High Dimensional Data Analysis Project

Recommendation System

Team 7: Sergeev Nikita, Turgunboev Dadakhon, Roman Konkov.

Table of contents

01

Introduction and Description of Used models

Introduction and Problem Statement

The goal of this project is to address the matrix completion problem, a fundamental task in recommender systems and collaborative filtering. The objective is to estimate the missing entries in a partially observed matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ by finding two smaller matrices $\boldsymbol{U} \in \mathbb{R}^{m \times r}$ and $\boldsymbol{V} \in \mathbb{R}^{r \times n}$ that minimize the reconstruction error for observed entries.
The optimization problem can be formulated as:

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

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 matrix factorization model assumes

$$\boldsymbol{X} \approx \boldsymbol{UV}$$

where: $\boldsymbol{U} \boldsymbol{\epsilon}\, \mathbb{R}^{\boldsymbol{m\,x\,r}}$ and $\boldsymbol{V\epsilon}\, \mathbb{R}^{\boldsymbol{n\,x\,r}}$

Two initialization methods were used to enhance the optimization process:

- **Average Initialization:** In this method, the matrix $\boldsymbol{V}$ is initialized as a matrix of ones. The matrix **U** *is initialized such that each row contains the average of the observed values in the corresponding column of the matrix **X**.*
- ***SVD Initialization:*** This method approximates the matrix X using SVD, expressed as:

$$\mathbf{X} \approx \boldsymbol{U}_{svd}\boldsymbol{\Sigma}_{svd}\boldsymbol{V}_{svd}^{\boldsymbol{T}}$$

To retain the eigenvalue information, the matrices are scaled by the square root of the **eigenvalues**:

$$\boldsymbol{U} = \boldsymbol{U}_{svd}\sqrt{\boldsymbol{\Sigma}_{svd}}$$
$$\boldsymbol{V} = \boldsymbol{V}_{svd}\sqrt{\boldsymbol{\Sigma}_{svd}}$$

Description of Algorithm 1

Algorithm 1 Block Coordinate Descent with Gradient Descent Updates

- **Block Coordinate Descent:** This method alternates between updating the matrices *U and V*, fixing one matrix while updating the other. This approach simplifies the optimization problem by breaking it into smaller subproblems.
- **Backtracking Line Search** After computing the gradient, this technique is used to find the optimal step size in gradient descent, ensuring that the new error is smaller than the previous one.

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

Initialize $U^{(0)} \in \mathbb{R}^{m\times r}$, $V^{(0)} \in \mathbb{R}^{n\times r}$, and $\text{step}_u > 0$.

Set iteration counter $k \leftarrow 1$.

Initialize the error $e_0 \leftarrow \infty$.

repeat

  Step 1: Update U while fixing V:

  Compute residual: $R = U^{(k)}V^{(k)\top} - X$.

  Mask residual: $\mathrm{WR} = W \odot R$.

  Compute gradient for $U^{(k)}$:

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

  Update U using line search:

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

  Perform gradient descent with backtracking:

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

  Adjust step_u using γ until the new error:

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

  satisfies $e_1 \le e_0$.

  Scale $\text{step}_u \leftarrow \beta \cdot \text{step}_u$.

  Step 2: Update V while fixing U:

  Repeat analogous updates for $V^{(k+1)}$ (X^T, W^T, switch U and V in all computations).

  Step 3: Compute the root mean squared error (RMSE):

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

  where $|\Omega|$ is the number of nonzero entries in W.

  Check convergence:

  if $\frac{e_{k-1}-e_k}{e_{k-1}} < \epsilon$ **then**

    Break.

  end if

  Increment iteration counter $k \leftarrow k+1$.

until $k >$ max_k

return Matrices $U^{(k)}, V^{(k)}$.

02

Hyperparameters and L_2 regularization

Hyperparameters

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

- $\boldsymbol{r}$: Rank of matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, indicating the amount of information that can be stored in these matrices.
- $\boldsymbol{\gamma}$: Scale of the gradient descent step used in the line search to determine the optimal step size.
- $\boldsymbol{\beta}$: Final scale of the step size after each iteration, adjusting the learning rate.
- $\boldsymbol{\epsilon}$: Minimal objective improvement considered significant.
- $\boldsymbol{k_{max}}$: Maximum number of iterations allowed; if the algorithm doesn't converge within this number of iterations, it stops.

L_2 regularization

This improvement adds an L_2 norm regularization term to the loss function to enhance robustness. It prevents overfitting by penalizing large values in matrices U and V. The regularization helps the model generalize better to missing values in the matrix, ensuring that the solution is more stable and less prone to overfitting. The regularization strength is controlled by the hyperparameter λ.

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

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

where:

- $f\,(\cdot)$ - objective function to minimize.
- $g(\cdot)$ - gradient of the objective function with respect to U or V.
- $\lambda \geq 0$ - regularization strength hyperparameter.

03

Accelerated Gradient Descent: Enhancing Optimization Efficiency

Accelerated Gradient Descent

Accelerated Gradient Descent (AGD) improves convergence speed and stability by adding a momentum term, which helps the algorithm "remember" its previous direction. The momentum term also helps avoid local minima by maintaining the algorithm's trajectory, even in regions where the gradient is weak.

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

where:

- $\boldsymbol{\alpha} > \mathbf{0}$: learning rate.
- $\boldsymbol{\mu} \in [\mathbf{0}, \mathbf{1})$: momentum coefficient hyperparameter.
- $\nabla_{\boldsymbol{U}}^{(\boldsymbol{k})}$ and $\nabla_{\boldsymbol{V}}^{(\boldsymbol{k})}$: gradients of the loss function with respect to ***U*** and ***V*** at iteration **k**,
- $\boldsymbol{U}^{(\boldsymbol{k}-\mathbf{1})}$ and $\boldsymbol{V}^{(\boldsymbol{k}-\mathbf{1})}$: parameter values from the previous iteration.

04

Comparison of Results Across Models and Techniques

Baseline Solution

- **Initial Approach**: Used Algorithm 1 as-is, with out modifications.
- **Matrix Initialization**: Applied straightforward average initialization for ***U*** and ***V***
- ***Hyperparameters:*** *No hyperparameter tuning*

Performance

- **Objective** = 0.92 on the observed values of X
- **RMSE** = 0.923 on test values of X

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

Table 1. Hyperparameter Values Used in Initial Solution

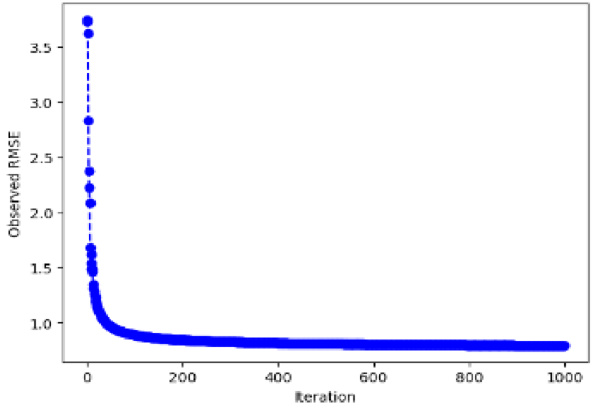

Figure 1. Metric convergence plot for baseline solution.

L_2 regularization

- **L_2 Regularization**: Added to the optimization objective to stabilize the training process and prevent overfitting.
- **Hyperparameters**:
  - Increased γ to make the line search for step size more aggressive, speeding up training.
  - Used a simple average approach for matrix initialization.
  - Regularization parameter λ was set to 0.1, a common value in machine learning tasks.

Performance

- **Objective** = 0.903 on the observed values of X (even with L_2 norm of matrix)
- **RMSE** = 0.913 on test values of X

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

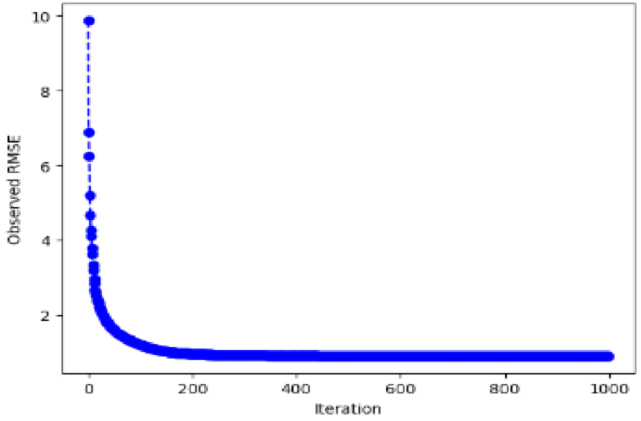

Figure 2. Objective convergence plot for L_2 solution

Applying SVD initialization

- **Improved Initialization**: Used advanced method for initializing U and V matrices
- **Hyperparameter Change**: The main difference was an increased rank of matrices, allowing U and V to capture more complex dependencies in X.
- **Training Impact**: While training took more time, the improved initialization resulted in a better starting point for optimization.

Performance

- **RMSE** = 0.792 on the observed values of X
- **RMSE** = 0.854 on test values of X

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

Table 3. Hyperparameter used for L_2 regularized solution

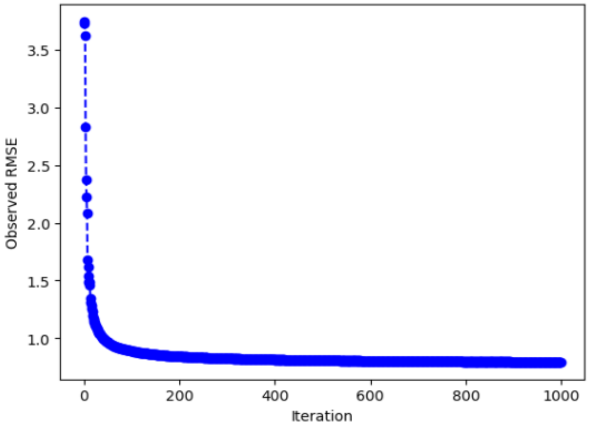

Figure 3. Objective convergence plot for SVD solution

Using Accelerated GD

- **Removal of Line Search:** Instead of using line search for step size adjustment, a fixed step size (α = 10^{-4})
- **Use of Momentum:** Added to improve convergence speed and prevent getting stuck in local minima.
- **Convergence**: The method converged faster and reached nearly optimal results (Fig. 4).

Performance

- **Objective** = 0.784 on the observed values of X (even with $\boldsymbol{L_2}$)
- **RMSE** = 0.862 on test values of X

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

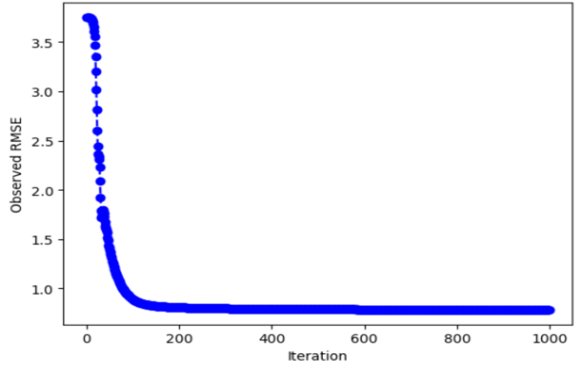

Figure 4. Objective convergence plot for Accelerated GD solution

Conclusion

- **Baseline model**: Simple starting point, struggled with convergence.
- **L_2 regularization**: Improved robustness and reduced RMSE.
- **SVD initialization**: Strongly improved performance by providing a better starting point.
- **Accelerated Gradient Descent (AGD)**: Faster convergence but caused slight overfitting, highlighting the need for further tuning.
- **Best results**: Achieved with SVD initialization + L_2 regularization.
- **Future work**: Focus on hybrid models, adaptive learning rates, and advanced regularization techniques to further enhance performance and generalization.

Thank you for your attention !