High Dimensional Data Analysis Project report

Sergeev Nikita
Turgunboev Dadakhon
n.sergeev@innopolis.university
d.turgunboev@innopolis.university

1 Introduction

The goal of this project is to implement a method for matrix completion for the given sparse matrix. This problem is fundamental in the field of recommender systems and is known as collaborative filtering. The matrix completion problem can be formulated as follows: given a partially observed matrix, the objective is to estimate the missing entries of the matrix.

Formally, it is presented as the following optimization problem: Given a sparse matrix $X \in \mathbb{R}^{m \times n}$, the goal is to find matrices $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{r \times n}$ that minimize the following objective function:

$$\min_{U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{r \times n}} \sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij} \left(X_{ij} - (UV)_{ij}\right)^2, \tag{1}$$

where

$$W_{ij} = \begin{cases} 1, & \text{if } X_{ij} \text{ is observed,} \\ 0, & \text{otherwise.} \end{cases}$$

The matrix completion problem is ill-posed, as there are infinitely many matrices U and V that exactly fit the observed entries. The objective is to find a matrix that fits the observed entries well and has some additional structure that allows generalization to the missing entries.

2 Description of Used Models

The task defined above assumes matrix factorization. The following model was used:

$$X \approx UV \tag{2}$$

The main adjustment introduced in this model is the initialization of the matrices U and V. Two different methods were utilized:

2.1 Average Initialization

In this case:

- Matrix $V \in \mathbb{R}^{n \times r}$ is initialized as a matrix of ones.
- Matrix $U \in \mathbb{R}^{m \times r}$ is initialized such that each row is the average of the observed values in the corresponding column of the matrix X.

This method avoids the problem of zero initialization, which can lead to zero gradients, causing the model to fail to learn. Additionally, average initialization of U captures client preferences, which is useful for recommender systems.

2.2 SVD Initialization

In this method, the matrix X is approximated using SVD decomposition, and U and V are initialized as follows:

$$X \approx U_{\text{svd}} \Sigma_{\text{svd}} V_{\text{svd}}^T, \tag{3}$$

where U_{svd}, Σ_{svd}, and V_{svd} are the matrices obtained from the SVD of X. To retain the information in Σ_{svd}, which contains the eigenvalues of X, the matrices U_{svd} and V_{svd} are scaled by the square root of the eigenvalues:

$$U = U_{\text{svd}} \sqrt{\Sigma_{\text{svd}}}, \quad V = V_{\text{svd}} \sqrt{\Sigma_{\text{svd}}}. \tag{4}$$

SVD initialization provides a good approximation of X, serving as an effective starting point for optimization. This initialization helps the optimization process avoid local minima and achieve better solutions.

3 Description of Used Algorithms

As an optimization algorithm for the objective function, block coordinate descent with gradient descent updates was employed. The overall algorithm can be described as follows:

Algorithm 1 Block Coordinate Descent with Gradient Descent Updates

Require: Matrix $X \in \mathbb{R}^{m \times n}$, weight matrix $W \in \{0,1\}^{m \times n}$, iterations max_k, tolerance $\epsilon > 0$.

1: Initialize $U^{(0)} \in \mathbb{R}^{m \times r}$, $V^{(0)} \in \mathbb{R}^{n \times r}$, and $\text{step}_u > 0$.
2: Set iteration counter $k \leftarrow 1$.
3: Initialize the error $e_0 \leftarrow \infty$.
4: **repeat**
5: **Step 1: Update U while fixing V:**
6: Compute residual: $R = U^{(k)} V^{(k)\top} - X$.
7: Mask residual: $\text{WR} = W \odot R$.
8: Compute gradient for $U^{(k)}$:

$$\nabla_U = \text{WR} \cdot V^{(k)}$$

9: Update U using line search:

$$\text{step}_u \leftarrow 2 \cdot \frac{\|U^{(k)}\|}{\|\nabla_U\|}$$

10: Perform gradient descent with backtracking:

$$U^{(k+1)} \leftarrow U^{(k)} - \text{step}_u \cdot \nabla_U$$

11: Adjust step_u using γ until the new error:

$$e_1 = \|\text{WR} \odot (X - U^{(k+1)} V^{(k)\top})\|^2$$

satisfies $e_1 \leq e_0$.
12: Scale $\text{step}_u \leftarrow \beta \cdot \text{step}_u$.
13: **Step 2: Update V while fixing U:**
14: Repeat analogous updates for $V^{(k+1)}$ (X^T, W^T, switch U and V in all computations).
15: **Step 3: Compute the root mean squared error (RMSE):**

$$\text{RMSE}_u = \sqrt{e_1 / |\Omega|}$$

where $|\Omega|$ is the number of nonzero entries in W.
16: Check convergence:
17: **if** $\frac{e_{k-1} - e_k}{e_{k-1}} < \epsilon$ **then**
18: **Break.**
19: **end if**
20: Increment iteration counter $k \leftarrow k + 1$.
21: **until** $k >$ max_k
22: **return** Matrices $U^{(k)}, V^{(k)}$.

This algorithm iteratively updates both matrices U and V to achieve optimal matrix X factorization with minimization of the initial objective (1).

3.1 Hyperparameters

Described algorithm have the following hyperparameters that can be tuned to have a better or faster convergence:

1. r - rank of matrices U and V. It represents how much information can be stored in these matrices.
2. γ - scale of the gradient descent step for the line search.
3. β - final scale of the stepsize at the end of the iteration.
4. ϵ - minimal objective improvement, which we count as significant. If improvement of the algorithm on certain step is less then ϵ we assume that algorithm achieved stationary point and converge.
5. k_{max} - maximum number of iteration for the algorithm, if convergence criteria doesn't holds.

3.2 L_2 regularization

This improvement for the algorithm which we applied for this task have the following idea:
Add L_2 norm term to the loss. This term will help to make the solution more robust. While finding optimal solution there may be the case then U and V makes good approximation for the training data (observable values of X) but have a greater error on unknown values of user-item matrix, which is actually, the main goal of this task. As, where is two sets of decision variables, which should be optimized, this normalization have the following form

$$\begin{aligned} V^{k+1} &\leftarrow \min_V f(U^{k+1}, V) + \lambda \|V\|_2 \\ V^{k+1} &\leftarrow V^{k+1} - g(V^k) + 2\lambda \|V\| \end{aligned} \tag{5}$$

$$\begin{aligned} U^{k+1} &\leftarrow \min_U f(U, V^{k+1}) + \lambda \|U\|_2 \\ U^{k+1} &\leftarrow U^{k+1} - g(U^k) + 2\lambda \|U\| \end{aligned} \tag{6}$$

where:

1. $f(\cdot)$ - objective function, we tends to minimize
2. $g(\cdot)$ - gradient of the objective function on certain variable (U or V)
3. $\lambda \geq 0$ - hyperparameter, which determines the regularization strength

This adjustment to optimization objective prevent the model to have to big values of in matrices U and $V \rightarrow$ make solution more robust to overfitting.

3.3 Accelerated gradient descent

Next improvement to the algorithm is connected with the way pf calculating the stepsize for the gradient descent. Accelerated Gradient Descent is a technique to stabilize the training procedure. The intuition behind this method comes from viewing the optimization process through the lens of physics, treating the parameter updates as a particle moving through a potential field defined by the loss function. The momentum term adds an "inertia" to the updates, allowing the algorithm to maintain its direction even when the gradient changes, thus accelerating convergence and add a potential possibility for the method to avoid local minima in some cases.

Main changes for implementing accelerated GD is applied to the process of updating the model's parameters (step 10

in algo. 1). It add a factor of the previous gradient to the changes of decision variables and updates for both U and V becomes as following:

$$\begin{aligned} U^{(k+1)} &= U^{(k)} - \alpha \nabla_U^{(k)} + \mu(U^{(k)} - U^{(k-1)}), \\ V^{(k+1)} &= V^{(k)} - \alpha \nabla_V^{(k)} + \mu(V^{(k)} - V^{(k-1)}), \end{aligned} \tag{7}$$

where:

- $\alpha > 0$ is the learning rate,
- $\mu \in [0, 1)$ is the momentum coefficient hyperparameter, it determines, how much previous gradient will affect the current stepsize.
- $\nabla_U^{(k)}$ and $\nabla_V^{(k)}$ are the gradients of the loss function with respect to U and V at iteration k,
- $U^{(k-1)}$ and $V^{(k-1)}$ are the parameter values from the previous iteration.

4 Discussion of results

This section describes all the approaches we tried in the flow of completing given project. Description of each approach consists of the following parts:

1. Brief description of the approach. It includes the unique features used in the certain approach and how they can possibly help to improve the solution.
2. Set of hyperparameters used to in the solution
3. Results obtained on the train and test set with corresponding convergence rate.

4.1 Baseline Solution

As a first approach to solve this task we just used an algo. 1 as it is. As for the matrices U and V initialization we used a straightforward average initialization described in 2.1. Considering hyperparameters, we didn't tune them for the initial solution, and use just a random ones:

Parameter	Value
r	5
γ	1.5
β	4
ϵ	10^{-6}
k_{max}	100

Table 1. Hyperparameter Values Used in Initial Solution

While training I got the convergence plot presented at Fig. 1. As it is visible from the plot, RMSE decreasing over all training process and stabilizes at the end. As a result I got the following metrics:

1. RMSE = 0.92 on the observed values of X
2. RMSE = 0.923 on test values of X

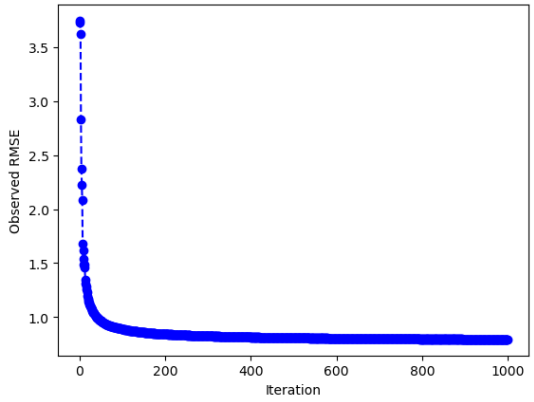

Figure 1. Metric convergence plot for baseline solution.

4.2 Adding L_2 regularization to loss

To improve the performance of solution, we decided add L_2 regularization to the optimization objective. This should make training process more stable. Full motivation for using it and description of the implementation described in 3.2. Considering the hyperparameters, in this case we used the following ones:

Parameter	Value
r	5
γ	3
β	4
ϵ	10^{-6}
k_{max}	500
λ	0.1

Table 2. Hyperparameter used for L_2 regularized solution

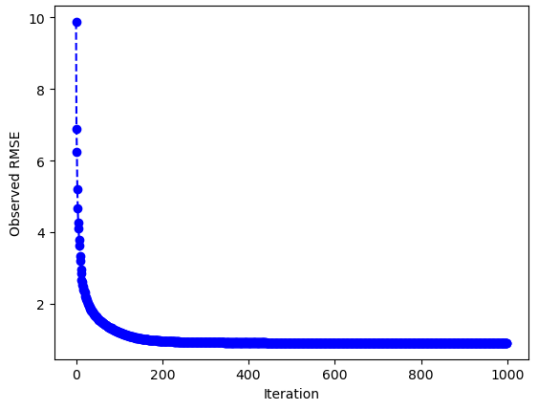

Figure 2. Objective convergence plot for L_2 solution

Using more steps for training might help to achieve better results. And greater γ make linesearch for the stepsize more

aggressive, and therefore training goes faster. For the initialization in this case we also used simple average approach. And the regularization parameter λ was set to 0.1, which quite standard value for this parameter in machine learning tasks.

Considering the results of training for this version of algorithm, Loss values during training presented at Fig. 2. As it is visible, initial objective is bigger then in the previous case because of L_2 term in loss. But at the end of training it achieves better results then baseline:

1. Objective = 0.903 on the observed values of X (even with L_2 norm of matrix)
2. RMSE = 0.913 on test values of X

4.3 Applying SVD initialization

To further improve algorithm we decided to use an improved method for U and V matrices initialization, described in 2.2. Set of hyperparameters used in this case not much differs from the used in the previous version.

Parameter	Value
r	7
γ	3
β	4
ϵ	10^{-6}
k_{max}	500
λ	0.1

Table 3. Hyperparameter used for SVD init solution

The only difference is increased rank of matrices. This way, the training will take more time and computation, but U and V can handle more complex dependencies of the initial matrix X.

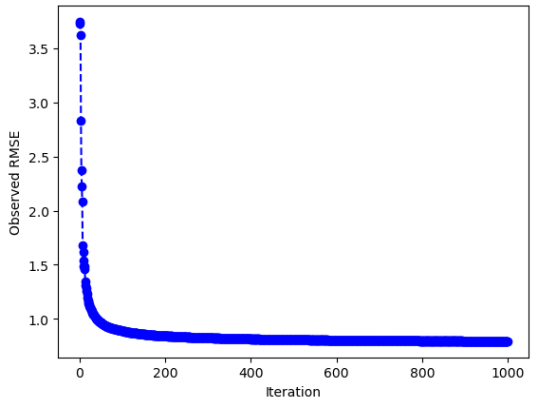

Figure 3. Objective convergence plot for SVD solution

Considering the results of this method, we can see what the initial objective function value decreased, compared with the previous solution. This is so, because the initial U and V is the closest approximation of the given matrix X. During the training, this method converges to the better objective value and got the following results:

1. Objective = 0.792 on the observed values of X (even with L_2 norm of matrix)
2. RMSE = 0.854 on test values of X

This specific initialization have much more impact on the final score, compared to the previous approach. From this we can conclude that this step is crucial for successful optimization of the given objective.

4.4 Using Accelerated GD

To further experiment with the different approaches, we decided to use a more advanced gradient descent technique - Accelerated gradient descent, described in 3.3. For it we used the following hyperparameters:

Parameter	Value
r	7
γ	-
β	-
ϵ	-
k_{max}	500
λ	0.1
μ	0.9
α	10^{-4}

Table 4. Hyperparameter used for Accelerated GD solution

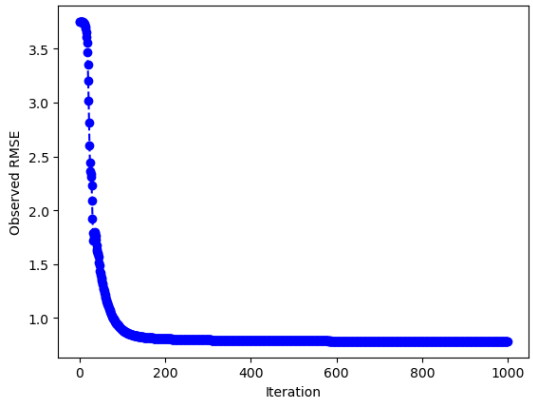

Figure 4. Objective convergence plot for Accelerated GD solution

Important to notice, that in this case we removed line search from the method, instead we used constant step size for $\alpha = 10^{-4}$ both U and V updates. This decision was made because with the line search stepsize approaches zero at some point, and parameters stops updating. Intuitively, this may happen, because momentum may move the solution to

the point with worse objective function. But this is done to achieve better solution in the future, and avoid possible local optimum.

Convergence plot presented in the Fig. 4. As it is visible from the plot, this method achieves near optimal value faster then previous ones. Moreover it achieves better results in the training set:

1. Objective = 0.784 on the observed values of X (even with L_2 norm of matrix)
2. RMSE = 0.862 on test values of X

However, on the test set with unobserved values of X, this method shows worse results, then the GD with linesearch. Possible reason for this may be the overfitting on the training data without capturing the general dependencies in the data.

5 Conclusion

In this project, we implemented and analyzed several approaches to solve the matrix completion problem using collaborative filtering techniques. The baseline solution, while straightforward, provided a good starting point but showed limitations in terms of convergence. Incorporating L_2 regularization improved the model's robustness by penalizing large values in the factorized matrices, which led to a lower RMSE on the test set. SVD initialization proved to be highly effective, significantly improving both training and test performance by providing a better starting point for optimization. Finally, the use of Accelerated Gradient Descent demonstrated faster convergence but at the cost of slightly higher overfitting, suggesting a need for further regularization or tuning.

Overall, the results indicate that careful initialization and regularization play a crucial role in matrix completion tasks. While SVD initialization combined with L_2 regularization yielded the best test results, the performance of Accelerated Gradient Descent highlights the potential for further exploration of advanced optimization techniques. Future work could focus on hybrid approaches, adaptive learning rates, or alternative regularization methods to enhance both convergence speed and generalization performance.