Low-Rank Matrix Completion via Deep Neural Networks

Mostafa Alkady

Machine Learning for Physicists and Mathematicians Summer School 2025
Northeastern University

1. Problem Statement
2. Dataset
3. Model
4. Results

1. Problem Statement
2. Dataset
3. Model
4. Results

Matrix Completion Problem Statement

- Given a partially observed matrix $M \in \mathbb{R}^{m\times n}$, where only a subset of entries $(i, j) \in \{1, \dots, m\} \times \{1, \dots, n\}$ are known, the goal is to recover the full matrix by estimating the missing entries.
- This is typically done under the assumption that the true underlying matrix is *low-rank*, i.e.,

$$\mathrm{rank}(M) \ll \min(m, n).$$

Our Problem

- In this project, we restrict our attention to square matrices $M \in \mathbb{R}^{n\times n}$.
- Number of independent matrix elements in a square matrix of size n and rank r is $2nr - r^2$.
- We mask $m < n^2 - 2nr + r^2$ entries from the matrix and have the neural network predict them.

Examples

2 × 2 example (rank 1):

$$\begin{pmatrix} 0.6 & -0.2 \\ 1.2 & * \end{pmatrix} \to \begin{pmatrix} 0.6 & -0.2 \\ 1.2 & -0.4 \end{pmatrix}$$

4 × 4 example (rank 2):

$$\begin{pmatrix} -0.303 & 0.388 & * & -0.64 \\ -4.413 & -2.146 & -0.12 & 0.776 \\ * & 0.362 & 1.1 & -0.307 \\ 1.375 & 0.96 & 1.717 & * \end{pmatrix} \to \begin{pmatrix} -0.303 & 0.388 & 3.082 & -0.64 \\ -4.413 & -2.146 & -0.12 & 0.776 \\ 0.356 & 0.362 & 1.1 & -0.307 \\ 1.375 & 0.96 & 1.717 & -0.619 \end{pmatrix}$$

1. Problem Statement
2. Dataset
3. Model
4. Results

We generate random matrices A of size (n, n) and rank r as follows:

- Generate two matrices U and V of sizes (n, r)
- $A = UV^T$
- The matrix A will have rank r.

We generate random matrices A of size (n, n) and rank r as follows:

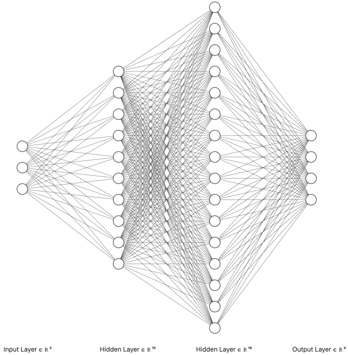
- Generate two matrices U and V of sizes (n, r)
- $A = UV^T$
- The matrix A will have rank r.

Proof.

Left as an exercise to the reader. $\square$

1. Problem Statement
2. Dataset
3. Model
4. Results

Architecture: Feed-Forward Neural Network, with two hidden layers and ReLU activations.[1]


Input Layer $\in \mathbb{R}^3$
Hidden Layer $\in \mathbb{R}^{10}$
Hidden Layer $\in \mathbb{R}^{16}$
Output Layer $\in \mathbb{R}^4$

$$\text{Proposed number of neurons} = \mathcal{O}(n^2)$$

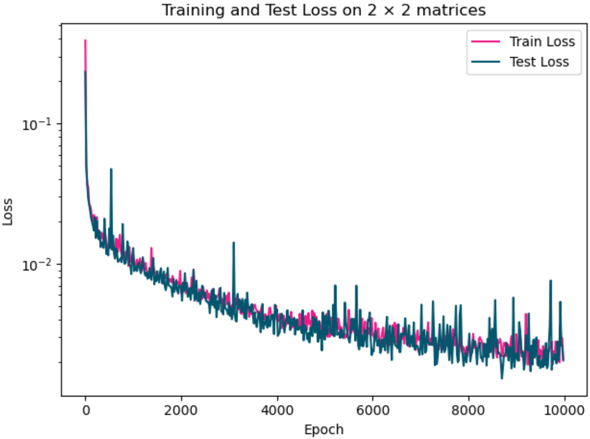
[1] Addition of the second hidden layer significantly improved the model.

Model Summary:

- ADAM optimizer with initial learning rate $\alpha = 0.001$
- Training epochs = 10,000
- Mini-batch Gradient Descent with batch size = 64
- 90-10 Data split (training/testing)

1. Problem Statement
2. Dataset
3. Model
4. Results

Performance on 2 × 2 matrices



Performance on 2×2 matrices

Input:

$$\begin{pmatrix} -0.0233782 & -0.0575182 \\ -0.12784362 & * \end{pmatrix}$$

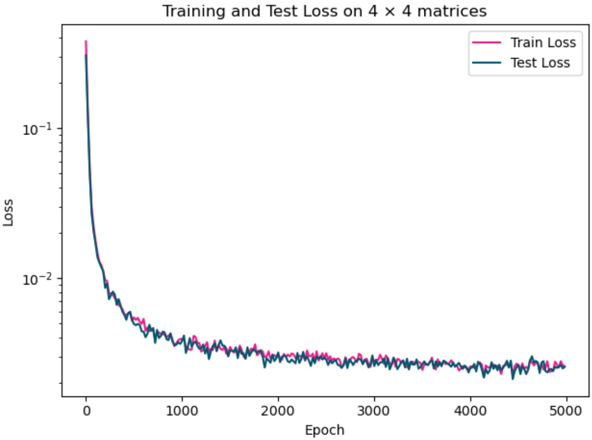
Ground truth:

$$\begin{pmatrix} -0.0233782 & -0.0575182 \\ -0.12784362 & -0.31453815 \end{pmatrix}$$

Prediction:

$$\begin{pmatrix} -0.02139766 & -0.05180938 \\ -0.13365555 & -0.30243173 \end{pmatrix}$$

Performance on 4×4 matrices



Performance on 4 × 4 matrices

Input:

$$\begin{pmatrix} -0.784 & -0.07 & -0.167 & 0.333 \\ -1.185 & 0.579 & 0.841 & * \\ * & * & -2.152 & -0.031 \\ 1.413 & -0.628 & -0.904 & -0.565 \end{pmatrix}$$

Ground truth:

$$\begin{pmatrix} -0.784 & -0.07 & -0.167 & 0.333 \\ -1.185 & 0.579 & 0.841 & 0.472 \\ 0.221 & -1.358 & -2.152 & -0.031 \\ 1.413 & -0.628 & -0.904 & -0.565 \end{pmatrix}$$

Prediction:

$$\begin{pmatrix} -0.766 & -0.074 & -0.143 & 0.313 \\ -1.187 & 0.603 & 0.854 & 0.388 \\ 1.625 & -0.602 & -2.103 & -0.028 \\ 1.378 & -0.627 & -0.876 & -0.572 \end{pmatrix}$$

Performance on 4 × 4 matrices

But how good are these predictions?

Analysis of 2×2 Results

Recall that

$$\begin{aligned} M &= UV^T && (1) \\ &= \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \begin{pmatrix} v_1 & v_2 \end{pmatrix} = \begin{pmatrix} u_1 v_1 & u_1 v_2 \\ u_2 v_1 & u_2 v_2 \end{pmatrix} && (2) \end{aligned}$$

Since $u_2 \sim \mathcal{N}(0,1)$ and $v_2 \sim \mathcal{N}(0,1)$, the distribution of the element $u_2 v_2$ follows

$$P_Z = \frac{1}{\pi} K_0(|Z|) \qquad (3)$$

where K_0 is the modified Bessel function of the second kind of order zero, **which also has variance = 1**.

Distribution of elements in $n \times n$ matrix (rank r)

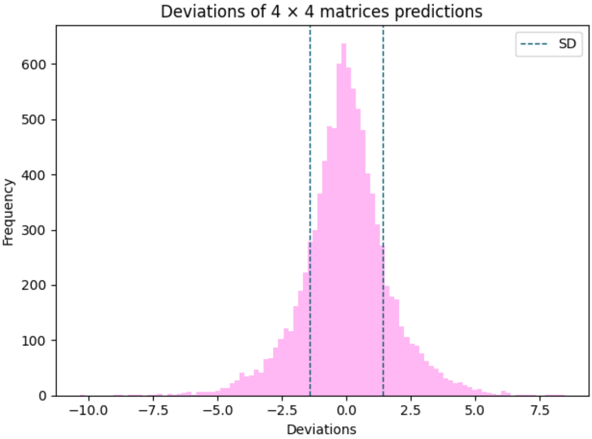
For a general $n \times n$ matrix of rank r, we have

$$M_{ij} = \sum_{k=1}^{r} U_{ik} V_{jk} \tag{4}$$

Since U_{ik} and $V_{jk} \sim \mathcal{N}(0,1)$, M_{ij} is a sum of r independent random variables with variance 1, therefore

$$\mathrm{Var}[P(M_{ij})] = r \tag{5}$$

Distribution of elements in $n \times n$ matrix



68.4 % lie within 1 σ from the mean![2]

[2] Thanks Aria for the idea!

Going forward..

- General $n \times m$ matrix completion problem
- Varying the masking with iterations
- Varying input size

Acknowledgements:

- Aria Masoomi
- Sean Gunn

  for their immense help in the project, and

- Paul Hand
- Fabian Ruehle
- Mahsa Bazzaz

  for all their efforts throughout the school.

Thank You!

Questions?