Stat 580 (Spring 2017): Final project

Due: May 2, 2017

1 Get started

In this project, we will focus on matrix completion, an increasingly popular topic in statistics and machine learning. In particular, we will study some computational aspects of a popular matrix completion algorithm called Soft-Impute. The first task is to get yourself familiar with this topic. Read

- Chaper 7.1 to 7.3 of Hastie *et al.* (2015)
- Mazumder *et al.* (2010)

and write a one-page summary of matrix completion. (Feel free to check other references by yourself. Note that some references are very theoretical, but we only focus on the computational aspects of matrix completion.)

2 Various SVD methods/implementations

1. Implement Soft-Impute (Algorithm 7.1. of Hastie *et al.* (2015)) in C via LAPACK. You have to describe briefly how to use your code (e.g. the format of the input data file.)
2. Implement Soft-Impute in R. There is an R package softimpute that performs the algorithm Soft-Impute. You are not allowed to use it. Then compare various singular value decomposition (SVD) algorithms / implementations:
   (a) R internal svd function
   (b) svd package (`https://cran.r-project.org/web/packages/svd/index.html`) (includes more than one methods)
   (c) RcppArmadillo (i.e. write an Rcpp function that utilizes C++ library Armadillo to perform SVD)
   (d) irlba package (`https://cran.r-project.org/web/packages/irlba/index.html`)

   You have to design your own simulation experiments to compare their performances. Report your findings.
3. (Bonus) Newly emerged randomized method could be used to form approximated SVD (e.g., Halko *et al.*, 2011). For Soft-Impute, we can focus on a variation that produces singular value thresholding (i.e. $\mathcal{S}_\lambda$ in Algorithm 7.1. of Hastie *et al.* (2015)) approximately. Read Oh *et al.* (2015) and implement their method to compute $\mathcal{S}_\lambda$ using RcppArmadillo. Design experiments to compare the speed of this method with the above methods. Report your findings.
4. (Bonus) When some observed entries are outliers, Soft-Impute (or the underlying estimator) could be severely affected. A robustification idea is disucssed in Wong and Lee (2016) and the robust modification of Soft-Impute is provided in Algorithm 2 of Wong and Lee (2016). Describe the idea (what is Huber function and why it helps?) behind such method and implement Algorithm 2 of Wong and Lee (2016) in C. (No numerical experiment is needed.)

3 Application

In the following two tasks, you have to apply Soft-Impute to perform matrix completions.

1. The Lena image is depicted in Figure 3. Such an image has been used extensively in the image processing literature. In Blackboard, a version of such image is uploaded. It consists of 256×256 pixels. Your task is to first choose 40% of the pixels randomly as missing pixels, and then reconstruct the image using the remaining 60% pixels. You only have to choose *one* implementation in Question 2 of Section 2, to perform this task. Here these 60% pixels are regarded as your data. To select the tuning parameter λ, you have to design a grid of λ (i.e. $\lambda_1, \ldots, \lambda_K$), and separate the data (these 60% pixels) further into two subsets: training set (70%) and validation set (30%). Use the validation set to perform hold-out validation for the selection of λ from the grid.

Figure 1: The Lena image.

Present both the image with only non-missing entries and the reconstructed image. Report the rank of the reconstructed image. Further, you could compare the reconstructed pixels and the original pixels at the missing locations.

2. The MovieLens dataset can be obtained from

   `http://grouplens.org/datasets/movielens/`

   Apply Soft-Impute with *various* implementations of SVD, described in Question 2 of Section 2, to the MovieLens 100K Dataset. Split the dataset randomly into three subsets: training set (70%), validation set (15%) and test set (15%). Use the validation set to perform hold-out validation for the selection of λ. You have to design a grid of λ (i.e., $\lambda_1, \dots, \lambda_K$). Compare the speed and performance (based on the test set) of Soft-Impute using various implementations of SVD.

4 (Bonus) Accelerating Soft-Impute?

Soft-Impute is a proximal gradient (PG) method. There exists an accelerated version of PG method (Algorithm 1). Compare this algorithm with Soft-Impute.

Algorithm 1 Accelerated PG version of Soft-Impute

```
1: Initialize $\mathbf{Z}^{(0)} = \mathbf{0}$ and create a decreasing grid $\lambda_1 > \cdots > \lambda_K$.
2: for each $k = 1, \dots, K$ do
3:     set $\lambda = \lambda_k$, $t_1 = 1$ and $\mathbf{A} = \mathbf{Z}^{(0)}$.
4:     repeat for $j = 1, 2, \dots,$ compute
5:         $\mathbf{Z}^{(j)} \leftarrow \mathcal{S}_\lambda \left( P_\Omega(\mathbf{Z}) + P_\Omega^\perp(\mathbf{A}) \right)$
6:         $t_{j+1} \leftarrow \left( 1 + \sqrt{1 + 4t_j^2} \right) / 2$
7:         $\mathbf{A} \leftarrow \mathbf{Z}^{(j)} + \{(t_j - 1)/t_{j+1}\}(\mathbf{Z}^{(j)} - \mathbf{Z}^{(j-1)})$
8:     until convergence
9:     Set $\widehat{\mathbf{Z}}_{\lambda_k} = \mathbf{Z}^{(j)}$.
10:    Set $\mathbf{Z}^{(0)} = \widehat{\mathbf{Z}}_{\lambda_k}$.
11: end for
12: Output the sequence of solution $\widehat{\mathbf{Z}}_{\lambda_1}, \dots, \widehat{\mathbf{Z}}_{\lambda_K}$.
```

References

Halko, N., Martinsson, P.-G. and Tropp, J. A. (2011) Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, **53**, 217–288.

Hastie, T., Tibshirani, R. and Wainwright, M. (2015) *Statistical Learning with Sparsity: The Lasso and Generalizations*. CRC Press.

Mazumder, R., Hastie, T. and Tibshirani, R. (2010) Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, **11**, 2287–2322.

Oh, T.-H., Matsushita, Y., Tai, Y.-W. and So Kweon, I. (2015) Fast randomized singular value thresholding for nuclear norm minimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4484–4493.

Wong, R. K. W. and Lee, T. C. M. (2016) Matrix completion with noisy entries and outliers. URL `http://arxiv.org/abs/1503.00214`. Submitted.