LAB : Advanced Smart Sensing - Matrix completion

Adrien Saumard

Due to February 6th, 2024

The main aim of this LAB is to implement and test the algorithms which enable rank minimization of a matrix in the context of matrix completion. You should produce a report explaining your results, comments, conclusions and implementation.

1 Dataset

Texte

Download the movieLens dataset :

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

You will work solely with the file `rating.csv`.

By using python / ipython, and given the fact that `rating.csv` is under `mypath`, you could easily load the dataset as it follows :

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

The data are represented by 100836 rows and 4 columns `called userId, movieId, rating` et `timestamp`. We will work only with the 3 first columns which are `userId, movieId, rating`. These data are made by 610 users rating more or less 10 movies each among 9724 movies.

TIPS : In order to work with the plain matrix movies$\times$users which is made of lots of zeros, you could use the sparse matrix representations with functions coo_matrix or csr_matrix :

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

2 Minimal rank choice of a matrix

Determine graphically and empirically the rank r of the matrix movies$\times$users as it follows :

For the matrix completion problem, in the first step of our SVP method, we compute singular values incrementally till we find no more significant gap between singular values : the gap between the rth and $r+1$th singular value should be small compared to the first ones.

3 Implementation and application

Implement and test on the dataset the 3 approaches proposed below :

3.1 Singular Value Projection algorithm

Implement the SVP algorithm described below :

Singular Value Projection (SVP)

Require: $\mathbf{y}$, $\mathcal{M}$, r

Initialize estimate: $\mathbf{X}_0 = \mathbf{0}$

while (some stopping criterion is met) **do**

  $\mathbf{X}_{n+1/2} = \mathbf{X}_n + \mathcal{M}^*(\mathbf{y} - \mathcal{M}(\mathbf{X}_n))$

  $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] = SVD(\mathbf{X}_{n+1/2})$

  $\mathbf{X}_{n+1} = \mathbf{U}\mathrm{diag}(H_r(\mathrm{diag}(\mathbf{\Sigma})))\mathbf{V}^T$

end while

return $\mathbf{X}_n$

The notations are the same as the ones you saw in class. In particular, the operator H_r is the hard thresholding operator, that The stopping criterion is the RMSE (Root Mean Square Error) defined as :

$$RMSE(y, \hat{y}) = \sqrt{\frac{\sum_{i,j \in \mathcal{S}} \|y_{ij} - \hat{y}_{ij}\|^2}{\text{card}(\mathcal{S})}}$$

with y_{ij} the rating of *movie* i by *user* j, $\hat{y}_{ij}$ the fitted rating and $\mathcal{S}$ the set of known ratings among the matrix movies$\times$users. The algorithm stops under one or the other condition :

- when the decrease of the RMSE is no longer observed between 2 consecutive iterations i.e. $RMSE^t - RMSE^{t+1} < 0$.
- when the decrease of the RMSE between 2 consecutive iterations is below a given threshold i.e. $RMSE^t - RMSE^{t+1} < \varepsilon$. We could state $\varepsilon = 0.01$.

You will apply your implementation to matrix movies$\times$users. You will plot the RMSE at each iteration of the algorithm and give the best RMSE obtained.

3.2 Convex relaxation algorithm

You will implement the following optimization algorithm, known as Singular Value Thresholding :

Singular Value Thresholding (SVT)

Require: $\mathbf{y}$, $\mathcal{M}$, r

Initialize estimate: $\mathbf{X}_0 = \mathbf{0}$

while (some stopping criterion is met) **do**

  $\mathbf{X}_{n+1/2} = \mathbf{X}_n + \mathcal{M}^*(\mathbf{y} - \mathcal{M}(\mathbf{X}_n))$

  $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] = SVD(\mathbf{X}_{n+1/2})$

  $\mathbf{X}_{n+1} = \mathbf{U}\mathrm{diag}(S_\lambda(\mathrm{diag}(\mathbf{\Sigma})))\mathbf{V}^T$

end while

return $\mathbf{X}_n$

The notations are the usual ones used in the course. In particular, the operator S_λ is applied at each entry of the vector $\mathrm{diag}(\Sigma)$ and gives the soft-thresholded value at level λ.

Note that the parameter λ should be carefully chosen. Note also that a gradient step size can be put in step 3, in the aim to improve the convergence of the algorithm.

Use your implementation to the matrix movies$\times$users and give the RMSE obtained.

3.3 ADMiRA algorithm

Implement ADMIRA algorithm by following the steps described below :

Admira pseudo-code
Require: y, $\mathcal{M}$, r
1: Initial estimate: $X_0 = 0,\ \Psi = \emptyset$
2: **while** (stopping criterion is met) **do:**
3: $\quad \Psi' \leftarrow \arg\max_\Psi \left\|\mathcal{P}_\Psi \mathcal{M}^*(y - \mathcal{M}\hat{X})\right\|_F : \lvert\Psi\rvert < 2r$
4: $\quad \tilde{\Psi} \leftarrow \Psi' \cup \Psi$
5: $\quad \tilde{X} \leftarrow \arg\min_X \lVert y - \mathcal{M}X)\rVert_2 : X \in \mathrm{span}(\tilde{\Psi})$
6: $\quad \hat{\Psi} \leftarrow \arg\max_\Psi \left\|\mathcal{P}_\Psi \hat{X}\right\|_F : \lvert\Psi\rvert \leq r$
7: $\quad \hat{X} \leftarrow \mathcal{P}_{\hat{\Psi}} \tilde{X}$
8: **end while**
9: **return** $\hat{X}$

The choice of the parameter r will be the same as the ones defined in question 2. The notations are the same than in the course notes.

The algorithm stops under one or the other condition :

- when the decrease of $\left\|y - \mathcal{M}\hat{X}\right\|_2 / \lVert y\rVert_2$ is no longer observed
- $\left\|y - \mathcal{M}\hat{X}\right\|_2 / \lVert y\rVert_2 < \varepsilon$ with $\varepsilon = 0.01$

Use your implementation to the matrix movies$\times$users. You will plot the RMSE at each iteration of the algorithm and give the best RMSE obtained.

4 Conclusion

Compare the results (RMSE, computational time) with the 3 approaches. Conclude about the best approach to keep in your example and give potential limits on the methodologies.