Lab - Dimension Reduction
Introduction to Matrix Completion

Ludovic Stephan

Due February 15th, 2025

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:

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

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.

Try to write *readable* code: it should not be production-grade, but I should be able to verify its correctness.

Don't hesitate to take a look at the referenced articles for details on the algorithms and/or their implementation.

2 Rank estimation for matrix completion

A popular rank estimation for matrix completion is as follows. We compute the singular values of $Y^{\mathcal{S}}$ incrementally until there is no more significant gap between singular values: the gap between the r-th and $(r+1)$-th singular value should be small compared to the first ones.

Use this procedure to determine empirically the rank of the matrix `users` $\times$ `movies`.

3 Algorithms for matrix completion

For each of the three algorithms below, implement and test them on the `users` $\times$ `movies` dataset. In all cases, the parameter r should be set to the value you chose in Section 2.

The error measure we choose is the relative error between $X^{\mathcal{S}}$ and $Y^{\mathcal{S}}$:

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

For each algorithm, you will plot the relative error at each step, as well as report the final error obtained. 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.

3.1 Singular Value Projection

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

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

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

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

3.2 Convex relaxation: Singular Value Thresholding (SVT) Algorithm

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: $\quad X_{t+1/2} \leftarrow X_n + (Y^{\mathcal{S}} - X^{\mathcal{S}})$
4: $\quad [U, \Sigma, V] \leftarrow \text{SVD}(X_{t+1/2})$
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

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.

3.3 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: $\quad [U, \_, V] \leftarrow \text{SVD}_{2r}(Y^{\mathcal{S}} - X^{\mathcal{S}})$
4: $\quad \Psi_{t+1/2} \leftarrow \Psi_t \cup \left\{ u_j \, v_j^\top \ : \ j \leq 2r \right\}$
5: $\quad X_{t+1/2} \leftarrow \arg\min \|Y^{\mathcal{S}} - X^{\mathcal{S}}\|_2 : X \in \text{vect}(\Psi_{t+1/2})$
6: $\quad (U', \Sigma, V') \leftarrow \text{SVD}_r(X_{t+1/2})$
7: $\quad X_{t+1} \leftarrow U' \Sigma (V')^\top$
8: $\quad t \leftarrow t + 1$
9: **end while**
10: **return** X_t

The set Ψ_t is a set of rank-one matrices, or *atoms*. Note that the minimization problem in step 5 can actually be rephrased as a least-squares regression problem.

4 Conclusion

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.

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.