Matrix Factorization and Completion

Archer Yang

McGill University

October 8, 2024

- References [1][2]
- **Reading assignment:** Probabilistic Machine Learning: An Introduction (PML) Chapter 22

[1] https://developers.google.com/machine-learning/recommendation/collaborative/basics
[2] CSC 311, University of Toronto


NETFLIX
2009
DATE 09.21.09
PAY TO THE ORDER OF: BellKor's Pragmatic Chaos
$ 1,000,000 00
AMOUNT: ONE MILLION 00/100
FOR The Netflix Prize
Reed Hastings
The Netflix Prize

Netflix Prize

- In 2006, Netflix released a large dataset of 100,480,507 movie ratings (on a scale of 1 to 5) from 480,189 users of 17,770 movies.
- The ratings matrix is 99% sparse (unknown).
- A prize of $1M, known as the **Netflix Prize.**
- The prize claimed on September 21, 2009 by a team known as "BellKor's Pragmatic Chaos"
- They proposed an ensemble of different methods[3]. We describe a key component in their ensemble.

[3] https://www.jstor.org/stable/41714795

MATRIX FACTORIZATION

A Movie Recommendation Example

Consider a feedback data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times p}$:

- N rows, each row represents a user.
- p columns, each column represents a movie.
- Binary feedback

$$x_{ij} = [\boldsymbol{X}]_{ij} = \begin{cases} 1 & \text{user } i \text{ interested in movie } j \\ 0 & \text{otherwise} \end{cases}$$

for $i = 1, \cdots, N$ and $j = 1, \cdots, p$.

A Movie Recommendation Example

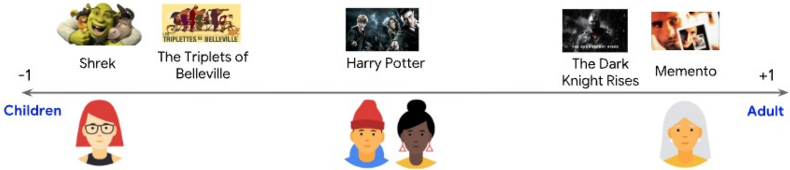
	 Harry Potter	 The Triplets of Belleville	 Shrek	 The Dark Knight Rises	 Memento
	✓		✓	✓	
		✓			✓
	✓	✓	✓		
				✓	✓

1D feature

- In a simplest example, the user feedback is explained by the product of **1D** user feature and **1D** movie feature

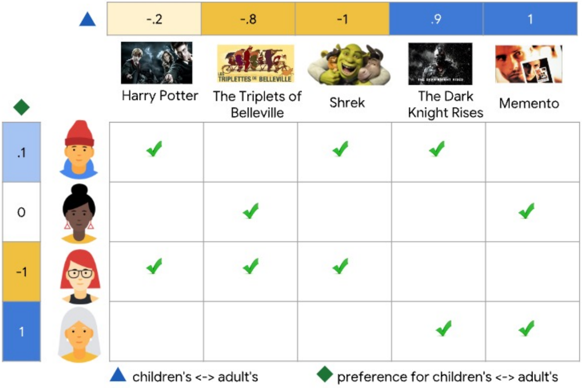
$$x_{ij} \approx u_{1i} v_{1j}$$

- **User feature:** $u_{1i} \in [-1, 1]$ describes **user** i's interest in children's movies (closer to -1) or adult movies (closer to +1).
- **Movie feature:** $v_{1j} \in [-1, 1]$ describes whether **movie** j is for children (closer to -1) or adults (closer to +1).



In a simplest example, the user feedback is approximated by the product of **1D** user feature and **1D** movie feature

$$x_{ij} \approx u_{1i} v_{1j}$$



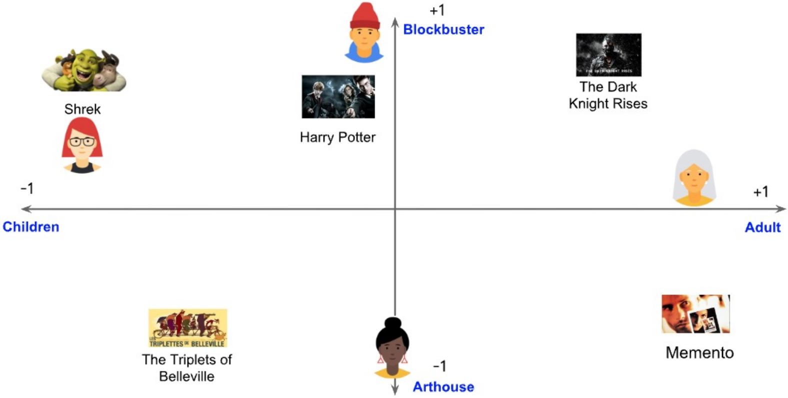
However 1D feature might not be sufficient enough to explain users' preferences. e.g. the first and second users' preferences

2D feature

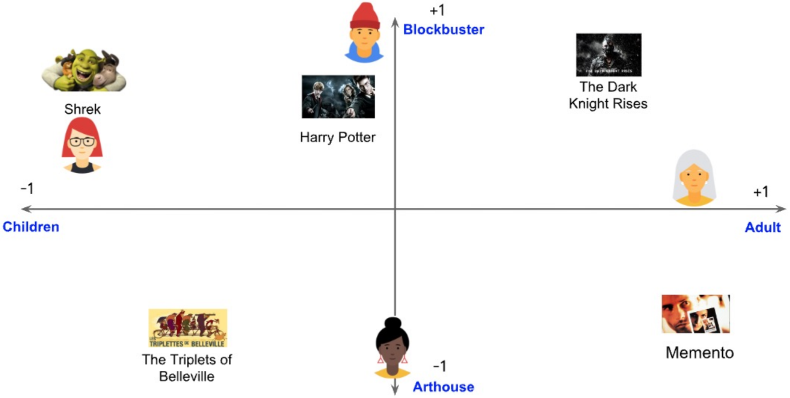
- If one feature was not enough, let's add a second one!
- **Addl. user feature:** $u_{2i} \in [-1, 1]$ describes **user** i's interest in arthouse movies (closer to -1) or blockbuster movies (closer to +1).
- **Addl. movie feature:** $v_{2j} \in [-1, 1]$ describes whether **movie** j is arthouse (closer to -1) or blockbuster (closer to +1).

With a second feature, the user and movie feature are two dimensional

- User i: column vector $U_i = (u_{1i}, u_{2i})^\top$
- Movie j: column vector $V_j = (v_{1j}, v_{2j})^\top$

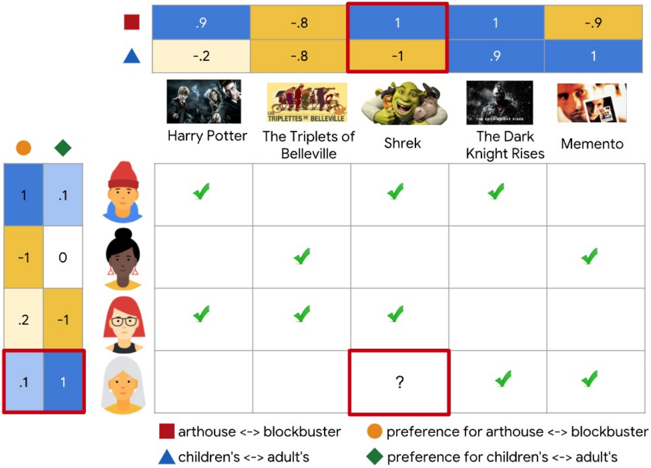


- Features of users with similar preferences will be close together.
- Features of movies liked by similar users will be close in the feature space.



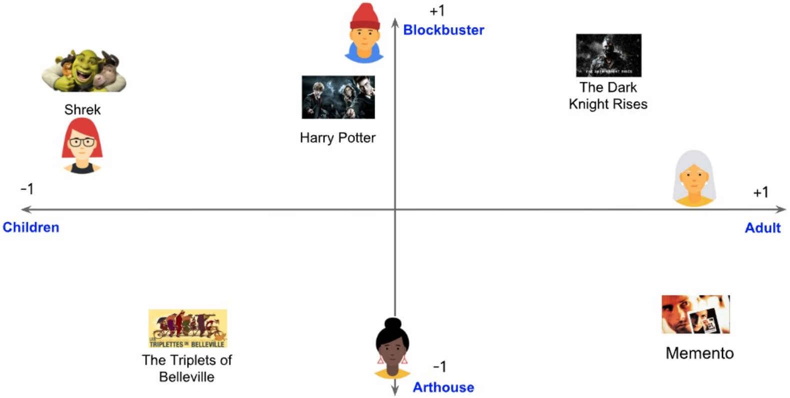
The user feedback is explained by **2D** user feature and movie feature

$$x_{ij} = [\boldsymbol{X}]_{ij} \approx u_{1i}v_{1j} + u_{2i}v_{2j} = U_i^\top V_j$$



Inner product $U_i^\top V_j$ measures similarity

- Inner product $U_i^\top V_j = u_{1i}v_{1j} + u_{2i}v_{2j}$ measures similarity between U_i and V_j, measures how much user i likes movie j
- Users and their liked movies will be close in the feature space.



Matrix factorization

- In this example, we hand-engineered the features. In practice, the latent features can be **learned automatically**.
- Matrix factorization solves

$$\min_{\{U_i, V_j\}} \sum_{(i,j)} (x_{ij} - U_i^\top V_j)^2 \tag{1}$$

Ex. Collaborative Filtering

Collaborative filtering:

- Uses similarities between users and items simultaneously to provide recommendations.
- Recommend an item to **user A** based on the interests of a similar **user B**.
- The embeddings can be learned automatically, without relying on hand-engineering of features.

Connection to rank-k matrix approximation

- The matrix factorization problem (with k-dimensional feature)

$$\min_{\{U_i, V_j\}} \sum_{(i,j)} (x_{ij} - U_i^\top V_j)^2$$

can also be written as

$$\min_{\boldsymbol{U}, \boldsymbol{V}} \|\boldsymbol{X} - \boldsymbol{U}\boldsymbol{V}^\top\|_F^2 \qquad \Longrightarrow \qquad \boldsymbol{X} \approx \boldsymbol{U}\boldsymbol{V}^\top \tag{2}$$

where

$$\boldsymbol{U} = \begin{pmatrix} U_1 & \cdots & U_N \end{pmatrix}^\top \in \mathbb{R}^{N \times k} \qquad \boldsymbol{V}^\top = \begin{pmatrix} V_1 & \cdots & V_p \end{pmatrix} \in \mathbb{R}^{k \times p}$$

Connection to rank-k matrix approximation

- One the other hand, the rank-k matrix approximation of $\boldsymbol{X}$ considers

$$\widehat{\boldsymbol{X}}(k) = \underset{\text{rank}(\boldsymbol{C})=k}{\arg\min} \|\boldsymbol{X} - \boldsymbol{C}\|_F^2 \qquad \Longrightarrow \qquad \boldsymbol{X} \approx \widehat{\boldsymbol{X}}(k) = \boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{V}_k^\top \tag{3}$$

- Unlike $\boldsymbol{U}_k$ and $\boldsymbol{V}_k$ in (3), $\boldsymbol{U}$ and $\boldsymbol{V}$ in (2) are not necessarily orthonormal.

Matrix completion via factorization

Matrix completion via factorization

- Sometimes, some entries of the matrix are missing, matrix completion can **predict** the missing values using the observed ones.
- Let $O = \{(i, j) : \text{entry } (i, j) \text{ of matrix } \boldsymbol{X} \text{ is observed}\}$
- Matrix completion solves

$$\min_{U_i, V_j : (i,j) \in O} \sum_{(i,j) \in O} (x_{ij} - U_i^\top V_j)^2$$

Matrix completion via factorization

- The dimension of U_i and V_j can be generalized to k in

$$\min_{U_i, V_j : (i,j) \in O} \sum_{(i,j) \in O} (x_{ij} - U_i^\top V_j)^2$$

  with $U_i = (u_{1i}, u_{2i}, \cdots, u_{ki})^\top$ and $V_j = (v_{1j}, v_{2j}, \cdots, v_{kj})^\top$
- The objective is non-convex in U_i and V_j jointly, and hard to find the global minimum.
- However, as a function of either U_i and V_j individually, the problem is convex and easy to optimize.

Alternating Least Squares

- Use alternating minimization to solve

$$\min_{U_i, V_j : (i,j) \in O} \sum_{(i,j) \in O} (x_{ij} - U_i^\top V_j)^2$$

- Alternating Least Squares (ALS): fix V_j and optimize U_i, followed by fix U_i and optimize V_j, and so on until convergence.

Alternating Least Squares

- Decompose the cost into a sum of independent terms:

only depends on U_i, fixing V_j

$$\sum_{(i,j)\in O} (x_{ij} - U_i^\top V_j)^2 = \sum_{i:(i,j)\in O} \boxed{\sum_{j:(i,j)\in O} (x_{ij} - U_i^\top V_j)^2} \qquad (4)$$

only depends on V_j, fixing U_i

$$= \sum_{j:(i,j)\in O} \boxed{\sum_{i:(i,j)\in O} (x_{ij} - U_i^\top V_j)^2} \qquad (5)$$

- Assume values of V_j are fixed (known), (4) can be minimized independently for U_i for each observed row $i : (i,j) \in O$

$$\min_{U_i:(i,j)\in O} \sum_{(i,j)\in O} (x_{ij} - U_i^\top V_j)^2 = \min_{U_i:(i,j)\in O} \sum_{i:(i,j)\in O} \sum_{j:(i,j)\in O} (x_{ij} - U_i^\top V_j)^2$$

$$= \sum_{i:(i,j)\in O} \min_{U_i} \sum_{j:(i,j)\in O} (x_{ij} - U_i^\top V_j)^2$$

Alternating Least Squares

- This can be minimized independently for U_i for each observed row $i : (i, j) \in O$

$$U_i^+ = \arg\min_{U_i} \sum_{j:(i,j)\in O} (x_{ij} - U_i^\top V_j)^2$$

- This is essentially a linear regression problem. Its optimal solution is:

$$U_i^+ = \left(\sum_{j:(i,j)\in O} V_j V_j^\top \right)^{-1} \sum_{j:(i,j)\in O} x_{ij} V_j \qquad (6)$$

Proof of (6)

Compute the derivative of the objective function in (6) and set it to zero

$$\frac{\partial}{\partial U_i^\top} \sum_{j:(i,j)\in O} (x_{ij} - U_i^\top V_j)^2 = -2 \sum_{j:(i,j)\in O} V_j (x_{ij} - U_i^\top V_j)$$

$$= -2 \sum_{j:(i,j)\in O} V_j (x_{ij} - V_j^\top U_i) = 0$$

which gives the equation

$$\sum_{j:(i,j)\in O} V_j x_{ij} = \sum_{j:(i,j)\in O} V_j V_j^\top U_i$$

Therefore the solution of (6) is

$$U_i = \left(\sum_{j:(i,j)\in O} V_j V_j^\top \right)^{-1} \sum_{j:(i,j)\in O} x_{ij} V_j$$

Alternating Least Squares

- Similarly, assuming values of U_i are known, (5) can be minimized independently for V_j for each observed column $j : (i, j) \in O$

$$V_j^+ = \arg\min_{V_j} \sum_{i:(i,j)\in O} (x_{ij} - U_i^\top V_j)^2$$

- Its optimal solution is:

$$V_j^+ = \left(\sum_{i:(i,j)\in O} U_i U_i^\top \right)^{-1} \sum_{i:(i,j)\in O} x_{ij} U_i$$

ALS for matrix completion problem

1. Initialize U_i and V_j randomly, for $(i,j) \in O$
2. Repeat step 3 and 4 until convergence
3. **for** $i = 1, \cdots, N$ **do**

$$U_i = \left(\sum_{j:(i,j)\in O} V_j V_j^\top \right)^{-1} \sum_{j:(i,j)\in O} x_{ij} V_j$$

4. **for** $j = 1, \cdots, p$ **do**

$$V_j = \left(\sum_{i:(i,j)\in O} U_i U_i^\top \right)^{-1} \sum_{i:(i,j)\in O} x_{ij} U_i$$

Matrix Completion in High-dimension

Matrix completion in high-dimension

- There might be an overfitting issue when the dimensions of U_i and V_j are very high. To overcome overfitting, we consider the regularized problem

  $$\min_{U_i, V_j : (i,j) \in O} \sum_{(i,j) \in O} (x_{ij} - U_i^\top V_j)^2 + \lambda \sum_{i:(i,j) \in O} \|U_i\|^2 + \lambda \sum_{j:(i,j) \in O} \|V_j\|^2$$

  with $U_i = (u_{1i}, u_{2i}, \cdots, u_{ki})^\top$ and $V_j = (v_{1j}, v_{2j}, \cdots, v_{kj})^\top$.
- Here $\lambda > 0$ is a tuning parameter for controlling strength of regularization.
- We use Alternating Least Squares (ALS): fix V_j and optimize U_i, followed by fix U_i and optimize V_j, and so on until convergence.

Alternating Least Squares

- This can be minimized independently for U_i for each observed row $i : (i, j) \in O$

$$U_i^+ = \arg\min_{U_i} \sum_{j:(i,j)\in O} (x_{ij} - U_i^\top V_j)^2 + \lambda ||U_i||^2$$

- This is essentially a linear regression problem. Its optimal solution is:

$$U_i^+ = \left(\sum_{j:(i,j)\in O} V_j V_j^\top + \lambda \boldsymbol{I}_k \right)^{-1} \sum_{j:(i,j)\in O} x_{ij} V_j \tag{7}$$

where $\boldsymbol{I}_k$ is a $k \times k$ identity matrix. Here k is the dimension of U_i and V_j.

Proof of (7)

Compute the derivative of the objective function in (7) and set it to zero

$$
\begin{aligned}
& \frac{\partial}{\partial U_i^\top} \sum_{j:(i,j)\in O} (x_{ij} - U_i^\top V_j)^2 + \lambda \|U_i\|^2 \\
= \quad & -2 \sum_{j:(i,j)\in O} V_j (x_{ij} - V_j^\top U_i) + 2\lambda U_i = 0
\end{aligned}
$$

which gives the equation

$$
\sum_{j:(i,j)\in O} V_j V_j^\top U_i + \lambda U_i = \left(\sum_{j:(i,j)\in O} V_j V_j^\top + \lambda \boldsymbol{I}_k \right) U_i = \sum_{j:(i,j)\in O} V_j x_{ij}
$$

Therefore the solution of (6) is

$$
U_i = \left(\sum_{j:(i,j)\in O} V_j V_j^\top + \lambda \boldsymbol{I}_k \right)^{-1} \sum_{j:(i,j)\in O} x_{ij} V_j
$$

Alternating Least Squares

- Similarly, assuming values of U_i are known, (5) can be minimized independently for V_j for each observed column $j : (i, j) \in O$

$$V_j^{+} = \arg\min_{V_j} \sum_{i:(i,j)\in O} (x_{ij} - U_i^{\top} V_j)^2 + \lambda \sum_{j:(i,j)\in O} \|V_j\|^2$$

- Its optimal solution is:

$$V_j^{+} = \left(\sum_{i:(i,j)\in O} U_i U_i^{\top} + \lambda \boldsymbol{I}_k \right)^{-1} \sum_{i:(i,j)\in O} x_{ij} U_i$$

ALS for high-dimensional matrix completion problem

1. Initialize U_i and V_j randomly, for $(i, j) \in O$
2. Repeat step 3 and 4 until convergence
3. **for** $i = 1, \cdots, N$ **do**

$$U_i = \left(\sum_{j:(i,j)\in O} V_j V_j^\top + \lambda \boldsymbol{I}_k \right)^{-1} \sum_{j:(i,j)\in O} x_{ij} V_j$$

4. **for** $j = 1, \cdots, p$ **do**

$$V_j = \left(\sum_{i:(i,j)\in O} U_i U_i^\top + \lambda \boldsymbol{I}_k \right)^{-1} \sum_{i:(i,j)\in O} x_{ij} U_i$$

Connection to Matrix Factorization

Connection to matrix factorization

- The matrix completion problem minimizes

$$\sum_{(i,j)\in O} (x_{ij} - U_i^\top V_j)^2$$

- On the other hand, the matrix factorization problem minimizes

$$\sum_{(i,j)} (x_{ij} - U_i^\top V_j)^2 = \sum_{(i,j)\in O} (x_{ij} - U_i^\top V_j)^2 + \sum_{(i,j)\notin O} (x_{ij} - U_i^\top V_j)^2 \quad (8)$$

Case Study: Netflix Prize

BellKor's approach

The team proposed an approximate $\tilde{x}_{ij}$ that also allows for **user-specific** and **item-specific baselines**

$$x_{ij} \approx \tilde{x}_{ij} = \mu + b_i + c_j + U_i^\top V_j$$

This can capture:

- Some users might always tend to give low ratings and others may give high ratings;
- Some items (e.g., very popular movies) might have unusually high ratings.

BellKor's approach

To avoid overfitting, they also added ℓ_2 regularization to the parameters to get the objective

$$L = \sum_{(i,j)\in O} [x_{ij} - \tilde{x}_{ij}]^2 + \lambda(b_i^2 + c_j^2 + ||U_i||^2 + ||V_j||^2)$$

The resulting optimization problem is

$$\min_{\mu, b_i, c_j, U_i, V_j : (i,j)\in O} L \tag{9}$$

Algorithm[4]

They solve problem (9) by use stochastic gradient descent (SGD).

1. Initialize μ, b_i, c_j, U_i, V_j randomly for $(i, j) \in O$
2. Repeat step 3 – 8 until convergence
3. Randomly sample an entry $(i, j) \in O$.
4. $b_i = b_i + \eta(e_{ij} - \lambda b_i)$
5. $c_j = c_j + \eta(e_{ij} - \lambda c_j)$
6. $U_i = U_i + \eta(e_{ij} V_j - \lambda U_i)$
7. $V_j = V_j + \eta(e_{ij} U_j - \lambda V_j)$
8. $\mu = \sum_{(i,j)\in O}(x_{ij} - (b_i + c_j + U_i^\top V_j)) / \sum_{(i,j)\in O} 1$

where $e_{ij} = x_{ij} - \tilde{x}_{ij}$ is the error term, and $\eta > 0$ is the learning rate.

[4] https://sifter.org/~simon/journal/20061211.html