\[ A \approx BC \]

Matrix Approximation Problems

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

(MPI für biologische Kybernetik, Tübingen)
What’s the course about?

\[ A \approx \hat{A} \]
What’s the course about?

$A \approx \hat{A}$
What’s the course about?

A \approx \hat{A}
What’s the course about?

\[ A \approx \hat{A} \]

Not quite!
What’s the course about?

\[ A \approx \hat{A} \]

Given an input matrix \( A \) compute a matrix \( \hat{A} \) that satisfies certain desired properties, e.g.,
What’s the course about?

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

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

Let's learn something!
Introduction – matrices all over

Images

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

---

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

---

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

■ Computer Science

The Internet Graph

1 Matrix Collage made from images on Wikipedia; Sci. Comp. images take from Tim Davis’ website; Internet graph from Wikipedia;
Introduction – Why approximate?

Measurements fail to satisfy expectation:

\[
\begin{bmatrix}
0 & 3 & 8 \\
2 & 8 & 0 \\
7 & 9 & 4
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 3 & 7 .5 \\
3 & 0 & 4 .5 \\
7 & 5 & 4 .5 & 0
\end{bmatrix}
\]

\[AC \neq CA\] and \[AC > AB + BC\]!

Rounding errors, noise confound:

Expected symmetric, orthogonal, real, posdef, etc., but obtained something else!
Introduction – Why approximate?

Measurements fail to satisfy expectation:

\[
\begin{bmatrix}
A & B & C \\
0 & 3 & 7.5 \\
3 & 0 & 4.5 \\
7 & 4.5 & 0 \\
\end{bmatrix}
\]

\[AC \neq CA \text{ and } AC > AB + BC!\]
Introduction – Why approximate?

Measurements fail to satisfy expectation:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>B</td>
<td>2.8</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>7.9</td>
<td>4.1</td>
<td>0</td>
</tr>
</tbody>
</table>

AC ≠ CA and AC > AB + BC!
Introduction – Why approximate?

Measurements fail to satisfy expectation:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>B</td>
<td>2.8</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>7.9</td>
<td>4.1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>3</td>
<td>7.5</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>0</td>
<td>4.5</td>
</tr>
<tr>
<td>C</td>
<td>7.5</td>
<td>4.5</td>
<td>0</td>
</tr>
</tbody>
</table>

AC \neq CA and AC > AB + BC!
Introduction – Why approximate?

Measurements fail to satisfy expectation:

\[
\begin{array}{ccc}
A & B & C \\
\hline
A & 0 & 3 & 8 \\
B & 2.8 & 0 & 4 \\
C & 7.9 & 4.1 & 0 \\
\end{array}
\]

\[
\begin{array}{ccc}
A & B & C \\
\hline
A & 0 & 3 & 7.5 \\
B & 3 & 0 & 4.5 \\
C & 7.5 & 4.5 & 0 \\
\end{array}
\]

\[AC \neq CA \text{ and } AC > AB + BC!\]

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
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 $A$ with several missing entries
- “Predict” missing entries to “complete” the matrix
Introduction – Why approximate?

For €€€ reasons!

- Netflix million-$ prize problem!
- Typical \textit{matrix completion} problem
- Input: matrix $A$ 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 $A$ 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
Suppose we wish to approx. matrix $\mathbf{A}$ by $\hat{\mathbf{A}}$. Ideally, $\hat{\mathbf{A}}$ is the “nearest” matrix satisfying a desired property (eg. $\hat{\mathbf{A}} \in \Omega$)?
Introduction – preliminary concepts

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

First define *nearest*!
Suppose we wish to approx. matrix $A$ by $\hat{A}$. Ideally, $\hat{A}$ is the “nearest” matrix satisfying a desired property (eg. $\hat{A} \in \Omega$)?

First define *nearest*!

We measure “distance” between two matrices using $\Delta$

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

First define *nearest*!

We measure “distance” between two matrices using $\Delta$

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

“Nearest” means: $\hat{A} \in \Omega$ having smallest $\Delta$ value
Introduction – preliminary concepts

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

First define *nearest*!

We measure “distance” between two matrices using $\Delta$

\[
\Delta(A, \hat{A})
\]

“Nearest” means: $\hat{A} \in \Omega$ having smallest $\Delta$ value

Commonly used: $\Delta(A, \hat{A}) = \| A - \hat{A} \|$
Digression: Matrix Norms

An (operator) *norm* of a matrix $A$ is defined as

$$\|A\| = \max_{\|x\| = 1} \|Ax\|$$

**Example:** Maximum singular value, $\sigma_1(A) = \|A\|_2$
Digression: Matrix Norms

An (operator) norm of a matrix $A$ is defined as

$$
\|A\| = \max_{\|x\|=1} \|Ax\|
$$

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

The Frobenius norm $\|A\|_F$ is defined as

$$
\|X\|_F = \sqrt{\sum_{ij} x_{ij}^2}
$$
An (operator) norm of a matrix $A$ is defined as

$$\|A\| = \max_{\|x\| = 1} \|Ax\|$$

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

The Frobenius norm $\|A\|_F$ is defined as

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

I. Exercise: prove $\|X\|_F^2 = \text{Tr}(X^T X)$ where $\text{Tr}(\mathbf{A}) \triangleq \sum_i A_{ii}$ II. Bonus: verify that $\sigma_1(A) = \|A\|_2$
An (operator) *norm* of a matrix $A$ is defined as

$$\|A\| = \max_{\|x\| = 1} \|Ax\|$$

**Example:** Maximum singular value, $\sigma_1(A) = \|A\|_2$

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

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

**I. Exercise:** prove $\|X\|_F^2 = \text{Tr}(X^T X)$ where $\text{Tr}(\mathbf{A}) \triangleq \sum_i A_{ii}$

**II. Bonus:** verify that $\sigma_1(A) = \|A\|_2$

*We will mostly use the Frobenius norm for convenience*
Warmup example

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

\[
\min \| A - \hat{A} \|_F \quad \text{s.t.} \quad \hat{A}^T = \hat{A}
\]
Warmup example

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

\[
\min_{\hat{A}} \| A - \hat{A} \|_F \quad \text{s.t.} \quad \hat{A}^T = \hat{A}
\]

**Solution:** FaHo55

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

1. Let \( X \) be any \( n \times n \) symmetric matrix
2. Prove that \( \| A - \hat{A} \|_F \leq \| A - X \|_F \)
Warmup example

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

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

**Solution:** FaHo55

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

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

$$\| A - \hat{A} \|_F = \frac{1}{2} \| A - X + X^T - A^T \|_F$$
Warmup example

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

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

Solution: FaHo55

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

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

$$
\|A - \hat{A}\|_F = \frac{1}{2} \|A - X + X^T - A^T\|_F \\
\leq \frac{1}{2} \|A - X\|_F + \frac{1}{2} \|(X - A)^T\|_F = \|A - X\|_F,
$$
Suppose $A \in \mathbb{R}^{n \times n}$. What is the nearest symmetric matrix?

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

**Solution:** FaHo55

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

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

$$\| A - \hat{A} \|_F = \frac{1}{2} \| A - X + X^T - A^T \|_F$$

$$\leq \frac{1}{2} \| A - X \|_F + \frac{1}{2} \| (X - A)^T \|_F = \| A - X \|_F,$$

since $\| X \|_F = \| X^T \|_F$. 
Suppose $A \in \mathbb{R}^{m \times n}$ (we assume throughout $m \geq n$). What is the nearest rank-$k$ matrix, where $k < r = \text{rank}(A)$?
More challenging example

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

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

$$A \approx BC$$
More challenging example

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

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

$$A \approx BC$$

“Factors” $B$, $C$ can be computed by solving

$$\min \frac{1}{2} \| A - BC \|_F^2$$

But How??
The SVD

Recall fundamental matrix *factorization*:

\[
\text{Singular Value Decomposition}
\]
The SVD

Recall fundamental matrix *factorization*:

\[
\text{Singular Value Decomposition}
\]

**SVD (Thm. 2.5.2 [GoLo96])**

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

\[
U^T AV = \text{Diag}(\sigma_1, \ldots, \sigma_p), \quad p = \min(m, n),
\]

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

\[
U^T A V = \text{Diag}(\sigma_1, \ldots, \sigma_p), \quad p = \min(m, n),
\]

where \( \sigma_1 \geq \sigma_2 \geq \cdots \geq 0 \).

\[
A_{m \times n} = U_{m \times m} \begin{bmatrix} \Sigma_{n \times n} & \end{bmatrix} \begin{bmatrix} \Sigma_{n \times n} & \end{bmatrix} V^T_{n \times n}
\]

Exercise: \( A = \sum_i \sigma_i u_i v_i^T \) \hspace{1cm} (\( U = [u_i] \) and \( V = [v_i] \))
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 \textit{rank}, \textit{range space}, \textit{null space} of $A$. 

\begin{align*}
\text{Theorem (Optimality of SVD)} \\
\text{Let } A \text{ have the SVD } U \Sigma V^T. \text{ If } k < \text{rank}(A) \text{ and } A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T, \text{ then } \\
\|A - A_k\|_2 \leq \|A - B\|_2, \text{ s.t. } \text{rank}(B) \leq k, \text{ and } \\
\|A - A_k\|_F \leq \|A - B\|_F, \text{ s.t. } \text{rank}(B) \leq k.
\end{align*}
Approximation example: truncated SVD

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

\textbf{Theorem (Optimality of SVD)}

\textit{Let} \( \mathbf{A} \) \textit{have the SVD} \( \mathbf{U} \Sigma \mathbf{V}^T \). \textit{If} \( k < \text{rank}(\mathbf{A}) \) \textit{and}

\[
\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T,
\]

\textit{then,}

\[
\| \mathbf{A} - \mathbf{A}_k \|_2 \leq \| \mathbf{A} - \mathbf{B} \|_2, \quad \text{s.t.} \quad \text{rank}(\mathbf{B}) \leq k
\]

\[
\| \mathbf{A} - \mathbf{A}_k \|_F \leq \| \mathbf{A} - \mathbf{B} \|_F, \quad \text{s.t.} \quad \text{rank}(\mathbf{B}) \leq k.
\]
Truncated SVD (TSVD) – Proof Sketch

Prove: TSVD yields “best” Rank-\(k\) approximation to matrix \(A\)

Proof: (2-norm).

1. First verify that \(\|A - A_k\|_2 = \sigma_{k+1}\)
Truncated SVD (TSVD) – Proof Sketch

Prove: TSVD yields “best” Rank-\( k \) approximation to matrix \( A \)

Proof: (2-norm).

1. First verify that \( \| A - A_k \|_2 = \sigma_{k+1} \)
2. Let \( B \) be any rank-\( k \) matrix
Truncated SVD (TSVD) – Proof Sketch

Prove: TSVD yields “best” Rank-\(k\) approximation to matrix \(A\)

Proof: (2-norm).

1. First verify that \(\|A - A_k\|_2 = \sigma_{k+1}\)
2. Let \(B\) be any rank-\(k\) matrix
3. Prove that \(\|A - B\|_2 \geq \sigma_{k+1}\)
Truncated SVD (TSVD) – Proof Sketch

Prove: TSVD yields “best” Rank-\( k \) approximation to matrix \( A \)

Proof: (2-norm).

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

Since rank(\( B \)) = \( k \), there are \( n - k \) vectors that span the null-space \( \mathcal{N}(B) \). But \( \mathcal{N}(B) \cap V_{k+1} \neq \{0\} \) (??), so we can pick a unit-norm vector \( z \in \mathcal{N}(B) \cap V_{k+1} \). Now \( Bz = 0 \), so
Truncated SVD (TSVD) – Proof Sketch

Prove: TSVD yields “best” Rank-\(k\) approximation to matrix \(A\)

Proof: (2-norm).

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

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

\[
|A - B|_2^2 \geq |(A - B)z|_2^2 = |Az|_2^2 = \sum_{i=1}^{k+1} \sigma_i^2 (v_i^T z)^2 \geq \sigma_{k+1}^2
\]
Truncated SVD (TSVD) – Proof Sketch

Prove: TSVD yields “best” Rank-

Proof: (2-norm).

\[ \text{First verify that } \| \mathbf{A} - \mathbf{A}_k \|_2 = \sigma_{k+1} \]
\[ \text{Let } \mathbf{B} \text{ be any rank-}k \text{ matrix} \]
\[ \text{Prove that } \| \mathbf{A} - \mathbf{B} \|_2 \geq \sigma_{k+1} \]

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

\[ \| \mathbf{A} - \mathbf{B} \|_2^2 \geq \| (\mathbf{A} - \mathbf{B}) \mathbf{z} \|_2^2 = \| \mathbf{A} \mathbf{z} \|_2^2 = \sum_{i}^{k+1} \sigma_i^2 (\mathbf{v}_i^T \mathbf{z})^2 \geq \sigma_{k+1}^2 \]

We used: \( \| \mathbf{A} \mathbf{z} \|_2 \leq \| \mathbf{A} \|_2 \| \mathbf{z} \|_2 \)
TSVD – Message

If we are seeking a rank-$k$ approximation to $A$

$$A \approx BC$$
If we are seeking a rank-$k$ approximation to $A$

\[ A \approx BC \]
If we are seeking a rank-$k$ approximation to $A$

$A \approx BC$

TSVD yields: $B = U_k \Sigma_k$, and $C = V_k^T$
Example Problems
Problems

1. Truncated SVD, PCA
Problems

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

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

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

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

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 and so on....
Problems

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 and so on....
Problems

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
Problems

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

and so on....
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....
Principal component analysis, aka PCA based on TSVD

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

Original matrix

\[
\begin{array}{ccc}
a & + & a & + & + \\
\circ & z & \circ & & \circ \\
a & + & a & + & + \\
* & * & * & & * \\
* & * & * & & * \\
\circ & z & \circ & & \circ \\
\end{array}
\]
### Clustering, Co-clustering

#### Clustered matrix

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>a</th>
<th>+</th>
<th>+</th>
<th>+</th>
</tr>
</thead>
<tbody>
<tr>
<td>z</td>
<td>z</td>
<td></td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>_</td>
<td>_</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>_</td>
<td>_</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Z</td>
<td>Z</td>
<td></td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>

After clustering and permutation
Clustering, Co-clustering

Co-clustered matrix

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>a</th>
<th>z</th>
<th>z</th>
<th>_</th>
<th>_</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>z</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td></td>
<td></td>
</tr>
<tr>
<td>z</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After co-clustering and permutation
Clustering, Co-clustering

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

We cluster *columns* of $X$.

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

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

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

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

We cluster *columns* of $X$

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

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

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

**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* $A$

$$\begin{array}{ll}
\min & \|A - \hat{A}\|_F \\
\text{s.t.} & \hat{A} \succeq 0
\end{array}$$
Nearest positive definite

Sometimes one needs to find for a symmetric $A$

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

Solution: BoXi06

$A = A_+ - A_-, \quad A_+ = A_+^T \succeq 0, \quad A_- = A_-^T \succeq 0, \quad A_+A_- = 0$. Moreover

$$\|A - A_+\|_F = \|A_-\|_F \leq \|A - X\|_F$$

for any $X \succeq 0$. (Observe, computing $A_-\,$ enough)
Nearest positive definite

Sometimes one needs to find for a symmetric $A$

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

**Solution:** BoXi06

$A = A_+ - A_-$, $A_+ = A_+^T \succeq 0$, $A_- = A_-^T \succeq 0$, $A_+A_- = 0$. Moreover

$$\| A - A_+ \|_F = \| A_- \|_F \leq \| A - X \|_F$$

for any $X \succeq 0$. (Observe, computing $A_-$ enough)

Modified Cholesky: $A + E$ with $\| E \|_2 = O(n)$
Nonnegative matrix approximation (aka NMF)

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

We could invoke SVD – but sometimes not desirable:
Nonnegative matrix approximation (aka NMF)

Say we are seeking a low-rank approx $A \approx BC$

We could invoke SVD – but sometimes not desirable:

- SVD yields dense $B$ and $C$
- $B$ and $C$ full of negative numbers, even if $A \geq 0$
- SVD decomposition might not be that easy to interpret
Nonnegative matrix approximation (aka NMF)

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

We could invoke SVD – but sometimes not desirable:

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

So why not impose $B \geq 0$, $C \geq 0$?
Problems

Nonnegative matrix approximation (aka NMF)

SVD

\[ \begin{array}{c}
\text{SVD} \\
\end{array} \]

\[ \times \]

\[ \begin{array}{c}
= \\
\end{array} \]

\[ \text{Image} \]
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
We consider the *NMA* problem:

\[ A \approx BC \quad \text{s.t.} \quad B, C \geq 0. \]
Measure quality of approximation using $\Delta$:

\[
\text{minimize } \Delta(A, BC) \quad \text{s.t. } B, C \geq 0
\]
Algorithms: NMA

Measure quality of approximation using $\Delta$:

\[
\text{minimize } \Delta(A, BC) \quad \text{s.t. } B, C \geq 0
\]

Instantiations: where $\Delta$ is

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

Measure quality of approximation using $\Delta$:

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

Instantiations: where $\Delta$ is

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

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

Is this problem solvable?
Least-squares NMA

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| A - BC \|_F^2 \\
\text{s.t.} & \quad B, C \geq 0.
\end{align*}
\]

- Is this problem solvable? Yes!
Least-squares NMA

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

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

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

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

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

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

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \| A - BC \|_F^2 \\
\text{s.t.} & \quad B, C \geq 0.
\end{align*}
\]

- 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

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

- 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: $A, k$

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

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 $B^0$, $t \leftarrow 0$
NMA Algorithm: Alternating Methods

Generic Iterative Alternating Descent

1. Initialize $B^0$, $t \leftarrow 0$
2. Compute $C^{t+1}$ s.t. $\Delta(A, B^t C^{t+1}) \leq \Delta(A, B^t C^t)$
## NMA Algorithm: Alternating Methods

#### Generic Iterative Alternating Descent

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

Generic Iterative Alternating Descent

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

For least-squares NMA

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

*Alternating Least Squares* computes

$$C = \operatorname{arg\,min}_C \| A - B^t C \|^2_F;$$
Alternating least-squares

*Alternating Least Squares* computes

\[ C = \arg\min_C \| A - B^t C \|_F^2; \quad C^{t+1} \leftarrow \max(0, C) \]
Alternating least-squares

*Alternating Least Squares* computes

\[
C = \arg\min_C \| A - B^t C \|_F^2; \quad C^{t+1} \leftarrow \max(0, C)
\]

\[
B = \arg\min_B \| A - B C^{t+1} \|_F^2;
\]
**Alternating Least Squares** computes

\[
\begin{align*}
C &= \arg\min_C \|A - B^t C\|_F^2; \\
B &= \arg\min_B \|A - BC^{t+1}\|_F^2; \\
C^{t+1} &\leftarrow \max(0, C) \\
B^{t+1} &\leftarrow \max(0, B)
\end{align*}
\]
Alternating least-squares

**Alternating Least Squares** computes

\[ C = \underset{C}{\operatorname{arg\,min}} \| A - B^t C \|_F^2; \]

\[ B = \underset{B}{\operatorname{arg\,min}} \| A - BC^{t+1} \|_F^2; \]

\[ C^{t+1} \leftarrow \max(0, C) \]

\[ B^{t+1} \leftarrow \max(0, B) \]

ALS is fast, simple, often effective, but ...
Alternating least-squares

*Alternating Least Squares* computes

\[
C = \text{argmin}_C \| A - B^t C \|_F^2 ;
\]

\[
B = \text{argmin}_B \| A - BC^{t+1} \|_F^2 ;
\]

\[
C^{t+1} \leftarrow \max(0, C)
\]

\[
B^{t+1} \leftarrow \max(0, B)
\]

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

Bad News!
Alternating least-squares

**Alternating Least Squares** computes

\[
C = \arg \min_C \| A - B^t C \|_F^2; \quad C^{t+1} \leftarrow \max(0, C)
\]

\[
B = \arg \min_B \| A - BC^{t+1} \|_F^2; \quad B^{t+1} \leftarrow \max(0, B)
\]

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

**Bad News!**

\[
\| A - B^{t+1} C^{t+1} \|_F^2 \leq \| A - B^t C^{t+1} \|_F^2 \leq \| A - B^t C^t \|_F^2
\]

is NOT guaranteed!
Alternating NNLS

“Simple” fix is to instead compute

$$C^{t+1} = \arg\min_C \| A - B^t C \|_F^2 \quad \text{s.t.} \quad C \geq 0$$
“Simple” fix is to instead compute

\[ C^{t+1} = \arg\min_C \| A - B^t C \|_F^2 \quad \text{s.t.} \quad C \geq 0 \]

\[ B^{t+1} = \arg\min_B \| A - BC^{t+1} \|_F^2 \quad \text{s.t.} \quad B \geq 0 \]
Alternating NNLS

“Simple” fix is to instead compute

\[ C^{t+1} = \arg\min_C \| A - B^t C \|_F^2 \quad \text{s.t.} \quad C \geq 0 \]

\[ B^{t+1} = \arg\min_B \| A - B C^{t+1} \|_F^2 \quad \text{s.t.} \quad B \geq 0 \]

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

“Simple” fix is to instead compute

\[
C^{t+1} = \arg\min_C \|A - B^t C\|_F^2 \quad \text{s.t.} \quad C \geq 0
\]

\[
B^{t+1} = \arg\min_B \|A - BC^{t+1}\|_F^2 \quad \text{s.t.} \quad B \geq 0
\]

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

\[
C^{t+1} = \arg\min_C \|A - B^t C\|_F^2 \quad \text{s.t.} \quad C \geq 0
\]

\[
B^{t+1} = \arg\min_B \|A - BC^{t+1}\|_F^2 \quad \text{s.t.} \quad B \geq 0
\]

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

How to solve the “argmin”??
The nonnegative least squares (NNLS) subproblem is

$$\min_{c \geq 0} \ 1/2 \| A - BC \|_F^2$$

Essentially the same as solving

$$\min_{c \geq 0} \ f(c) = 1/2 \| a - Bc \|_2^2$$
The *nonnegative least squares* (NNLS) subproblem is

\[
\min_{c \geq 0} \quad \frac{1}{2} \| A - BC \|_F^2
\]

Essentially the same as solving

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

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

$$\min \ f(c) = \frac{1}{2} \| a - Bc \|_2^2$$
Consider first the \textit{unconstrained} problem

\[
\min \quad f(c) = \frac{1}{2} \| a - Bc \|_2^2
\]

\[
\nabla f(c^*) = 0
\]

Familiar gradient descent
**Background – Gradient Methods**

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

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

- **Step-size** $\alpha_k \geq 0$
- **Descent direction** $-\nabla f(\mathbf{c}^k)$
Gradient descent: Vector $c^{k+1}$ is chosen as

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

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

More generally, *Gradient methods* iterate as

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

where the descent direction is

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

Gradient methods

\[ \mathbf{c}^{k+1} = \mathbf{c}^k + \alpha_k \mathbf{d}^k, \quad k = 0, 1, \ldots \]

- Different choices of \( \mathbf{d}^k \)
  - Scaled gradient \( \mathbf{d}^k = -\mathbf{D}^k \nabla f(\mathbf{c}^k), \mathbf{D}^k > 0 \)
  - Note: \( \mathbf{D}^k = \mathbf{I} \) gives *steepest descent*
  - Newton’s method, conjugate gradients, etc.
Gradient Methods

Gradient methods

\[ \mathbf{c}^{k+1} = \mathbf{c}^k + \alpha_k \mathbf{d}^k, \quad k = 0, 1, \ldots \]

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

- Different choices of \( \alpha_k \)
  - Limited minimization \( \alpha_k = \arg\min_{0 \leq \alpha \leq s} f(\mathbf{c}^k + \alpha \mathbf{d}^k) \)
  - Armijo-line-search, backtracking, etc.
Gradient Methods

Gradient methods

\[ \mathbf{c}^{k+1} = \mathbf{c}^k + \alpha_k \mathbf{d}^k, \quad k = 0, 1, ... \]

- Different choices of \( \mathbf{d}^k \)
  - Scaled gradient \( \mathbf{d}^k = -\mathbf{D}^k \nabla f(\mathbf{c}^k), \quad \mathbf{D}^k > 0 \)
  - Note: \( \mathbf{D}^k = \mathbf{I} \) gives steepest descent
  - Newton’s method, conjugate gradients, etc.

- Different choices of \( \alpha_k \)
  - Limited minimization \( \alpha_k = \arg\min_{0 \leq \alpha \leq s} f(\mathbf{c}^k + \alpha \mathbf{d}^k) \)
  - Armijo-line-search, backtracking, etc.

Step-sizes \( \alpha_k \) chosen to ensure descent

\[ f(\mathbf{c}^{k+1}) < f(\mathbf{c}^k) \]
Gradient Methods – Illustration

\[ f(c) = l_1 \]

\[ f(c) = l_2 < l_1 \]

\[ l_3 < l_2 \]

\[ \nabla f(c) \]

\[ c - \alpha_1 \nabla f(c) \]

\[ c + \alpha_1 d \]

\[ c + \alpha_2 d \]

\[ c - \delta_1 \nabla f(c) \]

\[ -\nabla f(c) \]

(adapted from Bertsekas, Nonlinear Programming)
Gradient Methods – Handling constraints

Our problem is constrained

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

Recall gradient-descent iteration

$$c^{k+1} = c^k - \alpha_k \nabla f(c^k), \quad k = 0, 1, \ldots$$
Gradient Methods – Handling constraints

Our problem is constrained

\[
\min_{c \geq 0} \quad f(c) = \frac{1}{2} \| a - Bc \|_F^2
\]

Replace it with \textit{Gradient-Projection}!

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

\(P_+ x = \max(0, x)\): projection to ensure \textit{non-negativity}
Gradient Methods – Handling constraints

Our problem is constrained

\[
\min_{c \geq 0} \quad f(c) = \frac{1}{2} \| a - Bc \|_F^2
\]

Replace it with *Gradient-Projection*!

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

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

Note: Step-size \(\alpha_k\) selected to ensure descent

\[f(c^{k+1}) < f(c^k)\]
Alternating NNLS – summary

\[
\text{minimize } \frac{1}{2} \| A - BC \|_F^2 \quad \text{s.t. } \quad B, C \geq 0.
\]
Alternating NNLS – summary

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

by alternating

\[
C^{t+1} = \arg\min_{C \geq 0} F(C) = \| A - B^t C \|_F^2
\]

\[
B^{t+1} = \arg\min_{B \geq 0} F(B) = \| A - BC^{t+1} \|_F^2,
\]
Alternating NNLS – summary

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

by alternating

$$C^{t+1} = \arg\min_{C \geq 0} F(C) = \| A - B^t C \|_F^2$$

$$B^{t+1} = \arg\min_{B \geq 0} F(B) = \| A - BC^{t+1} \|_F^2,$$

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

$$C^{k+1} = P_+ (C^k - \alpha_k \nabla F(C^k)), \quad k = 0, 1, \ldots$$
Alternating NNLS – summary

minimize  \[ \frac{1}{2} \| A - BC \|_F^2 \]  s.t.  \[ B, C \geq 0. \]

by alternating

\[ C^{t+1} = \arg\min_{C \geq 0} F(C) = \| A - B^t C \|_F^2 \]

\[ B^{t+1} = \arg\min_{B \geq 0} F(B) = \| A - BC^{t+1} \|_F^2, \]

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

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

So are we ready to implement this?
Alternating NNLS – summary

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

by alternating

\[ C^{t+1} = \text{argmin}_{C \geq 0} F(C) = \| A - B^t C \|_F^2 \]
\[ B^{t+1} = \text{argmin}_{B \geq 0} F(B) = \| A - BC^{t+1} \|_F^2, \]

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

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

So are we ready to implement this?
How to compute \( \nabla F(C^k) \)?
Background – Matrix Derivatives

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

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

I. Compute $\partial \text{Tr}(XY) / \partial X$
Derivative of \( f : \mathbb{R}^{m \times n} \rightarrow \mathbb{R} \) is defined as

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

I. Compute \( \partial \text{Tr}(XY) / \partial X \)

Recall \( \text{Tr}(XY) = \sum_{ij} x_{ij} y_{ji} \). Hence,

\( \partial \text{Tr}(XY) / \partial X = Y^T. \)
Background – Matrix Derivatives

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

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

II. Verify that: \( \frac{\partial \|X\|_F^2}{\partial X} = 2X \)
Derivative of \( f : \mathbb{R}^{m \times n} \rightarrow \mathbb{R} \) is defined as

\[
\frac{\partial f(X)}{\partial X} \triangleq \begin{bmatrix}
\frac{\partial f(X)}{\partial x_{pq}}
\end{bmatrix}
\]

II. Verify that: \( \frac{\partial \|X\|_F^2}{\partial X} = 2X \)

Solution:

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

\[
\frac{\partial \|X\|_F^2}{\partial X} = \frac{\partial \text{Tr}(X^T X)}{\partial x_{pq}} = \frac{\partial (\sum_{ij} x_{ij}^2)}{\partial x_{pq}} = 2x_{pq}.
\]
**Derivative** of \( f : \mathbb{R}^{m \times n} \rightarrow \mathbb{R} \) is defined as

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

III. Verify that: \( \frac{\partial \text{Tr}(X^TAX)}{\partial X} = (A + A^T)X \)
**Derivative** of $f : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}$ is defined as

$$\frac{\partial f(X)}{\partial X} \triangleq \begin{bmatrix} \frac{\partial f(X)}{\partial x_{pq}} \end{bmatrix}$$

III. Verify that: $\frac{\partial \text{Tr}(X^TAX)}{\partial X} = (A + A^T)X$

**Solution:** Brute force

$$\text{Tr}(X^TAX) = \sum_{ij} x_{ij} (AX)_{ji} = \sum_{ijk} x_{ij} a_{jk} x_{ki}$$
Derivative of $f : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}$ is defined as

$$
\frac{\partial f(X)}{\partial X} \triangleq \begin{bmatrix}
\frac{\partial f(X)}{\partial x_{pq}}
\end{bmatrix}
$$

**Exercise:** IV.

Let $F(C) = \frac{1}{2} \| A - BC \|_F^2$; compute $\partial F / \partial C$.
Background – Matrix Derivatives

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

\[
\frac{\partial f(X)}{\partial X} \triangleq \begin{bmatrix} \frac{\partial f(X)}{\partial x_{pq}} \end{bmatrix}
\]

**Exercise:** IV.

Let \( F(C) = \frac{1}{2} \| A - BC \|_F^2 \); compute \( \partial F / \partial C \)

**Solution:**

\[
F(C) = \| A \|_F^2 - 2 \text{Tr}(CA^T B) + \text{Tr}(C^T B^T BC)
\]

\[
\frac{\partial F(C)}{\partial C} = -2B^T A + 2B^T BC.
\]
In passing: The Fréchet derivative

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

$$f(X + E) - f(X) - L(X, E) = o(\|E\|)$$

The Fréchet derivative $D_f(X)$ (of $f$ at point $X$) identified via:

$$L(X, E) = D_f(X)(E)$$

Can be used to develop matrix calculus formally.
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

<table>
<thead>
<tr>
<th>Exercise: Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is the computational complexity per (major) iteration?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solution:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A lot! Especially since there might be many (inner) gradient projection iterations for each major iteration.</td>
</tr>
</tbody>
</table>

What to do?
Alternating descent

Idea! Do not insist on minimization

Recall that we originally wanted

\[ \| A - B \|_2^2 + C \|_2^2 \leq \| A - B \|_2^2 \]

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!
Alternating descent

Idea! Do not insist on minimization

Recall that we originally wanted descent

\[ \| A - B^{t+1} C^{t+1} \|_F^2 \leq \| A - B^t C^{t+1} \|_F^2 \leq \| A - B^t C^t \|_F^2 \]
Alternating descent

Idea! Do not insist on minimization

Recall that we originally wanted descent

\[ \| A - B^{t+1} C^{t+1} \|_F^2 \leq \| A - B^t C^{t+1} \|_F^2 \leq \| A - B^t C^t \|_F^2 \]

- 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

\[
\|A - B^{t+1} C^{t+1}\|_F^2 \leq \|A - B^t C^{t+1}\|_F^2 \leq \|A - B^t C^t\|_F^2
\]

- 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

\[ \|A - B^{t+1} C^{t+1}\|_F^2 \leq \|A - B^t C^{t+1}\|_F^2 \leq \|A - B^t C^t\|_F^2 \]

- 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”

\[
\begin{align*}
C' & \leftarrow C \odot \frac{B^T A}{B^T BC} \\
B' & \leftarrow B \odot \frac{A C'^T}{B C' C'^T}.
\end{align*}
\]

This algorithm’s simplicity made NMA popular.

Note: \(A \odot B = [a_{ij}b_{ij}]\) – *elementwise multiplication*
Lee & Seung (2000) proposed the following “algorithm”

\[ C' \leftarrow C \odot \frac{B^T A}{B^T BC} \]

\[ B' \leftarrow B \odot \frac{AC'^T}{BC'C'^T} \]

This algorithm’s simplicity made NMA popular.

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

- Easy to see that nonnegativity respected
Lee & Seung (2000) proposed the following “algorithm”

\[ C' \leftarrow C \odot \frac{B^T A}{B^T B C} \]

\[ B' \leftarrow B \odot \frac{A C'^T}{B C' C'^T} \]

This algorithm’s simplicity made NMA popular.

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

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

\[ \| A - B' C' \|_F^2 \leq \| A - BC' \|_F^2 \leq \| A - BC \|_F^2 \]
Let $c$ be an arbitrary column of $C$. Consider the subproblem:

\[
\min_{c \geq 0} f(c) = \frac{1}{2} \|a - Bc\|_F^2
\]

A general technique for deriving “descent” methods:
Multiplicative updates – preliminaries

Let $c$ be an arbitrary column of $C$. Consider the subproblem:

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

A general technique for deriving “descent” methods:

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

   $$g(c, c) = f(c), \quad \text{for all } c,$$
   $$g(c, \tilde{c}) \geq f(c), \quad \text{for all } c, \tilde{c}.$$
Multiplicative updates – preliminaries

Let \( \mathbf{c} \) be an arbitrary column of \( \mathbf{C} \). Consider the subproblem:

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

A general technique for deriving “descent” methods:

1. Find a function \( g(\mathbf{c}, \tilde{\mathbf{c}}) \) that satisfies:
   \[
   g(\mathbf{c}, \mathbf{c}) = f(\mathbf{c}), \quad \text{for all } \mathbf{c},
   \]
   \[
   g(\mathbf{c}, \tilde{\mathbf{c}}) \geq f(\mathbf{c}), \quad \text{for all } \mathbf{c}, \tilde{\mathbf{c}}.
   \]

2. Compute \( \mathbf{c}^{t+1} = \arg\min_{\mathbf{c}} g(\mathbf{c}, \mathbf{c}^t) \)
Let \( c \) be an arbitrary column of \( C \). Consider the subproblem:

\[
\min_{c \geq 0} \ f(c) = \frac{1}{2} \| a - Bc \|_F^2
\]

A general technique for deriving “descent” methods:

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

\[
g(c, c) = f(c), \quad \text{for all } c,
\]

\[
g(c, \tilde{c}) \geq f(c), \quad \text{for all } c, \tilde{c}.
\]

2. Compute \( c^{t+1} = \text{argmin}_c \ g(c, c^t) \)

3. Then we have descent

\[
f(c^{t+1})
\]
Multiplicative updates – preliminaries

Let $c$ be an arbitrary column of $C$. Consider the subproblem:

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

A general technique for deriving “descent” methods:

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

   $$g(c, c) = f(c), \quad \text{for all } c,$$

   $$g(c, \tilde{c}) \geq f(c), \quad \text{for all } c, \tilde{c}.$$

2. Compute $c^{t+1} = \arg\min_c g(c, c^t)$

3. Then we have descent

   $$f(c^{t+1}) \stackrel{\text{def}}{=} g(c^{t+1}, c^t)$$
Multiplicative updates – preliminaries

Let $c$ be an arbitrary column of $C$. Consider the subproblem:

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

A general technique for deriving “descent” methods:

1. Find a function $g(c, \tilde{c})$ that satisfies:
   
   \begin{align*}
   g(c, c) &= f(c), \quad \text{for all } c, \\
   g(c, \tilde{c}) &\geq f(c), \quad \text{for all } c, \tilde{c}.
   \end{align*}

2. Compute $c^{t+1} = \arg\min_c g(c, c^t)$

3. Then we have descent

$$f(c^{t+1}) \overset{\text{def}}{=} \min f(c^{t+1}, c^t) \leq g(c^{t+1}, c^t) \leq g(c^t, c^t)$$
Multiplicative updates – preliminaries

Let \( c \) be an arbitrary column of \( C \). Consider the subproblem:

\[
\min_{c \geq 0} f(c) = \frac{1}{2} \| a - Bc \|_F^2
\]

A general technique for deriving “descent” methods:

1. Find a function \( g(c, \tilde{c}) \) that satisfies:
   \[
   g(c, c) = f(c), \quad \text{for all } c,
   \]
   \[
   g(c, \tilde{c}) \geq f(c), \quad \text{for all } c, \tilde{c}.
   \]

2. Compute \( c^{t+1} = \arg\min_c g(c, c^t) \)

3. Then we have descent

\[
f(c^{t+1}) \overset{\text{def}}{=} g(c^{t+1}, c^t) \overset{\arg\min}{\leq} g(c^t, c^t) \overset{\text{def}}{=} f(c^t)
\]
Constructing $g$

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

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

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

$$h(\sum \lambda_i x_i) \leq \sum \lambda_i h(x_i), \text{ where } \lambda_i \geq 0, \sum \lambda_i = 1$$
Constructing $g$

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

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

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

Non-convex, and a convex set
Constructing $g$

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

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

\[ h(\sum_i \lambda_i x_i) \leq \sum_i \lambda_i h(x_i), \text{ where } \lambda_i \geq 0, \sum_i \lambda_i = 1 \]
Constructing $g$

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

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

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

$$f(c) = \frac{1}{2} \sum_i (a_i - b_i^T c)^2 =$$
Constructing $g$

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

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

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

\[
f(c) = \frac{1}{2} \sum_i (a_i - b_i^T c)^2 = \frac{1}{2} \sum_i a_i^2 - 2 a_i b_i^T c + (b_i^T c)^2
\]
Constructing $g$

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

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

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

\[
f(c) = \frac{1}{2} \sum_i (a_i - b_i^T c)^2 = \frac{1}{2} \sum_i a_i^2 - 2a_i b_i^T c + (b_i^T c)^2
\]
\[
= \frac{1}{2} \sum_i a_i^2 - 2a_i b_i^T c + \frac{1}{2} \sum_i (\sum_j b_{ij} c_j)^2
\]
Constructing $g$

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

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

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

\[
f(c) = \frac{1}{2} \sum_i (a_i - b_i^T c)^2 = \frac{1}{2} \sum_i a_i^2 - 2a_i b_i^T c + (b_i^T c)^2
\]

\[
= \frac{1}{2} \sum_i a_i^2 - 2a_i b_i^T c + \frac{1}{2} \sum_i (\sum_j b_{ij} c_j)^2
\]

\[
= \frac{1}{2} \sum_i a_i^2 - 2a_i b_i^T c
\]
Constructing $g$

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

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

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

$$f(c) = \frac{1}{2} \sum_i (a_i - b_i^T c)^2 = \frac{1}{2} \sum_i a_i^2 - 2 a_i b_i^T c + (b_i^T c)^2$$
$$= \frac{1}{2} \sum_i a_i^2 - 2 a_i b_i^T c + \frac{1}{2} \sum_i (\sum_j b_{ij} c_j)^2$$
$$= \frac{1}{2} \sum_i a_i^2 - 2 a_i b_i^T c + \frac{1}{2} \sum_i (\sum_j \lambda_{ij} b_{ij} c_j / \lambda_{ij})^2$$
Constructing $g$

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

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

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

\[
f(c) = \frac{1}{2} \sum_i (a_i - b_i^T c)^2 = \frac{1}{2} \sum_i a_i^2 - 2 a_i b_i^T c + (b_i^T c)^2
\]
\[
= \frac{1}{2} \sum_i a_i^2 - 2 a_i b_i^T c + \frac{1}{2} \sum_i (\sum_j b_{ij} c_j)^2
\]
\[
= \frac{1}{2} \sum_i a_i^2 - 2 a_i b_i^T c + \frac{1}{2} \sum_i (\sum_j \lambda_{ij} b_{ij} c_j / \lambda_{ij})^2
\]
\[
\leq \frac{1}{2} \sum_i a_i^2 - 2 a_i b_i^T c + \frac{1}{2} \sum_{ij} \lambda_{ij} (b_{ij} c_j / \lambda_{ij})^2
\]
Constructing $g$

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

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

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

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

$$= \frac{1}{2} \sum_i a_i^2 - 2a_i b_i^T c + \frac{1}{2} \sum_i (\sum_j b_{ij} c_j)^2$$

$$= \frac{1}{2} \sum_i a_i^2 - 2a_i b_i^T c + \frac{1}{2} \sum_i (\sum_j \lambda_{ij} b_{ij} c_j / \lambda_{ij})^2$$

$$\leq \frac{1}{2} \sum_i a_i^2 - 2a_i b_i^T c + \frac{1}{2} \sum_{ij} \lambda_{ij} (b_{ij} c_j / \lambda_{ij})^2$$

$$= g(c, \tilde{c}), \text{ where } \lambda_{ij} \text{ are convex coeffs}$$
Constructing $g$

In summary:

$$f(c) = \frac{1}{2} \| a - Bc \|_2^2$$

$$g(c, \tilde{c}) = \frac{1}{2} \| a \|_2^2 - \sum_i a_i b_i^T c + \frac{1}{2} \sum_{ij} \lambda_{ij} (b_{ij} c_j / \lambda_{ij})^2$$

Now we **pick** $\lambda_{ij}$
Constructing $g$

In summary:

$$f(c) = \frac{1}{2} \| a - Bc \|^2_2$$

$$g(c, \tilde{c}) = \frac{1}{2} \| a \|^2_2 - \sum_i a_i b_i^T c + \frac{1}{2} \sum_{ij} \lambda_{ij} (b_{ij} c_j / \lambda_{ij})^2$$

Now we pick $\lambda_{ij}$

$$\lambda_{ij} = \frac{b_{ij} \tilde{c}_j}{\sum_k b_{ik} \tilde{c}_k} = \frac{b_{ij} \tilde{c}_j}{b_i^T \tilde{c}}$$
Constructing $g$

In summary:

$$f(c) = \frac{1}{2} \| a - Bc \|^2_2$$

$$g(c, \tilde{c}) = \frac{1}{2} \| a \|^2_2 - \sum_i a_i b_i^T c + \frac{1}{2} \sum_{ij} \lambda_{ij} (b_{ij} c_j / \lambda_{ij})^2$$

Now we *pick* $\lambda_{ij}$

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

**Exercise:** Aux function

Verify that $g(c, c) = f(c)$;
Constructing $g$

In summary:

$$f(c) = \frac{1}{2} \| a - Bc \|^2$$

$$g(c, \tilde{c}) = \frac{1}{2} \| a \|^2 - \sum_i a_i b_i^T c + \frac{1}{2} \sum_{ij} \lambda_{ij} (b_{ij} c_j / \lambda_{ij})^2$$

Now we pick $\lambda_{ij}$

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

**Exercise:** Aux function

Verify that $g(c, c) = f(c)$;

**Exercise:** Richardson-Lucy

Let $f(c) = \sum_i a_i \log(a_i/(Bc)_i) - a_i + (Bc)_i$.

Derive an auxiliary function $g(c, \tilde{c})$ for this $f(c)$
Minimizing $g$

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

Solve $\partial g(c, c^t) / \partial c_p = 0$
Minimizing $g$

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

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

\[
\partial g / \partial c_p = - \sum_i a_i b_{ip} + \sum_i b_{ip} (b_i^T c^t) c_p / c_p
\]
Minimizing $g$

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

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

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

Which yields (verify!): $c_p = c^t_p \frac{[B^T a]_p}{[B^T B c^t]_p}$
Minimizing $g$

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

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

\[
\partial g / \partial c_p = - \sum_i a_i b_{ip} + \sum_i b_{ip} (b_i^T c^t) c_p / c_p^t
\]

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

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

\[
C^{t+1} = C^t \odot \frac{B^T A}{B^T B C^t}
\]
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 $\lambda_{ij}$
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 $\lambda_{ij}$
- Our technique one variant of repertoire of \textit{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 $\lambda_{ij}$
- 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 $\lambda_{ij}$
- 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

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

- We looked at least-squares NMA

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

- 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!
We looked at least-squares NMA:

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

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

Significant “eigengenes” $\implies$ independent biological processes and experimental artifacts.

Figure taken from: [http://www.bme.utexas.edu/research/orly/teaching/BME341](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: \( \mathbf{A} \in \mathbb{R}^{4857 \times 3891} \)
NMA – Text Analysis

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

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

<table>
<thead>
<tr>
<th>CISI</th>
<th>CRAN</th>
<th>MED</th>
</tr>
</thead>
<tbody>
<tr>
<td>retrieval</td>
<td>wing</td>
<td>patients</td>
</tr>
<tr>
<td>system</td>
<td>pressure</td>
<td>cells</td>
</tr>
<tr>
<td>systems</td>
<td>mach</td>
<td>growth</td>
</tr>
<tr>
<td>indexing</td>
<td>supersonic</td>
<td>hormone</td>
</tr>
<tr>
<td>scientific</td>
<td>shock</td>
<td>cancer</td>
</tr>
<tr>
<td>science</td>
<td>jet</td>
<td>treatment</td>
</tr>
<tr>
<td>index</td>
<td>lift</td>
<td>buckling</td>
</tr>
<tr>
<td>search</td>
<td>wings</td>
<td>blood</td>
</tr>
<tr>
<td>computer</td>
<td>body</td>
<td>cases</td>
</tr>
<tr>
<td>document</td>
<td>theory</td>
<td>cell</td>
</tr>
</tbody>
</table>
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 $1024 \times 256$
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 $1024 \times 256$
- Decompose the matrix into $1024 \times 17$ (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: $2.41 \times 10^7$
Image analysis – toy example

Via more advanced projection algorithm
Time: 62.3 seconds, Objective: $6.85 \times 10^{-4}$
Part of a face recognition system

- 143 images from MIT face image database
- Input matrix $A \in \mathbb{R}^{9216 \times 143}$
Part of a face recognition system

- A rank-20 approximation to the input
- The basis vectors (columns of $B$) 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

\[ y_t = a_t \star x + n_t \]

\[ y_0 = a_0 \star x_0 + n_0 \]
Our model of the video

\[ y_t = a_t \ast x + n_t \]

\[ y_0 = a_0 \ast x + n_0 \]

\[ y_1 = a_1 \ast x + n_1 \]

\[ y_2 = a_2 \ast x + n_2 \]

\[ y_k = a_k \ast x + n_k \]
Convolution operation may be written as

\[ a \star x = Ax = Xa \]
Convolution operation may be written as

\[ a \star x = Ax = Xa \]

\[
\begin{bmatrix}
  y_1 & y_2 & \cdots & y_t
\end{bmatrix}
\approx
\begin{bmatrix}
  a_1 & a_2 & \cdots & a_t
\end{bmatrix}
\]

\[ Y \approx XA \]
Multiframe blind deconvolution

We seek to minimize

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

We seek to minimize

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

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

\[ y_t = x_t \]
Double star epsilon lyrae

\[ y_t \approx a_t \star x_t \]

\[ 2 \]

\[ t \approx t^\star \]
Double star epsilon lyrae

\[ y_t \approx a_t \ast x_t \]

\text{time } t
Double star epsilon lyrae

time $t$

$y_t \approx a_t \ast x_t$

4
Double star epsilon lyrae

\[ t \approx a_t \times x_t \]
Double star epsilon lyrae

\[ y_t \approx a_t \ast x_t \]

6
Double star epsilon lyrae

Time $t$

$y_t \approx a_t \ast x_t$
Double star epsilon lyrae

time $t$

$y_t \approx a_t \star x_t$

8
Double star epsilon lyrae

time $t$

$y_t \approx a_t \star x_t$
Double star epsilon lyrae

\[
y_t \approx \alpha_t \ast x_t
\]

\[
\text{time } t
\]

10
Double star epsilon lyrae

time $t$

$y_t \approx a_t \star x_t$
Double star epsilon lyrae

\[ y_t \approx a_t \ast x_t \]
Double star epsilon lyrae

Time $t$ gives $y_t \approx a_t \star x_t$
Double star epsilon lyrae

Time $t$

$y_t \approx a_t \ast x_t$

14
Double star epsilon lyrae

\[ \text{time } t \]

\[ y_t \]

\[ \approx \]

\[ a_t \]

\[ \ast \]

\[ x_t \]
Double star epsilon lyrae

\[ y_t \approx a_t \ast x_t \]

16
Double star epsilon lyrae

\[ t \approx a_t \star x_t \]

17
Double star epsilon lyrae

\[
y_t \approx \alpha_t \ast x_t
\]

18

time
Double star epsilon lyrae

\[ \text{time } t \quad y_t \quad \approx \quad a_t \quad \ast \quad x_t \]
Double star epsilon lyrae

\[ y_t \approx a_t \ast x_t \]

image at time \( t \)
Double star epsilon lyrae

\[ t \approx a_t \star x_t \]
Double star epsilon lyrae

\[ y_t \approx a_t \ast x_t \]

22

time \( t \)
Double star epsilon lyrae

time $t$

$y_t \approx a_t \ast x_t$

23
Double star epsilon lyrae

Time $t$, $y_t \approx \alpha_t \ast x_t$
Double star epsilon lyrae

time $t$

$y_t \approx a_t \ast x_t$

25
Double star epsilon lyrae

$\mathbf{y}_t \approx a_t \star x_t$

Time $t$

26
Double star epsilon lyrae

Time $t$  

$y_t$  

$\approx$  

$a_t$  

$\star$  

$x_t$  

27
Double star epsilon lyrae

\[ y_t \approx a_t \star x_t \]

Time \( t \)

28
Double star epsilon lyrae

\[ \text{time } t \approx a_t \ast x_t \approx y_t \]
Double star epsilon lyrae

\[ t \approx a_t \ast x_t \]

30
Double star epsilon lyrae

\[
\begin{align*}
t &\approx t^* \\
y_t &\approx a_t \\
\star &\quad \star \\
x_t
\end{align*}
\]
Double star epsilon lyrae

time $t$ 

$y_t$ 

$\approx$ 

$a_t$ 

$\approx$ 

$x_t$ 

32
Double star epsilon lyrae

time $t$

$\mathbf{y}_t \approx a_t \star \mathbf{x}_t$

33
Double star epsilon lyrae

time $t$  

$y_t \approx a_t \star x_t$

34
Double star epsilon lyrae

time \( t \)

\[ y_t \approx a_t \ast x_t \]

35
Double star epsilon lyrae

time \( t \)  \[ y_t \] \( \approx \) \[ a_t \] \( \ast \) \[ x_t \]

36
Double star epsilon lyrae

time $t$

$37$

$y_t \approx a_t \ast x_t$
Double star epsilon lyrae

\[ y_t \approx a_t \star x_t \]
Double star epsilon lyrae

time $t$

$y_t \approx a_t \star x_t$

39
Double star epsilon lyrae

time \( t \)

\( y_t \)  \( \approx \)  \( a_t \)  \( * \)  \( x_t \)

40
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 \( \mathbf{A} \) of size \( 43.9M \times 769M \); total \( 4.38 \times 10^9 \) nonzeros (\( 1.2 \times 10^{-7} \) - density)
- 7 hours per iteration (dedicated cluster of 8 comps)
Closing: Huge Matrix Problems

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

- Input matrix $A$ of size $43.9M \times 769M$; total $4.38 \times 10^9$ nonzeros ($1.2 \times 10^{-7}$ - density)
- 7 hours per iteration (dedicated cluster of 8 comps)

I think YOU can do better!