Agenda

Talk on matrix decomposition (45 mins)

Questions & exercise (15 mins)

Talk on tensor decomposition (45 mins)

Questions & exercise (15 mins)

https://github.com/ahwillia/mit-tensor
Large-scale data analysis via matrix and tensor decompositions

Part 1: Matrix decomposition

Alex Williams
MIT, 09/05/2017
Examples of Matrix-Encoded Data

1. Gene Expression

- dissection
- dissociation
- single-cell sequencing
- data matrix holding transcript counts

(schematic adapted from La Manno & Gyllborg et al., 2016)

2. Neural Activity

- raw spikes
- neurons
- smoothed or trial-averaged neurons

3 mm
Examples of Matrix-Encoded Data

3. Fluorescence Images

Cortical neurons expressing YFP
(Kim & Zhang et al., 2016)

4. Spectrograms

Zebra Finch courtship song
(Provided by Emily Mackevicius)
Goal: extract simple structure from these large-scale datasets
Matrix Decomposition

A simple & general framework for extracting correlations and low-dimensional structure from matrix-coded datasets

\[ X \]

neurons

\[
\begin{bmatrix}
\vdots \\
\vdots \\
\vdots
\end{bmatrix}
\]

time

≈

time

trial factors

\[ \text{temporal factors} \]

\[ \text{neuron factors} \]
Matrix Decomposition

A simple & general framework for extracting correlations and low-dimensional structure from matrix-coded datasets
Matrix Decomposition

A simple & general framework for extracting correlations and low-dimensional structure from matrix-coded datasets

$$x_{ij} \approx \sum_{r=1}^{R} u_i^r v_j^r$$

$$X \approx UV^T$$

**X**

neurons

neurons

time

# components
Matrix Decomposition

A simple & general framework for extracting correlations and low-dimensional structure from matrix-coded datasets

\[ x_{ij} \approx \sum_{r=1}^{R} u_r^i v_r^j \]

\[ \begin{bmatrix} x_{ij} \end{bmatrix} \approx \begin{bmatrix} U & V^T \end{bmatrix} \]

\[ \text{neurons} \]

\[ \text{time} \]

\[ \text{# components} \]

\[ \text{# components} \]
Matrix Decomposition

A simple & general framework for extracting correlations and low-dimensional structure from matrix-coded datasets

\[ x_{ij} \approx \sum_{r=1}^{R} u^*_i v^*_j \]

\[ X \approx UV^T \]

neuron factors

temporal factors

\[ \text{cell #1} \quad \text{cell #6} \]

\[ \text{trial start} \quad \text{trial end} \]
Visualization of Matrix Decomposition

Original Data $\approx$ Factor Matrices $\times$ Rank-3 Reconstruction

Sum of Rank-1 Matrices $+$ $+$

Positive Numbers | Negative Numbers

zero
Talk Outline

1. Long list of matrix decomposition models
2. Optimization and model fitting
3. Visualization and model assessment
Talk Outline

1. Long list of matrix decomposition models
2. Optimization and model fitting
3. Visualization and model assessment
Matrix decomposition model, stated formally

\[
\begin{align*}
\text{minimize} & \quad \|X - UV^T\|_F^2 + \lambda_u f_u(U) + \lambda_v f_v(V) \\
\text{subject to} & \quad U \in \Omega_u, \ V \in \Omega_v
\end{align*}
\]
The simplest matrix decomposition is PCA

\[
\begin{align*}
\text{maximize} & \quad \|XVV^T\|_F^2 \\
(\text{subject to } V \text{ orthonormal})
\end{align*}
\]

\[
\begin{align*}
\text{minimize} & \quad \|X - UV^T\|_F^2 \\
(\text{subject to } U, V \text{ orthogonal})
\end{align*}
\]
There are an infinite # of solutions to PCA

\[ \hat{X} = UV^T \]

known as “the rotation problem”
There are an infinite # of solutions to PCA

known as “the rotation problem”

\[ \hat{X} = UV^T = UF^{-1}FV^T = U'V'T \]
There are an infinite # of solutions to PCA known as “the rotation problem”

\[ \hat{X} = UV^T = UF^{-1}FV^T = U'V'T \]
There are an infinite # of solutions to PCA known as “the rotation problem”

$$\hat{X} = UV^T = UF^{-1}FV^T = U'V'T$$
Nonnegative Matrix Factorization (NMF)

\[
\begin{align*}
\text{minimize} & \quad \|X - UV^T\|_F^2 \\
\text{subject to} & \quad U \geq 0, \; V \geq 0
\end{align*}
\]
Nonnegative Matrix Factorization (NMF)

\[
\begin{align*}
\text{minimize} & \quad \|X - UV^T\|_F^2 \\
\text{subject to} & \quad U \geq 0, \ V \geq 0
\end{align*}
\]
Nonnegative Matrix Factorization

\[
\begin{align*}
\text{minimize} & \quad \| \mathbf{X} - \mathbf{U} \mathbf{V}^T \|_F^2 \\
\text{subject to} & \quad \mathbf{U} \geq 0, \quad \mathbf{V} \geq 0
\end{align*}
\]

(Lee & Seung, 1999)
Nonnegative Matrix Factorization

\[
\begin{align*}
\text{minimize} \quad & ||X - UV^T||^2_F \\
\text{subject to} \quad & U \geq 0, \quad V \geq 0
\end{align*}
\]

NMF advantages:

- sparse factors
- additively combined
- can be "parts-based"
- can be unique (i.e. no rotation problem)

(Stodden & Donoho, 1999)
Sparse PCA*

\[
\text{minimize}_{U,V} \quad \|X - UV^T\|_F^2 + \lambda_u \sum_i \|u_i\|_1 + \lambda_v \sum_j \|v_j\|_2^2
\]

* Several variants of this model with different properties appear in the literature. Originally it was proposed by Zou et al. (2006).
Sparse PCA*

$$\begin{align*}
\text{minimize} \quad & \|X - UV^T\|_F^2 + \lambda_u \sum_i \|u_i\|_1 + \lambda_v \sum_j \|v_j\|_2^2 \\
\text{subject to} \quad & u_i, v_j \neq 0
\end{align*}$$

* Several variants of this model with different properties appear in the literature. Originally it was proposed by Zou et al. (2006).
Why L1 penalties result in sparse factors

$L_1$ penalty

$L_2$ penalty
Sparse PCA

(D’Aspremont et al., 2007)
PCA

minimize $\|X - UV^T\|_F^2$
subject to $U^T U = V^T V = I$

NMF

minimize $\|X - UV^T\|_F^2$
subject to $U \geq 0, V \geq 0$

Sparse NMF

minimize $\|X - UV^T\|_F^2 + \lambda_u \sum_i \|u_i\|_1$
subject to $U \geq 0, V \geq 0$

Semi-NMF

minimize $\|X - UV^T\|_F^2$
subject to $U \geq 0$

Sparse semi-NMF

minimize $\|X - UV^T\|_F^2 + \lambda_u \sum_i \|u_i\|_1$
subject to $U \geq 0$

K-means

minimize $\|X - UV^T\|_F^2$
subject to $u_i \in \{e_k\}, \forall i$
Matrix decomposition can be interpreted probabilistically, via Bayes Rule:

\[
p(\text{model} \mid \text{data}) = \frac{p(\text{data} \mid \text{model}) p(\text{model})}{p(\text{data})}
\]
Matrix decomposition can be interpreted probabilistically, via Bayes Rule:

\[
\begin{aligned}
\text{posterior} \\
p(\text{model} \mid \text{data}) &= \frac{\text{likelihood} \times \text{prior}}{p(\text{data})} \\
p(\text{model} \mid \text{data}) &= \frac{p(\text{data} \mid \text{model}) \times p(\text{model})}{p(\text{data})}
\end{aligned}
\]
Matrix decomposition can be interpreted probabilistically, via Bayes Rule:

\[
p_{\text{posterior}}(\text{model} | \text{data}) = \frac{p(\text{data} | \text{model}) \cdot p(\text{model})}{p(\text{data})}
\]

\[-\ln p_{\text{posterior}}(\text{model} | \text{data}) \propto -\ln p(\text{data} | \text{model}) - \ln p(\text{model})\]
Matrix decomposition can be interpreted probabilistically, via Bayes Rule:

\[
p(model | data) = \frac{p(data | model) \cdot p(model)}{p(data)}
\]

\[-\ln p(model | data) \propto -\ln p(data | model) - \ln p(model)\]

**Bottom Line:** *Standard matrix decomposition can be viewed as maximum a posteriori estimation*

Loss functions often map onto the negative log-likelihood

Regularizers often map onto the prior distributions
Using the appropriate loss function can make a difference

http://alexhwilliams.info/itsneuronalblog/2016/03/27/pca/
## Combinatorial menu of models

<table>
<thead>
<tr>
<th>Loss functions</th>
<th>Regularizers/constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>quadratic</td>
<td>L2 norm</td>
</tr>
<tr>
<td>(real data)</td>
<td>(small factors)</td>
</tr>
<tr>
<td>absolute</td>
<td>L1 norm (sparsity)</td>
</tr>
<tr>
<td>(robust to outliers)</td>
<td>(sparse factors)</td>
</tr>
<tr>
<td>logistic</td>
<td>Nonnegative</td>
</tr>
<tr>
<td>(binary data)</td>
<td>(additive factors)</td>
</tr>
<tr>
<td>Poisson</td>
<td>Derivative penalties</td>
</tr>
<tr>
<td>(integer data)</td>
<td>(smooth factors)</td>
</tr>
<tr>
<td>circular</td>
<td></td>
</tr>
<tr>
<td>(angular data)</td>
<td></td>
</tr>
</tbody>
</table>
Further Reading


Presents one of the most general matrix factorization frameworks that includes PCA, NMF, Sparse PCA, K-means, and many others as special cases.


A comprehensive overview of applications and extensions of NMF.


A very comprehensive thesis placing greater focus on the algorithmic aspects of NMF. Also see more recent work from Gillis.
Talk Outline

1. Long list of matrix decomposition models

2. Optimization and model fitting

3. Visualization and model assessment
Properties of PCA

Rotation problem limits interpretability. However, it also allows us to organize factors to have convenient properties.

Canonically, choose factors to be orthogonal and order them by variance explained.
Properties of PCA

Rotation problem limits interpretability. However, it also allows us to organize factors to have convenient properties.

Canonically, choose factors to be orthogonal and order them by variance explained.

**Eckart-Young Theorem:** solution given by truncated singular value decomposition (SVD)

**Consequence:** the solution with $R$ components is contained in the solution with $R+1$ components.
Properties of PCA

PCA is one of the few examples of a nonconvex problem* that can be provably solved in polynomial time

* with a bit of work you can formulate a convex optimization problem whose solution also solves the PCA problem:
http://www.stat.cmu.edu/~ryantibs/convexopt/lectures/nonconvex.pdf
Properties of PCA

PCA is one of the few examples of a nonconvex problem* that can be provably solved in polynomial time.

Can prove that all local minima are solutions.

All non-optimal critical points are saddle points or maxima.

* with a bit of work you can formulate a convex optimization problem whose solution also solves the PCA problem:
http://www.stat.cmu.edu/~ryantibs/convexopt/lectures/nonconvex.pdf
Properties of PCA

PCA is one of the few examples of a nonconvex problem* that can be provably solved in polynomial time.

Can prove that all local minima are solutions.

All non-optimal critical points are saddle points or maxima.

*Baldi & Hornik, 1989.

* with a bit of work you can formulate a convex optimization problem whose solution also solves the PCA problem: [http://www.stat.cmu.edu/~ryantibs/convexopt/lectures/nonconvex.pdf](http://www.stat.cmu.edu/~ryantibs/convexopt/lectures/nonconvex.pdf)
Consider the PCA loss for a single matrix element $x_{ij} \approx u_i v_j$.
Consider the PCA loss for a single matrix element

\[ \ell_{ij}(u_i, v_j) = (x_{ij} - u_i v_j)^2 \]
Consider the PCA loss for a single matrix element

\[ \ell_{ij}(u_i, v_j) = (x_{ij} - u_i v_j)^2 \]
Consider the PCA loss for a single matrix element.

\[ \ell_{ij}(u_i, v_j) = (x_{ij} - u_i v_j)^2 \]

Convex in \( u \) when \( v \) is fixed as constant (and vice versa).
Alternating Minimization

\[
\begin{align*}
\text{minimize} & \quad \|X - UV^T\|_F^2 \\
\text{subject to} & \quad U^T U = V^T V = I
\end{align*}
\]

Decompose the loss function into two, easy to solve subproblems:

**step 1:** \[\hat{U} \leftarrow \text{argmin } \|X - \hat{U} V^T\|_F^2\]

**step 2:** \[\hat{V} \leftarrow \text{argmin } \|X - U \hat{V}^T\|_F^2\]

Repeat until loss function converges.
Fitting PCA in 10 lines of MATLAB

1 - \( K = 3; \) \% number of components
2 - \( \text{data} = \text{randn}(100,K) \times \text{randn}(K, 101); \)
3 - \([M, N] = \text{size(data)};\)
4 - \( U = \text{randn}(M, K); \) \% initial guess for \( U \)
5 -
6 - \( \text{for} \) iteration = 1:10
7 - \( \text{Vt} = U \backslash \text{data}; \) \% Update \( V \) (fixed \( U \))
8 - \( U = \text{data} / \text{Vt}; \) \% Update \( U \) (fixed \( V \))
9 - \( \text{loss(iteration)} = \text{norm(data} - U \times Vt, 'fro'); \)
10 - \( \text{end} \)
Alternating minimization is super effective in practice. Generally, not that many iterations are needed.

$\sigma^2 = 0.1$

$\sigma^2 = 1$

$\sigma^2 = 10$

Simulated 100x100 data matrix, with 10 components.
Alternating minimization is super effective in practice.

For moderate data sizes, iterations are fast.

Time to perform 1 update of $U$ and $V$ on my MacBook Pro.
NMF can also be solved by alternating minimization

Each step is nonnegative least squares problem

\[ U \leftarrow \text{argmin}_{\tilde{U}} \| X - \tilde{U} V^T \|_F^2 \quad \tilde{U} \geq 0 \]

\[ V \leftarrow \text{argmin}_{\tilde{V}} \| X - U \tilde{V}^T \|_F^2 \quad \tilde{V} \geq 0 \]
NMF can also be solved by alternating minimization

Each step is *nonnegative least squares* problem

Convex problem

Specialized, fast optimization methods

(e.g. Kim & Park, 2008)

\[
U \leftarrow \text{argmin} \| X - \tilde{U} V^T \|_F^2 \quad \tilde{U} \geq 0
\]

\[
V \leftarrow \text{argmin} \| X - U \tilde{V}^T \|_F^2 \quad \tilde{V} \geq 0
\]
NMF can also be solved by alternating minimization

Each step is *nonnegative least squares* problem

Convex problem

Specialized, fast optimization methods
  (e.g. Kim & Park, 2008)

In MATLAB:

```
x = lsqnonneg(A, b);
```

In Python:

```
import scipy.optimize
x = scipy.optimize.nnls(A, b)
```
Further reading on optimization


A unified review that covers alternating minimization along with other specialized methods for fitting NMF.


An overview of a very simple, but powerful class of optimization methods for matrix optimization. Udell et al. (2016), cited earlier, make use of these methods.
Talk Outline

1. Long list of matrix decomposition models
2. Optimization and model fitting
3. Visualization and model assessment
Scree Plot — How well am I fitting the data?

**Interpretation:** NMF converges to similar error from different initializations, and nearly achieves the optimal lower bound on performance set by SVD.
Similarity Plot — Are there multiple solutions that fit the data equally well?

Define the similarity of two factor matrices as:

\[ S(U, U') = \max_{\Pi} \frac{1}{r} \text{Tr} [U^T U' \Pi] \]

where \( \Pi \), is an \( r \times r \) permutation matrix.
Holding out data at random for cross-validation draws a connection to the well-studied matrix completion problem (see e.g. Candès & Recht, 2009)
Further reading on model assessment


The subtle concepts behind the similarity plot are much better studied for clustering algorithms (rather than NMF). This review covers that literature.


An in-depth look at cross-validation procedures for PCA and other matrix factorization approaches.