Lab - Dimension Reduction
Introduction to Matrix Completion

Ludovic Stephan

Due Feb 19, 2026

1 Introduction and dataset

1.1 The matrix completion problem

The Netflix Prize was a competition organized by the video streaming website Netflix that ran from Oct 2, 2006 to Sep 18, 2009. The company provided a training dataset consisting of user's ratings of movies, and the goal was to infer user ratings outside of this dataset.

Figure 1: An illustration of the Netflix Prize task. Stars correspond to ratings provided in the training set, while question marks denote the entries to be predicted.

Formally, we can represent the set of ratings as a matrix $Y \in \mathbb{R}^{n \times m}$, where n is the number of users and m the number of movies. An entry Y_{ij} is the rating given to movie j by user i. In this formalism, the training data consists in the set $(Y_{ij})_{(i,j)\in\mathcal{S}}$, where $\mathcal{S}$ is the set of (`user` $\times$ `movie`) ratings present in the training data. We organize this data in a matrix by defining $Y_{\mathcal{S}}$ as

$$Y^{\mathcal{S}}_{ij} = \begin{cases} Y_{ij} & \text{if } (i,j) \in \mathcal{S} \\ 0 & \text{otherwise} \end{cases}. \tag{1}$$

Note that this matrix can be computed using only the training data.

A structural assumption to solve this problem is often to assume that the underlying matrix Y has low rank. As such, the optimization problem we aim to solve is the following:

$$\hat{X}_r = \underset{\text{rank}(X)\leq r}{\arg\min} \|X^{\mathcal{S}} - Y^{\mathcal{S}}\|_2^2, \tag{2}$$

where $X^{\mathcal{S}}$ denotes the restriction of X to $\mathcal{S}$ as in Eq. (1).

1.2 Dataset

We will use the MovieLens dataset, available at the following url:

https://files.grouplens.org/datasets/movielens/ml-latest-small.zip

You will only use the `ratings.csv` file. It consists in 100836 rows and 4 columns called `userId, movieId, rating` and `timestamp`. Only the first three columns (`userId, movieId, rating`) will be useful. This data comes from 610 users, who rated around 10 movies each among 9724 movies.

Using the `pandas` library, you can import the dataset as follows:

```
import pandas as pd
data = pd.read_csv('mypath/ratings.csv')
```

Since the data matrix has a lot of zeros, you can use the *scipy* classes specialized in sparse matrices, `coo_matrix` and `csr_matrix`:

```
from scipy.sparse import coo_matrix, csr_matrix
```

1.3 Instructions

The main aim of this Lab is to implement and test some algorithms that aim to solve the minimization problem (2). You should produce a report (`.ipynb` or `.py` + `.pdf`) explaining your results, comments, conclusions and implementation.

You will implement the following algorithms:

- Singular Value Projection (SVP) algorithm [1]
- Singular Value Thresholding (SVT) algorithm [2]
- ADMiRA algorithm [3]

Their pseudocode is given in Section 2.

Preliminary questions. Before implementing the algorithms, answer the following questions in your report:

(i) How is this problem related to the course? To which of the sparse linear regression algorithms (matching pursuit, iterative hard thresholding, convex relaxation) can you compare each of the three completion algorithms?

(ii) How does the error measure $\|\hat{X}_r^{\mathcal{S}} - Y^{\mathcal{S}}\|_2^2$ behave as we increase the rank r? To what phenomenon is this related, and how can we prevent it?

(iii) Plot the singular values of the matrix $Y^{\mathcal{S}}$. What does the plot look like? What does this tell you about the rank of the matrix Y?

Implementation Implement the three algorithms of Section 2. In all algorithms, the error measure we choose is the relative error between $X^{\mathcal{S}}$ and $Y^{\mathcal{S}}$:

$$\mathrm{err}(X, Y) = \frac{\|X^{\mathcal{S}} - Y^{\mathcal{S}}\|_2}{\|Y^{\mathcal{S}}\|_2}.$$

The stopping criterion we use for all algorithms is the following: we stop the algorithm whenever either:

- $\mathrm{err}(X_t, Y) < \epsilon$ for some $\epsilon > 0$ (e.g. $\epsilon = 10^{-2}$ or 10^{-3}),
- $\mathrm{err}(X_{t+1}, Y) < \mathrm{err}(X_t, Y)$, i.e. the algorithm performance stops improving.

Compare the results (relative error and computation time) of the three algorithms above. Which one seems the best suited for movie rating prediction? Give potential limits on the methodology used. Don't hesitate to take a look at the referenced articles for details on the algorithms and/or their implementation. Keep in mind the answers to the preliminary questions while implementing the algorithms: they should give you precious insights!

The grade will be based on several factors:

- quality and correctness of the implementations; the code should non-be production-ready, but should be clear and well-structured,
- quality of the statistical methodology and reflections on the results,
- quality of the writing and presentation of the report.

Remark: depending on your implementation and optimization skills, some algorithms may run slowly on your computer. You can either choose to wait it out (some real-life data science tasks can take ours or days to run!), try and optimize your code further, or run the algorithms on a smaller subset of the data. If you choose the latter, please mention it in your report.

2 Algorithms for matrix completion

2.1 Singular Value Projection

Algorithm 1 Singular Value Projection (SVP) algorithm [1]

Input: $\mathcal{S}, Y^{\mathcal{S}}, r$

1: Initialize estimate: $X_0 = 0$
2: **while** (stopping criterion is not met) **do**
3: $\quad X_{t+1/2} \leftarrow X_t + (Y^{\mathcal{S}} - X_t^{\mathcal{S}})$
4: $\quad [U, \Sigma, V] \leftarrow \mathrm{SVD}_r(X_{t+1/2})$
5: $\quad X_{t+1} \leftarrow U\Sigma V^\top$
6: $\quad t \leftarrow t + 1$
7: **end while**
8: **return** X_t

Here, the $\mathrm{SVD}_r(M)$ operator computes the truncated SVD of M, keeping only its top r singular values.

2.2 Singular Value Thresholding (SVT) Algorithm

The function S_λ is the soft thresholding operator seen in the course. The notation $S_\lambda(\Sigma)$ means that S_λ is applied to each diagonal entry of Σ. The parameter λ should be carefully chosen. In both this algorithm and the one above, a learning rate η can be introduced (and optimized) in step 3 to try and improve the performance.

Algorithm 2 Singular Value Thresholding (SVT) algorithm [2]

Input: $\mathcal{S}, Y^{\mathcal{S}}$

1: Initialize estimate: $X_0 = 0$
2: **while** (stopping criterion is not met) **do**
3: $\quad X_{t+1/2} \leftarrow X_t + (Y^{\mathcal{S}} - X_t^{\mathcal{S}})$
4: $\quad [U, \Sigma, V] \leftarrow \mathrm{SVD}(X_{t+1/2})$
5: $\quad X_{t+1} \leftarrow U S_\lambda(\Sigma) V^\top$
6: $\quad t \leftarrow t + 1$
7: **end while**
8: **return** X_t

2.3 ADMiRA algorithm

Algorithm 3 ADMiRA algorithm [3]

Input: $\mathcal{S}, Y^{\mathcal{S}}, r$

1: Initialize estimates: $X_0 = 0, \Psi_0 = \emptyset$
2: **while** (stopping criterion is not met) **do**
3: $\quad [U, \_, V] \leftarrow \mathrm{SVD}_{2r}(Y^{\mathcal{S}} - X_t^{\mathcal{S}})$
4: $\quad \Psi_{t+1/2} \leftarrow \Psi_t \cup \left\{ u_j \, v_j^\top \, : \, j \leq 2r \right\}$
5: $\quad X_{t+1/2} \leftarrow \arg\min \left\| Y^{\mathcal{S}} - X^{\mathcal{S}} \right\|_2 : X \in \mathrm{span}(\Psi_{t+1/2})$
6: $\quad (U', \Sigma, V') \leftarrow \mathrm{SVD}_r(X_{t+1/2})$
7: $\quad \Psi_{t+1} \leftarrow \left\{ u'_j \, (v'_j)^\top \, : \, j \leq r \right\}$
8: $\quad X_{t+1} \leftarrow U' \Sigma (V')^\top$
9: $\quad t \leftarrow t + 1$
10: **end while**
11: **return** X_t

The goal of ADMiRA is to write the low-rank matrix X_t as

$$X_t = \sum_{M \in \Psi_t} a_M M,$$

where the set Ψ_t is a set of rank-one matrices, or *atoms*. The resulting matrix X_t is always of rank less than $\mathrm{Card}(\Psi_t)$.

The rank-one matrices M are obtained from the (truncated) SVDs of X and $Y^{\mathcal{S}} - X^{\mathcal{S}}$ (steps 3-4 and 6-7). Step 5 optimizes the coefficients a_M to minimize the error $\|Y^{\mathcal{S}} - X^{\mathcal{S}}\|$; this step can be viewed as a least-squares linear regression in dimension n^2.

References

[1] Raghu Meka, Prateek Jain, and Inderjit S. Dhillon. Guaranteed rank minimization via singular value projection. 2009. URL https://arxiv.org/abs/0909.5457.

[2] Jian-Feng Cai, Emmanuel J. Candes, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. 2008. URL https://arxiv.org/abs/0810.3286.

[3] Kiryung Lee and Yoram Bresler. Admira: Atomic decomposition for minimum rank approximation. 2009. URL https://arxiv.org/abs/0905.0044.