SAPIENZA
UNIVERSITÀ DI ROMA

A comparison of recommender systems

Project for the “Big Data Computing” course, a.y. 2021/22.

Alessio Palma, palma.1837493@studenti.uniroma1.it
Davide Santoro, santoro.1843664@studenti.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

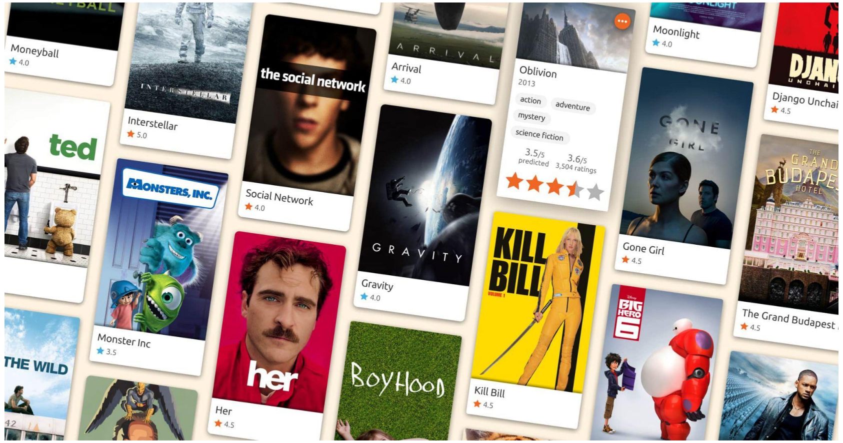
MovieLens 1M Dataset

SAPIENZA
UNIVERSITÀ DI ROMA

Description

MovieLens 1M is a dataset that includes about one million ratings made by about six thousand users on about four thousand movies. The ratings used in MovieLens 1M are explicit, they range from 1 to 5 stars and each user has at least 20 ratings. It was created by GroupLens Research, a research lab in the Department of Computer Science and Engineering at the University of Minnesota.

movielens
Non-commercial, personalized movie recommendations.



The dataset was split in train and test set, with a percentage of 80% and 20% respectively. From the former we also extracted a validation set (10% of the training set), with the aim of doing hyperparameter tuning.

SAPIENZA
UNIVERSITÀ DI ROMA

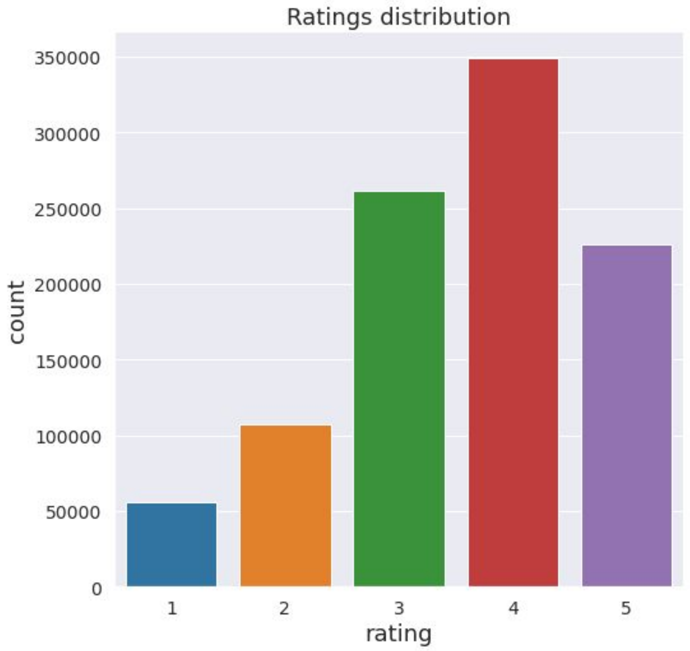
Preprocessing

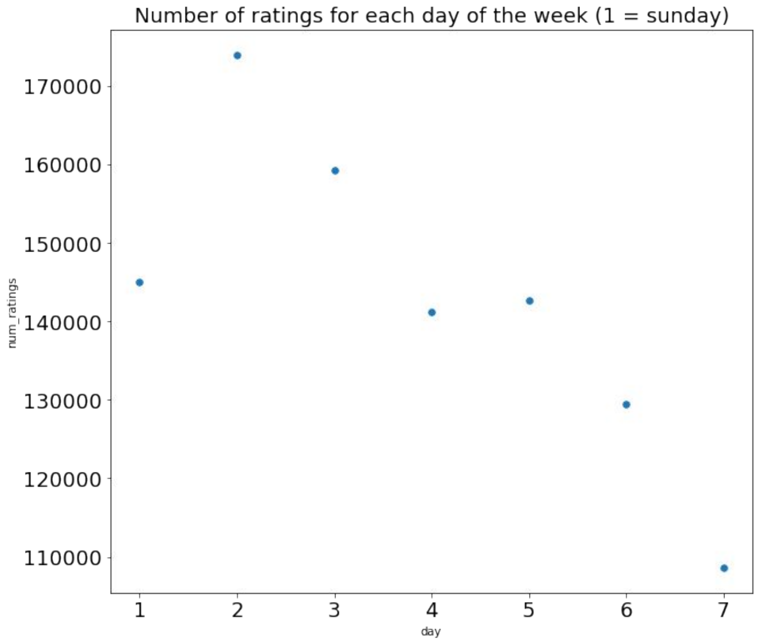
Since we noticed that movie ids are not contiguous and we need contiguous ids in order to reduce the dimensionalities of vector representations, we just remapped each id in the range [0, # items - 1].

Sparse vectors will be used by PySpark's K-means algorithm, which is the main component of the first paper we implemented. One-hot encodings will be used for the second paper.

SAPIENZA
UNIVERSITÀ DI ROMA

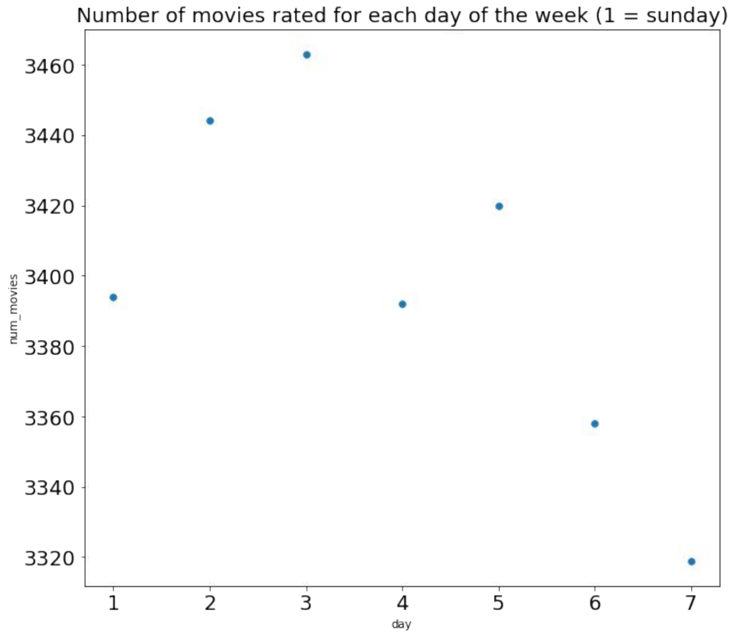
Data exploration

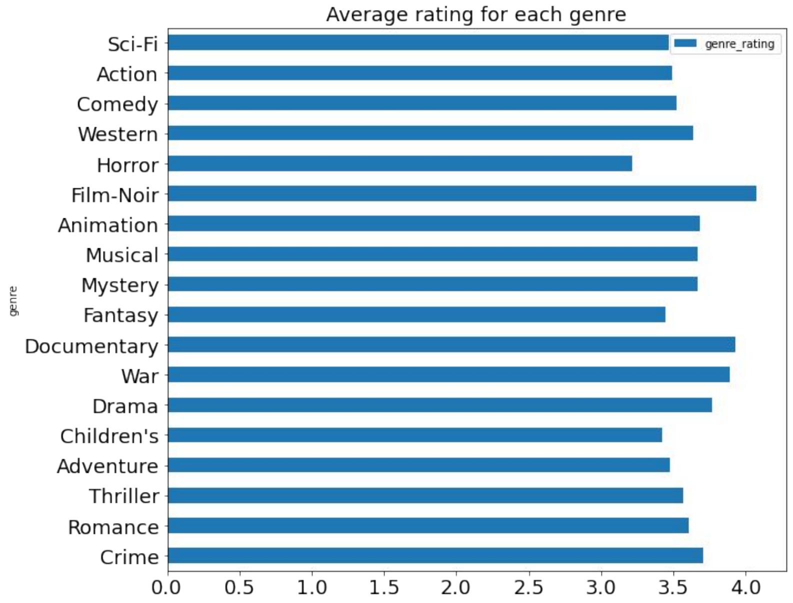




SAPIENZA
UNIVERSITÀ DI ROMA

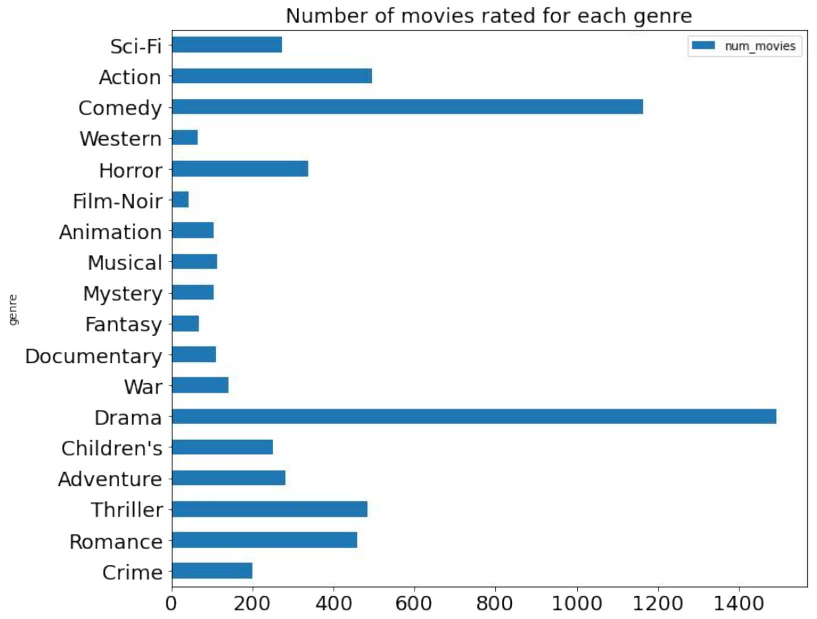
Data exploration (cont'd)

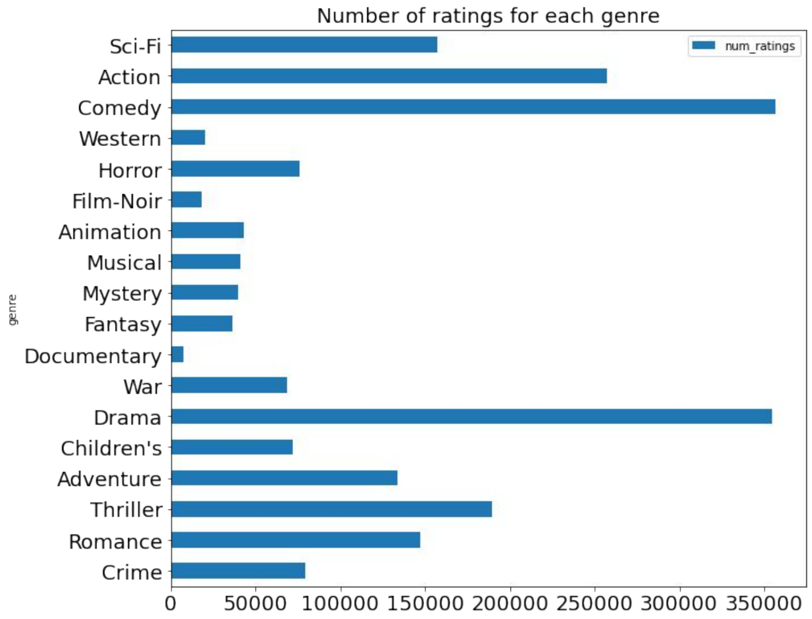




SAPIENZA
UNIVERSITÀ DI ROMA

Data exploration (cont'd)





“A New Collaborative Filtering Recommendation Algorithm Based on Dimensionality Reduction and Clustering Techniques”[1]

1. https://ieeexplore.ieee.org/document/8355449

Key ideas

- Collaborative filtering recommendation algorithms are highly valuable and play a vital role at the success of many companies;
- This paper proposes a new collaborative filtering algorithm with the aim of generating accurate recommendations;
- For this purpose, the K-means algorithm was adopted to cluster users in partitions according to their preferences and then a factorization technique was used to reconstruct each cluster;
- At prediction phase, we add as a bias the average rating of the user cluster. This approach outputs personalised recommendations while allowing us to compute just one bias for each cluster.

SAPIENZA
UNIVERSITÀ DI ROMA

Proposed approach

The construction of the recommender system proposed in the paper is divided into two main steps:

- The “offline model creation”;
- The “online model utilization”.

The first one consists of applying the K-means algorithm to the user-item matrix and then many SV decompositions:

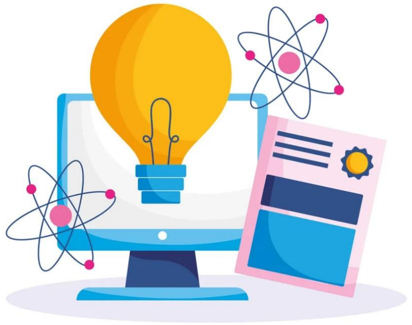
- The K-means algorithm is used to cluster similar users, the cosine similarity was used as the similarity measure:

$$\cos(x, y) = \frac{\sum_{i=1}^{n} p_{x,i} \times p_{y,i}}{\sqrt{\sum_{i=1}^{n} p_{x,i}^2}\sqrt{\sum_{i=1}^{n} p_{y,i}^2}}$$

- The SVD technique is applied by the authors on each cluster to obtain the decomposition matrices:

$$A_{m\times n} = U\Sigma V^T$$

$$\Sigma \in \mathbb{R}_{k\times k}$$
$$U \in \mathbb{R}_{m\times k}$$
$$V \in \mathbb{R}_{n\times k}$$



SAPIENZA
UNIVERSITÀ DI ROMA

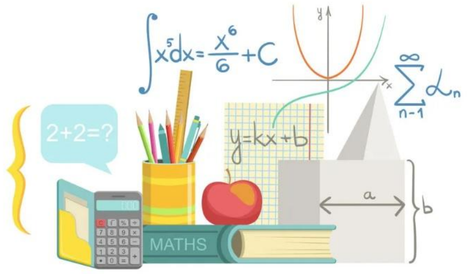
Proposed approach (cont'd)

- The second step involves the usage of the constructed model to produce recommendations for a given user. The rating given by the i-th user to the j-th item is computed with the following formula:

$$c_{ij} = \overline{c_k} + (U_k \cdot \Sigma_k)_i \cdot (V_k^T)_j$$

- The Root Mean Squared Error was used as the evaluation metric, it is widely used in evaluating recommender systems as it mimics a regression problem:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

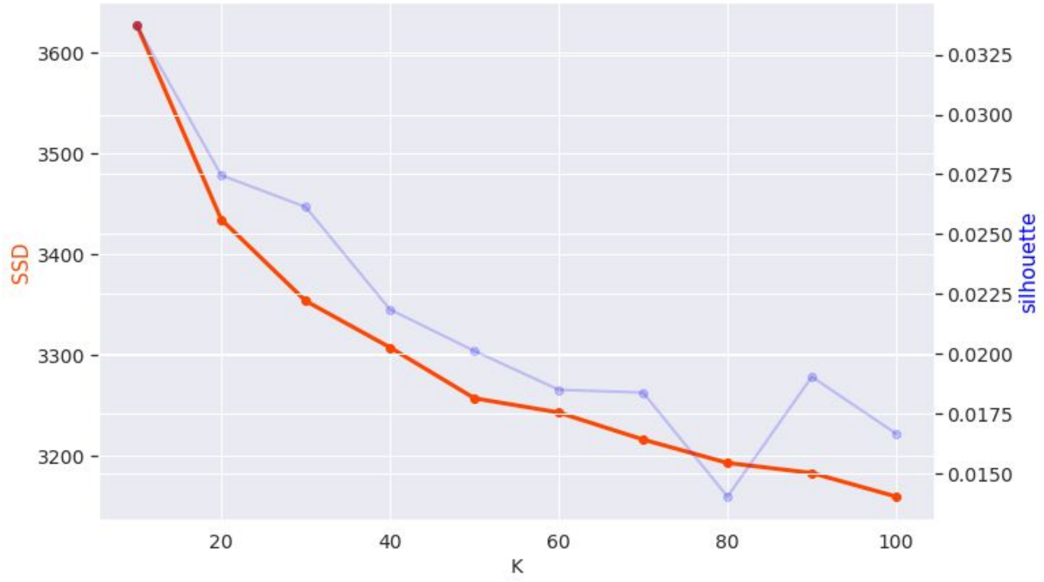


$\int x^5 dx = \frac{x^6}{6} + C$
2+2=?
y=kx+b
$\sum_{n=1}^{\infty} \alpha_n$
MATHS
a
b

SAPIENZA
UNIVERSITÀ DI ROMA

K-means clustering

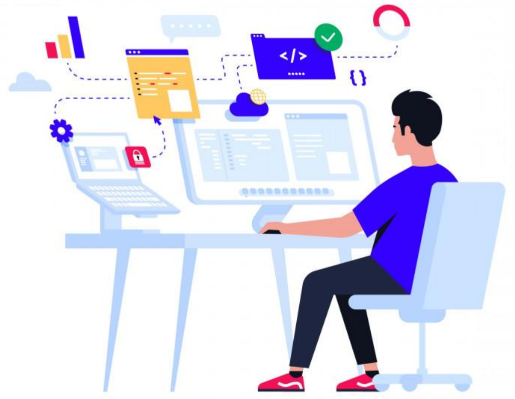
We used the elbow method to find the best K for the K-means algorithm:



- We can see that around K=50 there is an elbow point;
- This value of K was fixed to perform hyperparameter tuning;
- We also find out the best combination of hyperparameters: rank=25, regParam=0.1, maxIter=10.

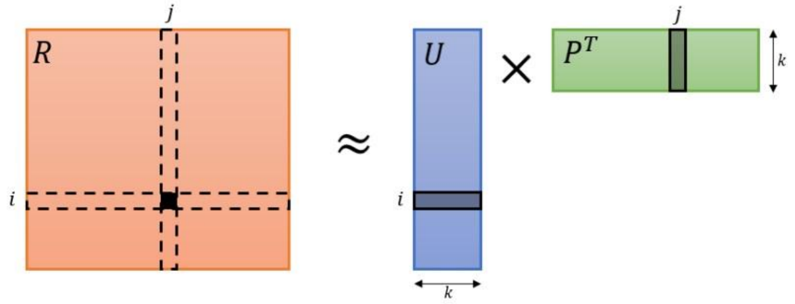
Implementation: K-means

- First of all we applied the K-means algorithm on the user-item matrix, in order to measure the similarity between vectors we adopted a non-euclidean distance: the **cosine distance**;
- To still obtain an optimal solution with the K-means algorithm we normalized the input points, since the cosine distance is directly correlated to L2-distance if we consider normalized vectors;
- After applying the algorithm, we divided both the training and the test set in a list of K dataframes, one for each cluster. Thanks to this, we can always know in which cluster a user falls;
- Then we computed the mean of the ratings for each cluster (c_k), and we used it to center the ratings column.



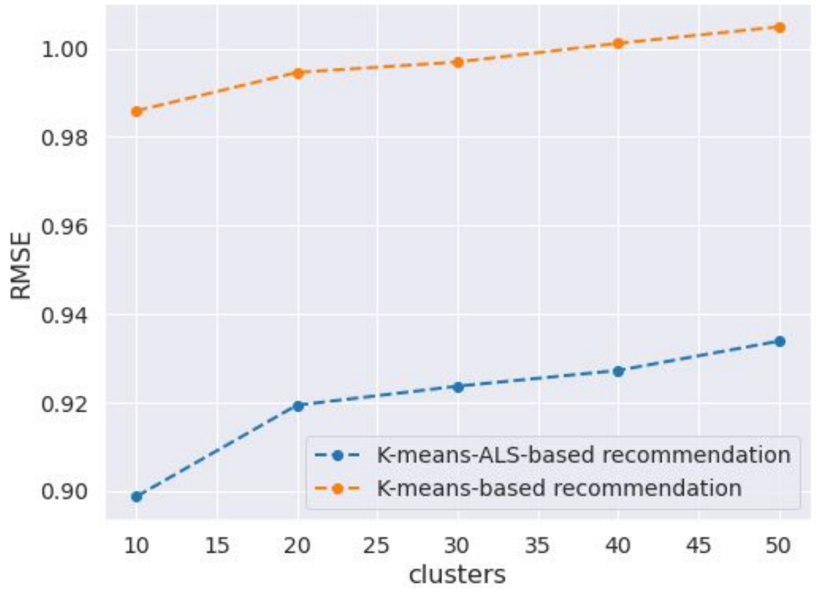
Implementation: SVD vs ALS

- Regarding the construction of the recommendation model, we used ALS instead of SVD. This choice is due to the fact that in PySpark there is no easy way to use SVD for recommender systems;
- SVD, as we showed before, decomposes a matrix into the product of three matrices, while ALS factorizes a matrix into two terms;
- Since they are just two approaches to achieve a factorization, we chose ALS. After the phase of hyperparameter tuning on the validation set, we trained several recommendation models (one for each cluster);
- Finally we used the fitted models to make predictions on the test set.



SAPIENZA
UNIVERSITÀ DI ROMA

Performance comparison



- We compared this method with a baseline which predicts the review that user u gives to movie i by computing the average of the reviews given to i by all the users in the same cluster of u (**K-means-based recommendation**);
- We compared the RMSE values obtained with these two methods on different values of K;
- The method proposed in the paper outperforms the one based only on K-means for all the tested Ks, with **0.899** as the best RMSE on **K=10**.

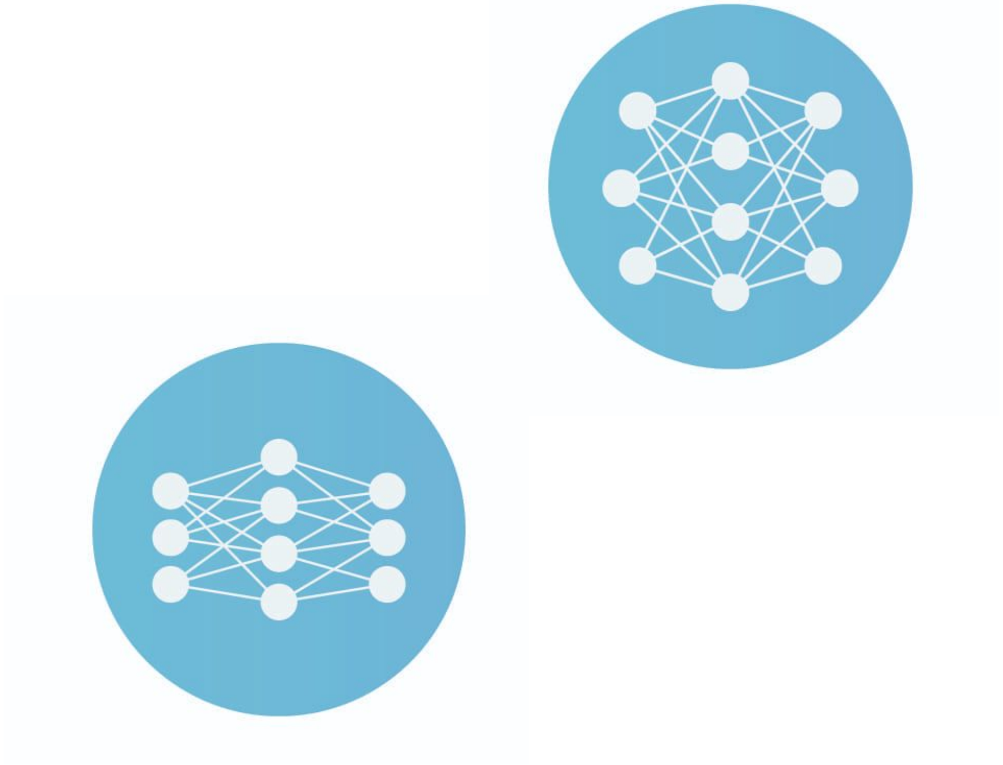
How was it possible?

- **Paperspace** is a high-performance cloud computing and ML development platform for building, training and deploying machine learning models;

- They launched **Gradient Notebook**, their ML Notebooks platform. It boasts a free-GPU instance with 30 GB RAM and we leveraged it since this first part was very memory demanding;

- **PySpark** for the preprocessing and implementation, since it allows distributed machine learning code;

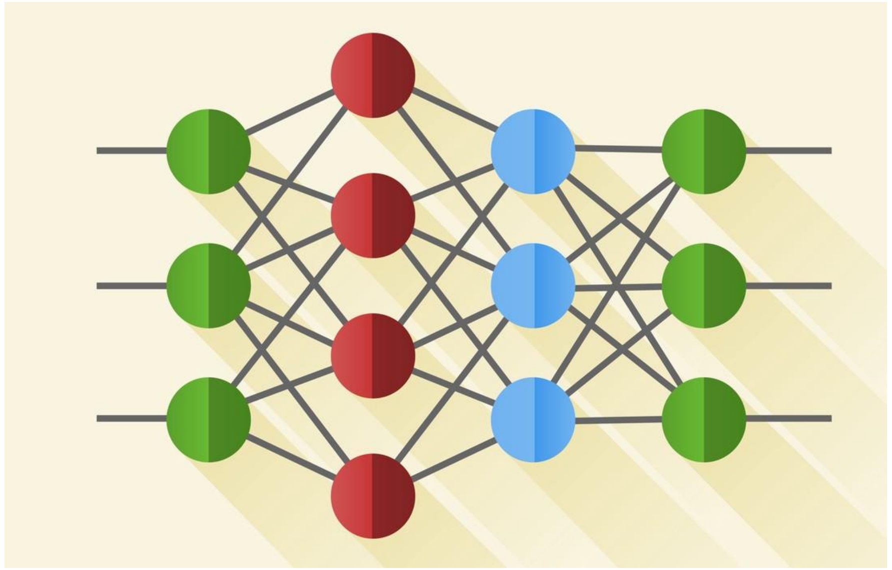
“Neural Collaborative Filtering”[2]



2. https://dl.acm.org/doi/10.1145/3038912.3052569

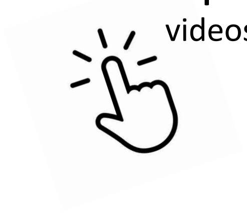
Key ideas

- Among the various collaborative filtering techniques, matrix factorization (MF) is the most popular one;
- A user's interaction on an item is modelled as the inner product of their latent vectors, but the performance can really be hindered by this simple choice of the inner product;
- By replacing this with a neural architecture that can learn an arbitrary function, they present a general framework named NCF;
- NCF is generic and can express and generalize matrix factorization under its framework. Also, it can learn nonlinear user-item interaction functions.



Implicit vs explicit feedback

- **Implicit feedback**: indirectly extract users' preferences through behaviours like watching videos, purchasing products and clicking items. Only 2 outcomes;

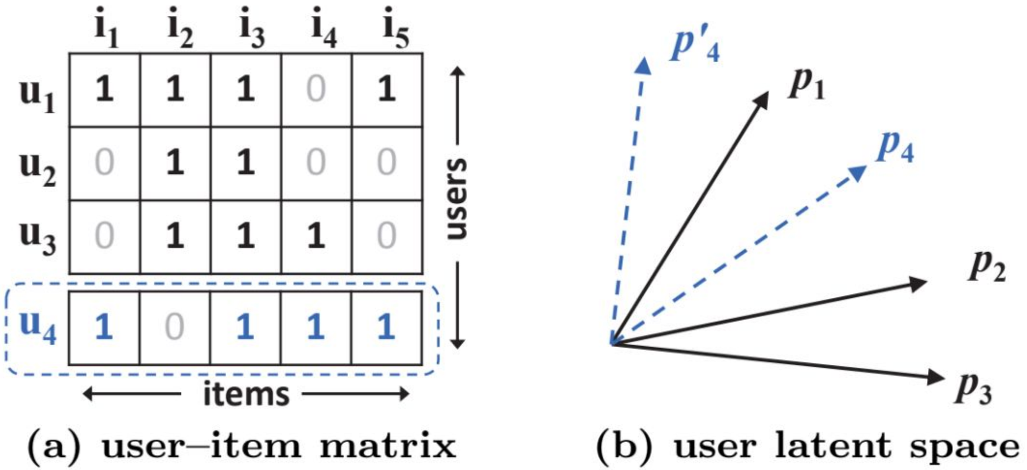


- **Explicit feedback**: explicit ratings and/or reviews on a scale;

- Implicit feedback can be tracked automatically and is thus much easier to collect for content providers, but it is more challenging to utilize since user satisfaction is not observed and there is a natural scarcity of negative feedback;

- **Binary vs multiclass classification**: in the paper they convert MovieLens' ratings into implicit feedback by thresholding, while we kept the problem harder by predicting ratings from 1 to 5;

The limits of dot product

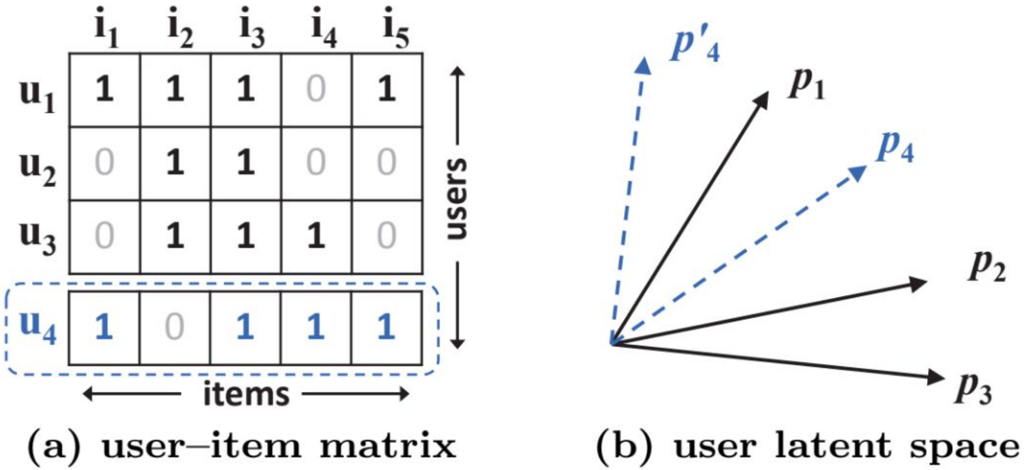


	i_1	i_2	i_3	i_4	i_5
u_1	1	1	1	0	1
u_2	0	1	1	0	0
u_3	0	1	1	1	0
u_4	1	0	1	1	1

(a) user–item matrix (b) user latent space

MF maps users and items to the same latent space P, the similarity between two users can then be measured with the inner product or equivalently, if latent vectors p∈P have unit length, the cosine of the angle between them.

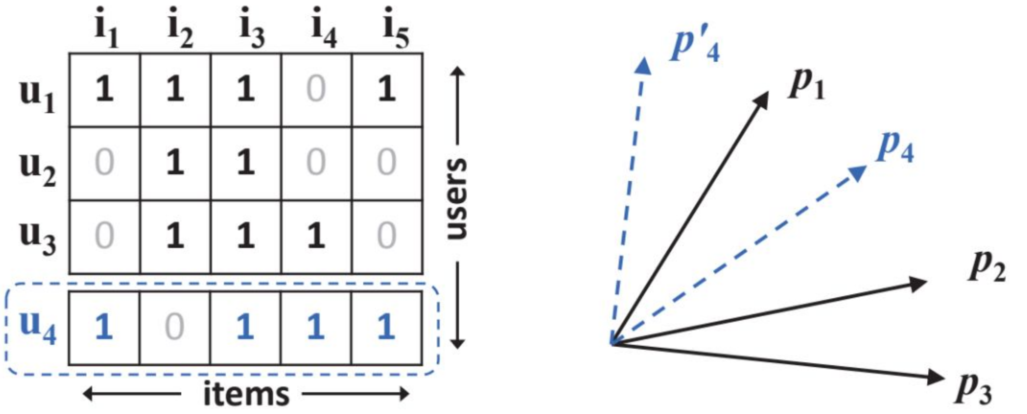
The limits of dot product



Without loss of generality, we can use the Jaccard coefficient J(A,B) as the **ground truth similarity of two users that MF needs to recover**:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

The limits of dot product



	i_1	i_2	i_3	i_4	i_5
u_1	1	1	1	0	1
u_2	0	1	1	0	0
u_3	0	1	1	1	0
u_4	1	0	1	1	1

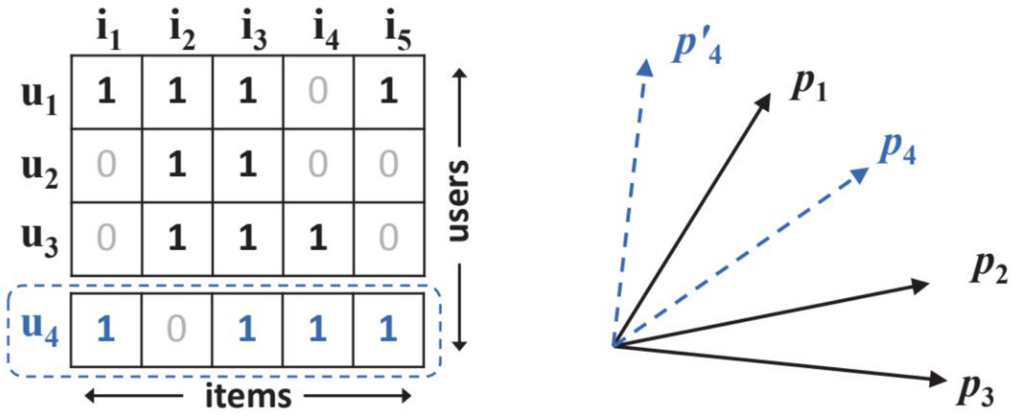
(a) user–item matrix (b) user latent space

Note: p_x is the normalized latent representation of user u_x.

It is easy to see that J(2,3) = 0.66 > J(1,2) = 0.5 > J(1,3) = 0.4. As such, the geometric relations of p_1, p_2, and p_3 in a 2D latent space can be plotted as in figure b.

Then let us consider a new user u_4, whose input is given as the dashed line, we then have J(4,1) = 0.6 > J(4,3) = 0.4 > J(4,2) = 0.2.

The limits of dot product



	i_1	i_2	i_3	i_4	i_5
u_1	1	1	1	0	1
u_2	0	1	1	0	0
u_3	0	1	1	1	0
u_4	1	0	1	1	1

(a) user–item matrix (b) user latent space

However, if a MF model places p_4 closest to p_1 (the two options are shown in figure b with dashed lines), it will result in p_4 closer to p_2 than p_3 according to the cosine of the angle, which is wrong!

One way to resolve the issue is to use a larger number of latent dimensions, but it may cause overfitting and we can fall into the curse of dimensionality. So, using a simple inner product in low-dimensional latent spaces can be very limiting.

The general NCF framework

The NCF predictive model is formulated as:

$$\hat{y}_{ui} = f(\mathbf{P}^T \mathbf{v}_u^U, \mathbf{Q}^T \mathbf{v}_i^I | \mathbf{P}, \mathbf{Q}, \Theta_f)$$

where $\mathbf{v}_u^U$ and $\mathbf{v}_i^I$ are one-hot encoding of user u and item i respectively, $\mathbf{P} \in R^{MxK}$ and $\mathbf{Q} \in R^{NxK}$ denote the latent factor matrix for users and items respectively, and Θ_f denotes the model parameters.

Since the function f is defined as a multi-layer neural network, it can be formulated as:

$$f(\mathbf{P}^T \mathbf{v}_u^U, \mathbf{Q}^T \mathbf{v}_i^I) = \phi_{out}(\phi_x(...\phi_2(\phi_1(\mathbf{P}^T \mathbf{v}_u^U, \mathbf{Q}^T \mathbf{v}_i^I))...))$$

where φ_{out} and φ_x respectively denote the mapping function for the output layer and the x-th layer.

SAPIENZA
UNIVERSITÀ DI ROMA

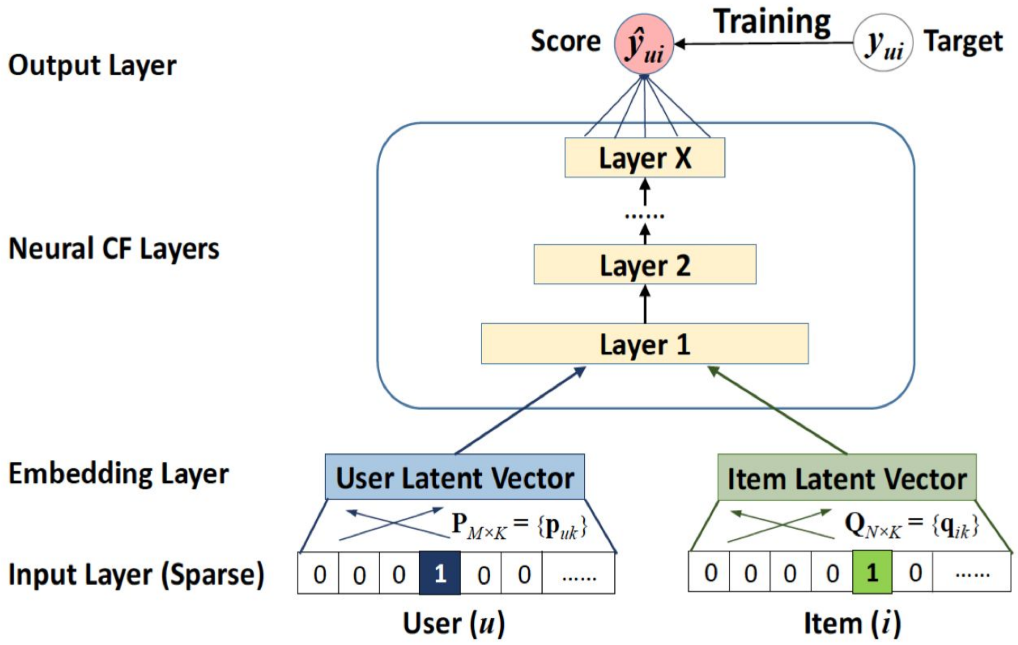
The MLP model

By concatenating the embeddings $\mathbf{p}_u = P^T v_u^U$ and $\mathbf{q}_i = Q^T v_i^I$ into $\mathbf{z}_1$, we can define the MLP model's layers as:

$$\phi_n(\mathbf{z}_{n-1}) = a_n(\mathbf{W}_n^T \mathbf{z}_{n-1} + \mathbf{b}_n)$$

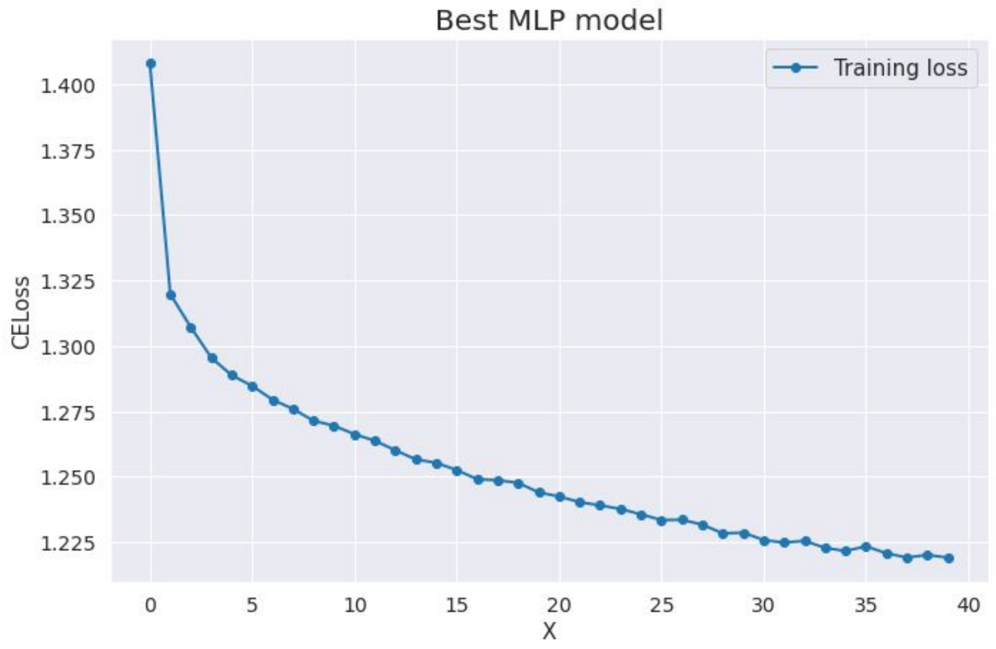
where $\mathbf{W}_n$, $\mathbf{b}_n$, and a_n denote the weight matrix, bias vector, and activation function for the n-th layer respectively. The ReLU function has been chosen as the activation function:

$$ReLU(x) = max(0, x)$$



SAPIENZA
UNIVERSITÀ DI ROMA

The MLP model (cont'd)



- We are addressing a multiclass classification problem, so we use the **cross entropy loss**;
- **Adam** is the chosen optimizer;
- After the implementation of the model and an extensive phase of hyperparameter tuning on the validation set, we considered the validation RMSE as the main metric. It turned out that the model with batch size = 256, L2 regularization = 0.0001, embeddings size = 32 and 4 linear layers is the best;
- We retrained it on both train and validation data, and then tested on the held-out test set. We obtained an RMSE of **0.975**.

Generalized Matrix Factorization

MF can be interpreted just as a special case of the NCF framework. Due to the one-hot encoding of user (item) ID at the input layer, the obtained embedding vector $\mathbf{p}_u$ ($\mathbf{q}_i$) can be seen as the latent vector of the user (item). We can now define the first neural layer as:

$$\phi_1(\mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u \odot \mathbf{q}_i$$

where $\odot$ denotes the element-wise product of vectors. We then define the output layer as:

$$\hat{y}_{ui} = \mathbf{h}^T \phi_1(\mathbf{p}_u, \mathbf{q}_i)$$

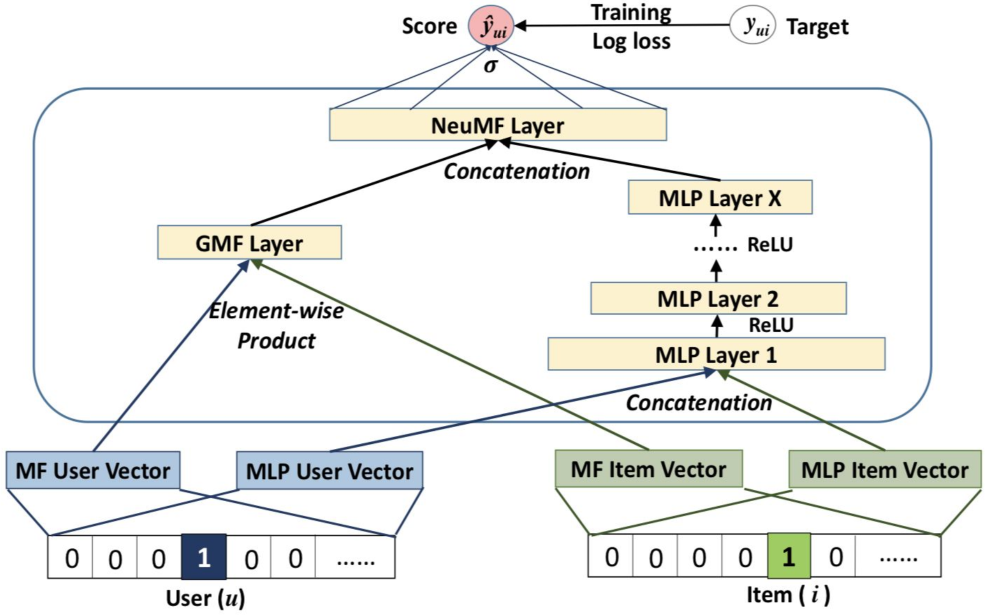
where **h** is a uniform vector of 1. Now it is easy to see that we exactly recovered the classical MF model.

Fusion of GMF and MLP: NeuMF

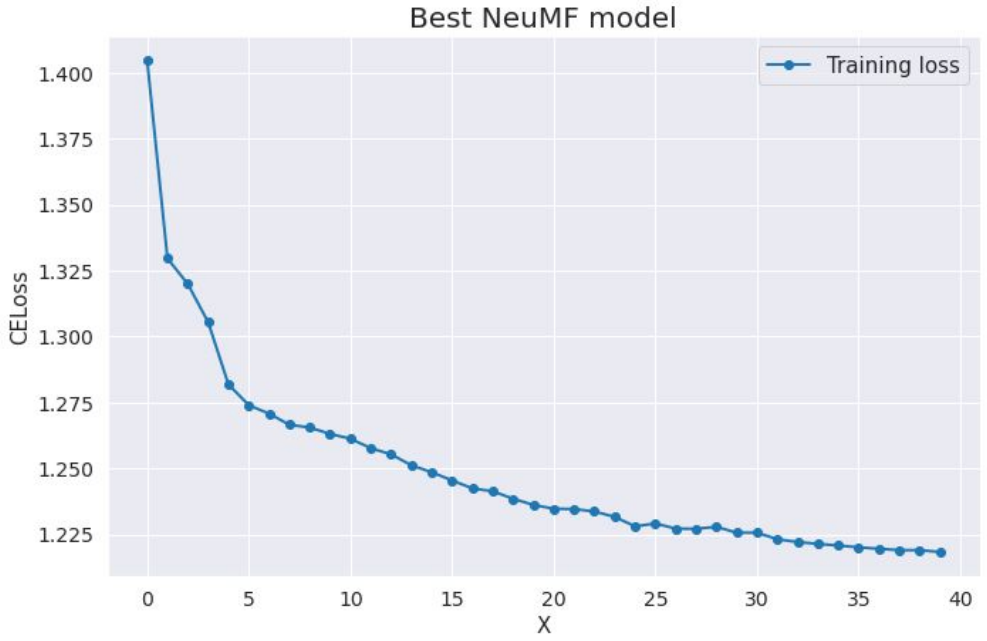
- Fuse GMF and MLP under the NCF framework so that they can mutually reinforce each other;

- Sharing same embeddings for GMF and MLP might limit the performance of the new model, e.g. it implies that GMF and MLP must use the same embeddings size, and this solution may fail to obtain the optimal ensemble;

- Allow GMF and MLP to learn separate embeddings, then combine the two models by concatenating their last hidden layers.



NeuMF (cont'd)



- After the implementation of the model and an extensive phase of hyperparameter tuning on the validation set, we considered the validation RMSE as the main metric. It turned out that (having fixed the best hyperparameters for the MLP part) the model with MF's embedding size = 20 and learning rate = 0.001 is the best;

- We retrained it on both train and validation data, and then tested on the held-out test set. We obtained an RMSE of **0.945**, so there is an actual improvement of **0.03** over the previous model.

How was it possible?

- **Google Colab** as the programming environment, since this part wasn't really memory consuming but required a good CPU to train the networks (Horovod actually doesn't support GPU training on Colab);
- **PyTorch** to implement the neural networks;
- **PySpark** for the preprocessing phase and to allow distributed machine learning code;
- **Horovod**, a distributed deep learning training framework, to allow training of neural networks on PySpark;

SAPIENZA
UNIVERSITÀ DI ROMA

Conclusions

- In the first paper we compared a weaker approach based only on K-means and a more advanced one based on K-means + ALS, which performs better;

- In the second paper we compared two different neural network approaches, with the NeuMF being an enhanced version of the MLP model;

- The K-means-ALS is the best approach, reaching an RMSE of **0.899**;

Model	Test RMSE
K-means	0.986
K-means-ALS	**0.899**
MLP	0.975
NeuMF	0.945

- It’s interesting to see that a classical but "clever" recommender system can still outperform a neural network, which is a more complicated model. Of course we are making some assumptions (e.g. we use the neural approach to perform multi-class classification meanwhile the authors perform binary classification), but still we managed to get some fascinating results.

Thank you for listening us!