IRLS Algorithm in Matrix Completion

Enkhzaya Enkhtaivan: enkhtaivan@wisc.edu
Yifan Hong: yhong84@wisc.edu
Xiaosheng Liu: xliu878@wisc.edu

Abstract

As a cost-effective alternative, various low-rank matrix completion (LRMC) algorithms have been proposed over the years. In class we learned ISVT algorithm that requires the rank of original matrix. In this activity, we'll discuss the Iteratively Reweighted Least Squares(IRLS) minimization technique that do not require the rank information.

Since the rank minimization problem is NP-hard, it is computationally intractable when the dimension of a matrix is large. In order to avoid computational issue, we can replace the non-convex objective function with its convex surrogate. One way of doing that is shrinking all singular values equally to approximate the matrix rank, which is called nuclear norm minimization (NNM). One computationally efficient way to solve it is IRLS minimization technique.

After completing this activity, students will be able to 1) understand the basic concept of NNM and IRLS; 2) use IRLS minimization technique to solve NNM problem; 3) understand the matrix completion problem and its application in high-level.

1 Introduction

Simply put, the rank minimization problem is a task to find a matrix of minimum rank satisfying a set of linear constraints e.g,:

$$\text{minimize rank}(X)$$
$$\mathcal{L}(X) = M$$

with $\mathcal{L}$ is a linear operator encoding the linear constraints and M is the observed data. In general, this problem is non-convex and has doubly-exponential running time both in theory and practice [3]; however, when both the matrix we seek to recover and the constraints are simple enough, one can solve the problem reliably fast. One such example is the task of matrix completion, which can be applied to collaborative filtering problems such as the Netflix problem. The reason why the unknown matrix is considered to have a low rank is because an individual user's ratings is typically a combination of only a few factors such as genre, running time, inclusion of favorite actors/actresses etc.

In this paper, we explore what is called the Nuclear Norm Minimization (NNM) - a convex relaxation of the NP-hard problem of rank minimization. The iterative solver we employ in order to solve a NNM problem is called Iteratively Reweighted Least Squares (IRLS). Finally, we will see a demonstration of this method on a problem of Matrix Completion, which we solved in class by using Iterative Singular Value Thresholding and compare their performances.

Here, we briefly note the advantages of using IRLS. First, this method does not require knowing the matrix of the matrix we wish to recover in contrast to ISVT. Furthermore, we solve a reweighted least squares problem at each iteration of the algorithm, which enables us to use various least squares problem solvers, such as Gauss-Newton, Gradient Descent etc.

On the other hand, the same fact that we use a least squares problem at each iteration might cause longer running time and sometimes unstability when condition numbers of the matrices involved become too high. These issues are fixed by method of stabilizing, which will be described in the section 1.2.

1.1 Formulation of the problem and mathematical foundation.

Given a matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ with only partially observed data $\Omega \subseteq \mathbb{R}$, we want to find its missing entries. In class, we learned about several different matrix norms such as the operator norm:

$$\begin{aligned} \|\boldsymbol{X}\| &= \sup\{\|\boldsymbol{X}\boldsymbol{w}\| : \boldsymbol{w} \in \mathbb{R}^n \text{ with } \|\boldsymbol{w}\| = 1\} \\ &= \sup\left\{\frac{\|\boldsymbol{X}\boldsymbol{w}\|}{\|\boldsymbol{w}\|} : \boldsymbol{w} \in \mathbb{R}^n \text{ with } \boldsymbol{w} \neq 0\right\}. \end{aligned}$$

where $\|\cdot\|$ is any vector norm on $\mathbb{R}^n$, and the entrywise p- norm:

$$\|\boldsymbol{X}\|_p = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} |x_{ij}|^p\right)^{\frac{1}{p}}.$$

In this paper, the norm of particular interest is the Schatten p - norm:

$$\|\boldsymbol{X}\|_{S_p} = \left(\sum_{i=1}^{r} \sigma_i^p(\boldsymbol{X})\right)^{\frac{1}{p}}$$

where $\sigma_1(\boldsymbol{X}) \geq \sigma_2(\boldsymbol{X}) \geq \ldots \sigma_r(\boldsymbol{X}) \geq 0$ are the singular values of the rank r matrix $\boldsymbol{X}$. We have seen this when $p = 2$:

$$\|\boldsymbol{X}\|_{S_2} = \left(\sum_{i=1}^{r} \sigma_i^2(\boldsymbol{X})\right)^{\frac{1}{2}} = \|\boldsymbol{X}\|_F,$$

the Frobenius norm. However, our main algorithm makes use of what is called the nuclear norm:

$$\|\boldsymbol{X}\|_* := \|\boldsymbol{X}\|_{S_1} = \sum_{i=1}^{r} \sigma_i(\boldsymbol{X}).$$

A property of the nuclear norm, or the Schatten p - norm in general, that enables us to solve the NP - hard problem of rank minimization is that the *smooth Schatten - p function*:

$$f_p(\boldsymbol{X}) = \left(\sum_{i=1}^{r} \sigma_i(\boldsymbol{X})^p + \gamma\right)^{\frac{p}{2}} = \text{Tr}(\boldsymbol{X}^T\boldsymbol{X} + \gamma\boldsymbol{I})^{\frac{p}{2}},$$

is convex for $p \geq 1$ and $\gamma \geq 0$, [2]. In particular, we deduce that the nuclear norm $f_1(\boldsymbol{X}) = \|\boldsymbol{X}\|_*$ with $p = 1$ and $\gamma = 0$ is convex.

Finally, under suitable conditions on the matrix $\boldsymbol{X}$, the solution to the convex relaxation:

$$\begin{aligned} &\text{minimize } \|\boldsymbol{X}\|_* \\ &\text{subject to } \mathcal{L}_\Omega(\boldsymbol{X}) = \boldsymbol{M} \end{aligned} \tag{1}$$

$$\begin{aligned} &\text{minimize } \operatorname{rank}(\boldsymbol{X}) \\ &\text{subject to } \mathcal{L}_\Omega(\boldsymbol{X}) = \boldsymbol{M} \end{aligned} \tag{2}$$

coincide. A sufficient condition for when this happens is called the Restricted Isometry Property and we refer the readers to [3] and [1] for the rigorous details.

1.2 IRLS Algorithms

In the general affine rank minimization problem, we are optimizing subject to the following constraint:

$$\mathcal{L}(\boldsymbol{X}) = \boldsymbol{M}$$

where $\boldsymbol{X} \in \mathbb{R}^{m\times n}$, $\boldsymbol{M}$ is either a matrix or a vector data and $\mathcal{L}$ is a linear operator on $\mathbb{R}^{m\times n}$. For the special case of matrix completion problem, the linear operator is simply:

$$(\mathcal{L}_\Omega(\boldsymbol{X}))_{ij} = \begin{cases} \boldsymbol{X}_{ij} \text{ if } (i,j) \in \Omega \\ 0 \text{ otherwise} \end{cases}$$

with Ω is the set of given entries of $\boldsymbol{X}$. We assume that the given entries of the matrix are being sampled from some random distribution.There are several implementations and variations of the IRLS algorithm, and let us first provide simple the IRLS - M the algorithm below, as implemented in [1]:

Algorithm 1 IRLS - M

```
1: Input: a constant $K \geq \operatorname{rank} \boldsymbol{X}$, a scaling paramter $\gamma > 0$
2: Initialize: $\boldsymbol{W}_0 = \boldsymbol{I} \in \mathbb{R}^{m \times n}$, iteration counter $k = 0$
3: while not converged do
4:     $\boldsymbol{X}_k = \underset{\mathcal{L}_\Omega(\boldsymbol{X})=M}{\operatorname{argmin}} \left\| (\boldsymbol{W}_{k-1})^{\frac{1}{2}} \boldsymbol{X} \right\|_F^2$
5:     $\varepsilon_k = \min \{\varepsilon_{k-1}, \gamma \sigma_{K+1}(\boldsymbol{X}_k)\}$
6:     $[\boldsymbol{U}_k, \boldsymbol{S}_k^2, \boldsymbol{U}_k^T] = \operatorname{svd}(\boldsymbol{X}_k \boldsymbol{X}_k^T)$
7:     $\boldsymbol{W}_k = \boldsymbol{U}_k (\boldsymbol{S}_{k,\varepsilon_k})^{-1} \boldsymbol{U}_k^T.$
8:     $k = k + 1;$
9: end while
```

Let us elaborate a bit on the algorithm above. Here we are solving a weighted least-squares problem at each iteration to obtain the k-th guess $\boldsymbol{X}^k$ and subsequently the k-th weight $\boldsymbol{W}^k$. The appearance of the Frobenius norm is explained by following simple observation:

$$\|\boldsymbol{X}\|_* = \operatorname{Tr}[(\boldsymbol{X}\boldsymbol{X}^T)^{-\frac{1}{2}}(\boldsymbol{X}\boldsymbol{X}^T)] = \left\| \boldsymbol{W}^{\frac{1}{2}} \boldsymbol{X} \right\|_F^2,$$

where $\boldsymbol{W} = (\boldsymbol{X}\boldsymbol{X}^T)^{-\frac{1}{2}}$. The square root of a matrix is uniquely defined when the matrix is PSD (positive semi-definite) and we indeed have a symmetric, PSD matrix since $\boldsymbol{X}\boldsymbol{X}^T$ is symmetric and PSD for any matrix $\boldsymbol{X}$.

About the stabilization parameters ε_k, the line 4 in the algorithm ensures that it is decreasing so our algorithm does not go on indefinitely. On the other hand, we define $\boldsymbol{W}_k$ in terms of the ε_k - stabilization of $\boldsymbol{X}_k\boldsymbol{X}_k^T$ in order to ensure that all of our matrices used in the operations are not ill-conditioned. That is, if some of the singular values of a matrix $\boldsymbol{Y}$ are too small, we bump up the singular values that are smaller than the threshold by doing the following:

$$\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T \implies \boldsymbol{Y}_\varepsilon = \boldsymbol{U}\boldsymbol{S}_\varepsilon\boldsymbol{V}^T,$$

where $\boldsymbol{S}_\varepsilon = \operatorname{diag}(\max\{\varepsilon, \sigma_j\})$ with σ_j's being the singular values of $\boldsymbol{Y}$ or the entries of the diagonal matrix $\boldsymbol{S}$. Finally, the line 5 uses the fact that given the Singular Value Decomposition:

$$\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T,$$

the matrix $\boldsymbol{Y}\boldsymbol{Y}^T$ has the Singular Value Decomposition:

$$\boldsymbol{Y}\boldsymbol{Y}^T = \boldsymbol{U}\boldsymbol{S}^2\boldsymbol{U}^T,$$

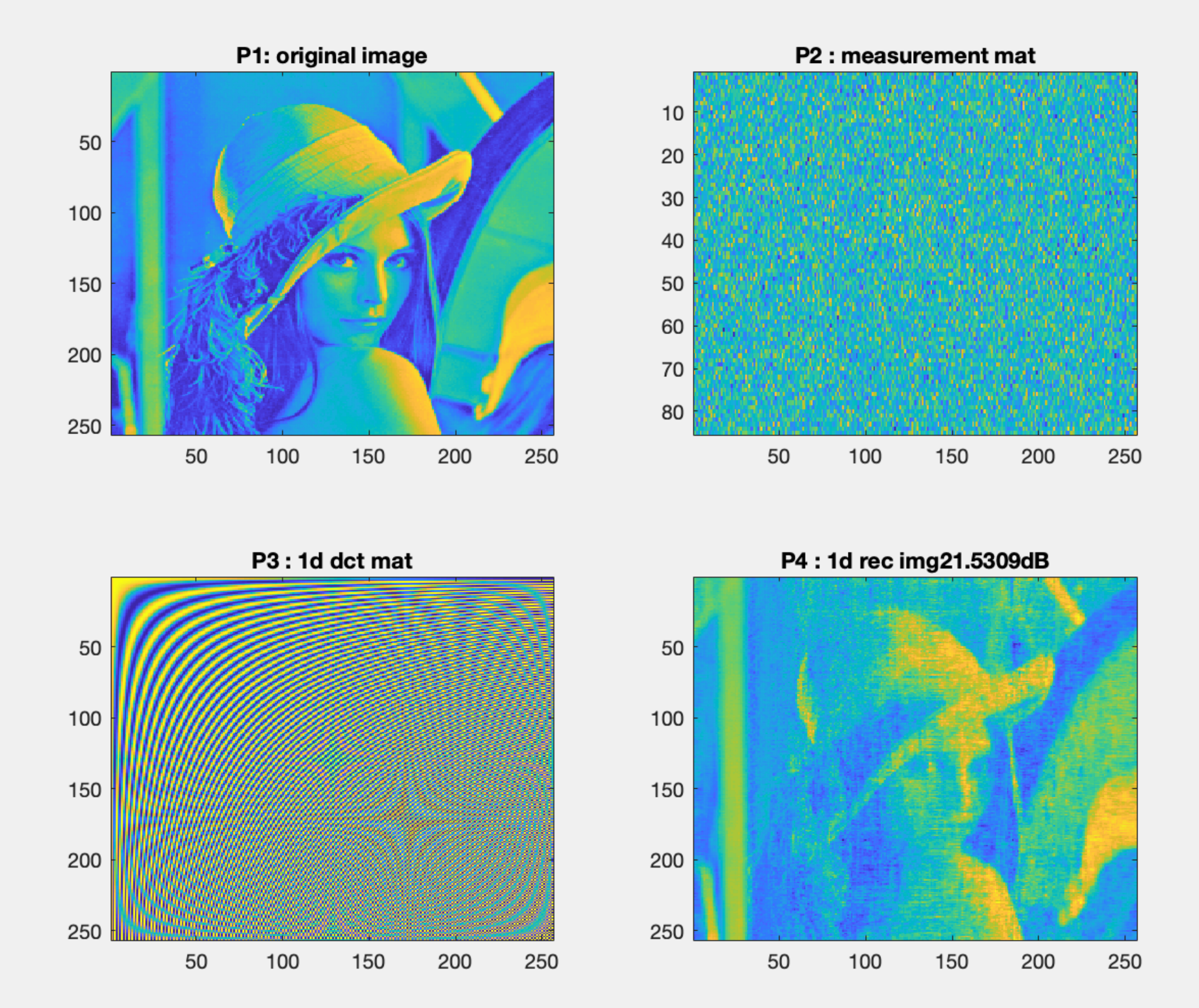
which also coincide with its eigendecomposition. For more rigorous explanation about the algorithm and proofs of its stability and convergence, we refer the readers to [1].

1.3 IRLS applications

There are many applications in the matrix completion problem solved by IRLS algorithm, and here we briefly mention one of them in the field of compressed sensing. The general problem is to recover the lost entries of a signal and if the signal is sparse in some appropriate meaning of the word, then it is possible to recover the signal efficiently by sampling only a portion of the matrix.

It was known for some time that we can do this due to what is called the Nyquist - Shannon sampling theorem [6]. However, the usage of IRLS made it clear that one can achieve even better efficiency as the theoretical was work pioneered in the work of Candes and Tao [5] showed. Below is a very brief example demonstrating the power of this method.

The following 4 pictures represent a simple progress of applying NNM - IRLS to matrix.



$$\min_{\boldsymbol{u}} \|\boldsymbol{u}\|_1, \quad \text{subject to } \boldsymbol{\Phi u} = \boldsymbol{b}$$

Based on the formula above, P1 is the original image I with a dimension of 256*256. We randomly pick one third rows of the original image as a new matrix Φ(P2) and then, reconstruct a new matrix Y = IΦ. We treat each column of Y as the b in the algorithm. For every column in Y, we will use IRLS to compute a u with minimum one norm. Finally, these u_i

are put in a matrix U(P4) which is a recovery of the original image. [4] P3 is a very common method to make image processing more accurately and efficiently.

Roughly speaking, we are applying IRLS - NNM algorithm successively to randomly chosen rows of the given image P2 and we recover the image P4. As one can clearly see, this is a remarkable recovery considering that the given image P2 hardly looks anything like the original image.

2 Warm-up

1. Based on the knowledge of rank minimization, which of the following is true?

a) Rank minimization is a special case of matrix completion.

b) When applying rank minimization, we don't limit the rank of original matrix.

c) Rank minimization problem can be solved in polynomial time.

d) The missing values position in matrix $\boldsymbol{M}$ is typically assumed to be randomly distributed.

2. Which of the following is/are true?

a) Semi - definite program can solve rank minimization efficiently in any kinds of matrix.

b) rank minimization problem is applied in many fields of science and technology, such as signal processing, collaborative filtering, compressed sensing.

c) One of the advantages of IRLS over linear programming and convex programming is that it can be solved by numerical iterative solvers such as Newton - Gauss method.

d) Each step of the proposed algorithm requires the computation of singular value thresholding and the solution of a (usually small) least squares problem.

3. We saw that the nuclear norm is a special case of the Schatten p - norm and is convex. In addition, we used the following identity in the algorithm:

$$\|\boldsymbol{X}\|_* = \|\boldsymbol{W}^{\frac{1}{2}}\boldsymbol{X}\|_F.$$

Prove the above identity by using the fact that $\boldsymbol{W} = (\boldsymbol{X}\boldsymbol{X}^T)^{-1/2}$ and its square root $\boldsymbol{W}^{\frac{1}{2}}$ is symmetric PSD and the fact that the trace function is invariant under cyclic permutations of its inputs; that is:

$$\mathrm{Tr}(\boldsymbol{X}\boldsymbol{A}) = \mathrm{Tr}(\boldsymbol{A}\boldsymbol{X}).$$

3 Main activity

1. **Weighted least squares(WLS)** WLS is a generalization of ordinary least squares and linear regression in which the errors covariance matrix is allowed to be different from an identity matrix. One can minimize the weighted sum of squares:

$$\arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{m} w_i \left| y_i - \sum_{j=1}^{n} X_{ij}\beta_j \right|^2 = \arg\min_{\boldsymbol{\beta}} \left\| W^{1/2}(\mathbf{y} - X\boldsymbol{\beta}) \right\|^2$$

The closed form solution is similar to least square problems:

$$\hat{\boldsymbol{\beta}} = (X^{\mathrm{T}} W X)^{-1} X^{\mathrm{T}} W \mathbf{y}$$

a) Consider the following matrix and vector:

$$\boldsymbol{X} = \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix}, \boldsymbol{y} = \begin{bmatrix} -1 \\ 2 \\ 1 \end{bmatrix}$$

Find the solution $\hat{\boldsymbol{\beta}}$ to $\arg\min_{\boldsymbol{\beta}} ||\boldsymbol{y} - \boldsymbol{X\beta}||^2$

b) Now let weight matrix

$$\boldsymbol{W} = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Find the solution $\hat{\boldsymbol{\beta}}$ to $\arg\min_{\boldsymbol{\beta}} \left\| W^{1/2}(\boldsymbol{y} - \boldsymbol{X\beta}) \right\|^2$

2. In this section we'll use the data file **incomplete.mat** from period 15 activity, which contains a rank-2, 16-by-16 matrix $\boldsymbol{X}_{true}$ with integer entries and three versions of this matrix ($\boldsymbol{Y_1}$, $\boldsymbol{Y_2}$ and $\boldsymbol{Y_3}$) with differing numbers of missing entries. The missing entries are indicated by NaN.
Attached python script provided IRLS-M algorithm for you to perform low-rank matrix completion. The function of this algorithm requires two inputs: i) the matrix with missing entries, and ii) the upper bound of matrix rank (recall that the precise rank is unnecessary in IRLS-M algorithm).

a) Complete the missing step in function IRLS_M.

b) Assuming the upper bound of rank is 3, apply IRLS-M algorithm to the three incomplete matrices. Compare your recovered completed matrices $\hat{\boldsymbol{Y}}_{\boldsymbol{i}}$ to $\boldsymbol{X}_{true}$ by computing the squared error $||\hat{\boldsymbol{Y}}_{\boldsymbol{i}} - \boldsymbol{X}_{true}||_F^2$. Does the number of missing entries affect the accuracy of the completed matrix?

c) Now try different γ $(= 3, 2, 1, 0.1, 0.01...)$ on incomplete matrix and also compute the squared error. Comment on the impact of using different γ in the completion process.

3. Performance evaluation In our script we also provided iterative singular value thresholding (ISVT) algorithm we learned in 4.6. File **Y.txt** and **Xtrue.txt** respectively contains a rank-5, 200-by-200 incomplete matrix $\boldsymbol{Y}$ and and its complete version $\boldsymbol{Xtrue}$. Perform ISVT and IRLS-M algorithm on $\boldsymbol{M}$ with different rank parameter, compare their running time, number of iterations and accuracy. Comment on your result. (Running the python script may take several minutes. Please be patient!)

References

[1] Massimo Fornasier, Holger Rauhut, and Rachel Ward. *Low-rank matrix recovery via iteratively reweighted least squares minimization.* `https://arxiv.org/pdf/1010.2471.pdf`

[2] Karthik Mohan and Maryam Fazel. *Iterative Reweigthed Algorithms for Matrix Rank Minimization.*

Journal of Machine Learning Research 13 (2012) 3441 - 3473.

[3] Benjamin Recht, Maryam Fazel and Pablo A. Parrilo. *Guaranteed Minimum Rank Solutions of Matrix Equations via Nuclear Norm Minimization.* `https://people.eecs.berkeley.edu/ brecht/papers/07.rfp.lowrank.pdf`

[4] Chartrand R, Yin W. Iteratively reweighted algorithms for compressive sensing[C]//2008 IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE, 2008: 3869-3872.

[5] Emmanuel Candes and Terence Tao. *The Dantiz Selector: Statistical Estimation when p is much larger than n.* Annals of Statistics 2007, Vol. 35, No. 6, 2313-2351.

[6] `https://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem`

4 Appendix

Warm-up solution

1. d.

2. b,c.

3.

$$\|\boldsymbol{W}^{1/2}\boldsymbol{X}\|_F = \mathrm{Tr}((\boldsymbol{W}^{1/2}\boldsymbol{X})^T\boldsymbol{W}^{1/2}\boldsymbol{X}) = \mathrm{Tr}(\boldsymbol{X}\boldsymbol{X}^T(\boldsymbol{W}^{1/2})^T\boldsymbol{W}^{1/2}) = \mathrm{Tr}(\boldsymbol{W}\boldsymbol{X}\boldsymbol{X}^T) =$$

$$= \mathrm{Tr}\left((\boldsymbol{X}\boldsymbol{X}^T)^{-1/2}\boldsymbol{X}\boldsymbol{X}^T\right) = \|\boldsymbol{X}\|_*.$$

Activity solution

1. Weighted least squares(WLS) WLS is a generalization of ordinary least squares and linear regression in which the errors covariance matrix is allowed to be different from an identity matrix. One can minimize the weighted sum of squares:

$$\arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{m} w_i \left| y_i - \sum_{j=1}^{n} X_{ij}\beta_j \right|^2 = \arg\min_{\boldsymbol{\beta}} \left\|\boldsymbol{W}^{1/2}(\mathbf{y} - \boldsymbol{X}\boldsymbol{\beta})\right\|^2$$

The closed form solution is similar to least square problems:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}W\mathbf{y}$$

a) $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} = \frac{1}{3}\begin{bmatrix} 1 \\ -1 \end{bmatrix}$

b) $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{y} \approx \begin{bmatrix} -0.43 \\ -1.28 \end{bmatrix}$

2. In this section we'll use the data file **incomplete.mat** from period 15 activity, which contains a rank-2, 16-by-16 matrix $\boldsymbol{X}_{true}$ with integer entries and three versions of this matrix ($\boldsymbol{Y_1}$, $\boldsymbol{Y_2}$ and $\boldsymbol{Y_3}$) with differing numbers of missing entries. The missing entries are indicated by NaN.
Attached python script provided IRLS-M algorithm for you to perform low-rank matrix

completion. The function of this algorithm requires two inputs: i) the matrix with missing entries, and ii) the upper bound of matrix rank (recall that the precise rank is unnecessary in IRLS-M algorithm).

a)

```
    W = U @ np.diag(s**(-1)) @ U.T
```

b) $\hat{\boldsymbol{Y}}_{\mathbf{1}}$ error: 12.7
$\hat{\boldsymbol{Y}}_{\mathbf{2}}$ error: $6.7 * 10^{-6}$
$\hat{\boldsymbol{Y}}_{\mathbf{3}}$ error: $3.4 * 10^{-7}$
Yes, when the number of missing entries getting larger, the errors are larger. The second and third example has the relatively small number of missing entries and recovery perfectly.

c) For $\hat{\boldsymbol{Y}}_{\mathbf{1}}$:
Gamma = 3 , Error: 12.6938091154
Gamma = 2 , Error: 12.6938091154
Gamma = 1 , Error: 12.6938091154
Gamma = 0.5 , Error: 12.1753757731
Gamma = 0.1 , Error: 11.8730828699
Gamma = 0.01 , Error: 11.8643645163
When γ getting smaller, the error also goes down. But a smaller γ will lead to longer running time, mostly we'll set $\gamma = 1$ in our IRLS-M algorithm.

3. **Performance evaluation** In our script we also provided iterative singular value thresholding (ISVT) algorithm we learned in 4.6. File **Y.txt** and **Xtrue.txt** respectively contains a rank-5, 200-by-200 incomplete matrix $\boldsymbol{Y}$ and and its complete version $\boldsymbol{Xtrue}$. Perform ISVT and IRLS-M algorithm on $\boldsymbol{M}$ with different rank parameter K, compare their running time and accuracy. Comment on your result.

r/K = 5
IRLS-M:
Running time: 104.54964399337769
Iter: 37
Error: 4.86705635669e-07

ISVT:
Running time: 0.9156858921051025
Iter: 10
Error: 0.0

r/K = 10
IRLS-M:

Running time: 105.59477090835571
Iter: 36
Error: 7.130343193e-07

ISVT:
Running time: 5.5990095138549805
Iter: 77
Error: 1462.07044974

Since solving the weighted least problem in IRLS-M costs much more computing resource, the running time of IRLS-M is usually longer than ISVT. When we know the precise rank of matrix, ISVT's performance is definitely better than IRLS-M. However, when the rank is unknown, IRLS-M algorithm performs much better than ISVT, needs less iteration and the error is almost zero.