Lecture 3 - Recommender Systems

Advanced Machine Learning

Miroslav Čepek, Zdeněk Buk, **Rodrigo da Silva Alves**, Vojtěch Rybář, Petr Šimánek

FIT CTU

9. 3. 2023

Recommender Systems

- Recommenders recommend:
  - Items to users (most common)
  - Users to Items
  - Items to Items
  - Users to Users
- Items are movies, products, news, music, books, recipes, ...

Working in pairs: try to find one example of each four recommender scenarios above.

Recommender Systems

- WLOG, we will focus on recommend **users** relevant **items**
  - Predictive modelling: predict the rate of item m by the user u
  - Retrieve modelling: learning a rank systems
- Typically based on past **interactions** and (or) **attributes** (from users and items)
- **Interactions:** normally modelled as interaction matrix
  - Explicit: a user rate a song with 4 stars in a scale from 0 to 5
  - Implicit: a user watched 80% of a movie
- **Attributes:** normally modelled as attribute matrices
  - Users: gender, age, location, $\cdots$
  - Items: text, video, meta-data, $\cdots$

Modelling Interactions: Explicit feedback

	m_1	m_2	...	m_n
u_1	?	2	...	3
u_2	5	1	...	?
u_3	?	3	...	1
...	...	...	...	...
u_m	4	4	...	?

Modelling Interactions: Implicit feedback

u_1	m_1	12/11/2021 09:01:21	Watch	25%
u_2	m_1	17/03/2021 14:27:09	Clicked	
u_2	m_4	17/03/2021 14:22:09	Clicked	Purchase
...	...	...	...	...
u_m	m_n	14/06/2020 23:14:46	Watch	100%

	m_1	m_2	...	m_n
u_1	1	0	...	0
u_2	1	0	...	0
u_3	0	1	...	1
...	...	...	...	...
u_m	0	1	...	1

From explicit to implicit feedback

	m_1	m_2	...	m_n
u_1	?	2	...	3
u_2	5	1	...	?
u_3	?	3	...	1
...	...	...	...	...
u_m	4	4	...	?

	m_1	m_2	...	m_n
u_1	?	0	...	0
u_2	1	0	...	?
u_3	?	0	...	0
...	...	...	...	...
u_m	1	1	...	?

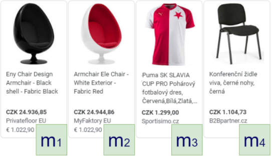
From implicit to explicit feedback

u_1	m_1	12/11/2021 09:01:21	Watch	25%
u_2	m_1	17/03/2021 14:27:09	Clicked	
u_2	m_4	17/03/2021 14:22:09	Clicked	Purchase
...	...	...	...	...
u_m	m_n	14/06/2020 23:14:46	Watch	100%

	m_1	m_2	...	m_n
u_1	0	0	...	0
u_2	0	0	...	0
u_3	0	1	...	1
...	...	...	...	...
u_m	0	0	...	1

Modelling Attributes

	Color	Price	Category
m_1	Black	24936	Chair
m_2	Red and White	24944	Chair
m_3	Red and White	1299	T-Shirt
m_4	Black	1104	Chair



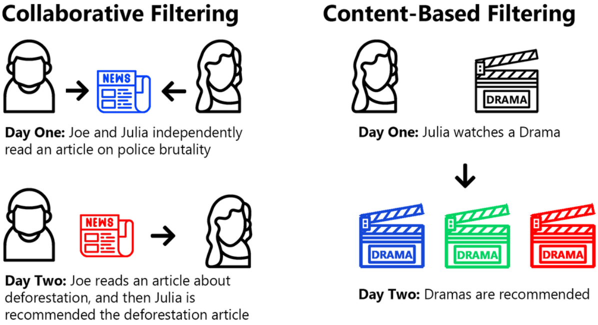
Personalised Machine Learning

- Personalisation *is not* a simple regression or classification problem
- A personalised model implies that: if the user have different interactions (or attribute) the recommendation should be different
- Suppose the vector a_u (a_m) are attribute vectors of user u (item m)
- We can use linear regression to predict how users u will like item m

$$r_{ui} = \omega^\top \times \begin{bmatrix} a_u \\ a_m \end{bmatrix}$$

- Is linear regression a personalised model for recommenders? No!

Recommendation Algorithms



Recommender as a Matrix

- As we saw, we can model recommenders as matrices.
- The ratings can be stored in a **ranking matrix** R of dimension $m \times n$ with elements from $\mathbb{R} \cup \{?\}$.
- An example of a rating matrix for $m = 4$ users and $n = 6$ items can read

$$R = \begin{pmatrix} 1 & ? & ? & 2 & ? & 1 \\ ? & 2 & 3 & ? & 2 & 1 \\ 1 & 5 & 5 & ? & ? & 5 \\ ? & ? & 2 & ? & ? & 3 \end{pmatrix}.$$

  Meaning, e.g., that user u_1 ranked items i_1 and i_6 with 1 star, item i_4 with 2 stars and had no interactions with items i_2, i_3 and i_4.
- **Our goal is to predict the unknown ratings** $r_{u,i} =?$ using the knowledge of the known ratings $r_{u,i} \neq ?$.

Idea of matrix factorization

- By **matrix factorization** we usually mean expressing a given matrix R as a matrix product of two (or more) matrices with some non-trivial properties. For example:

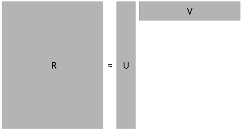
$$R = UV^\top$$

- These factorizations are a cornerstone of many algorithms and methods or are used to reach more numerically stable computations.

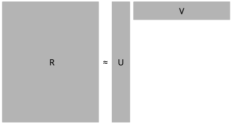
Do we need to know all the entries of a matrix R to factorize it, for example $R = UV^\top$?

Intuition of matrix factorization

- As for the recommendation systems, the inspiration comes mainly from the SVD as it can be used for constructing **latent features** or, in other words, **dimensionality reduction** using projections to lower dimensional space.
- The very basic idea of the lower dimensional approximation of an input matrix R of dimension $m \times n$ is based on this first-linear-algebra-lesson fact: Multiplying matrices U of dimension $m \times d$ and V of dimension $d \times n$ we get a matrix of dimension $m \times n$. This is true for any positive integer d.
- And this is the idea: **Given a rating matrix R, find lower dimensional matrices U and V so that the known elements of R are well approximated by the matrix $\mathbf{UV}^\top$.**



Matrix factorization for recommenders



- Let us denote:
  - The i-th row of U as u_i; the number of rows of U equals the number of users $|\mathcal{U}|$.
  - The j-th column of V as v_j; the number of columns of V equals the number of items $|\mathcal{I}|$.
  - Ω the subset of $\mathcal{U} \times \mathcal{I}$ of user-item pairs (i, j) such that $r_{i,j}$ is known, i.e., $r_{i,j} \neq ?$.
- The approximation of $r_{i,j}$ is given by the number $u_i^T v_j$, i.e., by the dot product of the two d-dimensional vectors.

Optmization Problem

- The **error of approximation** is usually measured by the squared residual:

$$(r_{i,j} - u_i^T v_j)^2.$$

- Hence, the matrices U and V are obtained by solving the optimization task:

$$\text{argmin}_{\mathbf{U},\mathbf{V}} \sum_{(i,j)\in\Omega} (r_{i,j} - u_i^T v_j)^2 + \lambda(\sum_x ||u_x||^2 + \sum_y ||v_y||^2).$$

Sparsity and prediction

- The matrices U and V are optimized only by considerung the known entries of R that are usually only a minority of entries.
- E.g. in the Netflix prize in 2006 there were $n = 17K$ movies and $m = 500K$ users, meaning that the matrix R had $8500M$ entries. But only $100M$ was given by Netflix!
- Still, the result of the matrix multiplication $UV^\top$ is a matrix having the same dimensions as R **with all entries known**!
- The unknown rating $r_{i,j} =?$ is estimated as $\hat{r}_{i,j} = u_i^T v_j$..

Example

- Consider our toy example matrix from above:

$$R = \begin{pmatrix} 1 & ? & ? & 2 & ? & 1 \\ ? & 2 & 3 & ? & 2 & 1 \\ 1 & 5 & 5 & ? & ? & 5 \\ ? & ? & 2 & ? & ? & 3 \end{pmatrix}.$$

- Assume that we chose the hyperparameter $d = 2$, i.e., we look for approximation matrices U and V with dimensions 4×2 and 2×6, respectively.
- Let us pretend that the matrices resulting from the optimization are

$$U = \begin{pmatrix} 0.3 & 0.7 \\ 0.3 & 0.5 \\ 0.2 & 0.4 \\ 0.2 & 0.1 \end{pmatrix} \quad \text{and} \quad V^\top = \begin{pmatrix} 1 & 10 & 11 & 10 & 4 & 20 \\ 1 & -1 & -2 & -1 & 1 & -4 \end{pmatrix}.$$

Example

- The resulting approximation is

$$\mathbf{UV}^\top = \begin{pmatrix} 0.3 & 0.7 \\ 0.3 & 0.5 \\ 0.2 & 0.4 \\ 0.2 & 0.1 \end{pmatrix} \begin{pmatrix} 1 & 10 & 11 & 10 & 4 & 20 \\ 1 & -1 & -2 & -1 & 1 & -4 \end{pmatrix} =$$

$$= \begin{pmatrix} 1 & \color{red}{2.3} & \color{red}{1.9} & 2.3 & \color{red}{1.9} & 3.2 \\ \color{red}{0.8} & 2.5 & 2.3 & \color{red}{2.5} & 1.7 & 4 \\ 0.6 & 1.6 & 1.4 & \color{red}{1.6} & \color{red}{1.2} & 2.4 \\ \color{red}{0.3} & \color{red}{1.9} & 2 & \color{red}{1.9} & \color{red}{0.9} & 3.6 \end{pmatrix},$$

where the red numbers are the desired predictions!

- E.g. the 3rd user predicted rating of the 4th item is $\hat{r}_{3,4} = 1.6$.

Supervised learning task

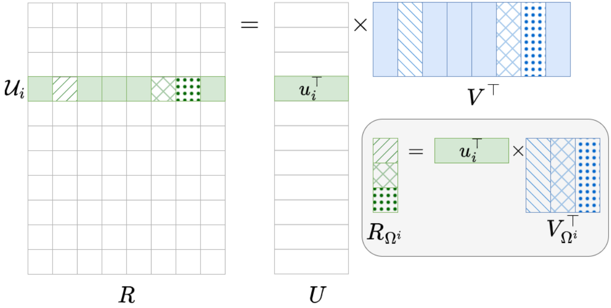
- The learning parameters: $U \in \mathbb{R}^{m\times d}$ and $V \in \mathbb{R}^{n\times d}$
- The hyperparameters:
  - the regularization constant $\lambda > 0$,
  - the matrix dimension d, which is a positive integer (significantly smaller than $\min\{m, n\}$).
- These hyperparameters can be tuned in the usual way via crossvalidation
- Therefore we would like to learn U and V, given d and λ by

$$\mathrm{argmin}_{\mathbf{U},\mathbf{V}} \sum_{(i,j)\in\Omega} (r_{i,j} - u_i^T v_j)^2 + \lambda(\sum_x ||u_x||^2 + \sum_y ||v_y||^2).$$

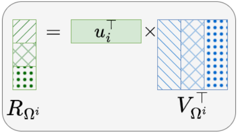
Alternating least squares (ALS)

- The idea of ALS is to fix alternately the matrix U and V. The non-fixed matrix is then considered learning variable and a subject to minimization.
- With one of the matrices fixed, the optimization problem becomes convex and very similar to the linear regression problem.
- Let's try ti understand how the mechanism works

Alternating least squares (ALS)



Alternating least squares (ALS)



- Then we have the following optimization problem

$$\min_{u_i} ||R_{\Omega^i} - {u_i}^\top {V_{\Omega^i}}^\top||^2 + \lambda ||u_i||^2$$

- Convex problem with closed-form

$$\hat{u}_i = (V_{\Omega^i} {V_{\Omega^i}}^\top + \lambda I)^{-1} V_{\Omega^i}^\top R_{\Omega^i}$$

Alternating least squares (ALS)

Randomly initialize U and V

- **WHILE** does not converge
  - $\forall i \in \mathcal{U}$, $\min_{u_i} ||R_{\Omega^i} - {u_i}^\top {V_{\Omega^i}}^\top||^2 + \lambda ||u_i||^2$
  - $\forall j \in \mathcal{I}$, $\min_{v_j} ||R_{\Omega^j} - {v_j}^\top {U_{\Omega^j}}^\top||^2 + \lambda ||v_j||^2$

MF for Implicit Feedback

- In real-world applications, we often observe more implicit feedback than explicit feedback.
- In fact, explicit feedback is sometimes considered implicit.
- Suppose user i watched 35% of movie A and 85% of movie B.

Does this mean that the user likes A more than B? If so, does it mean that the user likes A more than twice as much as B?

- The method we learned last class is more appropriate for explicit feedback. Why?

Modelling Implicit Feedback

- Let's understand a more appropriate method
- Assume the binary interaction matrix P:

$$P = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

- That is, if user-i interact with item-j, than $P_{ij} = 1$, otherwise $P_{ij} = 0$.
- Now let C be a matrix of confidence regarding the interaction:

$$C = \begin{pmatrix} 0.85 & 0 & 0 & 0.34 & 0 & 0.98 \\ 0 & 0.37 & 0.10 & 0 & 0.63 & 0.01 \\ 0.45 & 0.42 & 0.43 & 0 & 0 & 0.23 \\ 0 & 0 & 0.26 & 0 & 0 & 0.88 \end{pmatrix}.$$

Collaborative Filtering for Implicit Feedback

- Then we propose the following optimisation problem:

$$\text{min}_{U,V} \sum_{i,j} C_{ij}(P_{ij} - u_i^\top v_j)^2 + \lambda||u_i||^2 + \lambda||v_j||^2$$

- Two main differences from previous MF method:
  - We need to account for the varying confidence levels
  - Optimization should account for all possible j, j pairs, rather than only those corresponding to observed data.
- We can use gradient descent to solve it.
- And ALS? By fixing V, can we find u_i?

Closed form

- Assume V being fix and let's find u_i.
- Then we need to minimize the following loss

$$\mathcal{L}_i = \text{min}_{u_i} \sum_j C_{ij}(P_{ij} - u_i^\top v_j)^2 + \lambda ||u_i||^2$$

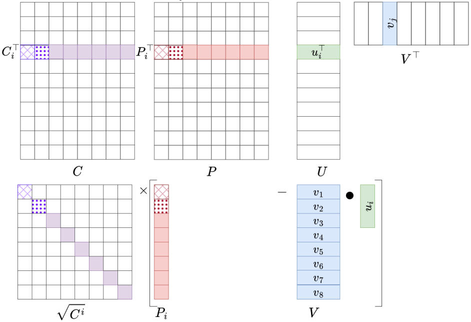
That is the same of:

$$\mathcal{L}_i = \text{min}_{u_i} \sum_j (\sqrt{C_{ij}}(P_{ij} - u_i^\top v_j))^2 + \lambda ||u_i||^2$$

Exercise: Find the closed form.

Alternating least squares (ALS)

$$\mathcal{L}_i = \min_{u_i} \sum_j (\sqrt{C_{ij}}(P_{ij} - u_i^\top v_j))^2 + \lambda ||u_i||^2$$



Closed form

- Therefore is the same of solving:

$$\mathcal{L}_i = ||\sqrt{C^i} P_i - \sqrt{C^i} V u_i||^2 + \lambda + ||u_i||^2$$

- Taking the derivative

$$\nabla u_i = -2(\sqrt{C^i} V)^\top (\sqrt{C^i} P_i - \sqrt{C^i} V u_i) + 2\lambda u_i$$

- Remind if D is diagonal $D = \sqrt{D} \times \sqrt{D}$ is trivial and $D = D^\top$
- Therefore, with just some algebraic derivations

$$u_i = (V^\top C^i V + \lambda I)^{-1} V^\top C^i P_i$$