CS-4053 Recommender System

Spring 2023

Lecture 7: Matrix Factorization

Course Instructor: Syed Zain Ul Hassan

National University of Computer and Emerging Sciences, Karachi

Email: zain.hassan@nu.edu.pk

Flow of this lecture

- ❑ Features
- ❑ Factorizing interaction matrix
- ❑ Storage
- ❑ Optimization
- ❑ Predictions

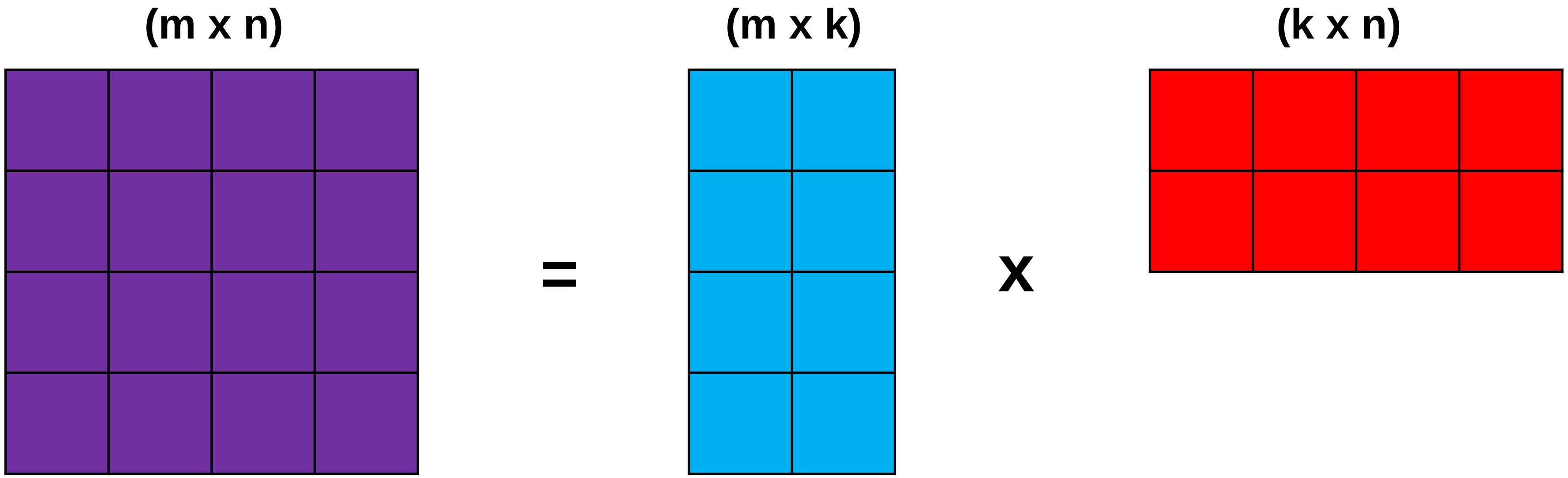
Factorization

- **Factorization** is a mathematical technique that allows a term to be decomposed into a product of two or more smaller terms

$$\mathbf{16} = \mathbf{8} \times \mathbf{2}$$

Matrix Factorization

- **Matrix Factorization** is a technique in which user-item interaction matrix is decomposed into a product of two or three matrices



Matrix Factorization

- We have a set of users and their interaction matrix for anime

User 1

User 2

User 3

User 4

Matrix Factorization

		I1	I2	I3	I4	I5
User 1		2	1	4	1	2
User 2		1	4	1	2	2
User 3		3	5	5	3	4
User 4		2	3	?	2	3

Matrix Factorization

- Let us consider some anime, their features, and **User 1**

User 1

Action	Sci-fi
1	0

Action	Sci-fi
2	1

Action	Sci-fi
1	4

Action	Sci-fi
4	1

Matrix Factorization

- ❑ Let us consider some anime, their features, and User 1

User 1

Action	Sci-fi
1	0

1 x 2 + 0 x 1

= 2

(User 1's rating for I1)

Action	Sci-fi
2	1

Action	Sci-fi
1	4

Action	Sci-fi
4	1

Matrix Factorization

- Let us try this with **User 2**

User 2

Action	Sci-fi
0	1

0 x 2 + 1 x 1

= 1

(User 2's rating for I1)

Action	Sci-fi
2	1

Action	Sci-fi
1	4

Action	Sci-fi
4	1

Matrix Factorization

		F1	F2
User 1		1	0
User 2		0	1
User 3		1	1
User 4		1	0.5

(m x k)

(k x n)

	I1	I2	I3	I4	I5
F1	2	1	4	1	2
F2	1	4	1	2	2

	I1	I2	I3	I4	I5
User 1	2	1	4	1	2
User 2	1	4	1	2	2
User 3	3	5	5	3	4
User 4	2	3	?	2	3

(m x n)

Matrix Factorization: Prediction

		F1	F2
User 1		1	0
User 2		0	1
User 3		1	1
User 4		1	0.5

(m x k)

(k x n)

	I1	I2	I3	I4	I5
F1	2	1	4	1	2
F2	1	4	1	2	2

1 x 4 + 0.5 x 1 ≈ 4

	I1	I2	I3	I4	I5
User 1	2	1	4	1	2
User 2	1	4	1	2	2
User 3	3	5	5	3	4
User 4	2	3	4	2	3

(m x n)

Matrix Factorization: **Finding Factors**

(k x n)

	I1	I2	I3	I4	I5
F1	2	1	4	1	2
F2	1	4	1	2	2

		F1	F2
User 1		1	0
User 2		0	1
User 3		1	1
User 4		1	0.5

(m x k)

Issue: *Where do **these** values come from? In other words, how do we find these two matrices (factors)?*

Matrix Factorization: Finding Factors

(k x n)

	I1	I2	I3	I4	I5
F1	2	1	4	1	2
F2	1	4	1	2	2

	F1	F2
User 1	1	0
User 2	0	1
User 3	1	1
User 4	1	0.5

(m x k)

Issue: *Where do **these** values come from? In other words, how do we find these two matrices (factors)?*

Answer: ***Gradient Descent***

Matrix Factorization: Gradient Descent

		F1	F2
User 1		0.2	0.5
User 2		0.4	0.6
User 3		0.1	0.05
User 4		0.9	0.3

	I1	I2	I3	I4	I5
F1	0.1	1.5	0.9	3.4	1.2
F2	0.8	3.1	2.5	4.5	0.6

Start with a rough guess

0.2 x 0.1 + 0.5 x 0.8

= 0.42

(Do the same to fill all cells)

We are “off” by 1.58

	I1	I2	I3	I4	I5
User 1	0.42 2	1.85 1	4	1	2
User 2	1	4	1	2	2
User 3	3	5	5	3	4
User 4	2	3	?	2	3

Matrix Factorization: Gradient Descent

		F1	F2
User 1		0.2	0.5
User 2		0.4	0.6
User 3		0.1	0.05
User 4		0.9	0.3

Start with a rough guess

	I1	I2	I3	I4	I5
F1	0.1	1.5	0.9	3.4	1.2
F2	0.8	3.1	2.5	4.5	0.6

0.2 x 0.1 + 0.5 x 0.8
= 0.42

	I1	I2	I3	I4	I5
User 1	0.42 2	1.85 1	4	1	2
User 2	1	4	1	2	2
User 3	3	5	5	3	4
User 4	2	3	?	2	3

Error = $(2 - 0.42)^2$
$+ (1 - 1.85)^2$
+ …

Matrix Factorization: Gradient Descent

		F1	F2
User 1		0.14	0.45
User 2		0.32	0.5
User 3		0.43	0.07
User 4		0.8	0.3

Let’s try this again with different values

$$0.14 \times 0.2 + 0.45 \times 1.1 = 0.52$$

Keep repeating until error can be minimized

	I1	I2	I3	I4	I5
F1	0.2	1.7	0.8	3.7	1.9
F2	1.1	3.3	3.2	4.0	0.5

	I1	I2	I3	I4	I5
User 1	0.52 2	1.72 1	4	1	2
User 2	1	4	1	2	2
User 3	3	5	5	3	4
User 4	2	3	?	2	3

Matrix Factorization: Finding Factors

(k x n)

	I1	I2	I3	I4	I5
F1	2	1	4	1	2
F2	1	4	1	2	2

		F1	F2
User 1		1	0
User 2		0	1
User 3		1	1
User 4		1	0.5

(m x k)

To find these values we can also use:

- ***Genetic Algorithm***
- ***Linear Programming***
- ***PSO***

any other optimization technique

Matrix Factorization: Pros and Cons

Pros

- It can take much less storage e.g. a **2000x1000** interaction matrix can be stored as two matrices of **2000x100** and **100x1000** dimensions
  Storage taken by 2000x1000 matrix **= 2M**
  Storage taken by separate matrices **=** ***200k + 100k i.e., 300k***
- Predictions can be calculated quickly and easily

Cons

- The cost of optimization during training is non-deterministic