Chapter 9

Matrix completion

Contents (class version)

9.0 Introduction

This chapter discusses the important problem of **matrix completion**, where we know some, but not all, elements of a matrix and want to "complete" the matrix by filling in the missing entries. Obviously this problem is **ill posed** in general because one could assign arbitrary values to the missing entries, unless one assumes some **model** for the matrix elements. The most common model is that the matrix is **low rank**.

A particularly famous application of **low-rank matrix completion** (**LRMC**) is the **Netflix problem**; this topic is also relevant to dynamic MR image reconstruction, and numerous other applications with missing data (incomplete observations).

Successful applications of LRMC include:

- Recommender systems (netflix, spotify, amazon, ...)
- Imaging: denoising, reconstruction in medical, hyper-spectral imaging.
- Anomaly detection in network flows
- Source localization and target tracking in radar and sonar
- Computer vision: background subtraction, object tracking, and to represent a single scene under varying illuminations
- Environmental monitoring of soil and crop conditions, water contamination, and air pollution, also sensor calibration
- Seismological activity and modal estimation in materials and human-made structures
- ...

9.1 Measurement model

If $\boldsymbol{X}$ is a **latent** $M \times N$ "low-rank" matrix with rank $r \leq \min(M, N)$, then

$$\boldsymbol{X} = \sum_{k=1}^{r} \sigma_k \boldsymbol{u}_k \boldsymbol{v}_k'.$$

Suppose we observe only *some* of the entries of $\boldsymbol{X}$ in a sampling set Ω, *i.e.*,

$$Y_{ij} = \begin{cases} \quad, & \\ \quad, & \text{otherwise,} \end{cases} \qquad \Omega \subset \{1, \dots, M\} \times \{1, \dots, N\},$$

where "?" denotes a value is "missing," *e.g.*, the ith user has not rated the jth movie.

Key questions:

- For a given (known) sampling set Ω, can we recover all of $\boldsymbol{X}$ from just the observed entries $\boldsymbol{Y}$? The answer will depend on the rank r, where typically $r \ll \min(M, N)$, and on the sampling set.
- How do we do it?
- How well does it work (*e.g.*, in the presence of noise or if $\boldsymbol{X}$ is not exactly low rank)?

Practical implementation

This topic is so important to modern data analysis problems that JULIA has its own data type, `Missing`, to represent missing values. Try this:

```
Y = [2 0; missing 4; 5 missing]
```

The type of `Y` is `3x2 Array{Union{Missing, Int64},2}`
because it is a 3×2 array of values that are either 64-bit integers or the special type `Missing`.

We cannot use any particular value like "0" to represent a missing value because 0 might be one of the observed values!

What is the minimum possible rank of $\boldsymbol{Y}$ in this example?

A: 0 B: 1 C: 2 D: 3 E: None of these. **??**

Sampling conditions for LRMC

The first question to ask is whether it should even be possible to recover $\boldsymbol{X}$ from $\boldsymbol{Y}$ in the noiseless case.

If $\boldsymbol{X}$ has rank r, then we can write $\boldsymbol{X}$ as the product of a $M \times r$ matrix (think $\tilde{\boldsymbol{U}}_r = \boldsymbol{U}_r \boldsymbol{\Sigma}_r$ for example) with the transpose of a $N \times r$ matrix $\tilde{\boldsymbol{V}}_r$. The number of **degrees of freedom** (**DoF**) in this representation looks to be $Mr + Nr = (M+N)r$. However, actually there are fewer DoF because once we fix $\tilde{\boldsymbol{U}}_r$ there are constraints on the values that $\tilde{\boldsymbol{V}}_r$ can take and still have $\tilde{\boldsymbol{U}}_r \tilde{\boldsymbol{V}}_r' = \boldsymbol{X}$. So $(M+N)r$ is an upper bound on the DoF. The total number of elements of $\boldsymbol{X}$ is MN, so if the matrix is low rank, then $(M+N)r \ll MN$ so it seems plausible that having around $(M+N)r$ samples might suffice.

A more careful analysis of the DoF is

$$\text{DoF} =$$

because

- the first r columns (of length M) are linearly independent,
- and the next $N-r$ columns depend linearly on those first r columns and we need r coefficients per column to describe the linear combination.

Note that if $r = N \leq M$ then $(M+N)r - r^2 = (M+N)N - N^2 = MN$, as expected.
When $M = N$ and $r \leq N/2$, a **lower bound on the number of samples** is $4Nr - 4r^2$ [1].

If $M \approx N$, then the DoF $\approx 2Nr$. So we need at least $O(Nr)$ samples to have any chance of recovery (unless we have additional information such as knowing that $\boldsymbol{U}_r$ and/or $\boldsymbol{V}_r$ is sparse).
Modern algorithms need $O(N\, r\, \textbf{polylog}(N))$ random samples under certain assumptions about $\boldsymbol{X}$.

Approximately how many observations *per column* are the minimum needed to recover a rank-r low-rank matrix of size $N \times N$?

A: r B: $2r$ C: r^2 D: N E: $2N$ **??**

Sampling mask

Define a $M \times N$ binary **sampling mask** matrix $\boldsymbol{M}$ in terms of the sampling set Ω by:

$$M_{ij} \triangleq \begin{cases} 1, & (i,j) \in \Omega, \\ 0, & \text{otherwise.} \end{cases} \tag{9.1}$$

In words, $\boldsymbol{M}$ is zero wherever we are missing a value in $\boldsymbol{Y}$.

Example. For the earlier 3×2 example we have `M = Int.(.~ismissing.(Y))` , *i.e.*, $\boldsymbol{M} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$

Hereafter, let $\boldsymbol{Y} \in \mathbb{F}^{M \times N}$ be an matrix where

$$Y_{ij} = \begin{cases} X_{ij}, & (i,j) \in \Omega \\ 0, & \text{otherwise.} \end{cases} \tag{9.2}$$

Now we *can* store a 0 in the locations of missing values because we have the separate mask $\boldsymbol{M}$ to keep track of which values are sampled and which are missing.

9.2 LRMC: noiseless case

Noiseless problem statement

The (noiseless) LRMC problem is to find a matrix $\hat{\boldsymbol{X}}$ that is **low-rank** and that agrees with the measurements $\boldsymbol{Y}$ on Ω, *i.e.*, we want

where $\odot$ denotes element-wise multiplication, *i.e.*, `M .* X` in JULIA and MATLAB, also known as the **Hadamard product** or **array multiplication** (as opposed to **matrix multiplication**) and is applicable to arrays of any dimension.

(In Python, `M * X` is the Hadamard product for `array` objects in `NumPy`, but is ordinary matrix multiplication for `matrix` objects.)

One "ideal" (noiseless) optimization formulation for the LRMC problem is [2]:

$$\hat{\boldsymbol{X}} = \arg\min_{\boldsymbol{X}} \tag{9.3}$$

This is a **non-convex** problem and is **NP hard** (impractical). No fast algorithm exists that is guaranteed to solve this problem formulation. Nevertheless, there are practical algorithms that often work reasonably well.

Alternating projection approach

An alternative to (9.3) is to pick a rank $1 \le K \ll \min(M,N)$ and seek a matrix $\hat{\boldsymbol{X}}$ that satisfies

$$\hat{\boldsymbol{X}} \in \quad \mathcal{C} \triangleq \left\{ \boldsymbol{X} \in \mathbb{F}^{M \times N} : \mathsf{rank}(\boldsymbol{X}) \le K \right\}, \quad \mathcal{D} \triangleq \left\{ \boldsymbol{X} \in \mathbb{F}^{M \times N} : \boldsymbol{M} \odot \boldsymbol{X} = \boldsymbol{M} \odot \boldsymbol{Y} \right\}.$$

In words, we seek a matrix $\hat{\boldsymbol{X}}$ having rank at most K and that agrees with the measurements $\boldsymbol{Y}$ on Ω.

The **projections onto convex sets** (**POCS**) approach is a classic signal processing tool for finding points in the intersection of two or more convex sets.

Which (if any) of the two sets $\mathcal{C}$ and $\mathcal{D}$ above is **convex**?
A: neither $\mathcal{C}$ nor $\mathcal{D}$ B: $\mathcal{C}$ but not $\mathcal{D}$ C: $\mathcal{D}$ but not $\mathcal{C}$ D: both $\mathcal{C}$ and $\mathcal{D}$ **??**

The **alternating projection** method alternates between projecting onto $\mathcal{C}$ and $\mathcal{D}$:

$$\boldsymbol{X}_{k+1} =$$

where $\mathcal{P}_{\mathcal{D}}(\boldsymbol{X})$ denotes the **projection** of its argument $\boldsymbol{X}$ onto the set $\mathcal{D}$, *i.e.*, the closest point in $\mathcal{D}$:

$$\mathcal{P}_{\mathcal{D}}(\boldsymbol{X}) = \arg\min_{\boldsymbol{Z} \in \mathcal{D}} \|\boldsymbol{X} - \boldsymbol{Z}\|_{\mathrm{F}}.$$

The algorithm designer gets to choose the norm for quantifying "closest." Here we are using matrices so we work with the simplest choice, the Frobenius norm.

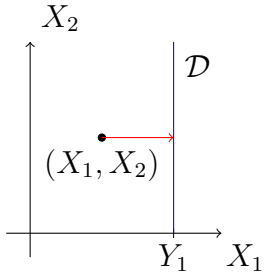
For the "data" set $\mathcal{D}$ above, the projection operation is very simple:

$$[\mathcal{P}_{\mathcal{D}}(\boldsymbol{X})]_{i,j} = \begin{cases} Y_{i,j}, & (i,j) \in \Omega \\ X_{i,j}, & \text{otherwise.} \end{cases}$$

Example.

Suppose $(M, N) = (2, 1)$ and $\boldsymbol{M} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ so $\boldsymbol{Y} = \begin{bmatrix} Y_1 \\ ? \end{bmatrix}$.

Then $\mathcal{D}$ is just $\{(X_1, X_2) \in \mathbb{R}^2 : X_1 = Y_1\}$.
For any point (X_1, X_2), the projection onto $\mathcal{D}$ is simply (Y_1, X_2).



If `M` is a logical array, then we can implement this operation "in place" in JULIA as follows:

```
X[M] .= Y[M]
```

Even though $\mathcal{C}$ is not convex, finding a projection onto $\mathcal{C}$ is fairly easy using Ch. 6:

$$\mathcal{P}_{\mathcal{C}}(\boldsymbol{Z}) = \underset{\boldsymbol{L} \,:\, \text{rank}(\boldsymbol{X}) \le K}{\arg\min} \|\boldsymbol{Z} - \boldsymbol{L}\|_{\text{F}} = \boldsymbol{U}_K \boldsymbol{\Sigma}_K \boldsymbol{V}_K', \quad \boldsymbol{Z} = \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{V}'.$$

This is simply the low-rank approximation problem of Ch. 6. It requires an SVD of the input argument.

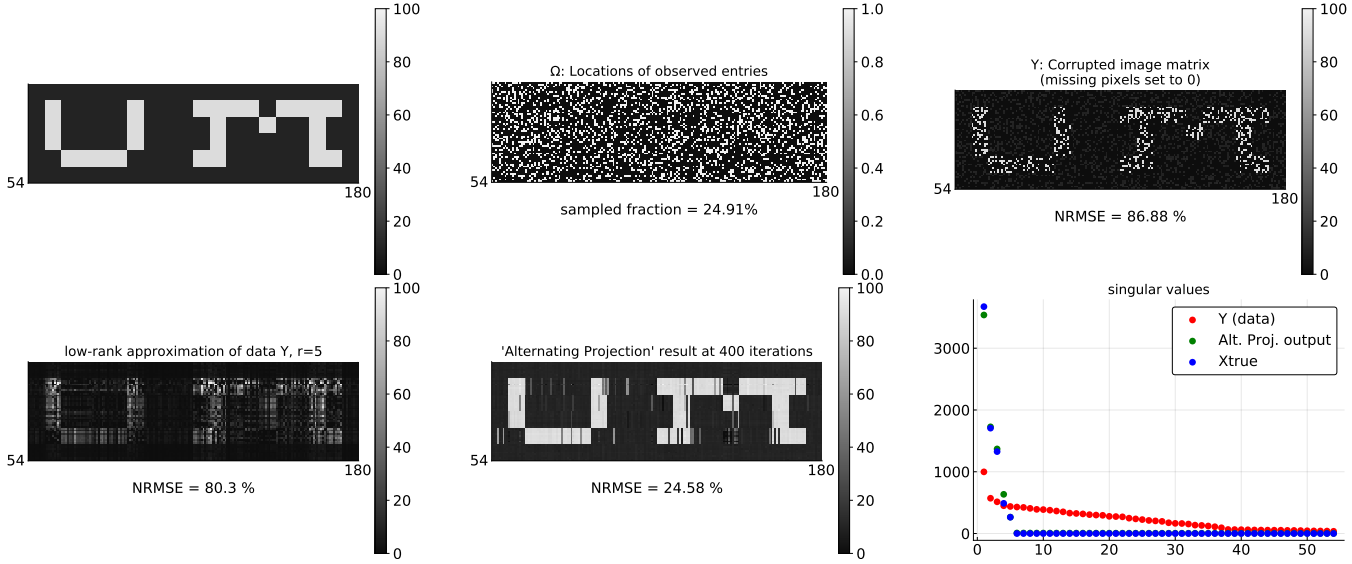
The projection $\mathcal{P}_{\mathcal{C}}(\boldsymbol{Z})$ is **unique** for any $M \times N$ input matrix $\boldsymbol{Z}$. (?)

A: True B: False **??**

Example. This demo shows matrix completion via alternating projection with 25% sampling.

https://web.eecs.umich.edu/~fessler/course/551/julia/demo/09_matcomp_altpro.html

https://web.eecs.umich.edu/~fessler/course/551/julia/demo/09_matcomp_altpro.ipynb



The alternating projection method works better than simply making a low-rank approximation to $\boldsymbol{Y}$, but it does not account for noise in the data, so we can do better.

Convergence of alternating projection method

Because the set of rank $\leq K$ matrices is nonconvex, it is hard to say much about the convergence of the alternating projection method. Choice of initial guess $\boldsymbol{X}_0$ can affect the convergence.

Example. Consider $\boldsymbol{Y} = \begin{bmatrix} 1 & 1 \\ ? & ? \end{bmatrix}$ with rank $K = 1$.

If $\boldsymbol{X}_0 = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$ then $\boldsymbol{X}_k = \boldsymbol{X}_0$ for all $k \in \mathbb{N}$.

If $\boldsymbol{X}_0 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ then $\boldsymbol{X}_k = \boldsymbol{X}_0$ for all $k \in \mathbb{N}$.

These two "limiting" solutions are both global optima.

Challenge. Find a more interesting case where one of the limits is a local optima and the other is a global optima.

Choice of rank

In practice one must choose the rank K to perform low-rank matrix completion.

One can use **cross validation** or **minimum description length** (**MDL**) methods [3] among others [4].

See survey papers [5] [6].

9.3 LRMC: noisy case

Noisy problem statement

In practice, often there is noise in the data, so the model (9.2) is often unrealistic. A more realistic model is

$$Y_{ij} = \begin{cases} X_{ij} + \varepsilon_{ij}, & (i,j) \in \Omega \\ 0, & \text{otherwise.} \end{cases}$$

Again, it is fine to store a 0 (or any other value) in the missing locations because we have the sampling mask $\boldsymbol{M}$ available.
In this case, it would be unreasonable to insist on exact data equality $\boldsymbol{M} \odot \boldsymbol{X} = \boldsymbol{M} \odot \boldsymbol{Y}$.

So now we want to find $\hat{\boldsymbol{X}}$ where $\boldsymbol{M} \odot \hat{\boldsymbol{X}} \approx \boldsymbol{M} \odot \boldsymbol{Y}$ and also $\hat{\boldsymbol{X}}$ is low-rank.

One possible LRMC formulation uses a (non-convex) **rank constraint**:

$$\hat{\boldsymbol{X}} = \qquad (9.4)$$

Another possible LRMC formulation uses a (non-convex) **rank regularizer**:

$$\hat{\boldsymbol{X}} = \qquad (9.5)$$

The following LRMC optimization formulation is the **convex relaxation** of the rank regularizer, using instead a **nuclear norm** regularizer:

$$\hat{\boldsymbol{X}} = \arg\min_{\boldsymbol{X}} \frac{1}{2} \| \boldsymbol{M} \odot (\boldsymbol{X} - \boldsymbol{Y}) \|_{\mathrm{F}}^2 + \qquad (9.6)$$

There are fast and practical algorithms for all three of these formulations, and there are convergence guarantees for the convex formulation.

Unfortunately, the optimization tools presented in Ch. 8 (GD, PGD, PSD) are inapplicable here because none of the cost functions above are differentiable! So we need something more than gradient-based methods.

Even the cost function that uses the (convex) nuclear norm $\|\boldsymbol{X}\|_*$ is not differentiable.

What is nuclear norm $\|x\|_*$ of a 1×1 "matrix" x ?

A: x B: $|x|$ C: $\mathbb{I}_{\{x \geq 0\}}$ D: $\max(0, x)$ E: x^2 **??**

Majorize-minimize (MM) iterations

To solve optimization problems like (9.6), we first consider **majorize-minimize** (**MM**) algorithms.

To solve an optimization problem like

$$\arg\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$$

we first design a **majorizer** or **surrogate function** $\phi_k(\boldsymbol{x})$ that satisfies the following two conditions:

$$f(\boldsymbol{x}_k) =$$
$$f(\boldsymbol{x}) \leq$$

Then the MM algorithm update is simply:

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x} \in \mathcal{X}} \phi_k(\boldsymbol{x}).$$

Any MM algorithm will monotonically decrease the cost function because

$$f(\boldsymbol{x}_{k+1}) \leq \phi_k(\boldsymbol{x}_{k+1}) \leq \phi_k(\boldsymbol{x}_k) = f(\boldsymbol{x}_k)\,.$$

Next we design a MM algorithm for LRMC problems like (9.6).

MM methods for LRMC

First define the complement of the mask matrix as follows:

$$\tilde{\boldsymbol{M}} \triangleq \boldsymbol{1}_M \boldsymbol{1}_N' - \boldsymbol{M}, \qquad \tilde{M}_{i,j} = \begin{cases} 0, & (i,j) \in \Omega \\ 1, & (i,j) \notin \Omega. \end{cases}$$

We focus now on the quadratic data-fit term in (9.4) (9.5) (9.6):

$$q(\boldsymbol{X}) \triangleq \| \boldsymbol{M} \odot (\boldsymbol{X} - \boldsymbol{Y}) \|_{\mathrm{F}}^2.$$

Now define the following function:

$$Q(\boldsymbol{X}; \boldsymbol{Z}) = \| \boldsymbol{X} - \boldsymbol{Z} + \boldsymbol{M} \odot (\boldsymbol{Z} - \boldsymbol{Y}) \|_{\mathrm{F}}^2 = \qquad (9.7)$$

This function is a **majorizer** of $q(\boldsymbol{X})$ because

$$q(\boldsymbol{X}) =$$

$$q(\boldsymbol{X}) \leq$$

Proof of majorization inequality: (Read)

$$\begin{aligned} Q(\boldsymbol{X}; \boldsymbol{Z}) &= \|\boldsymbol{X} - \boldsymbol{Z} + \boldsymbol{M} \odot (\boldsymbol{Z} - \boldsymbol{Y})\|_{\mathrm{F}}^2 = \left\|(\tilde{\boldsymbol{M}} + \boldsymbol{M}) \odot (\boldsymbol{X} - \boldsymbol{Z}) + \boldsymbol{M} \odot (\boldsymbol{Z} - \boldsymbol{Y})\right\|_{\mathrm{F}}^2 \\ &= \left\|\tilde{\boldsymbol{M}} \odot (\boldsymbol{X} - \boldsymbol{Z}) + \boldsymbol{M} \odot (\boldsymbol{X} - \boldsymbol{Y})\right\|_{\mathrm{F}}^2 = \left\|\tilde{\boldsymbol{M}} \odot (\boldsymbol{X} - \boldsymbol{Z})\right\|_{\mathrm{F}}^2 + \|\boldsymbol{M} \odot (\boldsymbol{X} - \boldsymbol{Y})\|_{\mathrm{F}}^2 \\ &\geq \|\boldsymbol{M} \odot (\boldsymbol{X} - \boldsymbol{Y})\|_{\mathrm{F}}^2 = q(\boldsymbol{X}), \end{aligned}$$

where the 4th equality holds because $\tilde{\boldsymbol{M}} \odot \boldsymbol{M} = \boldsymbol{0}$.

I designed the function (9.7) by making a 2nd-order Taylor expansion of $q(\boldsymbol{X})$ about $\boldsymbol{Z}$ and then using the fact that the elements of $\boldsymbol{M}$ are all 0 or 1. The above proof verifies that Q is a majorizer, which is all that is needed here. ♦♦

Now we use the majorizer (9.7) to develop a few different LRMC algorithms.

LRMC by iterative low-rank approximation

First consider LRMC using the rank constraint (9.4):

$$\hat{\boldsymbol{X}} = \underset{\boldsymbol{X} \,:\, \mathsf{rank}(\boldsymbol{X}) \le K}{\arg\min} \left\| \boldsymbol{M} \odot (\boldsymbol{X} - \boldsymbol{Y}) \right\|_{\mathrm{F}}^2.$$

The MM algorithm update for this formulation is simply

$$\begin{aligned}
\boldsymbol{X}_{k+1} &= \underset{\boldsymbol{X} \,:\, \mathsf{rank}(\boldsymbol{X}) \le K}{\arg\min} \; Q(\boldsymbol{X}; \boldsymbol{X}_k) \\
&= \underset{\boldsymbol{X} \,:\, \mathsf{rank}(\boldsymbol{X}) \le K}{\arg\min} \\
&= \underset{\boldsymbol{X} \,:\, \mathsf{rank}(\boldsymbol{X}) \le K}{\arg\min} \left\| \boldsymbol{X} - \tilde{\boldsymbol{X}}_k \right\|_{\mathrm{F}}^2, \quad \tilde{\boldsymbol{X}}_k \triangleq \tilde{\boldsymbol{M}} \odot \boldsymbol{X}_k + \boldsymbol{M} \odot \boldsymbol{Y} = \begin{cases} Y_{i,j}, & (i,j) \in \Omega \\ [\boldsymbol{X}_k]_{i,j}, & (i,j) \notin \Omega. \end{cases}
\end{aligned}$$

This algorithm alternates between two steps:

- Take the current guess $\boldsymbol{X}_k$ and replace all the values at sampled locations with the measurements from $\boldsymbol{Y}$ to get $\tilde{\boldsymbol{X}}_k$. (As mentioned earlier, this can be done "in place" using `X[M] .= Y[M]`)
- Perform low-rank (rank at most K) approximation (using **SVD**) to $\tilde{\boldsymbol{X}}_k$ to get the next iterate $\boldsymbol{X}_{k+1}$.

This process is exactly the same as the alternating projection method described earlier. So now we know that it is a MM method that decreases the Frobenius norm cost function monotonically.

(There is still no guarantee of convergence of $\{\boldsymbol{X}_k\}$ to a global minimizer because the rank constraint set is nonconvex.)

LRMC by iterative singular value hard thresholding

Now consider LRMC using the rank regularizer (9.5):

$$\hat{\boldsymbol{X}} = \underset{\boldsymbol{X} \in \mathbb{F}^{M \times N}}{\arg\min} \left\| \boldsymbol{M} \odot (\boldsymbol{X} - \boldsymbol{Y}) \right\|_{\mathrm{F}}^2 + \beta \, \mathsf{rank}(\boldsymbol{X}) \, .$$

The MM algorithm update for this formulation is simply

$$\begin{aligned}
\boldsymbol{X}_{k+1} &= \underset{\boldsymbol{X} \in \mathbb{F}^{M \times N}}{\arg\min} \, Q(\boldsymbol{X}; \boldsymbol{X}_k) + \\
&= \underset{\boldsymbol{X} \in \mathbb{F}^{M \times N}}{\arg\min} \left\| \boldsymbol{X} - (\tilde{\boldsymbol{M}} \odot \boldsymbol{X}_k + \boldsymbol{M} \odot \boldsymbol{Y}) \right\|_{\mathrm{F}}^2 + \beta \, \mathsf{rank}(\boldsymbol{X}) \\
&= \underset{\boldsymbol{X} \in \mathbb{F}^{M \times N}}{\arg\min} \left\| \boldsymbol{X} - \tilde{\boldsymbol{X}}_k \right\|_{\mathrm{F}}^2 + \beta \, \mathsf{rank}(\boldsymbol{X}) \, .
\end{aligned}$$

This algorithm alternates between two steps:

- Take the current guess $\boldsymbol{X}_k$ and replace all the values at sampled locations with the measurements from $\boldsymbol{Y}$ to get $\tilde{\boldsymbol{X}}_k$. (This can be done "in place.")
- Apply **singular value hard thresholding** to $\tilde{\boldsymbol{X}}_k$ to get the next iterate $\boldsymbol{X}_{k+1}$.

This MM method decreases the rank-regularized Frobenius norm cost function monotonically.

LRMC by iterative singular value soft thresholding

Now consider LRMC using the **convex nuclear norm** regularizer (9.6):

$$\hat{\boldsymbol{X}} = \arg\min_{\boldsymbol{X} \in \mathbb{F}^{M \times N}} \frac{1}{2} \|\boldsymbol{M} \odot (\boldsymbol{X} - \boldsymbol{Y})\|_{\mathrm{F}}^2 + \beta \|\boldsymbol{X}\|_*.$$

The MM algorithm update for this formulation is simply

$$\begin{aligned}
\boldsymbol{X}_{k+1} &= \arg\min_{\boldsymbol{X} \in \mathbb{F}^{M \times N}} \frac{1}{2} Q(\boldsymbol{X}; \boldsymbol{X}_k) + \beta \|\boldsymbol{X}\|_* \\
&= \arg\min_{\boldsymbol{X} \in \mathbb{F}^{M \times N}} \frac{1}{2} \left\| \boldsymbol{X} - (\tilde{\boldsymbol{M}} \odot \boldsymbol{X}_k + \boldsymbol{M} \odot \boldsymbol{Y}) \right\|_{\mathrm{F}}^2 + \beta \|\boldsymbol{X}\|_* \\
&= \arg\min_{\boldsymbol{X} \in \mathbb{F}^{M \times N}} \frac{1}{2} \left\| \boldsymbol{X} - \tilde{\boldsymbol{X}}_k \right\|_{\mathrm{F}}^2 + \beta \|\boldsymbol{X}\|_*.
\end{aligned}$$

This algorithm alternates between two steps:

- Take the current guess $\boldsymbol{X}_k$ and replace all the values at sampled locations with the measurements from $\boldsymbol{Y}$ to get $\tilde{\boldsymbol{X}}_k$. (This can be done "in place.")
- Apply **singular value soft thresholding** to $\tilde{\boldsymbol{X}}_k$ to get the next iterate $\boldsymbol{X}_{k+1}$.

This MM method decreases its cost function monotonically. Because the cost function is convex, with some additional work one can show that the sequence $\{\boldsymbol{X}_k\}$ converges to a global minimizer.

Iterative soft-thresholding algorithm (ISTA) ♦♦

In many modern applications, including LRMC, we have a **composite cost function** of the form:

$$\arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) + g(\boldsymbol{x}) \tag{9.8}$$

where $f(\boldsymbol{x})$ is convex and has a Lipschitz gradient (smooth) but $g(\boldsymbol{x})$ is convex but *not necessarily* smooth.

To develop an algorithm for such problems, we first reinterpret the GD step for minimizing $f(\boldsymbol{x})$ as follows:

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} q(\boldsymbol{x}; \boldsymbol{x}_k) = \boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k)$$

$$q(\boldsymbol{x}; \boldsymbol{x}_k) \triangleq f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x} - \boldsymbol{x}_k \rangle + \frac{1}{2\alpha} \|\boldsymbol{x} - \boldsymbol{x}_k\|_2^2 .$$

Proof (by completing the square):

$$\begin{aligned} q(\boldsymbol{x}; \boldsymbol{x}_k) &= f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x} - \boldsymbol{x}_k \rangle + \frac{1}{2\alpha} \|\boldsymbol{x} - \boldsymbol{x}_k\|_2^2 \\ &= f(\boldsymbol{x}_k) - \frac{\alpha}{2} \|\nabla f(\boldsymbol{x}_k)\|_2^2 + \frac{1}{2\alpha} \left(\|\alpha \nabla f(\boldsymbol{x}_k)\|_2^2 + 2 \langle \alpha \nabla f(\boldsymbol{x}_k), \boldsymbol{x} - \boldsymbol{x}_k \rangle + \|\boldsymbol{x} - \boldsymbol{x}_k\|_2^2 \right) \\ &= \left(f(\boldsymbol{x}_k) - \frac{\alpha}{2} \|\nabla f(\boldsymbol{x}_k)\|_2^2 \right) + \frac{1}{2\alpha} \|\boldsymbol{x} - (\boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k))\|_2^2 \\ \implies \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k). \end{aligned}$$

This alternate perspective on GD is the key to extending the GD approach to composite cost functions like (9.8). Consider the following iteration:

$$\begin{aligned}\boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \left\{ f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x} - \boldsymbol{x}_k \rangle + \frac{1}{2\alpha} \|\boldsymbol{x} - \boldsymbol{x}_k\|_2^2 + g(\boldsymbol{x}) \right\} \\ &= \arg\min_{\boldsymbol{x}} \left\{ \frac{1}{2\alpha} \|\boldsymbol{x} - (\boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k))\|_2^2 + g(\boldsymbol{x}) \right\}.\end{aligned}$$

This algorithm is called **iterative soft-thresholding algorithm** (**ISTA**), aka the **iterative shrinkage thresholding algorithm**, or more generally, the **proximal gradient method** (**PGM**).

More concisely, ISTA uses the following two steps per iteration:

$$\begin{aligned}\tilde{\boldsymbol{x}}_k &\triangleq \boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k) \text{ (usual GD step)} \\ \boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \frac{1}{2\alpha} \|\boldsymbol{x} - \tilde{\boldsymbol{x}}_k\|_2^2 + g(\boldsymbol{x}) = \operatorname{prox}_{\alpha g}(\tilde{\boldsymbol{x}}_k)\end{aligned} \tag{9.9}$$

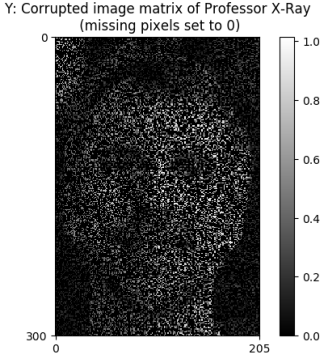
The first step is a conventional GD step. The second step is called a **proximity operation**.

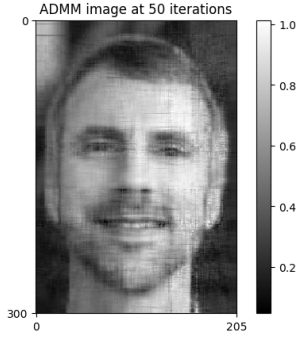
The method is convergent [7] (in the sense of p. 5.30 in Ch. 5) when the convex part $f(\boldsymbol{x})$ is smooth, *i.e.*, its gradient $\nabla f(\boldsymbol{x})$ has Lipschitz constant L, and we choose $0 < \alpha < 2/L$. We do not need $f(\boldsymbol{x})$ to be quadratic! Convex and smooth (Lipschitz gradient) is sufficient.

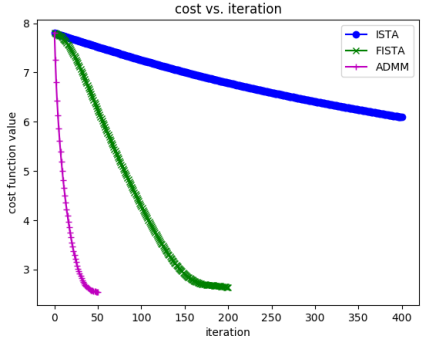
Demo

Recall that Nesterov's accelerated gradient method converges faster than GD for problems with smooth cost functions. There are also accelerated versions of ISTA including (**FISTA**) [8] and (**POGM**) [9].
An alternative approach to the LRMC optimization problem is the ADMM method [10]. ADMM requires a tuning parameter for fast convergence. When well tuned, ADMM can be faster than FISTA.
Historically, the original work on this problem used an **EM algorithm** perspective [11].

See the demo notebook created by Dr. Greg Ongie:
https://web.eecs.umich.edu/~fessler/course/551/julia/demo/09_lrmc_nuc.html
https://web.eecs.umich.edu/~fessler/course/551/julia/demo/09_lrmc_nuc.ipynb







9.4 Summary

Matrix completion using low-rank models is a rich area with numerous applications.

Low-rank matrix *approximation* is an easier problem and Ch. 6 provided nice SVD-based solutions.

LRMC is a more challenging problem (due to the missing data) so algorithms for LRMC typically are iterative.

Interestingly, LRMC algorithms usually use ingredients from low-rank matrix approximation. This pattern of using methods for simpler problems as part of some iterative approach for solving more complicate problem is quite common.

For image processing applications, often low-rank models are applied to collections of image patches, rather than to the entire image, *e.g.*, [12–16].

This topic is a very active research area, with a growing body of work related to convergence and performance guarantees, *e.g.*, [17] [18].

Bibliography

[1] Z. Xu. "The minimal measurement number for low-rank matrix recovery". In: *Applied and Computational Harmonic Analysis* 44.2 (Mar. 2018), 497–508 (cit. on p. 9.5).

[2] E. Candes and B. Recht. "Exact matrix completion via convex optimization". In: *Found. Comp. Math.* 9.6 (2009), 717–72 (cit. on p. 9.7).

[3] I. Ramirez and G. Sapiro. "An MDL framework for sparse coding and dictionary learning". In: *IEEE Trans. Sig. Proc.* 60.6 (June 2012), 2913–27 (cit. on p. 9.11).

[4] R. H. Keshavan and S. Oh. *A gradient descent algorithm on the Grassman manifold for matrix completion*. 2009 (cit. on p. 9.11).

[5] L. Balzano, Y. Chi, and Y. M. Lu. "Streaming PCA and subspace tracking: the missing data case". In: *Proc. IEEE* 106.8 (Aug. 2018), 1293–310 (cit. on p. 9.11).

[6] N. Vaswani, Y. Chi, and T. Bouwmans. "Rethinking PCA for modern data sets: theory, algorithms, and applications". In: *Proc. IEEE* 106.8 (Aug. 2018), 1274–6 (cit. on p. 9.11).

[7] K. Bredies and D. A. Lorenz. "Linear convergence of iterative soft-thresholding". In: *J. Fourier Anal. and Appl.* 14.5 (Dec. 2008), 813–37 (cit. on p. 9.21).

[8] A. Beck and M. Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM J. Imaging Sci.* 2.1 (2009), 183–202 (cit. on p. 9.22).

[9] A. B. Taylor, J. M. Hendrickx, and Francois Glineur. "Exact worst-case performance of first-order methods for composite convex optimization". In: *SIAM J. Optim.* 27.3 (Jan. 2017), 1283–313 (cit. on p. 9.22).

[10] J. Cai, E. Candes, and Z. Shen. "A singular value thresholding algorithm for matrix completion". In: *SIAM J. Optim.* 20.4 (2010), 1956–82 (cit. on p. 9.22).

[11] N. Srebro and T. Jaakkola. "Weighted low-rank approximations". In: *Proc. Intl. Conf. Mach. Learn*. 2003, 720–7 (cit. on p. 9.22).

[12] H. Ji, C. Liu, Z. Shen, and Y. Xu. "Robust video denoising using low rank matrix completion". In: *Proc. IEEE Conf. on Comp. Vision and Pattern Recognition*. 2010, 1791–8 (cit. on p. 9.23).

[13] J. D. Trzasko and A. Manduca. "Calibrationless parallel MRI using CLEAR". In: *Proc., IEEE Asilomar Conf. on Signals, Systems, and Comp.* 2011, 75–9 (cit. on p. 9.23).

[14] W. Dong, G. Shi, and X. Li. "Nonlocal image restoration with bilateral variance estimation: A low-rank approach". In: *IEEE Trans. Im. Proc.* 22.2 (Feb. 2013), 700–11 (cit. on p. 9.23).

[15] W. Dong, G. Shi, X. Li, Y. Ma, and F. Huang. "Compressive sensing via nonlocal low-rank regularization". In: *IEEE Trans. Im. Proc.* 23.8 (Aug. 2014), 3618–32 (cit. on p. 9.23).

[16] S. Ravishankar, B. E. Moore, R. R. Nadakuditi, and J. A. Fessler. "Low-rank and adaptive sparse signal (LASSI) models for highly accelerated dynamic imaging". In: *IEEE Trans. Med. Imag.* 36.5 (May 2017), 1116–28 (cit. on p. 9.23).

[17] Y. Cai and S. Li. "Convergence and stability of iteratively reweighted least squares for low-rank matrix recovery". In: *Inverse Prob. and Imaging* 11.4 (Aug. 2017), 643–61 (cit. on p. 9.23).

[18] R. Ge, C. Jin, and Y. Zheng. "No spurious local minima in nonconvex low rank problems: A unified geometric analysis". In: *Proc. Intl. Conf. Mach. Learn*. Vol. 70. 2017, 1233–42 (cit. on p. 9.23).