Recommender Systems

Authors: Felipe Perez, Ella Chen, Serena McDonnell, and Werner Chao

Nov 2019

aggregate intellect

Who are we?

Why are we doing this?

What we will learn in these 9 hours?

- What are recommender systems?
- Types of recommender systems?
- Evaluation of recommender systems.
- Ranking.
- Deep learning on recommender systems?

Hands on sessions

- Small coding questions to improve skills.
- Understanding the pain points, that is from theory to practice.
- Need of efficient and clever coding.
- Ideas for scaling.

What and why of recommender systems?

User:

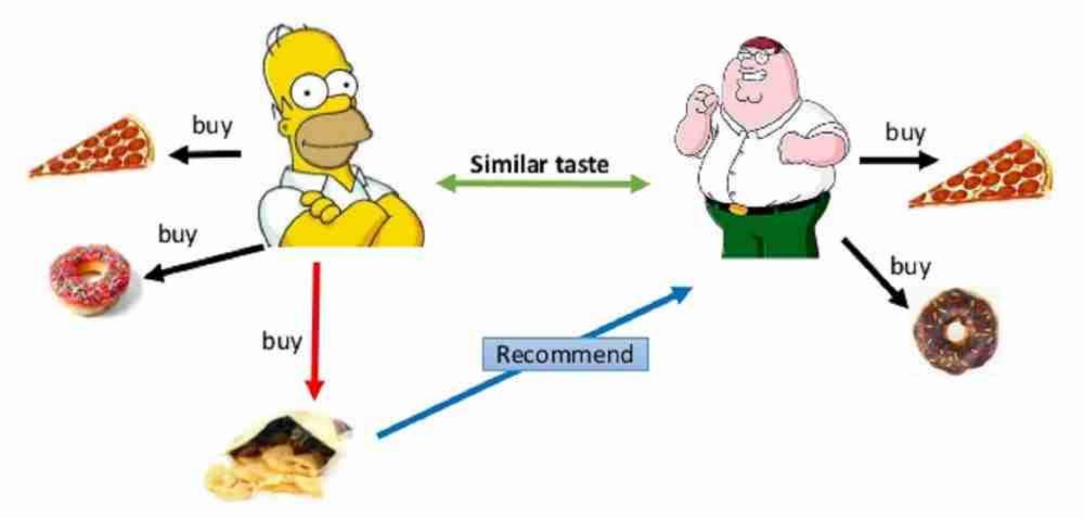
- Understanding user behaviour.
- Predicting desires and patterns.
- Understanding populations needs.

Business:

- Recommend products.
- Personalize.
- Keep users engaged.

Collaborative filtering

1. Identify similar users.
2. Use one customer behaviour to make recommendation that are relevant to the other.



Memory Based Collaborative filtering

- No parameters are found.
- Notion of similar users is needed.

Our setting is a user x item matrix collecting ratings that each user has given to some items.

$$\begin{pmatrix} ? & \cdots & r_{1i} & \cdots & \cdots & ? & r_{1m} \\ r_{21} & \cdots & ? & \cdots & r_{2j} & \cdots & ? \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ r_{u1} & \cdots & r_{ui} & \cdots & ? & \cdots & ? \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Rating of user u to item i

Memory Based Collaborative filtering

We represent an user by the ratings he has done

$$\text{user } u = \begin{pmatrix} ? & \cdots & r_{u'i} & \cdots & ? & \cdots & ? \end{pmatrix}$$

$$\text{user } u' = \begin{pmatrix} ? & \cdots & r_{u'i} & \cdots & r_{u'j} & \cdots & ? \end{pmatrix}$$

Memory Based Collaborative filtering

And create a similarity of users, among many options the most basic one is:

$$d(u, u') = \frac{\sum_i r_{ui} r_{u'i}}{\sqrt{\sum_i r_{ui}^2} \sqrt{\sum_i r_{u'i}^2}}$$

Memory Based Collaborative filtering (inference)

Given user $u = \begin{pmatrix} ? & \cdots & r_{u'i} & \cdots & ? & \cdots & ? \end{pmatrix}$

We find the closest k users and infer via:

$$\hat{r}_{ui} = \frac{\sum_{u'} d(u, u') r_{u'i}}{\sum_{u'} |d(u, u')|}$$

Collaborative filtering (Evaluation)

Erase the rankings on some users.

$$\left(r_{u1}, r_{u2}, r_{u3}, \ldots, r_{u(m-1)}, r_{um}\right)$$

$$\left(r_{u1}, ?, ?, \ldots, r_{u(m-1)}, ?\right)$$

Collaborative filtering (Evaluation)

Use a measure of distance between the prediction and the ground truth

$$MSE = \frac{1}{N}\sum_i (r_{ui} - r'_{ui})^2$$

Where r_{ui} is the ground truth and r'_{ui} is the prediction.

Hands on:

Go over the first notebook

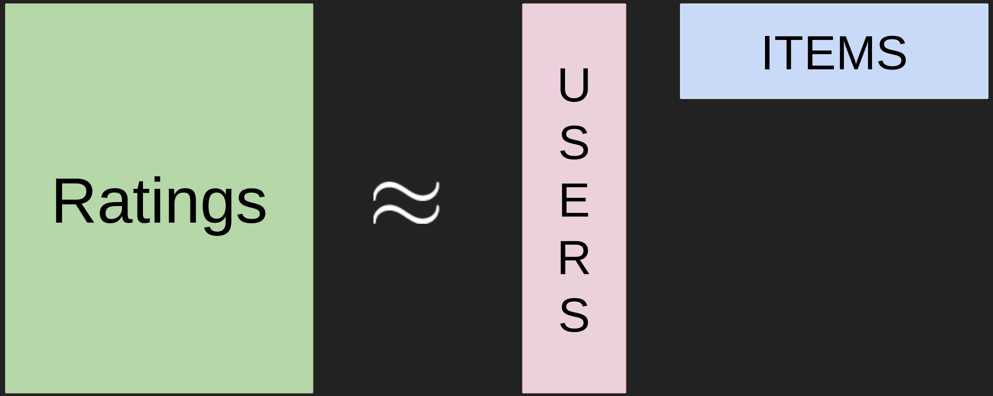
Where to go next?

1. Reducing bias.
2. Cold start problem.
3. Parallelizing.
4. Simple approaches.

Model based Collaborative filtering

Matrix Factorization

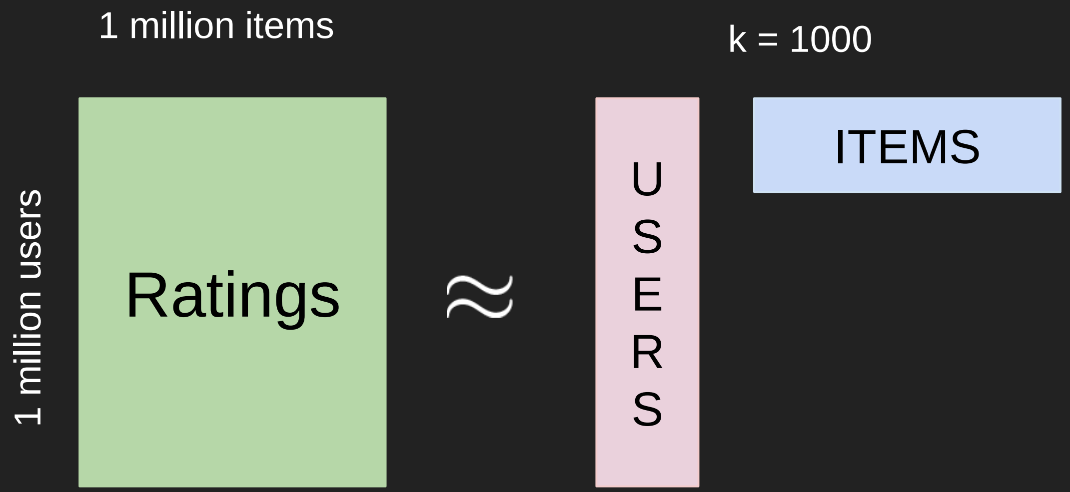
Representing the user item matrix



$$R \approx UV^T \text{ where } R \in \mathbb{R}^{n\times m}, U \in \mathbb{R}^{n\times k}, V \in \mathbb{R}^{n\times k}$$

Matrix factorization

Ratings matrices are usually sparse and extremely large.



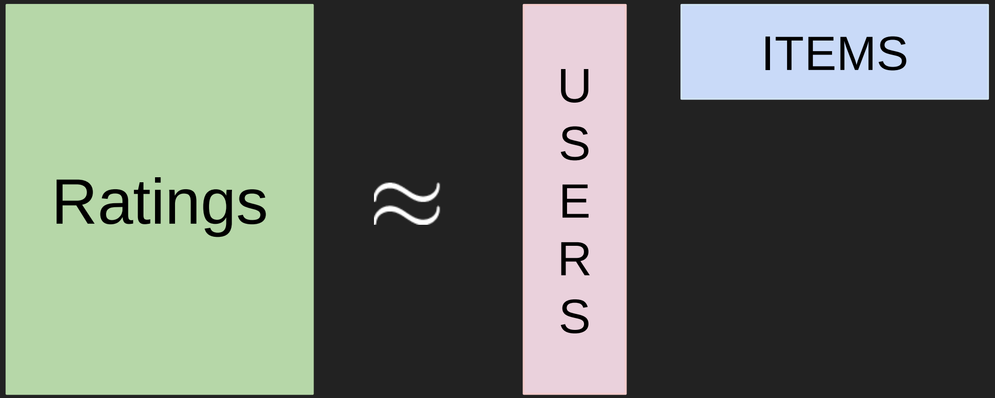
$10^{12}\text{entries} = 4TB$

$2 * 10^{9}\text{entries} = 8GB$

Matrix factorization

VANILLA

$$\text{Loss} = \sum_{(i,j)\in I} \left(r_{ij} - U_i V_j^T\right)^2$$



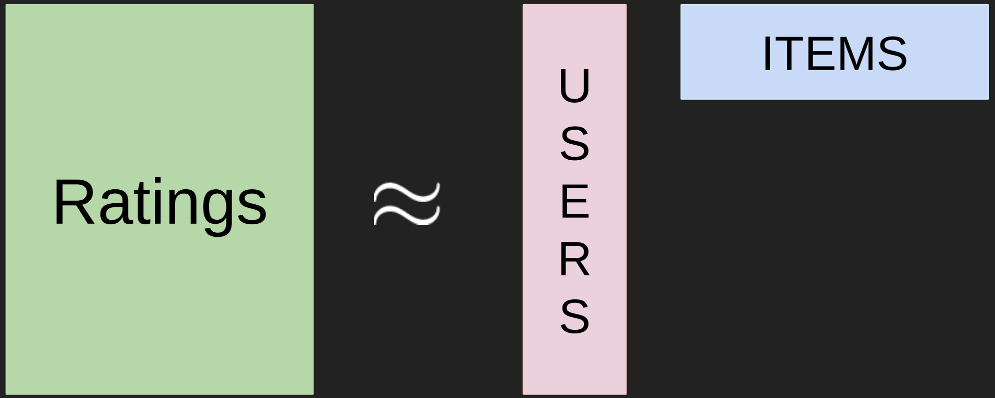
$R \approx UV^T$ where $R \in \mathbb{R}^{n\times m}, U \in \mathbb{R}^{n\times k}, V \in \mathbb{R}^{m\times k}$

Matrix factorization

REGULARIZED

$$\text{Loss} = \sum_{(i,j)\in I} \left(r_{ij} - U_i V_j^T\right)^2$$

$$+\lambda_1 \sum_i \|U_i\|^2 + \lambda_2 \sum_j \|V_j\|^2$$



$R \approx UV^T$ where $R \in \mathbb{R}^{n\times m}, U \in \mathbb{R}^{n\times k}, V \in \mathbb{R}^{m\times k}$

Alternating least squares

- Alternate on loss function
- Can be parallelized
- Great performance
- Many variations

Loss

$$L(U,V) =$$

$$\sum_{i,j}\left(r_{ij} - U_i V_j^T\right)^2 + \lambda_1 \sum_i \|U_i\|^2 + \lambda_2 \sum_j \|V_j\|^2$$

Alternating least squares

- Alternate on loss function
- Can be parallelized
- Great performance
- Many variations

Minimize for U

$$L(U, V) =$$

$$\sum_{i,j} \left(r_{ij} - U_i V_j^T\right)^2 + \lambda_1 \sum_i \|U_i\|^2 + \lambda_2 \sum_j \|V_j\|^2$$

Alternating least squares

- Alternate on loss function
- Can be parallelized
- Great performance
- Many variations

Minimize for V

$$L(U, V) =$$

$$\sum_{i,j} \left(r_{ij} - U_i V_j{}^T\right)^2 + \lambda_1 \sum_i \|U_i\|^2 + \lambda_2 \sum_j \|V_j\|^2$$

Alternating least squares

$$L(U,V) = \sum_{(i,j)\in I} \left(r_{ij} - U_i V_j^T\right)^2 + \lambda_1 \sum_i \|U_i\|^2 + \lambda_2 \sum_j \|V_j\|^2$$

We want to minimize this!

Alternating least squares

$$L(U,V) = \sum_{(i,j)\in I} \left(r_{ij} - U_i V_j^T\right)^2 + \lambda_1 \sum_i \|U_i\|^2 + \lambda_2 \sum_j \|V_j\|^2$$

$$\frac{1}{2}\frac{\partial L}{\partial U_{is}} = 0, \forall i, s$$

So we look for optimal via gradient equals zero!

Alternating least squares

$$L(U,V) = \sum_{(i,j)\in I} \left(r_{ij} - U_i V_j^T\right)^2 + \lambda_1 \sum_i \|U_i\|^2 + \lambda_2 \sum_j \|V_j\|^2$$

$$\frac{1}{2}\frac{\partial L}{\partial U_{is}} = 0, \forall i, s$$

$$-\sum_{j\in I_i} \left(r_{ij} - U_i V_j^T\right) V_{js} + \lambda_1 U_{is} = 0$$

The equation then becomes!

Alternating least squares

$$L(U,V) = \sum_{(i,j)\in I} \left(r_{ij} - U_i V_j^T\right)^2 + \lambda_1 \sum_i \|U_i\|^2 + \lambda_2 \sum_j \|V_j\|^2$$

$$\frac{1}{2}\frac{\partial L}{\partial U_{is}} = 0, \forall i, s$$

$$-\sum_{j\in I_i} \left(r_{ij} - U_i V_j^T\right) V_{js} + \lambda_1 U_{is} = 0$$

$$\sum_{j\in I_i} U_i V_j^T V_{js} + \lambda_1 U_{is} = \sum_{j\in I_i} r_{ij} V_{js}$$

We reorganize since we want to solve for U

Alternating least squares

$$L(U,V) = \sum_{(i,j)\in I} \left(r_{ij} - U_i V_j^T\right)^2 + \lambda_1 \sum_i \|U_i\|^2 + \lambda_2 \sum_j \|V_j\|^2$$

$$\frac{1}{2}\frac{\partial L}{\partial U_{is}} = 0, \forall i, s$$

$$-\sum_{j\in I_i} \left(r_{ij} - U_i V_j^T\right) V_{js} + \lambda_1 U_{is} = 0$$

$$\sum_{j\in I_i} U_i V_j^T V_{js} + \lambda_1 U_{is} = \sum_{j\in I_i} r_{ij} V_{js}$$

$$U_i \left(V_{I_i}^T V_{I_i} + \lambda_1 \mathbb{I}\right) = R(i, I_i) V_{I_i}$$

Which vectorizing becomes

Alternating least squares

$$L(U,V) = \sum_{(i,j)\in I} \left(r_{ij} - U_i V_j^T\right)^2 + \lambda_1 \sum_i \|U_i\|^2 + \lambda_2 \sum_j \|V_j\|^2$$

$$\frac{1}{2}\frac{\partial L}{\partial U_{is}} = 0, \forall i, s$$

$$-\sum_{j\in I_i} \left(r_{ij} - U_i V_j^T\right) V_{js} + \lambda_1 U_{is} = 0$$

$$\sum_{j\in I_i} U_i V_j^T V_{js} + \lambda_1 U_{is} = \sum_{j\in I_i} r_{ij} V_{js}$$

$$U_i \left(V_{I_i}^T V_{I_i} + \lambda_1 \mathbb{I}\right) = R(i, I_i) V_{I_i}$$

$$\boxed{U_i = R(i, I_i) V_{I_i} \left(V_{I_i}^T V_{I_i} + \lambda_1 \mathbb{I}\right)^{-1}}$$

Solving for U we obtain the update rule!

Hands on exercise

1. Implement ALS.
2. Use a linear solver for ALS.
3. Use gradient descent.
4. Compare the results.
5. Can you use gradient descent from the beginning? Try it.
6. (Optional) Try this is with the larger dataset
7. (Optional) Find the best possible k