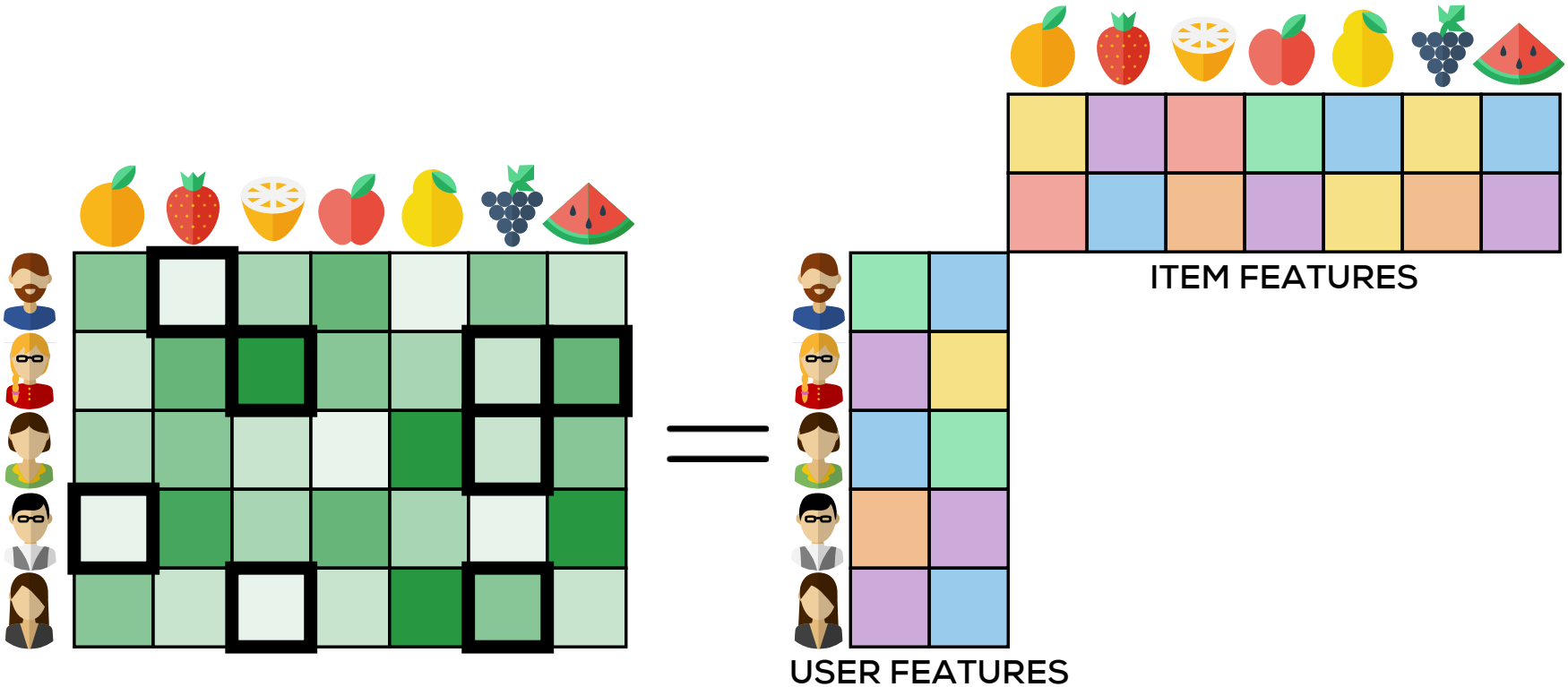
CPSC 8420 Advanced Machine Learning
Week 11: Data Completion

Dr. Kai Liu

October 27, 2020

Recommendation System



How does Amazon or other website recommend what you may like?

Low Rank Property for Data Recovery

Given a matrix $\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & ? \\ 3 & ? & ? \end{bmatrix}$, what's your guess for unobserved data?

Netflix Competition

NETFLIX

Netflix Prize

COMPLETED

Home | Rules | Leaderboard | Update

Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 **leaders.**

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11
Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in BigChaos				
13	xiangliang	0.8642	9.27	2009-07-15 14:53:22
14	Gravity	0.8643	9.26	2009-04-22 18:31:32
15	Ces	0.8651	9.18	2009-06-21 19:24:53
16	Invisible Ideas	0.8653	9.15	2009-07-15 15:53:04
17	Just a guy in a garage	0.8662	9.06	2009-05-24 10:02:54
18	J Dennis Su	0.8666	9.02	2009-03-07 17:16:17
19	Craig Carmichael	0.8666	9.02	2009-07-25 16:00:54
20	acmehill	0.8668	9.00	2009-03-21 16:20:50
Progress Prize 2007 - RMSE = 0.8723 - Winning Team: KorBell				
Cinematch score - RMSE = 0.9525				

There are currently 51051 contestants on 41305 teams from 186 different countries.
We have received 44014 valid submissions from 5169 different teams; 0 submissions in the last 24 hours.

Questions about interpreting the leaderboard? Please read this.

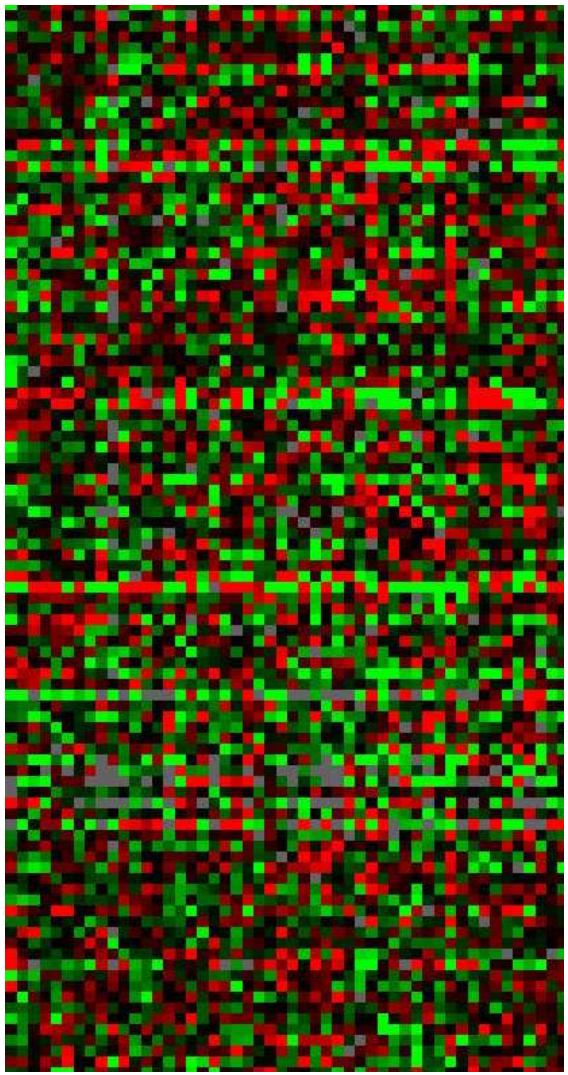
Data Completion

	movie I	movie II	movie III	movie IV	⋯
User A	1	?	5	4	⋯
User B	?	2	3	?	⋯
User C	4	1	2	?	⋯
User D	?	5	1	3	⋯
User E	1	2	?	?	⋯
⋮	⋮	⋮	⋮	⋮	⋱

- Training Data:
  $480K$ users, $18K$ movies,
  100 M ratings
  ratings 1-5
  (99 % ratings missing)
- Goal:
  \$ 1 M prize for 10 % reduction
  in RMSE over Cinematch
- BellKor's Pragmatic Chaos
  declared winners on 9/21/2009

  used ensemble of models, an
  important ingredient being
  low-rank factorization

Data Completion



- The rows are genes (variables)
- the columns are observations (samples, DNA arrays).
- Typical numbers are 6-10K genes, 50-150 samples.
- Refer to *eigengenes* and *eigenarrays*.
- Often 10–15% NAs.

SVD for Approximation

Given data Z, how to optimize rank-r matrix problems:

$$\min_{rank(M)=r} \|Z - M\|_F \tag{1}$$

What if Missing Data?

Assume we observe all entries of matrix Z indexed by the subset $\Omega \subset \{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\}$. How can we formulate the objective?

What if Missing Data?

Assume we observe all entries of matrix Z indexed by the subset $\Omega \subset \{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\}$. How can we formulate the objective?

$$\min rank(M) \quad s.t. \quad m_{ij} = z_{ij}, \quad ((i,j) \in \Omega) \tag{2}$$

The constraint is too strong, thus we can relax it as:

$$\min rank(M) \quad s.t. \sum_{(i,j)\in\Omega} (m_{ij} - z_{ij})^2 \leq \delta \tag{3}$$

or its equivalent from:

$$\min_{rank(M)\leq r} \sum_{(i,j)\in\Omega} (m_{ij} - z_{ij})^2 \tag{4}$$

Do we have some heuristic algorithm for the non-convex problem?

Funk MF

The original algorithm proposed by Simon Funk in his blog post factorized the user-item rating matrix as the product of two lower dimensional matrices, the first one has a row for each user, while the second has a column for each item.

$$M_{m \times n} = P^T_{m \times k} Q_{k \times n} \tag{5}$$

each element m_{ij} can be approximated by $q_j^T p_i$:

$$\sum_{i,j} (m_{ij} - q_j^T p_i)^2 \tag{6}$$

Maximum Margin Matrix Factorization

To avoid overfitting, we add regularization:

$$\underbrace{arg\ min}_{p_i, q_j} \sum_{i,j} (m_{ij} - q_j^T p_i)^2 + \lambda(||p_i||_2^2 + ||q_j||_2^2) \tag{7}$$

where we can utilize gradient descent method to optimize:

$$\begin{aligned} \frac{\partial J}{\partial p_i} &= -2(m_{ij} - q_j^T p_i)q_j + 2\lambda p_i \\ \frac{\partial J}{\partial q_j} &= -2(m_{ij} - q_j^T p_i)p_i + 2\lambda q_j \end{aligned} \tag{8}$$

therefore, we obtain the updating algorithm:

$$\begin{aligned} p_i &= p_i + \alpha((m_{ij} - q_j^T p_i)q_j - \lambda p_i) \\ q_j &= q_j + \alpha((m_{ij} - q_j^T p_i)p_i - \lambda q_j) \end{aligned} \tag{9}$$

Relaxation

Consider the objective function min $\sum_{(i,j)\in\Omega}(m_{ij}-z_{ij})^2+\lambda * rank(M)$ is NP-hard, we would like to relax it to be convex one:

$$\min \frac{1}{2}\sum_{(i,j)\in\Omega}(m_{ij}-z_{ij})^2+\lambda\|M\|_* \tag{10}$$

where $\|M\|_*$ is nuclear norm and defined as the sum of all singular value, which is the tightest convex relaxation of $rank(M)$.

Singular Value Thresholding (SVT)

Let (fully observed) $X_{n\times m}$ have SVD

$$X = U \cdot \mathrm{diag}[\sigma_1, \ldots, \sigma_m] \cdot V'$$

Consider the convex optimization problem

$$\underset{Z}{\text{minimize}} \quad \tfrac{1}{2}\|X - Z\|_F^2 + \lambda\|Z\|_*$$

Solution is *soft-thresholded SVD*

$$\mathbf{S}_\lambda(X) \;:=\; U \cdot \mathrm{diag}[(\sigma_1 - \lambda)_+, \ldots, (\sigma_m - \lambda)_+] \cdot V'$$

Like *lasso* for SVD: singular values are shrunk to zero, with many set to zero. Smooth version of best-rank approximation.

Soft-Impute

Recall $\min \frac{1}{2} \sum_{(i,j)\in\Omega} (m_{ij} - z_{ij})^2 + \lambda \|M\|_*$ and define:

$$[\mathcal{P}_\Omega(\mathbf{Z})]_{ij} = \begin{cases} z_{ij} & \text{if } (i,j) \in \Omega \\ 0 & \text{if } (i,j) \notin \Omega, \end{cases}$$

Given the singular value decomposition[3] $\mathbf{W} = \mathbf{UDV}^T$ of a rank-r matrix $\mathbf{W}$, we define its soft-thresholded version as

$$S_\lambda(\mathbf{W}) \equiv \mathbf{U}\mathbf{D}_\lambda\mathbf{V}^T \quad \text{where} \quad \mathbf{D}_\lambda = \text{diag}\left[(d_1 - \lambda)_+, \ldots, (d_r - \lambda)_+\right] \quad (7.13)$$

The following algorithm will recover the data:

Algorithm 7.1 SOFT-IMPUTE FOR MATRIX COMPLETION.

1. Initialize $\mathbf{Z}^{\text{old}} = \mathbf{0}$ and create a decreasing grid $\lambda_1 > \ldots > \lambda_K$.
2. For each $k = 1, \ldots, K$, set $\lambda = \lambda_k$ and iterate until convergence:
   Compute $\widehat{\mathbf{Z}}_\lambda \leftarrow \mathcal{S}_\lambda\big(P_\Omega(\mathbf{Z}) + P_\Omega^\perp(\mathbf{Z}^{\text{old}})\big)$.
   Update $\mathbf{Z}^{\text{old}} \leftarrow \widehat{\mathbf{Z}}_\lambda$
3. Output the sequence of solutions $\widehat{\mathbf{Z}}_{\lambda_1}, \ldots, \widehat{\mathbf{Z}}_{\lambda_K}$.

Hard-Impute

$$\underset{\text{rank}(Z)=r}{\text{minimize}} ||P_\Omega(X) - P_\Omega(Z)||_F$$

This is not convex in Z, but by analogy with SOFT-IMPUTE, an iterative algorithm gives good solutions.

Replace step:

(2a) Compute $Z^{\text{new}} \leftarrow \mathbf{S}_\lambda(P_\Omega(X) + P_\Omega^\perp(Z^{\text{old}}))$

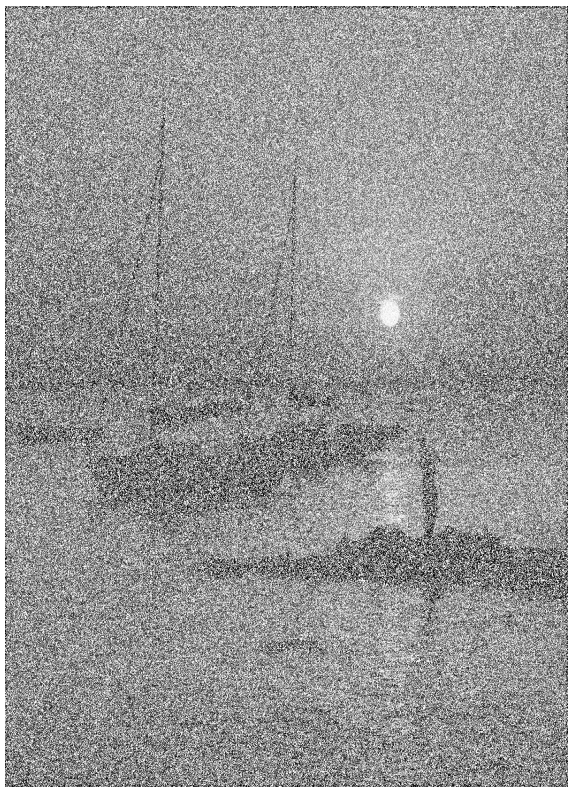
with

(2a') Compute $Z^{\text{new}} \leftarrow \mathbf{H}_r(P_\Omega(X) + P_\Omega^\perp(Z^{\text{old}}))$

Here $\mathbf{H}_r(X^*)$ is the best rank-r approximation to X^* — i.e. the rank-r truncated SVD approximation.

Experimental Results

(a) Original Image



(b) Noisy Image

(c) Soft-Impute

Tensor data Recovery

What about the data is of high dimension instead of 2D, say 3D?

$$\underset{\mathcal{T}\in\mathbb{R}^{n_1\times n_2\times n_3}}{\text{minimize}} \ \|\mathbf{y}-\mathrm{A}(\mathcal{T})\|)_2^2+\lambda\|\mathcal{T}\|_*.$$

where $\mathcal{T}$ is the data we want to recover and y is what we can observe. We can reformulate the objective as:

$$\underset{\{(\mathbf{u}_p,\mathbf{v}_p,\mathbf{w}_p)\}_{p=1}^{\tilde{r}}}{\text{minimize}} \ \|\mathbf{y}-\mathrm{A}(\sum_{p=1}^{\tilde{r}}\mathbf{u}_p\circ\mathbf{v}_p\circ\mathbf{w}_p)\|_2^2$$

$$+\lambda\sum_{p=1}^{\tilde{r}}\Big[(\|\mathbf{u}_p\|_2^2+\|\mathbf{v}_p\|_2^2\|\mathbf{w}_p\|_2^2)+(\|\mathbf{v}_p\|_2^2+\|\mathbf{u}_p\|_2^2\|\mathbf{w}_p\|_2^2)$$

$$+(\|\mathbf{w}_p\|_2^2+\|\mathbf{u}_p\|_2^2\|\mathbf{v}_p\|_2^2)\Big]$$

Algorithm

Data can be recovered by utilizing the following algorithm:

Algorithm 1 $\mathrm{AltMin}^{\lambda}$

Initialization: $k = 1$, λ, and $\mathbf{V}_0 \in \mathbb{R}^{n_2 \times r}, \mathbf{W}_0 \in \mathbb{R}^{n_3 \times r}$.
while stop criterion not meet **do**
  $\mathbf{U}_k = \arg\min_{\mathbf{U} \in \mathbb{R}^{n_1 \times r}} f(\mathbf{U}, \mathbf{V}_{k-1}, \mathbf{W}_{k-1})$;
  $\mathbf{V}_k = \arg\min_{\mathbf{V} \in \mathbb{R}^{n_2 \times r}} f(\mathbf{U}_k, \mathbf{V}, \mathbf{W}_{k-1})$;
  $\mathbf{W}_k = \arg\min_{\mathbf{W} \in \mathbb{R}^{n_3 \times r}} f(\mathbf{U}_k, \mathbf{V}_k, \mathbf{W})$;
  $k = k + 1$.
end while
Output: factorization $(\mathbf{U}^{\lambda}, \mathbf{V}^{\lambda}, \mathbf{W}^{\lambda})$.

Closed Solution

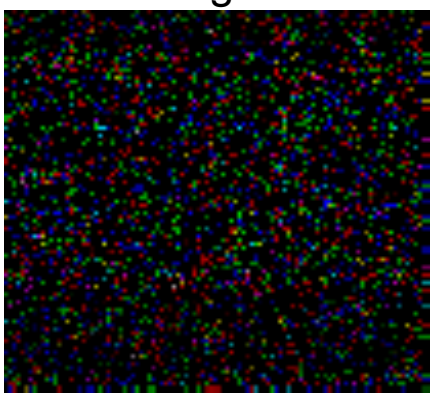
Where in each iteration we have exact closed solution as:

$$\begin{aligned}
\mathrm{vec}(\mathbf{U}_k) &= (\lambda \mathbf{I}_{n_1} \otimes \mathrm{diag}(\mathbf{1}_r + \mathrm{diag}(\mathbf{V}_{k-1}^\top \mathbf{V}_{k-1} + \mathbf{W}_{k-1}^\top \mathbf{W}_{k-1})) \\
&+ [(\mathbf{W}_{k-1} \circledast \mathbf{V}_{k-1}) \otimes \mathbf{I}_{n_1}]^\top \mathbf{A}^\top \mathbf{A} [(\mathbf{W}_{k-1} \circledast \mathbf{V}_{k-1}) \otimes \mathbf{I}_{n_1}])^{-1} \\
&\qquad [(\mathbf{W}_{k-1} \circledast \mathbf{V}_{k-1}) \otimes \mathbf{I}_{n_1}]^\top \mathbf{A}^\top \mathbf{y}; \\
\mathrm{vec}(\mathbf{V}_k) &= (\lambda \mathbf{I}_{n_2} \otimes \mathrm{diag}(\mathbf{1}_r + \mathrm{diag}(\mathbf{W}_{k-1}^\top \mathbf{W}_{k-1} + \mathbf{U}_k^\top \mathbf{U}_k)) \\
&+ [(\mathbf{W}_{k-1} \circledast \mathbf{U}_k) \otimes \mathbf{I}_{n_2}]^\top \mathbf{A}_{[2,1,3]}^\top \mathbf{A}_{[2,1,3]} [(\mathbf{W}_{k-1} \circledast \mathbf{U}_k) \otimes \mathbf{I}_{n_2}])^{-1} \\
&\qquad [(\mathbf{W}_{k-1} \circledast \mathbf{U}_k) \otimes \mathbf{I}_{n_2}]^\top \mathbf{A}_{[2,1,3]}^\top \mathbf{y}; \\
\mathrm{vec}(\mathbf{W}_k) &= (\lambda \mathbf{I}_{n_3} \otimes \mathrm{diag}(\mathbf{1}_r + \mathrm{diag}(\mathbf{V}_k^\top \mathbf{V}_k + \mathbf{U}_{k-1}^\top \mathbf{U}_{k-1})) \\
&+ [(\mathbf{U}_{k-1} \circledast \mathbf{V}_k) \otimes \mathbf{I}_{n_4}]^\top \mathbf{A}_{[3,2,1]}^\top \mathbf{A}_{[3,2,1]} [(\mathbf{U}_{k-1} \circledast \mathbf{V}_k) \otimes \mathbf{I}_{n_4}])^{-1} \\
&\qquad [(\mathbf{U}_{k-1} \circledast \mathbf{V}_k) \otimes \mathbf{I}_{n_3}]^\top \mathbf{A}_{[3,2,1]}^\top \mathbf{y}
\end{aligned}$$

Data Recovered

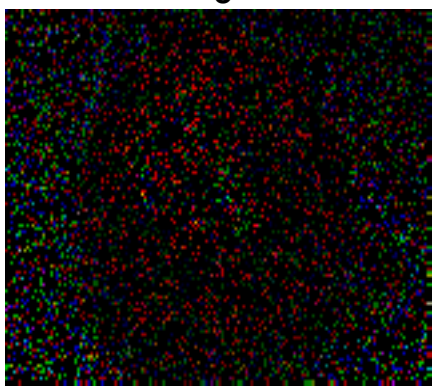
$\mathcal{T}^\star$

missing=90%

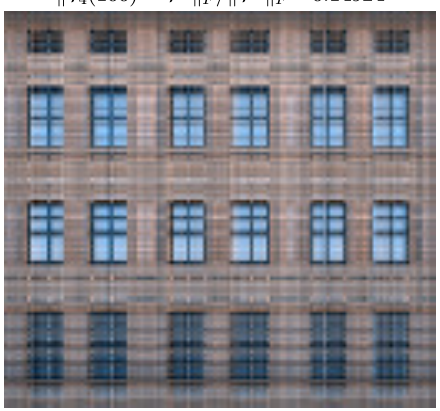


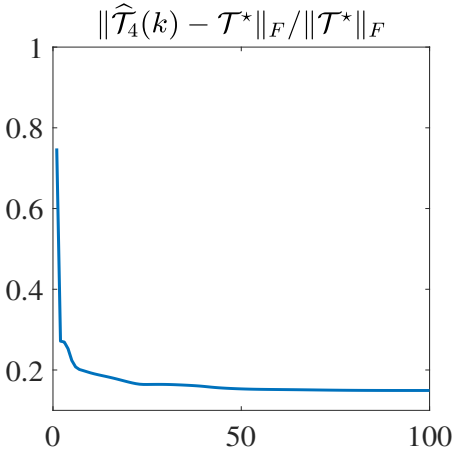
$\mathcal{T}^\star$

missing=90%

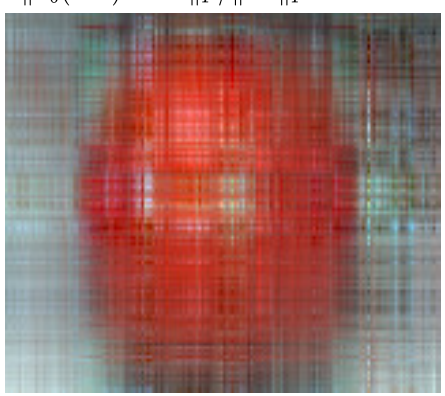


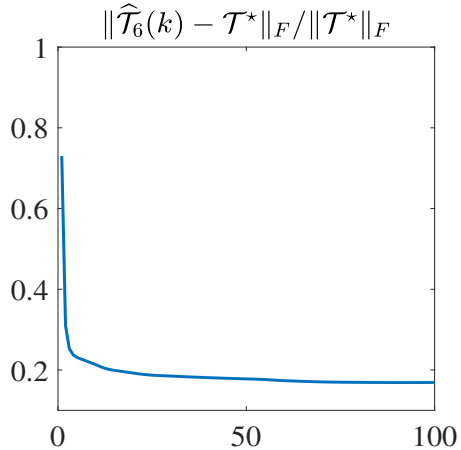
$\|\widehat{\mathcal{T}}_4(100) - \mathcal{T}^\star\|_F / \|\mathcal{T}^\star\|_F = 0.14924$





$\|\widehat{\mathcal{T}}_6(100) - \mathcal{T}^\star\|_F / \|\mathcal{T}^\star\|_F = 0.16912$





Pros & Cons

- The algorithm is unsupervised-based, which doesn't require any training datasets.
- Certain assumption is made that the data is of low-rank. If this is not true, the result won't be guaranteed.