsymbol{c}}. \end{aligned}$$

2. Compute $\boldsymbol{c}^{t+1} = \operatorname{argmin}_{\boldsymbol{c}}\, g(\boldsymbol{c}, \boldsymbol{c}^t)$

Multiplicative updates – preliminaries

Let $\boldsymbol{c}$ be an arbitrary column of $\boldsymbol{C}$. Consider the subproblem:

$$\boxed{\min_{\boldsymbol{c}\geq 0} \quad f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_F^2}$$

A general technique for deriving “descent” methods:

1. Find a function $g(\boldsymbol{c}, \tilde{\boldsymbol{c}})$ that satisfies:

$$\begin{aligned} g(\boldsymbol{c}, \boldsymbol{c}) &= f(\boldsymbol{c}), \quad \text{for all} \quad \boldsymbol{c}, \\ g(\boldsymbol{c}, \tilde{\boldsymbol{c}}) &\geq f(\boldsymbol{c}), \quad \text{for all} \quad \boldsymbol{c}, \tilde{\boldsymbol{c}}. \end{aligned}$$

2. Compute $\boldsymbol{c}^{t+1} = \operatorname{argmin}_{\boldsymbol{c}}\, g(\boldsymbol{c}, \boldsymbol{c}^t)$
3. Then we have descent

$$f(\boldsymbol{c}^{t+1})$$

Multiplicative updates – preliminaries

Let $\boldsymbol{c}$ be an arbitrary column of $\boldsymbol{C}$. Consider the subproblem:

$$\boxed{\min_{\boldsymbol{c}\geq 0} \quad f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_F^2}$$

A general technique for deriving "descent" methods:

1. Find a function $g(\boldsymbol{c}, \tilde{\boldsymbol{c}})$ that satisfies:

$$\begin{aligned} g(\boldsymbol{c}, \boldsymbol{c}) &= f(\boldsymbol{c}), \quad \text{for all} \quad \boldsymbol{c}, \\ g(\boldsymbol{c}, \tilde{\boldsymbol{c}}) &\geq f(\boldsymbol{c}), \quad \text{for all} \quad \boldsymbol{c}, \tilde{\boldsymbol{c}}. \end{aligned}$$

2. Compute $\boldsymbol{c}^{t+1} = \operatorname{argmin}_{\boldsymbol{c}}\, g(\boldsymbol{c}, \boldsymbol{c}^t)$
3. Then we have descent

$$f(\boldsymbol{c}^{t+1}) \overset{\text{def}}{\leq} g(\boldsymbol{c}^{t+1}, \boldsymbol{c}^t)$$

Multiplicative updates – preliminaries

Let $\boldsymbol{c}$ be an arbitrary column of $\boldsymbol{C}$. Consider the subproblem:

$$\boxed{\min_{\boldsymbol{c}\geq 0} \quad f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_F^2}$$

A general technique for deriving "descent" methods:

1. Find a function $g(\boldsymbol{c}, \tilde{\boldsymbol{c}})$ that satisfies:

$$\begin{aligned} g(\boldsymbol{c}, \boldsymbol{c}) &= f(\boldsymbol{c}), \quad \text{for all} \quad \boldsymbol{c}, \\ g(\boldsymbol{c}, \tilde{\boldsymbol{c}}) &\geq f(\boldsymbol{c}), \quad \text{for all} \quad \boldsymbol{c}, \tilde{\boldsymbol{c}}. \end{aligned}$$

2. Compute $\boldsymbol{c}^{t+1} = \operatorname{argmin}_{\boldsymbol{c}} g(\boldsymbol{c}, \boldsymbol{c}^t)$
3. Then we have descent

$$f(\boldsymbol{c}^{t+1}) \overset{\text{def}}{\leq} g(\boldsymbol{c}^{t+1}, \boldsymbol{c}^t) \overset{\text{argmin}}{\leq} g(\boldsymbol{c}^t, \boldsymbol{c}^t)$$

Multiplicative updates – preliminaries

Let $\boldsymbol{c}$ be an arbitrary column of $\boldsymbol{C}$. Consider the subproblem:

$$\boxed{\min_{\boldsymbol{c}\geq 0} \quad f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_F^2}$$

A general technique for deriving "descent" methods:

1. Find a function $g(\boldsymbol{c}, \tilde{\boldsymbol{c}})$ that satisfies:

$$\begin{aligned} g(\boldsymbol{c}, \boldsymbol{c}) &= f(\boldsymbol{c}), \quad \text{for all} \quad \boldsymbol{c}, \\ g(\boldsymbol{c}, \tilde{\boldsymbol{c}}) &\geq f(\boldsymbol{c}), \quad \text{for all} \quad \boldsymbol{c}, \tilde{\boldsymbol{c}}. \end{aligned}$$

2. Compute $\boldsymbol{c}^{t+1} = \operatorname{argmin}_{\boldsymbol{c}}\, g(\boldsymbol{c}, \boldsymbol{c}^t)$
3. Then we have descent

$$f(\boldsymbol{c}^{t+1}) \overset{\text{def}}{\leq} g(\boldsymbol{c}^{t+1}, \boldsymbol{c}^t) \overset{\text{argmin}}{\leq} g(\boldsymbol{c}^t, \boldsymbol{c}^t) \overset{\text{def}}{=} f(\boldsymbol{c}^t)$$

Constructing g

- Main difficulty for $f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_2^2$ due to $\boldsymbol{B}\boldsymbol{c}$
- We need to decouple $\boldsymbol{B}\boldsymbol{c}$ — let's see how.

Constructing g

- Main difficulty for $f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_2^2$ due to $\boldsymbol{B}\boldsymbol{c}$
- We need to decouple $\boldsymbol{B}\boldsymbol{c}$ — let's see how.

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

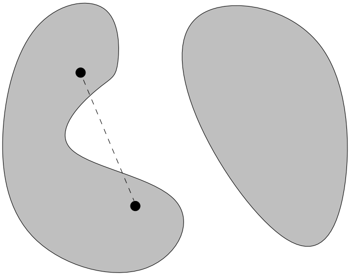
$$\boxed{h(\sum_i \lambda_i x_i) \leq \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \geq 0, \sum_i \lambda_i = 1}$$

Constructing g

- Main difficulty for $f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_2^2$ due to $\boldsymbol{B}\boldsymbol{c}$
- We need to decouple $\boldsymbol{B}\boldsymbol{c}$ — let's see how.

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h(\sum_i \lambda_i x_i) \leq \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \geq 0, \sum_i \lambda_i = 1}$$



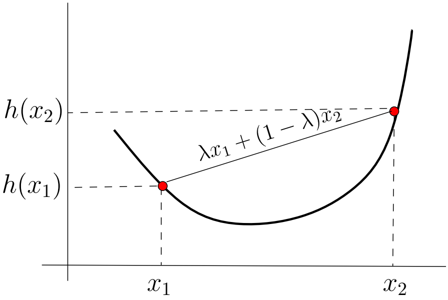
Non-convex, and a convex set

Constructing g

- Main difficulty for $f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{Bc}\|_2^2$ due to $\boldsymbol{Bc}$
- We need to decouple $\boldsymbol{Bc}$ — let's see how.

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h(\textstyle\sum_i \lambda_i x_i) \leq \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \geq 0, \sum_i \lambda_i = 1}$$



A convex function

Constructing g

- Main difficulty for $f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_2^2$ due to $\boldsymbol{B}\boldsymbol{c}$
- We need to decouple $\boldsymbol{B}\boldsymbol{c}$ — let's see how.

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h(\sum_i \lambda_i x_i) \le \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \ge 0, \sum_i \lambda_i = 1}$$

$$f(\boldsymbol{c}) = \frac{1}{2}\sum_i (a_i - \boldsymbol{b}_i^T \boldsymbol{c})^2 =$$

Constructing g

- Main difficulty for $f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_2^2$ due to $\boldsymbol{B}\boldsymbol{c}$
- We need to decouple $\boldsymbol{B}\boldsymbol{c}$ — let's see how.

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h(\textstyle\sum_i \lambda_i x_i) \leq \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \geq 0, \sum_i \lambda_i = 1}$$

$$f(\boldsymbol{c}) = \frac{1}{2}\sum_i (a_i - \boldsymbol{b}_i^T\boldsymbol{c})^2 = \frac{1}{2}\sum_i a_i^2 - 2a_i\boldsymbol{b}_i^T\boldsymbol{c} + (\boldsymbol{b}_i^T\boldsymbol{c})^2$$

Constructing g

- Main difficulty for $f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_2^2$ due to $\boldsymbol{B}\boldsymbol{c}$
- We need to decouple $\boldsymbol{B}\boldsymbol{c}$ — let's see how.

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h(\textstyle\sum_i \lambda_i x_i) \leq \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \geq 0, \sum_i \lambda_i = 1}$$

$$\begin{aligned} f(\boldsymbol{c}) &= \tfrac{1}{2}\textstyle\sum_i (a_i - \boldsymbol{b}_i^T\boldsymbol{c})^2 = \tfrac{1}{2}\sum_i a_i^2 - 2a_i\boldsymbol{b}_i^T\boldsymbol{c} + (\boldsymbol{b}_i^T\boldsymbol{c})^2 \\ &= \tfrac{1}{2}\textstyle\sum_i a_i^2 - 2a_i\boldsymbol{b}_i^T\boldsymbol{c} + \tfrac{1}{2}\sum_i \big(\sum_j b_{ij}c_j\big)^2 \end{aligned}$$

Constructing g

- Main difficulty for $f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_2^2$ due to $\boldsymbol{B}\boldsymbol{c}$
- We need to decouple $\boldsymbol{B}\boldsymbol{c}$ — let's see how.

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h(\sum_i \lambda_i x_i) \leq \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \geq 0,\ \sum_i \lambda_i = 1}$$

$$\begin{aligned}
f(\boldsymbol{c}) &= \frac{1}{2}\sum_i (a_i - \boldsymbol{b}_i^T\boldsymbol{c})^2 = \frac{1}{2}\sum_i a_i^2 - 2a_i\boldsymbol{b}_i^T\boldsymbol{c} + (\boldsymbol{b}_i^T\boldsymbol{c})^2 \\
&= \frac{1}{2}\sum_i a_i^2 - 2a_i\boldsymbol{b}_i^T\boldsymbol{c} + \frac{1}{2}\sum_i \Big(\sum_j b_{ij}c_j\Big)^2 \\
&= \frac{1}{2}\sum_i a_i^2 - 2a_i\boldsymbol{b}_i^T\boldsymbol{c}
\end{aligned}$$

Constructing g

- Main difficulty for $f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_2^2$ due to $\boldsymbol{B}\boldsymbol{c}$
- We need to decouple $\boldsymbol{B}\boldsymbol{c}$ — let's see how.

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h(\sum_i \lambda_i x_i) \leq \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \geq 0, \sum_i \lambda_i = 1}$$

$$\begin{aligned} f(\boldsymbol{c}) = \frac{1}{2}\sum_i (a_i - \boldsymbol{b}_i^T\boldsymbol{c})^2 &= \frac{1}{2}\sum_i a_i^2 - 2a_i\boldsymbol{b}_i^T\boldsymbol{c} + (\boldsymbol{b}_i^T\boldsymbol{c})^2 \\ &= \frac{1}{2}\sum_i a_i^2 - 2a_i\boldsymbol{b}_i^T\boldsymbol{c} + \frac{1}{2}\sum_i \Big(\sum_j b_{ij}c_j\Big)^2 \\ &= \frac{1}{2}\sum_i a_i^2 - 2a_i\boldsymbol{b}_i^T\boldsymbol{c} + \frac{1}{2}\sum_i \Big(\sum_j \lambda_{ij}b_{ij}c_j/\lambda_{ij}\Big)^2 \end{aligned}$$

Constructing g

- Main difficulty for $f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{Bc}\|_2^2$ due to $\boldsymbol{Bc}$
- We need to decouple $\boldsymbol{Bc}$ — let's see how.

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h(\textstyle\sum_i \lambda_i x_i) \le \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \ge 0,\ \sum_i \lambda_i = 1}$$

$$\begin{aligned}
f(\boldsymbol{c}) &= \tfrac{1}{2}\textstyle\sum_i (a_i - \boldsymbol{b}_i^T\boldsymbol{c})^2 = \tfrac{1}{2}\sum_i a_i^2 - 2a_i\boldsymbol{b}_i^T\boldsymbol{c} + (\boldsymbol{b}_i^T\boldsymbol{c})^2 \\
&= \tfrac{1}{2}\textstyle\sum_i a_i^2 - 2a_i\boldsymbol{b}_i^T\boldsymbol{c} + \tfrac{1}{2}\sum_i \left(\sum_j b_{ij}c_j\right)^2 \\
&= \tfrac{1}{2}\textstyle\sum_i a_i^2 - 2a_i\boldsymbol{b}_i^T\boldsymbol{c} + \tfrac{1}{2}\sum_i \left(\sum_j \lambda_{ij} b_{ij}c_j/\lambda_{ij}\right)^2 \\
&\overset{\text{cvx}}{\le} \tfrac{1}{2}\textstyle\sum_i a_i^2 - 2a_i\boldsymbol{b}_i^T\boldsymbol{c} + \tfrac{1}{2}\sum_{ij} \lambda_{ij}\left(b_{ij}c_j/\lambda_{ij}\right)^2
\end{aligned}$$

Constructing g

- Main difficulty for $f(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_2^2$ due to $\boldsymbol{B}\boldsymbol{c}$
- We need to decouple $\boldsymbol{B}\boldsymbol{c}$ — let's see how.

We exploit that $h(x) = \frac{1}{2}x^2$ is a *convex function*

$$\boxed{h(\textstyle\sum_i \lambda_i x_i) \le \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \ge 0,\ \sum_i \lambda_i = 1}$$

$$\begin{aligned}
f(\boldsymbol{c}) &= \tfrac{1}{2}\textstyle\sum_i (a_i - \boldsymbol{b}_i^T\boldsymbol{c})^2 = \tfrac{1}{2}\sum_i a_i^2 - 2a_i\boldsymbol{b}_i^T\boldsymbol{c} + (\boldsymbol{b}_i^T\boldsymbol{c})^2 \\
&= \tfrac{1}{2}\textstyle\sum_i a_i^2 - 2a_i\boldsymbol{b}_i^T\boldsymbol{c} + \tfrac{1}{2}\sum_i (\sum_j b_{ij}c_j)^2 \\
&= \tfrac{1}{2}\textstyle\sum_i a_i^2 - 2a_i\boldsymbol{b}_i^T\boldsymbol{c} + \tfrac{1}{2}\sum_i (\sum_j \lambda_{ij} b_{ij}c_j/\lambda_{ij})^2 \\
&\overset{\text{cvx}}{\le} \tfrac{1}{2}\textstyle\sum_i a_i^2 - 2a_i\boldsymbol{b}_i^T\boldsymbol{c} + \tfrac{1}{2}\sum_{ij} \lambda_{ij}(b_{ij}c_j/\lambda_{ij})^2 \\
&= g(\boldsymbol{c}, \tilde{\boldsymbol{c}}), \quad \text{where} \quad \lambda_{ij} \quad \text{are convex coeffts}
\end{aligned}$$

Constructing g

In summary:

$$\begin{aligned} f(\boldsymbol{c}) &= \tfrac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_2^2 \\ g(\boldsymbol{c}, \tilde{\boldsymbol{c}}) &= \tfrac{1}{2}\|\boldsymbol{a}\|_2^2 - \textstyle\sum_i a_i \boldsymbol{b}_i^T \boldsymbol{c} + \tfrac{1}{2}\textstyle\sum_{ij} \lambda_{ij}\left(b_{ij}c_j/\lambda_{ij}\right)^2 \end{aligned}$$

Now we *pick* λ_{ij}

Constructing g

In summary:

$$
\begin{aligned}
f(\boldsymbol{c}) &= \tfrac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_2^2 \\
g(\boldsymbol{c}, \tilde{\boldsymbol{c}}) &= \tfrac{1}{2}\|\boldsymbol{a}\|_2^2 - \sum_i a_i \boldsymbol{b}_i^T \boldsymbol{c} + \tfrac{1}{2}\sum_{ij} \lambda_{ij}\left(b_{ij} c_j / \lambda_{ij}\right)^2
\end{aligned}
$$

Now we *pick* λ_{ij}

$$
\lambda_{ij} = \frac{b_{ij}\tilde{c}_j}{\sum_k b_{ik}\tilde{c}_k} = \frac{b_{ij}\tilde{c}_j}{\boldsymbol{b}_i^T \tilde{\boldsymbol{c}}}
$$

Constructing g

In summary:

$$
\begin{aligned}
f(\boldsymbol{c}) &= \tfrac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_2^2 \\
g(\boldsymbol{c}, \tilde{\boldsymbol{c}}) &= \tfrac{1}{2}\|\boldsymbol{a}\|_2^2 - \sum_i a_i \boldsymbol{b}_i^T \boldsymbol{c} + \tfrac{1}{2}\sum_{ij} \lambda_{ij}(b_{ij}c_j/\lambda_{ij})^2
\end{aligned}
$$

Now we *pick* λ_{ij}

$$
\lambda_{ij} = \frac{b_{ij}\tilde{c}_j}{\sum_k b_{ik}\tilde{c}_k} = \frac{b_{ij}\tilde{c}_j}{\boldsymbol{b}_i^T \tilde{\boldsymbol{c}}}
$$

Exercise: Aux function

Verify that $g(\boldsymbol{c}, \boldsymbol{c}) = f(\boldsymbol{c})$;

Constructing g

In summary:

$$f(\boldsymbol{c}) = \tfrac{1}{2}\|\boldsymbol{a} - \boldsymbol{B}\boldsymbol{c}\|_2^2$$
$$g(\boldsymbol{c}, \tilde{\boldsymbol{c}}) = \tfrac{1}{2}\|\boldsymbol{a}\|_2^2 - \textstyle\sum_i a_i \boldsymbol{b}_i^T \boldsymbol{c} + \tfrac{1}{2}\sum_{ij} \lambda_{ij} (b_{ij} c_j / \lambda_{ij})^2$$

Now we *pick* λ_{ij}

$$\lambda_{ij} = \frac{b_{ij}\tilde{c}_j}{\sum_k b_{ik}\tilde{c}_k} = \frac{b_{ij}\tilde{c}_j}{\boldsymbol{b}_i^T \tilde{\boldsymbol{c}}}$$

Exercise: Aux function

Verify that $g(\boldsymbol{c}, \boldsymbol{c}) = f(\boldsymbol{c})$;

Exercise: Richardson-Lucy

Let $f(\boldsymbol{c}) = \sum_i a_i \log(a_i/(\boldsymbol{B}\boldsymbol{c})_i) - a_i + (\boldsymbol{B}\boldsymbol{c})_i$.
Derive an auxiliary function $g(\boldsymbol{c}, \tilde{\boldsymbol{c}})$ for this $f(\boldsymbol{c})$

Minimizing g

Recall,core step: $\boldsymbol{c}^{t+1} = \operatorname{argmin} g(\boldsymbol{c}, \boldsymbol{c}^t)$

Solve $\partial g(\boldsymbol{c}, \boldsymbol{c}^t)/\partial c_p = 0$

Minimizing g

Recall,core step: $\boldsymbol{c}^{t+1} = \operatorname{argmin} g(\boldsymbol{c}, \boldsymbol{c}^t)$
Solve $\partial g(\boldsymbol{c}, \boldsymbol{c}^t)/\partial c_p = 0$

$$\partial g/\partial c_p = -\sum_i a_i b_{ip} + \sum_i b_{ip}(\boldsymbol{b}_i^T \boldsymbol{c}^t) c_p / c_p^t$$

Minimizing g

Recall,core step: $\boldsymbol{c}^{t+1} = \operatorname{argmin} g(\boldsymbol{c}, \boldsymbol{c}^t)$
Solve $\partial g(\boldsymbol{c}, \boldsymbol{c}^t)/\partial c_p = 0$

$$\partial g/\partial c_p = -\sum_i a_i b_{ip} + \sum_i b_{ip}(\boldsymbol{b}_i^T \boldsymbol{c}^t) c_p / c_p^t$$

Which yields **(verify!)** : $c_p = c_p^t \dfrac{[\boldsymbol{B}^T \boldsymbol{a}]_p}{[\boldsymbol{B}^T \boldsymbol{B} \boldsymbol{c}^t]_p}$

Minimizing g

Recall,core step: $\boldsymbol{c}^{t+1} = \operatorname{argmin} g(\boldsymbol{c}, \boldsymbol{c}^t)$
Solve $\partial g(\boldsymbol{c}, \boldsymbol{c}^t)/\partial c_p = 0$

$$\partial g/\partial c_p = -\sum_i a_i b_{ip} + \sum_i b_{ip}(\boldsymbol{b}_i^T \boldsymbol{c}^t) c_p / c_p^t$$

Which yields **(verify!)** : $c_p = c_p^t \dfrac{[\boldsymbol{B}^T \boldsymbol{a}]_p}{[\boldsymbol{B}^T \boldsymbol{B} \boldsymbol{c}^t]_p}$

Extending to matrices, we obtain Lee & Seung's update

$$\boldsymbol{C}^{t+1} = \boldsymbol{C}^t \odot \frac{\boldsymbol{B}^T \boldsymbol{A}}{\boldsymbol{B}^T \boldsymbol{B} \boldsymbol{C}^t}$$

Some remarks regarding g

- We exploited convexity of x^2

Some remarks regarding g

- We exploited convexity of x^2
- Expectation Maximization (EM) algorithm exploits convexity of $-\log x$

Some remarks regarding g

- We exploited convexity of x^2
- Expectation Maximization (EM) algorithm exploits convexity of $-\log x$
- Richardson-Lucy (Astronomy), or EMML / MLEM (Tomography) exploits $x \log x$

Some remarks regarding g

- We exploited convexity of x^2
- Expectation Maximization (EM) algorithm exploits convexity of $-\log x$
- Richardson-Lucy (Astronomy), or EMML / MLEM (Tomography) exploits $x \log x$
- Other choices possible, e.g., by varying λ_{ij}

Some remarks regarding g

- We exploited convexity of x^2
- Expectation Maximization (EM) algorithm exploits convexity of $-\log x$
- Richardson-Lucy (Astronomy), or EMML / MLEM (Tomography) exploits $x \log x$
- Other choices possible, e.g., by varying λ_{ij}
- Our technique one variant of repertoire of *Majorization-Minimization* (MM) algorithms

Some remarks regarding g

- We exploited convexity of x^2
- Expectation Maximization (EM) algorithm exploits convexity of $-\log x$
- Richardson-Lucy (Astronomy), or EMML / MLEM (Tomography) exploits $x \log x$
- Other choices possible, e.g., by varying λ_{ij}
- Our technique one variant of repertoire of *Majorization-Minimization* (MM) algorithms
- Related to *d.c. programming*

Some remarks regarding g

- We exploited convexity of x^2
- Expectation Maximization (EM) algorithm exploits convexity of $-\log x$
- Richardson-Lucy (Astronomy), or EMML / MLEM (Tomography) exploits $x \log x$
- Other choices possible, e.g., by varying λ_{ij}
- Our technique one variant of repertoire of *Majorization-Minimization* (MM) algorithms
- Related to *d.c. programming*
- MM algorithms subject of a separate lecture!

Summary

- We looked at least-squares NMA

$$\min \quad \frac{1}{2}\|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}\|_F^2, \quad \text{s.t.} \quad \boldsymbol{B}, \boldsymbol{C} \geq 0.$$

Summary

- We looked at least-squares NMA

$$\min \quad \frac{1}{2}\|\boldsymbol{A} - \boldsymbol{BC}\|_F^2, \quad \text{s.t.} \quad \boldsymbol{B}, \boldsymbol{C} \geq 0.$$

- We derived two algorithms: (i) Gradient-Projection; (ii) multiplicative updates

Summary

- We looked at least-squares NMA

$$\min \quad \frac{1}{2}\|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}\|_F^2, \quad \text{s.t.} \quad \boldsymbol{B}, \boldsymbol{C} \geq 0.$$

- We derived two algorithms: (i) Gradient-Projection; (ii) multiplicative updates

Take home message: The methods, techniques that we saw, are general. You can use them for many other problems!

Applications & Practical Concerns

Applications – example areas

1. Statistics
2. Data mining, Machine learning
3. Signal processing (images, speech, music, etc.)
4. Computer graphics
5. Chemometrics
6. Remote Sensing
7. Scientific computing
8. …

TSVD

- Statistics
- Psychometrics
- Data Mining, Machine learning
- Information Retrieval
- Biology, Bioinformatics
- In general, exploratory data analysis

Bioinformatics – gene microarray analysis

Biologists measure *activity* (aka gene-expression) of different genes under various conditions (time, temperature, etc.).

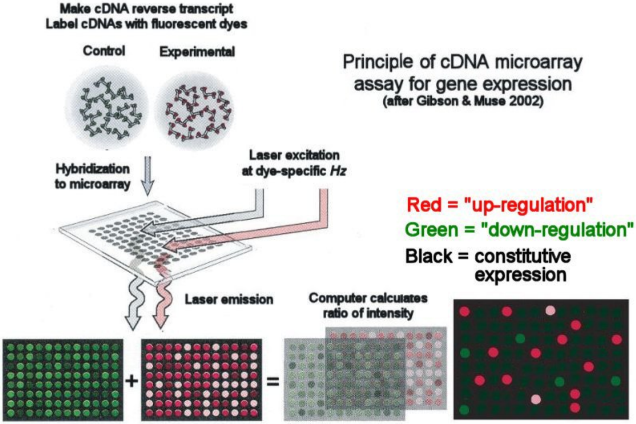
Bioinformatics – gene microarray analysis

Biologists measure *activity* (aka gene-expression) of different genes under various conditions (time, temperature, etc.).

Activity recorded using *gene microarray*

Bioinformatics – gene microarray analysis

Biologists measure *activity* (aka gene-expression) of different genes under various conditions (time, temperature, etc.). Activity recorded using *gene microarray*



Bioinformatics – gene microarray analysis

Biologists measure *activity* (aka gene-expression) of different genes under various conditions (time, temperature, etc.).
Activity recorded using *gene microarray*
Activities across numerous "conditions" or experiments

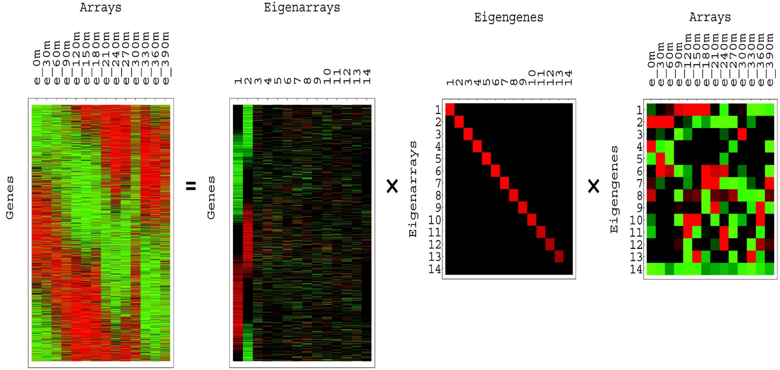
We measure an $m \times n$ ($m \gg n$) *genes* $\times$ *array* matrix.

Some "cleaning" (pre-processing) etc. needed.

Truncated SVD on this gene-expression matrix is performed.

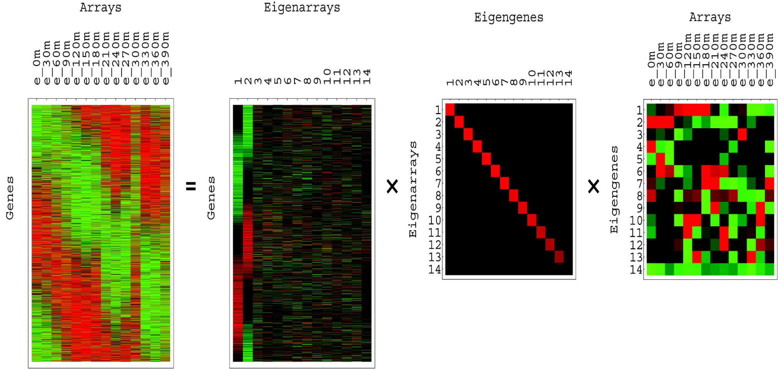
Bioinformatics – gene microarray analysis

Biologists measure *activity* (aka gene-expression) of different genes under various conditions (time, temperature, etc.).



Bioinformatics – gene microarray analysis

Biologists measure *activity* (aka gene-expression) of different genes under various conditions (time, temperature, etc.).



Significant "eigengenes" $\Longrightarrow$ independent biological processes and experimental artifacts.

Figure taken from: http://www.bme.utexas.edu/research/orly/teaching/BME341

NMA

- Chemometrics
- Document modeling, text-analysis
- Spam modeling
- Bioinformatics
- Music analysis
- Computer Vision
- Image processing
- Remote sensing (hyperspectral imaging)
- Dimensionality reduction
- Computer graphics
- Collaborative filtering
- Multiframe blind deconvolution

NMA – Text Analysis

- Dataset: Collection of 3891 documents
- Each document represented as a 4857 dimensional vector

NMA – Text Analysis

- Dataset: Collection of 3891 documents
- Each document represented as a 4857 dimensional vector
- Data matrix: $\boldsymbol{A} \in \mathbb{R}_{+}^{4857 \times 3891}$

NMA – Text Analysis

- Dataset: Collection of 3891 documents
- Each document represented as a 4857 dimensional vector
- Data matrix: $\boldsymbol{A} \in \mathbb{R}_{+}^{4857 \times 3891}$
- Three "human" defined categories CISI, CRAN and MED

NMA – Text Analysis

- Dataset: Collection of 3891 documents
- Each document represented as a 4857 dimensional vector
- Data matrix: $\boldsymbol{A} \in \mathbb{R}_{+}^{4857 \times 3891}$
- Three "human" defined categories CISI, CRAN and MED
- NMA: $\boldsymbol{A} \approx \boldsymbol{BC}$, where $\boldsymbol{B}$ has 3 columns — representing "topics"

NMA – Text Analysis

- Dataset: Collection of 3891 documents
- Each document represented as a 4857 dimensional vector
- Data matrix: $\boldsymbol{A} \in \mathbb{R}_+^{4857 \times 3891}$
- Three “human” defined categories CISI, CRAN and MED
- NMA: $\boldsymbol{A} \approx \boldsymbol{BC}$, where $\boldsymbol{B}$ has 3 columns — representing “topics”

CISI	CRAN	MED
retrieval	wing	patients
system	pressure	cells
systems	mach	growth
indexing	supersonic	hormone
scientific	shock	cancer
science	jet	treatment
index	lift	buckling
search	wings	blood
computer	body	cases
document	theory	cell

Image analysis – toy example

"Swimmer" database – 256, 32 x 32 images [DoSt03]

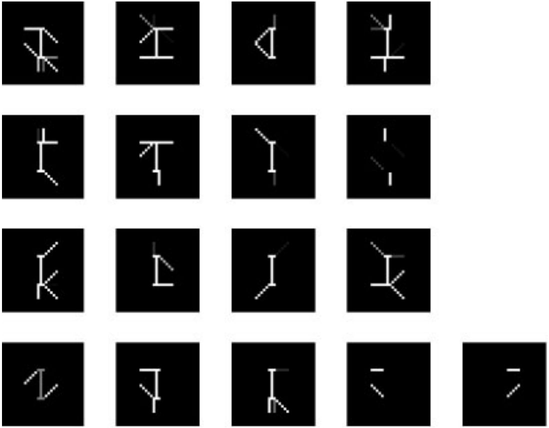
- Stick figures showing different configurations of the limbs of a swimmer
- Data matrix of size 1024×256

Image analysis – toy example

"Swimmer" database – 256, 32 x 32 images [DoSt03]

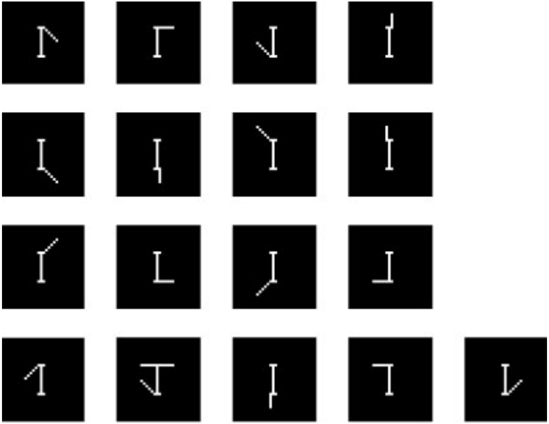
- Stick figures showing different configurations of the limbs of a swimmer
- Data matrix of size 1024×256
- Decompose the matrix into 1024×17 (17 seemed to be the "true" nonnegative rank)

Image analysis – toy example



Rank-17 decomposition via Lee/Seung's algo
Time: 182.4 seconds, Objective: 2.41×10^7

Image analysis – toy example



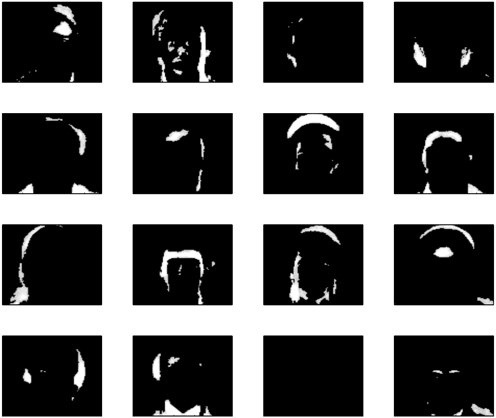
Via more advanced projection algorithm

Time: *62.3* seconds, Objective: 6.85×10^{-4}

Part of a face recognition system

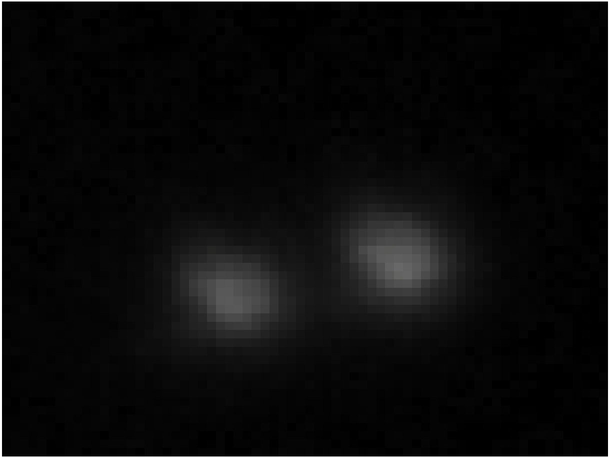
- 143 images from MIT face image database
- Input matrix $\boldsymbol{A} \in \mathbb{R}_{+}^{9216 \times 143}$

Part of a face recognition system



- A rank-20 approximation to the input
- The basis vectors (columns of $\boldsymbol{B}$) approximately correspond to important "*parts*" describing the faces.

Multiframe blind deconvolution – astronomy

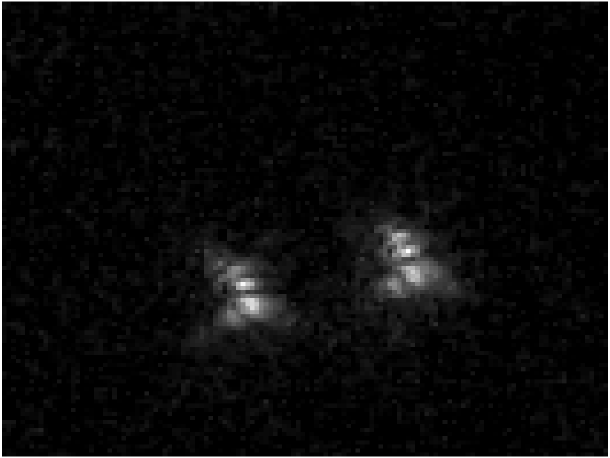


long-time exposure (approx. 1 s)

Problem: Atmospheric turbulence

Courtesy of Karl-Ludwig Bath, IAS, Hakos, Namibia

Multiframe blind deconvolution – astronomy



short-time exposure (approx. 10ms)

Problem: Atmospheric turbulence

Courtesy of Karl-Ludwig Bath, IAS, Hakos, Namibia

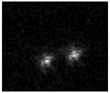
Multiframe blind deconvolution – astronomy

real-time video (15 fps)

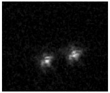
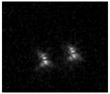
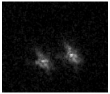
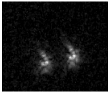
Problem: Atmospheric turbulences

Courtesy of Karl-Ludwig Bath, IAS, Hakos, Namibia

Our model of the video

time t	$\boldsymbol{y}_t$	$=$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}$	$+$	$\boldsymbol{n}_t$
0		$=$		$\star$		$+$	$\boldsymbol{n}_0$

Our model of the video

time t	$\boldsymbol{y}_t$	$=$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}$	$+$	$\boldsymbol{n}_t$
0		$=$		$\star$		$+$	$\boldsymbol{n}_0$
1		$=$		$\star$		$+$	$\boldsymbol{n}_1$
2		$=$		$\star$		$+$	$\boldsymbol{n}_2$
k		$=$		$\star$		$+$	$\boldsymbol{n}_k$

$$\begin{bmatrix} | & \vdots & | \\ \boldsymbol{y}_1 & | & \boldsymbol{y}_n \\ | & \vdots & | \end{bmatrix} \approx \begin{bmatrix} | & \vdots & | \\ \boldsymbol{a}_1 & | & \boldsymbol{a}_t \\ | & \vdots & | \end{bmatrix} \star \boldsymbol{x}$$

Convolution operation may be written as

$$\boxed{\boldsymbol{a} \star \boldsymbol{x} = \boldsymbol{A}\boldsymbol{x} = \boldsymbol{X}\boldsymbol{a}}$$

$$\begin{bmatrix} | & \vdots & | \\ \mathbf{y}_1 & | & \mathbf{y}_n \\ | & \vdots & | \end{bmatrix} \approx \begin{bmatrix} | & \vdots & | \\ \mathbf{a}_1 & | & \mathbf{a}_t \\ | & \vdots & | \end{bmatrix} \star \mathbf{x}$$

Convolution operation may be written as

$$\boxed{\mathbf{a} \star \mathbf{x} = \mathbf{A}\mathbf{x} = \mathbf{X}\mathbf{a}}$$

$$\begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_t \end{bmatrix} \approx \begin{bmatrix} \mathbf{A}_1 \\ \cdots \\ \mathbf{A}_t \end{bmatrix} \mathbf{x}$$

$$\begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_t \end{bmatrix} \approx \mathbf{X} \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_t \end{bmatrix}$$

$$\boxed{\mathbf{Y} \approx \mathbf{X}\mathbf{A}}$$

Multiframe blind deconvolution

We seek to minimize

$$\frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{A}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{X}, \boldsymbol{A} \geq 0$$

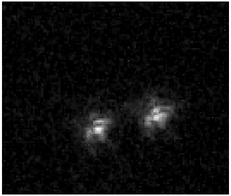
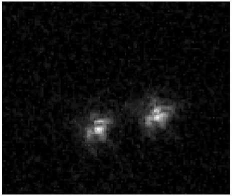
Multiframe blind deconvolution

We seek to minimize

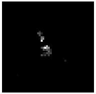
$$\frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{A}\|_F^2 \quad \text{s.t.} \quad \boldsymbol{X}, \boldsymbol{A} \geq 0$$

Note 1: $\boldsymbol{X}$ and $\boldsymbol{A}$ are the *unknowns*
Note 2: Additional constraints may be present on $\boldsymbol{X}$ or $\boldsymbol{A}$
Note 3: Looks like an NMA problem (except $\boldsymbol{X}$ or $\boldsymbol{A}$ have special structure due to the convolution $\boldsymbol{a} \star \boldsymbol{x}$)

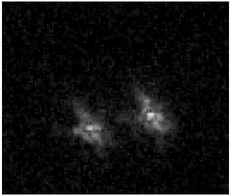
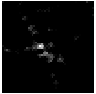
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$=$	$\boldsymbol{x}_t$
1		$=$	

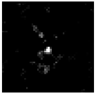
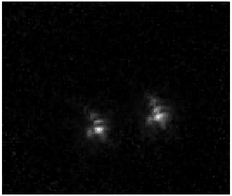
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
2		$\approx$		$\star$	

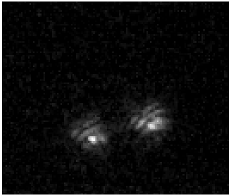
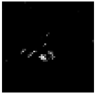
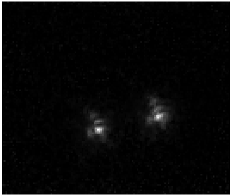
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
3		$\approx$		$\star$	

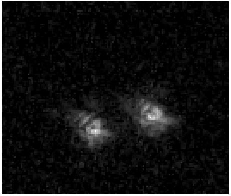
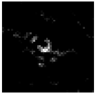
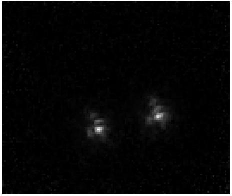
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
4		$\approx$		$\star$	

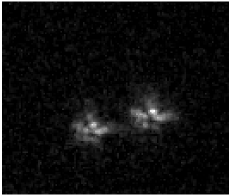
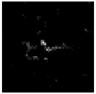
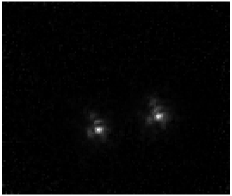
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
5		$\approx$		$\star$	

Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
6		$\approx$		$\star$	

Double star epsilon lyrae

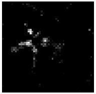
time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
7		$\approx$		$\star$	

Double star epsilon lyrae

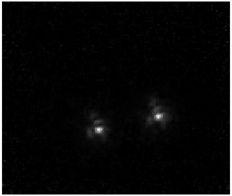
time t $\quad \boldsymbol{y}_t \approx \boldsymbol{a}_t \star \boldsymbol{x}_t$

8

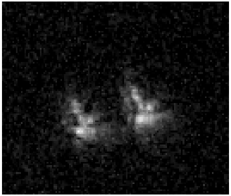
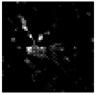
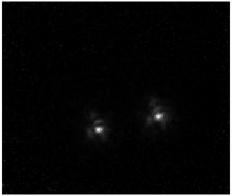
$\approx$



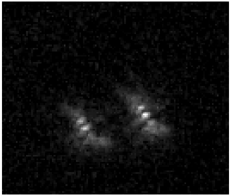
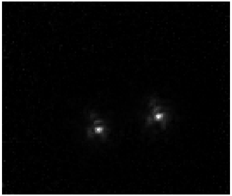
$\star$



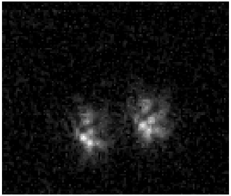
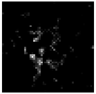
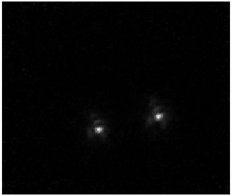
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
9		$\approx$		$\star$	

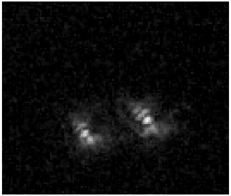
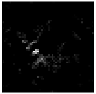
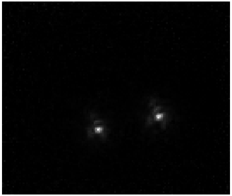
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
10		$\approx$		$\star$	

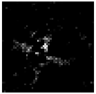
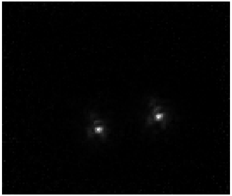
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
11		$\approx$		$\star$	

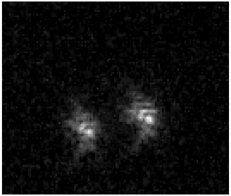
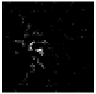
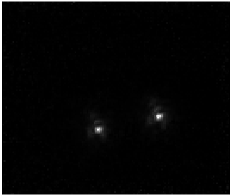
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
12		$\approx$		$\star$	

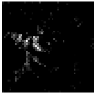
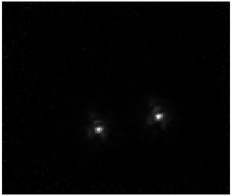
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
13		$\approx$		$\star$	

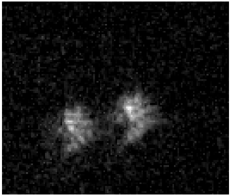
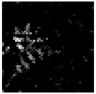
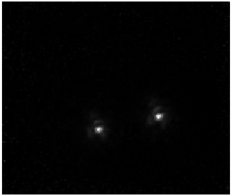
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
14		$\approx$		$\star$	

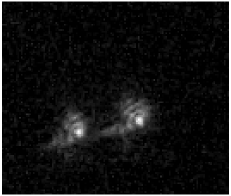
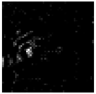
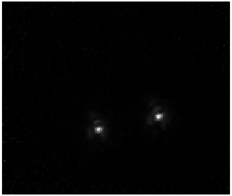
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
15		$\approx$		$\star$	

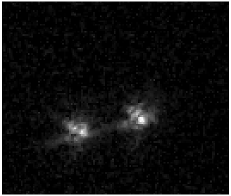
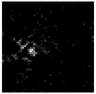
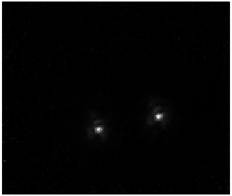
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
16		$\approx$		$\star$	

Double star epsilon lyrae

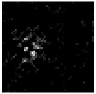
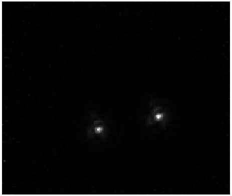
time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
17		$\approx$		$\star$	

Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
18		$\approx$		$\star$	

Double star epsilon lyrae

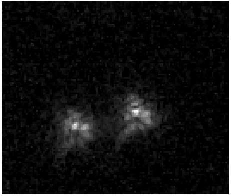
time t $\quad \boldsymbol{y}_t \approx \boldsymbol{a}_t \star \boldsymbol{x}_t$

19  $\approx$  $\star$ 

Double star epsilon lyrae

time t $\quad \boldsymbol{y}_t \approx \boldsymbol{a}_t \star \boldsymbol{x}_t$

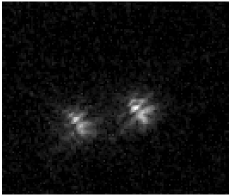
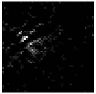
20



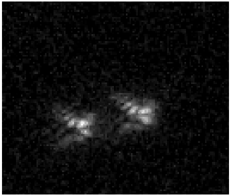
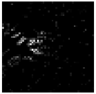
$\approx$

$\star$

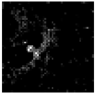
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
21		$\approx$		$\star$	

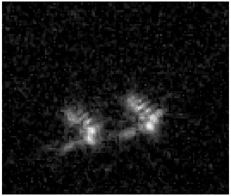
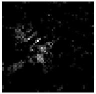
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
22		$\approx$		$\star$	

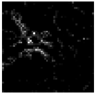
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
23		$\approx$		$\star$	

Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
24		$\approx$		$\star$	

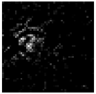
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
25		$\approx$		$\star$	

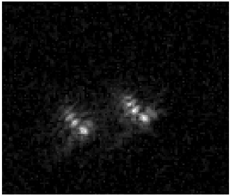
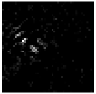
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
26		$\approx$		$\star$	

Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
27		$\approx$		$\star$	

Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
28		$\approx$		$\star$	

Double star epsilon lyrae

time t $\quad \boldsymbol{y}_t \approx \boldsymbol{a}_t \star \boldsymbol{x}_t$

29

$\approx$

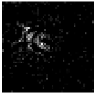
$\star$

Double star epsilon lyrae

time t $\quad \boldsymbol{y}_t \approx \boldsymbol{a}_t \star \boldsymbol{x}_t$

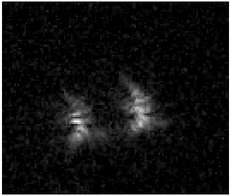
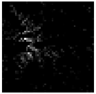
30

$\approx$

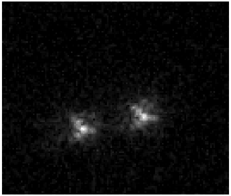
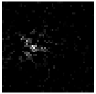


$\star$

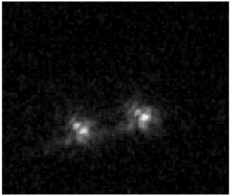
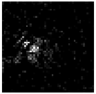
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
31		$\approx$		$\star$	

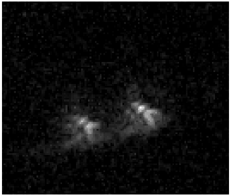
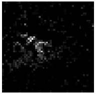
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
32		$\approx$		$\star$	

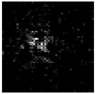
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
33		$\approx$		$\star$	

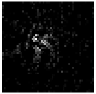
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
34		$\approx$		$\star$	

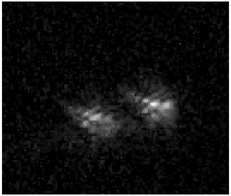
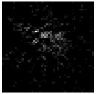
Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
35		$\approx$		$\star$	

Double star epsilon lyrae

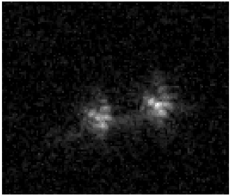
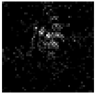
time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
36		$\approx$		$\star$	

Double star epsilon lyrae

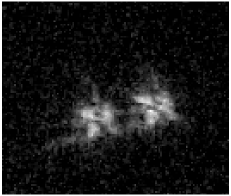
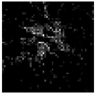
time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
37		$\approx$		$\star$	

Double star epsilon lyrae

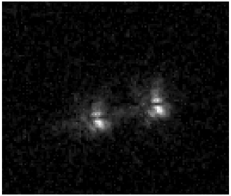
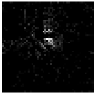
time t $\quad \boldsymbol{y}_t \approx \boldsymbol{a}_t \star \boldsymbol{x}_t$

38  $\approx$  $\star$ 

Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
39		$\approx$		$\star$	

Double star epsilon lyrae

time t	$\boldsymbol{y}_t$	$\approx$	$\boldsymbol{a}_t$	$\star$	$\boldsymbol{x}_t$
40		$\approx$		$\star$	

MFBD Video

Video example

Discussion & Wrap-up

Summary

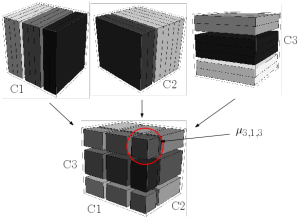
1. Introduction to matrix approximation problems
   - Background, motivation
   - Truncated SVD; its properties
   - List of some popular problems, e.g., NMA
2. Algorithms for NMA
   - Alternating minimization
   - Alternating descent
   - Gradient Projection
   - Multiplicative updates
3. Applications
   - Bioinformatics app of SVD
   - Image processing, astronomy, etc. of NMA

Challenges, other stuff

- Theoretical: Non-convex optimization
- Analysis, new algorithms, new problems
- Practical: Large-scale, sparse data
- Cluster, multi-core, GPU, etc.
- Efficient SVD (PROPACK, SLEPc, etc.)
- Methods based on random projections
- Numerous other *matrix nearness* problems exist
- Tensor approximations

Challenges, other stuff

- Theoretical: Non-convex optimization
- Analysis, new algorithms, new problems
- Practical: Large-scale, sparse data
- Cluster, multi-core, GPU, etc.
- Efficient SVD (PROPACK, SLEPc, etc.)
- Methods based on random projections
- Numerous other *matrix nearness* problems exist
- Tensor approximations



Closing: Huge Matrix Problems

Distributed Nonnegative Matrix Factorization for Web-Scale Dyadic Data Analysis on MapReduce by Chao Liu et al.

- Input matrix $\boldsymbol{A}$ of size $43.9M \times 769M$; total 4.38×10^9 nonzeros (1.2×10^{-7} - density)
- 7 hours per iteration (dedicated cluster of 8 comps)
- http://research.microsoft.com/pubs/119077/DNMF.pdf

Closing: Huge Matrix Problems

Distributed Nonnegative Matrix Factorization for Web-Scale Dyadic Data Analysis on MapReduce by Chao Liu et al.

- Input matrix $\boldsymbol{A}$ of size $43.9M \times 769M$; total 4.38×10^9 nonzeros (1.2×10^{-7} - density)
- 7 hours per iteration (dedicated cluster of 8 comps)
- http://research.microsoft.com/pubs/119077/DNMF.pdf

I think YOU can do better!