# Recommender Systems in Action

**Ángel González Prieto**

Universidad Complutense de Madrid

New Bridges between
Mathematics and Data Science

# Joint work with the KNODIS research group

(a) Jesús Bobadilla

(b) Raúl Lara

(c) Fernando Ortega

# What Netflix series should I watch?

**The problem**

Given a collection of users and items, what is the best way of matching items to users according to their preferences?

# Collaborative filtering

**The 'solution'**

Recommend items to a user that are 'similar' to other items that he/she liked before.

# Recommender systems as a matrix completion problem

**Data**

The users have rated the items that they previously consumed in a certain scale (typically, $1, 2, \dots, 5$).

These data are collected in the so-called **rating matrix** (typically, very sparse)

$$R = N \text{ users} \left\{ \underbrace{\begin{pmatrix} 1 & {\color{red}?} & 4 & 3 & \dots & 3 \\ {\color{red}?} & 5 & 4 & 3 & \dots & {\color{red}?} \\ \vdots & & & & \ddots & \vdots \\ 2 & 3 & {\color{red}?} & 5 & \dots & {\color{red}?} \end{pmatrix}}_{M \text{ items}} \right.$$

**Goal**: Fill up the matrix.

# Collaborative filtering in action

**Collaborate to fill up the matrix**

Use the preferences of other users in similar items to complete the unknowns.

$$R = \underbrace{N \text{ users} \left\{ \begin{pmatrix} 1 & \boxed{\color{red}{?}} & 4 & 3 & \dots & 3 \\ ? & 5 & 4 & 3 & \dots & ? \\ \vdots & & & & \ddots & \vdots \\ 2 & 3 & ? & 5 & \dots & ? \end{pmatrix} \right.}_{M \text{ items}}$$

# Collaborative filtering in action

**Collaborate to fill up the matrix**

Use the preferences of other users in similar items to complete the unknowns.

$$R = N \text{ users} \left\{ \underbrace{\begin{pmatrix} 1 & \boxed{\color{red}?} & \mathbf{4} & \mathbf{3} & \dots & 3 \\ ? & 5 & \mathbf{4} & \mathbf{3} & \dots & ? \\ \vdots & & & & \ddots & \vdots \\ 2 & 3 & ? & 5 & \dots & ? \end{pmatrix}}_{M \text{ items}} \right.$$

# Collaborative filtering in action

## Collaborate to fill up the matrix

Use the preferences of other users in similar items to complete the unknowns.

$$R = \underbrace{N\text{ users}\left\{\begin{pmatrix} 1 & \boxed{{\color{red}?} \rightsquigarrow {\color{green}5}} & \mathbf{4} & \mathbf{3} & \dots & 3 \\ ? & {\color{blue}\mathbf{5}} & \mathbf{4} & \mathbf{3} & \dots & ? \\ \vdots & & & & \ddots & \vdots \\ 2 & 3 & ? & 5 & \dots & ? \end{pmatrix}\right.}_{M\text{ items}}$$

# Collaborative filtering models

Two different approaches coexist in the literature:

- **K-Nearest Neighbors (KNN)**: Find the closest items/users and copy the known values.
- **Matrix Factorization (MF)**: Decompose the matrix $R$ into the product of two matrices.

$$R = \underbrace{\begin{pmatrix} \star & ? & \star & \dots & ? \\ \star & \star & ? & \dots & \star \\ \vdots & & & \ddots & \vdots \\ ? & \star & \star & \dots & ? \end{pmatrix}}_{N \times M} \approx \underbrace{\begin{pmatrix} \leftarrow & \mathbf{p}_1 & \rightarrow \\ \leftarrow & \mathbf{p}_2 & \rightarrow \\ \vdots & \vdots & \vdots \\ \leftarrow & \mathbf{p}_N & \rightarrow \end{pmatrix}}_{N \times K} \cdot \underbrace{\begin{pmatrix} \uparrow & \uparrow & \dots & \uparrow \\ \mathbf{q}_1 & \mathbf{q}_2 & \dots & \mathbf{q}_M \\ \downarrow & \downarrow & \dots & \downarrow \end{pmatrix}}_{K \times M} = P \cdot Q^t$$

$\mathbb{R}^K$ = Latent space of hidden factors.

# Probabilistic Matrix Factorization

**Salakhutdinov - Mnih**

Compute the hidden factors $\mathbf{p}_i$ and $\mathbf{q}_j$ through a Maximum Likelihood Estimator (MLE).

Let's model the rating of the $i$-th user to the $j$-th item as a random variable of mean $\mathbf{p}_i \cdot \mathbf{q}_j$ and variance $\sigma^2$ (fixed).

$$R_{i,j} \sim \mathcal{N}(\mathbf{p}_i \cdot \mathbf{q}_j, \sigma^2).$$

Denote by $\Delta$ the set of pairs $(i,j)$ with known vote $r_{i,j}$. The likelihood function is

$$\mathcal{L}(\mathbf{p}_1, \dots, \mathbf{p}_N, \mathbf{q}_1, \dots, \mathbf{q}_M) = \prod_{(i,j)\in\Delta} f_{\mathcal{N}(\mathbf{p}_i \cdot \mathbf{q}_j, \sigma^2)}(r_{i,j}),$$

where $f_{\mathcal{N}(\mathbf{p}_i \cdot \mathbf{q}_j, \sigma^2)}(r) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-(r - \mathbf{p}_i \cdot \mathbf{q}_j)^2 / 2\sigma^2\right)$ is the pdf of the normal distribution.

# Probabilistic Matrix Factorization

**Salakhutdinov - Mnih**

Compute the hidden factors $\mathbf{p}_i$ and $\mathbf{q}_j$ through a Maximum Likelihood Estimator (MLE).

Let's model the rating of the $i$-th user to the $j$-th item as a random variable of mean $\mathbf{p}_i \cdot \mathbf{q}_j$ and variance $\sigma^2$ (fixed).

$$R_{i,j} \sim \mathcal{N}(\mathbf{p}_i \cdot \mathbf{q}_j, \sigma^2).$$

Denote by $\Delta$ the set of pairs $(i, j)$ with known vote $r_{i,j}$. The likelihood function is

$$\mathcal{L}(\mathbf{p}_1, \ldots, \mathbf{p}_N, \mathbf{q}_1, \ldots, \mathbf{q}_M) = \prod_{(i,j)\in\Delta} f_{\mathcal{N}(\mathbf{p}_i \cdot \mathbf{q}_j, \sigma^2)}(r_{i,j}),$$

where $f_{\mathcal{N}(\mathbf{p}_i \cdot \mathbf{q}_j, \sigma^2)}(r) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-(r - \mathbf{p}_i \cdot \mathbf{q}_j)^2/2\sigma^2\right)$ is the pdf of the normal distribution.

# Optimizing the likelihood

The log-likelihood is thus

$$\ell(\mathbf{p}_i, \mathbf{q}_j) = \log \mathcal{L}(\mathbf{p}_i, \mathbf{q}_j) = C - \frac{1}{2\sigma^2} \sum_{(i,j)\in\Delta} (r_{i,j} - \mathbf{p}_i \cdot \mathbf{q}_j)^2.$$

Hence, its partial derivatives are

$$\frac{\partial \ell}{\partial \mathbf{p}_{i_0}} = \frac{1}{\sigma^2} \sum_{(i_0,j)\in\Delta} (r_{i_0,j} - \mathbf{p}_{i_0} \cdot \mathbf{q}_j)\mathbf{q}_j, \qquad \frac{\partial \ell}{\partial \mathbf{q}_{j_0}} = \frac{1}{\sigma^2} \sum_{(i,j_0)\in\Delta} (r_{i,j_0} - \mathbf{p}_i \cdot \mathbf{q}_{j_0})\mathbf{p}_i.$$

**PMF algorithm**

Iteratively optimize the likelihood by (stochastic) gradient ascend.

**Remark**: Gaussian priors can be added to include a regularization factor.

# Key ingredients in the recipe

$$\begin{pmatrix} \star & \color{red}{?} & \cdots & \color{red}{?} \\ \star & \star & \cdots & \star \\ \vdots & & \ddots & \vdots \\ \color{red}{?} & \star & \cdots & \color{red}{?} \end{pmatrix} \approx \begin{pmatrix} \leftarrow & \mathbf{p}_1 & \rightarrow \\ \leftarrow & \mathbf{p}_2 & \rightarrow \\ \vdots & \vdots & \vdots \\ \leftarrow & \mathbf{p}_N & \rightarrow \end{pmatrix} \cdot \begin{pmatrix} \uparrow & \uparrow & \cdots & \uparrow \\ \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_M \\ \downarrow & \downarrow & \cdots & \downarrow \end{pmatrix}$$

Underlying distribution — Comparison method — Aggregation function

- **Underlying distribution**: Distribution supposed for the votes.
- **Comparison method**: Matrix norm taken to measure difference between $R$ and $P \cdot Q^t$.
- **Aggregation function**: Map $\mathbb{R}^K \times \mathbb{R}^K \to \mathbb{R}$ chosen to compute the predicted score from the hidden vectors.

# Bernoulli Matrix Factorization

**Changing the underlying distribution**

Use a Bernoulli distribution for each of the possible ratings.

1. Split the rating matrix with votes in $1, \ldots, S$ into $S$ **binary matrices**
$$R \rightsquigarrow R^1, R^2, \ldots, R^S$$
with $R^s$ binary matrices.
2. Factorize each $R^s$ into users' and items' hidden factors
$$R^s = P^s \cdot (Q^s)^t.$$
3. Suppose that each entry of $R^s = (R^s_{i,j})$ is a Bernouilli distribution with success probability $\sigma(\mathbf{p}^s_i \cdot \mathbf{q}^s_j)$,
$$p_{\mathrm{Bern}(\mathbf{p}^s_i \cdot \mathbf{q}^s_j)}(s) = \begin{cases} \sigma(\mathbf{p}^s_i \cdot \mathbf{q}^s_j) & \text{if } s = 1, \\ \sigma(1 - \mathbf{p}^s_i \cdot \mathbf{q}^s_j) & \text{if } s = 0. \end{cases}$$

# Implementing Bernoulli Matrix Factorization

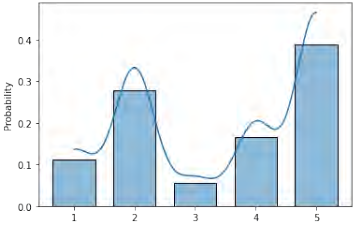
4. The log-likelihood function for factorizing the matrix $R^s$ is

$$\ell(\mathbf{p}_i^s, \mathbf{q}_j^s) = \sum_{R_{i,j}^s=1} \log \sigma(\mathbf{p}_i^s \cdot \mathbf{q}_j^s) + \sum_{R_{i,j}^s=0} \log \sigma(1 - \mathbf{p}_i^s \cdot \mathbf{q}_j^s)$$

5. Optimize through stochastic gradient ascend.

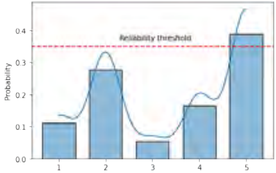
**Advantage 1**

The output can be interpreted as a probability distribution.

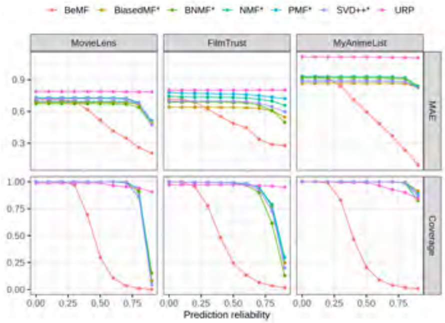


## Advantage 2

It provides a natural **reliability** measure of the prediction.




Tip: Recommend only those items with a **high probability** of being liked.



# Evolutionary Matrix Factorization

**Key idea**

Substitute the aggregation method

$$(\mathbf{p}_i, \mathbf{q}_j) \mapsto \mathbf{p}_i \cdot \mathbf{q}_j$$

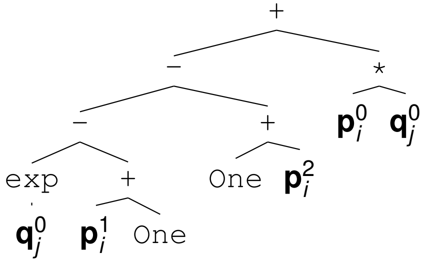
by a more complex method.

**Solution**: Use other analytic functions as aggregations methods.
**Example**:

$$f(\mathbf{p}_i, \mathbf{q}_j) = \Big(\Big(\exp(\mathbf{q}_j^0) - (\mathbf{p}_i^1 + 1)\Big) - (\mathbf{p}_i^2 + 1)\Big) + \mathbf{p}_i^0 \cdot \mathbf{q}_j^0$$

# Applying evolution

1. Codify the analytic function as a formal grammar ($\sim$ derivation tree).




2. Compute the log-likelihood function and its gradient to train the model ($\Leftarrow$ automatic derivation).
3. Apply a **genetic algorithm** to find the best derivation tree.

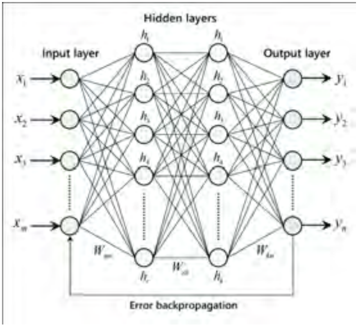
# The vertices of the tree

| Operator | Arity | Function | Symbol |
|---|---|---|---|
| Sine | 1 | $\sin(x)$ | `sin` |
| Cosine | 1 | $\cos(x)$ | `cos` |
| Arctangent | 1 | $\arctan(x)$ | `atan` |
| Exponential | 1 | $\exp(x)$ | `exp` |
| Logarithm | 1 | $\log(x)$ | `log` |
| Inverse | 1 | $\frac{1}{x}$ | `inv` |
| Sign | 1 | $-x$ | **–** |
| Addition | 2 | $x + y$ | `+` |
| Substraction | 2 | $x - y$ | `-` |
| Multiplication | 2 | $x \times y$ | `*` |
| Power | 2 | $x^y$ | `pow` |

# Deep Learning Matrix Factorization

**Xue et al.**

Use a neural network to create a complex and very flexible aggregation function.



# Present and future fun

- Train the neural network to not only achieve good results but also **fair** results.
- Enable an **inverse problem**: Train a neural network to detect teenagers $\rightsquigarrow$ Discover the 'archetypal movie' liked by the teenagers.
- Add noise to the hidden factors to increase the Shannon information of the dataset ($\sim$ VAE).
- Can we improve the results for certain underrepresented groups?
- Can we propose quality metrics beyond simple accuracy?

# Present and future fun

- Train the neural network to not only achieve good results but also **fair** results.
- Enable an **inverse problem**: Train a neural network to detect teenagers $\rightsquigarrow$ Discover the 'archetypal movie' liked by the teenagers.
- Add noise to the hidden factors to increase the Shannon information of the dataset ($\sim$ VAE).
- Can we improve the results for certain underrepresented groups?
- Can we propose quality metrics beyond simple accuracy?

***Thank you very much***
***for your attention!***