

Matlab Project on Least Squares and More

Recovery of Low-Rank Matrix Data From Heavily Contaminated Observations

1 The Problem

We are given a matrix $M \in \mathbb{R}^{n \times n}$ that is supposed to be an observed version of a rank- r matrix with $r \ll n$. Unfortunately, many, but not all, elements of M are simply wrong (erroneous observations or contaminations). Can we still recover the rank- r matrix using the poor-quality observation matrix M ?

Essentially, we want to do the separation

$$M = XY + Z$$

where $X \in \mathbb{R}^{n \times r}$, $Y \in \mathbb{R}^{r \times n}$ and $Z \in \mathbb{R}^{n \times n}$. Hopefully, the product XY has rank r and is our “true” data, and Z is somewhat “sparse” (having many zero elements), representing the contaminations present in the observation matrix M . After such a separation we may throw away Z and retain XY .

To accomplish the separation, we consider solving the following optimization model:

$$\min_{X,Y,Z} \sum_{i=1}^n \sum_{j=1}^n \left(\frac{1}{2} (XY + Z - M)_{ij}^2 + \mu |Z_{ij}| \right), \quad (1)$$

for some balancing parameter $\mu > 0$. The first term enforces the closeness between the given data M and our model $XY + Z$. The second term encourages Z to be sparse. Using the so-called Frobenius norm for a matrix (i.e., the square root of the sum of squares), we can rewrite

$$\min_{X,Y,Z} \frac{1}{2} \|XY + Z - M\|_F^2 + \mu \sum_{i=1}^n \sum_{j=1}^n |Z_{ij}|. \quad (2)$$

For convenience we will denote the function in (2) by $f(X, Y, Z)$, i.e.,

$$f(X, Y, Z) := \frac{1}{2} \|XY + Z - M\|_F^2 + \mu \|\text{vec}(Z)\|_1,$$

where $\|\cdot\|_F$ is the matrix Frobenius norm (in Matlab `norm(S, 'fro')` for matrix S), $\|\cdot\|_1$ is the vector 1-norm (in Matlab `norm(v, 1)` for vector v), and $\text{vec}(Z)$ is the vector formed by stacking the columns of Z (in Matlab `Z(:)`).

2 The Algorithm

The minimization problem is nonlinear and rather difficult. What we will do is to divide and conquer — an alternating minimization approach. That is, we will minimize with respect to one variable, X , Y or Z , at a time while fixing the other two, and do many many cycles. More specifically, we will have a loop of cycles where inside the loop a single cycle consists of updates the 3 matrix variables X, Y, Z one by one:

1. Update X : $\min_X \|XY - (M - Z)\|_F^2 \implies X$,

2. Update Y : $\min_Y \|XY - (M - Z)\|_F^2 \implies Y$,
3. Update Z : $\min_Z f(X, Y, Z) \implies Z$,

where each minimization is done with respect to one variable while fixing the other two (for example, in the last step, X and Y are fixed and the new Z is the minimizer of f with respect to Z only). A key observation is that the first two problems above becomes linear least squares problems that we know how to solve.

2.1 Formula for Updating X

You should derive this formula by yourself.

2.2 Formula for Updating Y

Let us write

$$Y := [y_1 \ y_2 \ \cdots \ y_n] \quad \text{and} \quad M - Z = B := [b_1 \ b_2 \ \cdots \ b_n]$$

where y_j and b_j are columns of Y and B , respectively. Since

$$XY - B = X[y_1 \ y_2 \ \cdots \ y_n] - [b_1 \ b_2 \ \cdots \ b_n] = [Xy_1 - b_1 \ Xy_2 - b_2 \ \cdots \ Xy_n - b_n],$$

it can be seen that

$$\|XY - (M - Z)\|_F^2 = \|XY - B\|_F^2 = \sum_{j=1}^n \|Xy_j - b_j\|^2.$$

Hence the problem for updating Y can be written as

$$\min_{y_1, \dots, y_n} \sum_{j=1}^n \|Xy_j - b_j\|^2,$$

which can be further broken into n separate linear least squares problems of more familiar form

$$\min_{y_j} \|Xy_j - b_j\|^2, \quad j = 1, 2, \dots, n. \tag{3}$$

All these problems have the same fixed coefficient matrix X but different right-hand sides b_j 's and different variables y_j 's. In Matlab we have the solution for the j -th column of Y

$$Y(:, j) = X \setminus B(:, j);$$

Or collectively and more efficiently,

$$Y = X \setminus (M - Z);$$

recalling that $B = M - Z$. You may use a more sophisticated formula if you believe that it is faster.

2.3 Formula for Updating Z

When we fix X and Y and minimize with respect to Z , we see from (1) that we can actually minimize with respect to each element Z_{ij} separately so that each minimization sub-problem has the identical form of

$$\min_{t \in \mathbb{R}} \frac{1}{2}(t-a)^2 + \mu|t|. \quad (4)$$

where t can stand for any element of Z , and a for the corresponding element of $A := M - XY$.

A formula for the minimizer of the above function can be derived by considering the two cases: $t \geq 0$ and $t \leq 0$, separately. By doing so, we can get rid of the absolute value $|\cdot|$ and do differentiation. From Calculus, a differentiable function must attain its minimum, if unrestricted, at a point where its derivative vanishes. However, when restricted to an interval (that is our case), a minimum may occur at an endpoint where the derivative may not be zero.

A close examination of the two cases ($t \geq 0$ and $t \leq 0$) gives us a closed-form formula for the minimizer t^* , that is,

$$t^* = \text{sign}(a) \max(0, |a| - \mu). \quad (5)$$

Applying the above formula simultaneously to all Z_{ij} and a_{ij} , we have the following collective Matlab formulas:

```
A = M - X*Y;
Z = sign(A) .* max(0, abs(A) - mu);
```

3 Program Specification

Write a Matlab function for solving problem (2) that implements the alternating minimization scheme.

```
function [X, Y, Z, iter] = MYcode(M, r, mu, tol)
% Output:
%       X, Y, Z -- computed solutions
%       iter -- number of iterations taken
% Input:
%       M -- Observed data
%       r -- rank of true data
%       mu -- balancing parameter
%       tol -- tolerance for stopping
```

You can initialize the matrices X and Y as random matrices, and set Z to the zeros matrix.

Stop the loop of cycles when the relative reduction in the function value of $f(X, Y, Z)$ in (2) falls below the given tolerance, i.e.

$$|f_p - f_c| \leq \text{tol} * f_p \quad (6)$$

where f_c is the function value evaluated at the current iteration while f_p is the function value at the previous iteration. Download the test script and the instructor's code and run the test script.

4 Requirements

Write and typeset a project report (not hand-written) that includes:

1. an introduction section to describe the problem in your own words and by your understanding;
2. a section containing a derivation of your formula for updating X (that should be almost as simple as that for Y), and a derivation for the formula (5) that solves (4);
3. a section describing your implementation and important programming details if any;
4. a results section to describe the results you obtain, including a discussion on the performance of your code in comparison to the instructor's code;
5. a conclusion section.

The report should be no more than 3 or 4 pages. Also attach a copy of your code and copies of pictures produced by running the downloaded test script which will call your function `MYcode.m`. (See the course website for more detailed information if any.)

5 Further Information

5.1 Continuation

The algorithm described above is not very stable when the parameter μ is small. To stabilize it, we consider a strategy of continuation: starting from a larger value μ_1 and gradually reducing it to the prescribed final value μ through a sequence μ_1, μ_2, \dots, μ . This would introduce a gradual and smooth transition from the “easy μ_1 ” to the “hard μ ”. For each μ_k value, we do a number of iterations as described above. Essentially, we add an outer loop that wraps around our original inner loop. In the outer loop, μ_k value decreasing for $k = 1, 2, \dots$ until it reached its final value μ .

In continuation, when μ_k is reduced, we always start iterations from the existing approximate solutions. Do not reset X, Y, Z because they are already increasingly good approximations.

Exact details of implementing this continuation strategy can be determined through experiments. For example, we may reduce μ_k by a factor of 2 (cut into half every time) so that $\mu_{k+1} = 0.5\mu_k$ and do a number of times. If we decide to do it 10 times, then $\mu_1 = 2^9\mu$.

5.2 Useful Matlab Commands

The following Matlab commands may be useful in this projects (if you choose to use them). Use the Matlab help facility to learn more about them.

`size, randn, zeros, for, break, norm, fprintf, sign, max, abs`