# Movie Recommendation System

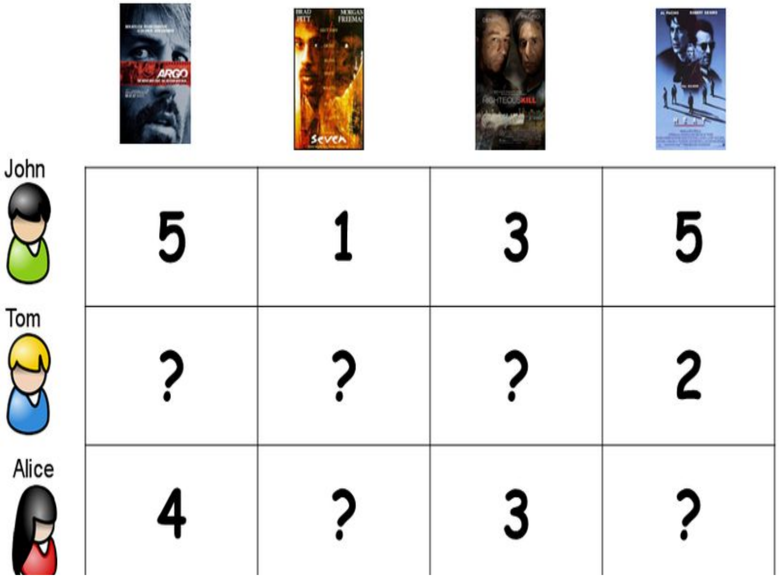
Realised by

Matrix Brigade
Medjaouri Insaf , Bouikni Lyna , Marius Roger

17/10/2023

# 1 / The problem

How can we effectively address the challenge of **sparse** data in recommendation systems ?



| | | | | |
|---|---|---|---|---|
| John | 5 | 1 | 3 | 5 |
| Tom | ? | ? | ? | 2 |
| Alice | 4 | ? | 3 | ? |

**User-Item rating matrix**

# 2/ The process  of the solution

1. Data Exploration
2. Build our first method as a baseline
3. Build the second approach
4. Result comparison

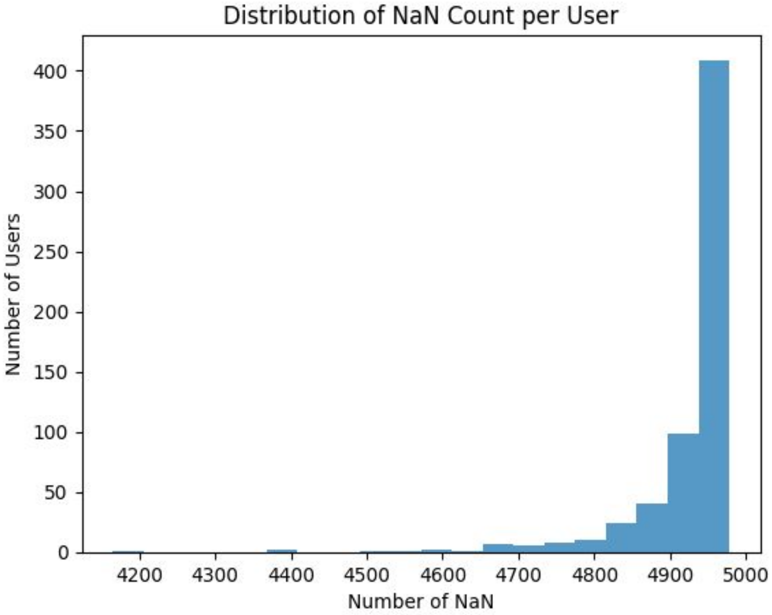
# 2.1/ Data Exploration

**Data set :** ratings_train.py + ratings_test.py

**Data shape :** 610 (users) * 4980 (movies)

**Percentage of Nan values :** ~ 98 %

**Zipf's law :** low-rank should work better



# 2.2 / Matrix Factorisation With Gradient Descent

Masking is used to avoid useless computations

```
R_m = np.ma.MaskedArray(R, nan_mask_R)
```

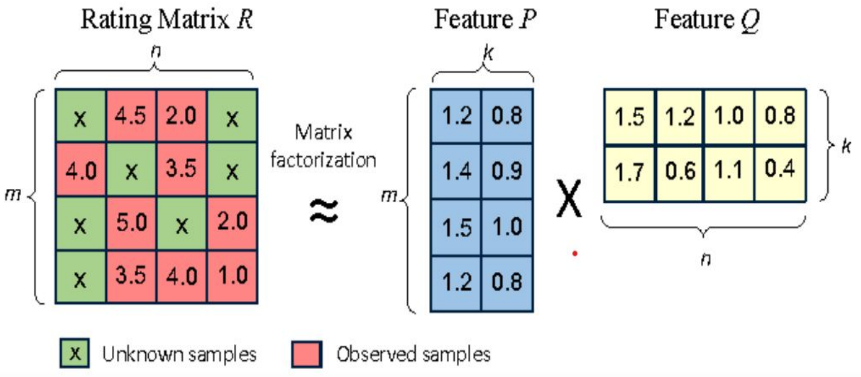
Deal with NaNs using “weights 0 on undefined loss values”

```
S = R - I @ U.T
S = np.where(R_m.mask, 0, S)
```

Compute several `I @ U.T` in parallel ➜ aggregate

Voting is expensive, and worse than averaging for MF ensembling

```
Re_flat = np.apply_along_axis(mode, 0, list_Re)
```



| | | | |
|---|---|---|---|
| x | 4.5 | 2.0 | x |
| 4.0 | x | 3.5 | x |
| x | 5.0 | x | 2.0 |
| x | 3.5 | 4.0 | 1.0 |

| | |
|---|---|
| 1.2 | 0.8 |
| 1.4 | 0.9 |
| 1.5 | 1.0 |
| 1.2 | 0.8 |

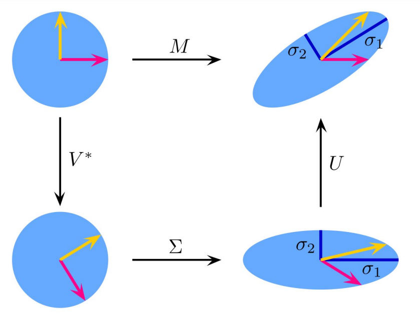
| | | | |
|---|---|---|---|
| 1.5 | 1.2 | 1.0 | 0.8 |
| 1.7 | 0.6 | 1.1 | 0.4 |

# 2.2.1/ Hyperparameters : First approach

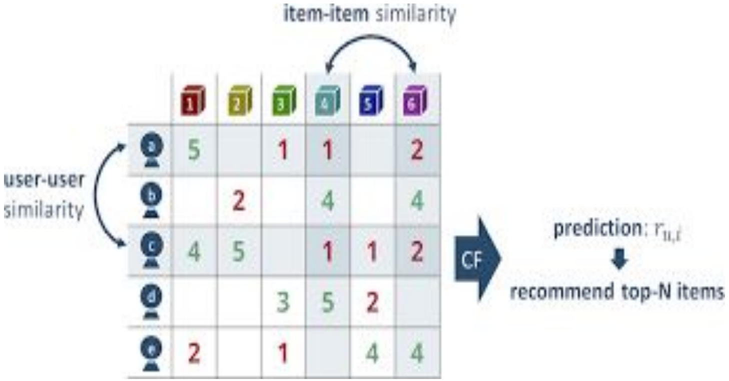
- Number of latent factors (K) = **5**
- Learning rate = **$1.4\times10^{-4}$**
- Regularization parameter **$\lambda$** = **0.1**
- Regularization parameter **$\mu$** = **1**
- Number of parallel models (first ensembling approach) = **20**
- Number of training epochs (first ensembling approach) = **100**

# 2.3/ SVD & Knn with cosine similarity

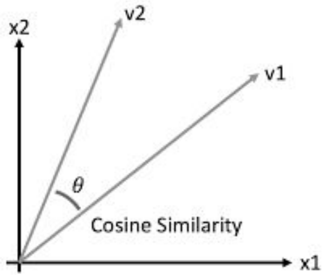
$$\boxed{C_{m\times n}} = \boxed{U_{m\times r}} \times \boxed{\Sigma_{r\times r}} \times \boxed{V^1_{r\times n}}$$



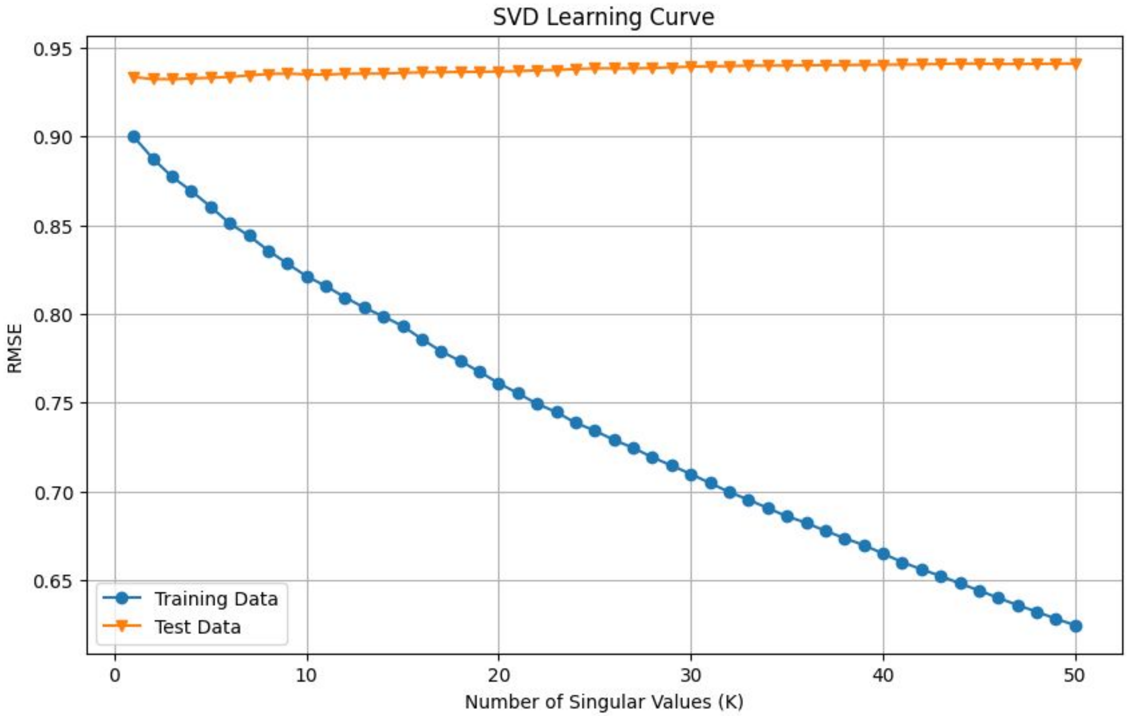
$$M = U \cdot \Sigma \cdot V^*$$



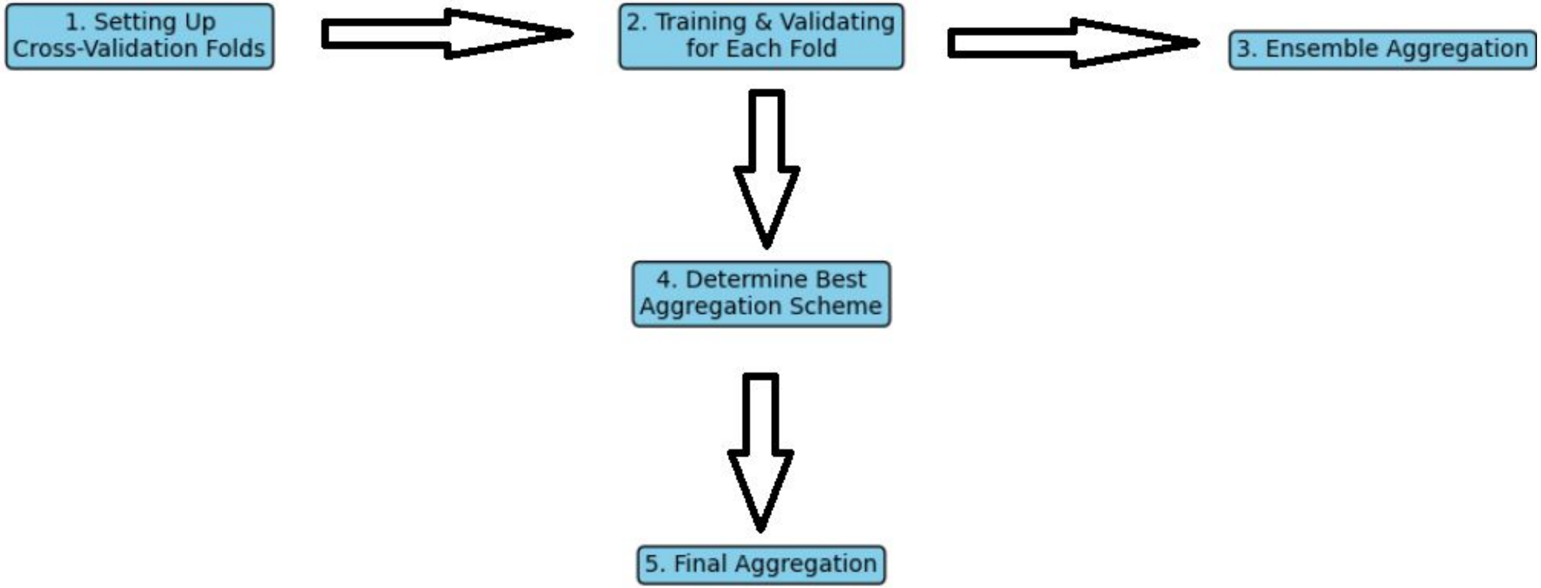
|   | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| a | 5 |   | 1 | 1 |   | 2 |
| b |   | 2 |   | 4 |   | 4 |
| c | 4 | 5 |   | 1 | 1 | 2 |
| d |   |   | 3 | 5 | 2 |   |
| e | 2 |   | 1 |   | 4 | 4 |



# 2.3/ SVD & Knn with cosine similarity



# 2.3/ Ensemble method



Flow Diagram for the Ensemble Method with Cross-Validation

1. Setting Up Cross-Validation Folds

2. Training & Validating for Each Fold

3. Ensemble Aggregation

4. Determine Best Aggregation Scheme

5. Final Aggregation

# 3/ Result comparison

| Evaluation Metrics | Baseline (MF, K=5, 3k epochs) | First Approach | Second Approach |
|---|---|---|---|
| **Rmse** | 0.969 | 0.89 | 0.992 |
| **Time** | 188.15 | 101.37 | 81.65 |
| **Accuracy** | 24.4 % | 24.89 % | 27.73 % |

Values of 2nd approach ∈ [2.5, 4.5] : too many models ➜ st-div of predictions is too small

# 3.2/ Hyperparameters : Second approach

- K (MF) = **5**
- Number of parallel models (MF) = **3**

(compute time & model diversity)

- Number of training epochs (MF) = **70**
- K (K-NN) = **50** (40-80)
- K (SVD) = **5** (tradeoff rmse-accuracy)

# 4/ Reference:

Petersen, K.B. and Pedersen, M.S., 2008. The matrix cookbook. *Technical University of Denmark*, *7*(15), p.510.

Thank you for your attention !