

COLLABORATIVE FILTERING

DATA LAB — ASSIGNMENT 1

Alexandre Vérine

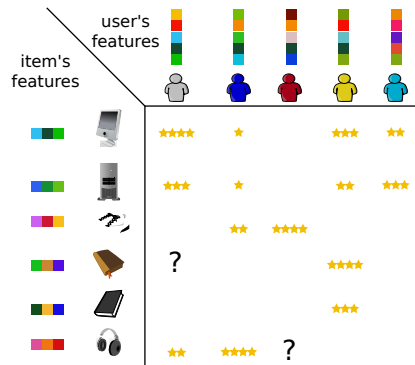
Université Paris Dauphine – PSL / ENS – PSL

1 October 2026



Recommendation: the problem

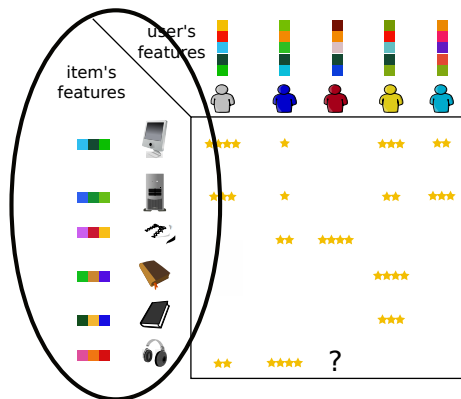
RECOMMENDATION: GENERAL SETTING



Goal: Predict grades & recommend items
with the highest predicted grade

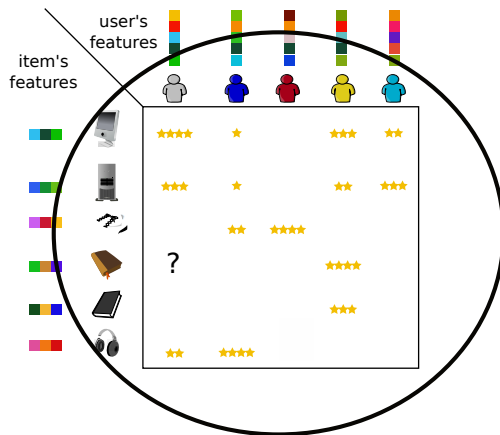
CONTENT BASED FILTERING

Recommendations based on **intra-user or intra-object relationships**



COLLABORATIVE FILTERING

Recommendations based on **user-object relationship**



CF vs. CBF IN RECOMMENDER SYSTEMS

Content-Based Filtering

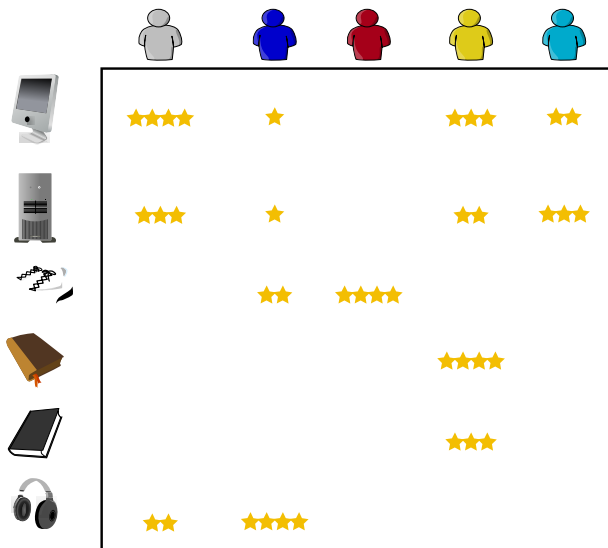
- ▶ Able to deal with **cold start**
- ▶ Requires user/item features
- ▶ Performance depends on the available features

Collaborative filtering

- ▶ Works with ratings only
 - ▶ Good performance in practice
 - ▶ Needs observations for new users and items
-
- ▶ The first assignment will focus on *collaborative filtering*

Neighborhood-based collaborative filtering

RATING MATRIX



→

$$R = \begin{bmatrix} 4 & 1 & 0 & 3 & 2 \\ 3 & 1 & 0 & 2 & 3 \\ 0 & 2 & 4 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 2 & 4 & 0 & 0 & 0 \end{bmatrix}$$

Remark

- ▶ Value 0 is ambiguous: not rated or rated as zero
particularly relevant with unitary ratings (e.g. online store)

NEIGHBORHOOD-BASED CF

Basic principle

1. compute similarity between **users** based on **preferred items**
2. predict unobserved ratings by combining grades of the **nearest users**

NEIGHBORHOOD-BASED CF

Basic principle

1. compute similarity between **users** based on **preferred items**
2. predict unobserved ratings by combining grades of the **nearest users**

Possible algorithm

How to predict unobserved rating $\hat{R}_{iu}$:

1. For all users v compute $Sim(u, v)$
2. Retain up to k nearest neighbors who rated item i : $N_k(u; i)$
3. predict $\hat{R}_{iu} = \frac{\sum_{v \in N_k(u; i)} Sim(u, v) R_{iv}}{\sum_{v \in N_k(u; i)} |Sim(u, v)|}$, using neighbors who rated item i ; use a baseline if the denominator is zero

NEIGHBORHOOD-BASED CF

Basic principle

1. compute similarity between **users** based on **preferred items**
2. predict unobserved ratings by combining grades of the **nearest users**

Possible algorithm

How to predict unobserved rating $\hat{R}_{iu}$:

1. For all users v compute $Sim(u, v)$
2. Retain up to k nearest neighbors who rated item i : $N_k(u; i)$
3. predict $\hat{R}_{iu} = \frac{\sum_{v \in N_k(u; i)} Sim(u, v) R_{iv}}{\sum_{v \in N_k(u; i)} |Sim(u, v)|}$, using neighbors who rated item i ; use a baseline if the denominator is zero

Example of similarity measure: *Jaccard similarity*

$$X_u = \{i : R_{iu} \geq 3\}$$

$$Sim(u, v) = Jaccard(X_u, X_v) = \frac{|X_u \cap X_v|}{|X_u \cup X_v|}$$

PEARSON CORRELATION COEFFICIENT

$$M_u = \{i : R_{iu} \text{ is observed} \}$$

► $M_u \cap M_v$ indexes of items rated by u and v

PEARSON CORRELATION COEFFICIENT

$$M_u = \{i : R_{iu} \text{ is observed} \}$$

► $M_u \cap M_v$ indexes of items rated by u and v

$$\begin{aligned} \text{Sim}(u, v) &= \text{Pearson}(u, v) \\ &= \\ &= \frac{\sum_{k \in M_u \cap M_v} (r_{ku} - \mu_u) \cdot (r_{kv} - \mu_v)}{\sqrt{\sum_{k \in M_u \cap M_v} (r_{ku} - \mu_u)^2} \cdot \sqrt{\sum_{k \in M_u \cap M_v} (r_{kv} - \mu_v)^2}} \end{aligned}$$

Where: $\mu_u = \frac{\sum_{k \in M_u} r_{ku}}{|M_u|}$

COMPUTATIONAL CONSIDERATIONS

Computational complexity for k recommendation

Performing k recommendation: $\mathcal{O}(n \cdot k)$

(assuming upper-bounded number of ratings per user)

- ▶ With $n = 58\,000$ and $k = 28\,000$,
an $\mathcal{O}(n \cdot k)$ algorithm running at 10 000 step / second takes ≈ 2 days to process.

Locality sensitive hashing functions

Properties:

- ▶ Similar users should collide with high probability.
- ▶ Dissimilar users should collide with low probability; collisions are not guarantees.

RECOMMENDATION WITH LSH

Algorithm

1. **offline:** For all users u , compute $h(u_i)$
2. **online:**
For all users u s.t. $h(u) = h(v)$,
Compute $Sim(v, u)$

c.f. **Mining massive datasets** (Free book) <http://infolab.stanford.edu/~ullman/mmds/ch3.pdf>
Chapter 3: Finding similar items

Model-based collaborative filtering

MATRIX FACTORIZATION FOR CF

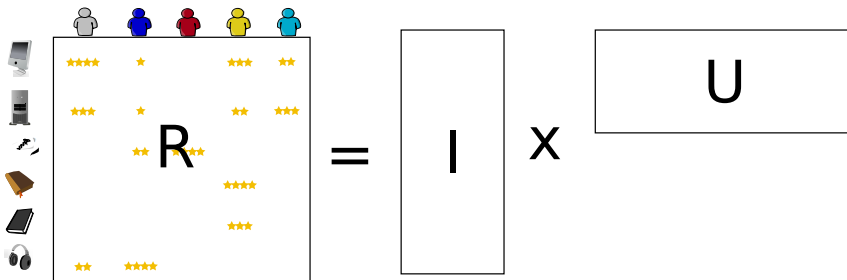
Idea: if ratings are correlated, then R can be approximated with a low rank matrix

Low rank matrix factorization

$$R \approx I \times U^T$$

Where

- ▶ $R \in \mathbb{R}^{m \times n}$
- ▶ $I \in \mathbb{R}^{m \times k}$
- ▶ $U \in \mathbb{R}^{n \times k}$



MATRIX FACTORIZATION DEMO

We search for $I \times U^T \approx R$

							$U^\top$
I		u_1	u_2	u_3	u_4	u_5	
	i_1	2	4				
	i_2	3	6				
	i_3	5		5	5	-5	
	i_4	1	2				

MATRIX FACTORIZATION DEMO

We search for $I \times U^T \approx R$

		<table><tr><td>1</td><td>2</td><td>1</td><td>1</td><td>-1</td></tr></table> $U^{\top}$					1	2	1	1	-1
1	2	1	1	-1							
I		u_1	u_2	u_3	u_4	u_5					
	2 i_1	2	4								
	3 i_2	3	6								
	5 i_3	5		5	5	-5					
	1 i_4	1	2								

MATRIX FACTORIZATION DEMO

We search for $I \times U^T \approx R$

		<table><tr><td>1</td><td>2</td><td>1</td><td>1</td><td>-1</td></tr></table> U^T					1	2	1	1	-1
1	2	1	1	-1							
I		u_1	u_2	u_3	u_4	u_5					
	2 i_1	2	4								
	3 i_2	3	6								
	5 i_3	5	10	5	5	-5					
	1 i_4	1	2								

MATRIX FACTORIZATION DEMO

We search for $I \times U^T \approx R$

		1	2	1	1	-1	U^T
I	i_1	2	4	2	2	-2	
	i_2	3	6	3	3	-3	
	i_3	5	10	5	5	-5	
	i_4	1	2	1	1	-1	

MATRIX FACTORIZATION DEMO

We search for $I \times U^\top \approx R$

		<table><tr><td>1</td><td>2</td><td>1</td><td>1</td><td>-1</td></tr></table> $U^\top$					1	2	1	1	-1
1	2	1	1	-1							
I		u_1	u_2	u_3	u_4	u_5					
	2	i_1	2	4	2	2	-2				
	3	i_2	3	6	3	3	-3				
	5	i_3	5	10	5	5	-5				
	1	i_4	1	2	1	1	-1				

Remark: I and U can be interpreted.

MATRIX FACTORIZATION DEMO: TWO LATENT DIMENSIONS

Observed preferences (users $\times$ movies)

	N	J	C	S	P	Ca
User 1	1	1	1	0	0	0
User 2	1	1	1	0	0	0
User 3	1	1	1	0	0	0
User 4	1	1	1	1	1	1
User 5	-1	-1	-1	1	1	1
User 6	-1	-1	1	1	1	1
User 7	-1	-1	1	1	1	1

N: Nero; J: Julius Caesar; C: Cleopatra; S: Sleepless in Seattle; P: Pretty Woman; Ca: Casablanca.

Adapted from the example in Charu C. Aggarwal, *Recommender Systems*. Here the matrix is transposed relative to the item-by-user notation used in this lecture. The fit is approximate.

Two factors: history and romance

$$P = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix}, \quad Q^T = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$R_{\text{users} \times \text{movies}} \approx PQ^T$$

MACHINE LEARNING FORMULATION

Regularized loss minimization on fixed-rank matrices

First consider a fully observed matrix, using the Frobenius matrix norm $\|\cdot\|_{\mathcal{F}}$

$$\min_{I,U} \|R - IU^{\top}\|_{\mathcal{F}}^2$$

Where:

- ▶ $I \in \mathbb{R}^{m,k}$
- ▶ $U \in \mathbb{R}^{n,k}$
- ▶ $\|X\|_{\mathcal{F}}^2 = \text{tr}(X^{\top}X)$

MACHINE LEARNING FORMULATION

Regularized loss minimization on fixed-rank matrices

First consider a fully observed matrix, using the Frobenius matrix norm $\|\cdot\|_{\mathcal{F}}$

$$\min_{I,U} \|R - IU^{\top}\|_{\mathcal{F}}^2 + \lambda \|I\|_{\mathcal{F}}^2 + \mu \|U\|_{\mathcal{F}}^2$$

Where:

- ▶ $I \in \mathbb{R}^{m,k}$
- ▶ $U \in \mathbb{R}^{n,k}$
- ▶ $\|X\|_{\mathcal{F}}^2 = \text{tr}(X^{\top}X)$

Role of regularization

(same as always)

- ▶ avoid overfitting ("learning by heart")
- ▶ improve generalization ("prediction to unseen data")

MACHINE LEARNING FORMULATION

Regularized loss minimization on fixed-rank matrices

First consider a fully observed matrix, using the Frobenius matrix norm $\|\cdot\|_{\mathcal{F}}$

$$\min_{I,U} \underbrace{\|R - IU^{\top}\|_{\mathcal{F}}^2 + \lambda\|I\|_{\mathcal{F}}^2 + \mu\|U\|_{\mathcal{F}}^2}_{C(I,U)}$$

Where:

- ▶ $I \in \mathbb{R}^{m,k}$
- ▶ $U \in \mathbb{R}^{n,k}$
- ▶ $\|X\|_{\mathcal{F}}^2 = \text{tr}(X^{\top}X)$

Role of regularization

(same as always)

- ▶ avoid overfitting ("learning by heart")
- ▶ improve generalization ("prediction to unseen data")

HOW TO OPTIMIZE

Cost function C

$$C(I, U) = \|R - IU^\top\|_{\mathcal{F}}^2 + \lambda \|I\|_{\mathcal{F}}^2 + \mu \|U\|_{\mathcal{F}}^2$$

Properties of C

- ▶ non-convex in U and I
- ▶ convex in U (with I fixed)
- ▶ convex in I (with U fixed)

Optimization strategy

- ▶ aim for any local minimum
- ▶ alternated minimization over U and I (while the other is fixed)

DERIVATIVES

Cost function C

$$C(I, U) = \|R - IU^\top\|_{\mathcal{F}}^2 + \lambda \|I\|_{\mathcal{F}}^2 + \mu \|U\|_{\mathcal{F}}^2$$

$$C(I, U) = \text{tr}(R^\top R) - 2\text{tr}(R^\top IU^\top) + \text{tr}(UI^\top IU^\top) + \lambda \text{tr}(I^\top I) + \mu \text{tr}(U^\top U)$$

Partial derivatives

$$\begin{aligned} \blacktriangleright \frac{\partial C}{\partial U}(I, U) &= -2R^\top I + 2UI^\top I + 2\mu U \\ \blacktriangleright \frac{\partial C}{\partial I}(I, U) &= -2RU + 2IU^\top U + 2\lambda I \end{aligned}$$

c.f. **The matrix cookbook**

<https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf>
Section 2.5 derivatives of Traces.

ALGORITHM

First approach: Gradient descent

at every step t ,

- ▶ $I_{t+1} = I_t - \eta_t \frac{\partial C}{\partial I}(I_t, U_t)$
- ▶ $U_{t+1} = U_t - \xi_t \frac{\partial C}{\partial U}(I_t, U_t)$

ALGORITHM

First approach: Gradient descent

at every step t ,

- ▶ $I_{t+1} = I_t - \eta_t \frac{\partial C}{\partial I}(I_t, U_t)$
- ▶ $U_{t+1} = U_t - \xi_t \frac{\partial C}{\partial U}(I_t, U_t)$

Second approach: Alternating Least Squares (ALS)

Setting the partial derivatives to zero, we have :

$$\begin{aligned}\frac{\partial C}{\partial I}(I, U) &= -2RU + 2IU^\top U + 2\lambda I &= 0 \\ \frac{\partial C}{\partial U}(I, U) &= -2R^\top I + 2UI^\top I + 2\mu U &= 0\end{aligned}$$

ALGORITHM

First approach: Gradient descent

at every step t ,

- ▶ $I_{t+1} = I_t - \eta_t \frac{\partial C}{\partial I}(I_t, U_t)$
- ▶ $U_{t+1} = U_t - \xi_t \frac{\partial C}{\partial U}(I_t, U_t)$

Second approach: Alternating Least Squares (ALS)

Setting the partial derivatives to zero, we have :

$$\begin{aligned}\frac{\partial C}{\partial I}(I, U) &= -2RU + 2IU^\top U + 2\lambda I &= 0 \\ \frac{\partial C}{\partial U}(I, U) &= -2R^\top I + 2UI^\top I + 2\mu U &= 0\end{aligned}$$

ALGORITHM

First approach: Gradient descent

at every step t ,

- ▶ $I_{t+1} = I_t - \eta_t \frac{\partial C}{\partial I}(I_t, U_t)$
- ▶ $U_{t+1} = U_t - \xi_t \frac{\partial C}{\partial U}(I_t, U_t)$

Second approach: Alternating Least Squares (ALS)

Setting the partial derivatives to zero, we have :

$$\begin{aligned} I &= RU(U^\top U + \lambda \mathbb{I})^{-1} \\ U &= R^\top I(I^\top I + \mu \mathbb{I})^{-1} \end{aligned}$$

ALGORITHM

First approach: Gradient descent

at every step t ,

- ▶ $I_{t+1} = I_t - \eta_t \frac{\partial C}{\partial I}(I_t, U_t)$
- ▶ $U_{t+1} = U_t - \xi_t \frac{\partial C}{\partial U}(I_t, U_t)$

Second approach: Alternating Least Squares (ALS)

Setting the partial derivatives to zero, we have :

$$\begin{aligned} I_{t+1} &= RU_t(U_t^\top U_t + \lambda \mathbb{I})^{-1} \\ U_{t+1} &= R^\top I_{t+1}(I_{t+1}^\top I_{t+1} + \mu \mathbb{I})^{-1} \end{aligned}$$

MISSING VALUES

$$S = \{i, j : r_{ij} \text{ is observed}\}$$

$$\frac{\partial C}{\partial i_{iq}}(I, U) = 2 \cdot \sum_{j: (i,j) \in S} \left(r_{ij} - \sum_{s=1}^k i_{is} \cdot u_{js} \right) (-u_{jq}) + 2\lambda i_{iq}$$

$$\frac{\partial C}{\partial u_{jq}}(I, U) = 2 \cdot \sum_{i: (i,j) \in S} \left(r_{ij} - \sum_{s=1}^k i_{is} \cdot u_{js} \right) (-i_{iq}) + 2\mu u_{jq}$$

OTHER ALTERNATIVES

- ▶ PCA and variants (including non-linear PCA) Generalized Principal Component Analysis by René Vidal Yi Ma and S.Shankar Sastry
See section *PCA with Robustness to Missing Entries*.
- ▶ Laplacian Embedding - Dimension reduction with a local only approach
- ▶ Deep Matrix Factorization
- ▶ Sparse k-NN
- ▶ Optimal Transport based methods
- ▶ Neural Collaborative Filtering
- ▶ ...

INCORPORATING BACKGROUND KNOWLEDGE

Items/users often come with features, how to incorporate them

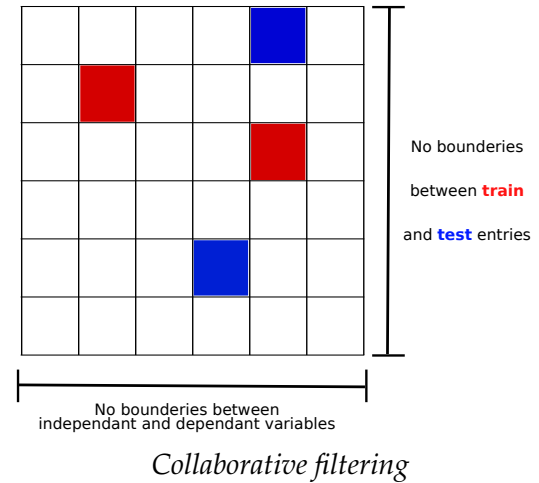
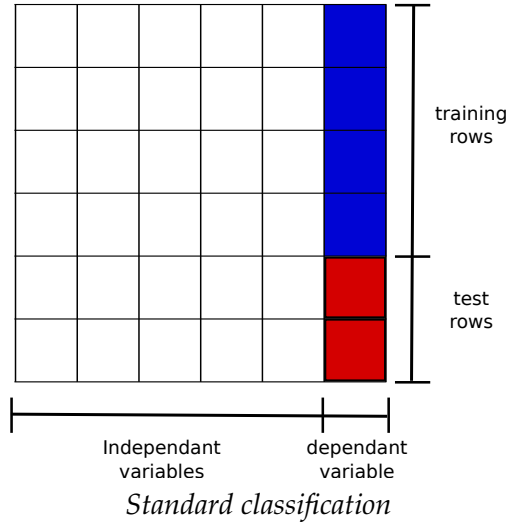
- ▶ Train multiple independent models
- ▶ Incorporate features into embedding
- ▶ ...

Evaluation

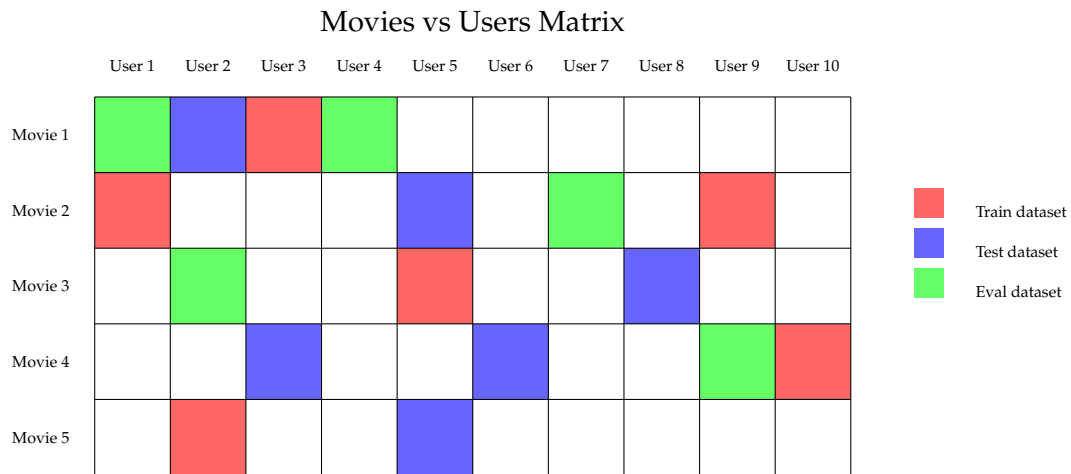
A COPY OF THE MOVIE LENS DATASET

- ▶ **ratings_train.npy**: 1600 Movies and 600 users. ~ 30k ratings
- ▶ **ratings_test.npy**: 1600 Movies and 600 users. ~ 30k ratings, to test your own method.
- ▶ **ratings_eval.npy**: Our dataset for the platform.
- ▶ **namesgenre.npy**: Movie names and genres to conduct an analysis on the dataset.

CLASSIFICATION VS. COLLABORATIVE FILTERING



OUR SETTING



EVALUATION METRIC

To evaluate the quality of your predictions, we will use the *Root Mean Squared Error* (RMSE):

$$RMSE(R, \hat{R}, T) = \sqrt{\frac{\sum_{(i,u) \in T} (R_{iu} - \hat{R}_{iu})^2}{|T|}}$$

Where:

- ▶ R is the **eval** rating matrix (sparse)
- ▶ $\hat{R}$ is the estimated rating matrix (dense) based on **train+test**
- ▶ $T = \{(i, u) \mid R_{iu} \text{ is in the eval set}\}$

LOCAL TESTING VS. PLATFORM EVALUATION

You have access to **train** and **test**. A third set, **eval**, is held out by the instructor.

	Input to your method	Scored on
Your local experiments	train	test (kept out of the input)
The platform	train + test in one matrix	eval (hidden from you)

- ▶ The starter example uses `ratings_train.npy`; the platform supplies **all available ratings** from train and test.
- ▶ Load the file passed through `-name`; do not hard-code the training filename.

Different scores are expected

The input contains more ratings and the score is computed on a different held-out set. Your local score and the platform score need not match.

THE EVALUATION PLATFORM

The platform measures prediction quality and computational cost:

- ▶ RMSE
- ▶ Accuracy for exact ratings
- ▶ Running time

Use these measurements to compare methods and interpret your results.

Code is not graded

Your report and presentation should explain the experimental protocol and what the results teach you.

Assignment 1: your work

GITHUB REGISTRATION: COMPLETE EVERY STEP IN ORDER

1. Fill in your GitHub ID in the registration sheet.

2. Accept the invitation to join Master-IASD.

If you have already added your ID, you should have received an invitation email. Check your inbox.

3. Register for the DataLab classroom.

4. Register for assignment A1.

Every student must complete all four steps. Each link above is clickable.

FORM YOUR GROUP BEFORE MONDAY 5 OCTOBER

Exactly three students per group

Agree on your partners and group name before registering your group.

1. **One student creates the group** when accepting A1.
2. **The other two students join that same group.**
3. Check that all three members can access the same GitHub repository.

Do not create three separate groups for the same team.

A1 AND A2: FIVE-MINUTE PRESENTATIONS

Your presentation lasts exactly 5 minutes

An alarm rings at 5 minutes. Finish your talk when it rings.

Current A1 count: 37 groups (7 October).

Session	Groups	Talk per group	Total with questions
Preliminary	19	5 minutes	133 minutes
Final	18	5 minutes	126 minutes

- ▶ Allow a further 2 minutes per group for questions and transition.
- ▶ Within 2 h 15: 2 minutes of margin in preliminary; 9 minutes in final.
- ▶ Provisional allocation: it may change after incomplete groups are reorganised.

A3: with 37 groups, these slots require 4 h 19. Its presentation format will be announced separately.

ASSIGNMENT 1: EXPECTED WORK

- ▶ Implement matrix factorization with ALS or gradient descent.
- ▶ Study and implement one or more alternative approaches.
- ▶ Compare against a baseline using a clear experimental protocol.
- ▶ Discuss hyperparameters, regularization, data splits and computational cost.
- ▶ Explain what you learned in your report and presentation.

THE STARTER REPOSITORY: INPUT AND OUTPUT

Local example: input/output interface

```
python3 generate.py --name ratings_train.npy
```

The script must produce `output.npy`, containing the completed rating matrix.

- ▶ A simple notebook is acceptable for exploration; provide `generate.py` for automated tests.
- ▶ Preserve the input/output interface of the starter code.
- ▶ List your dependencies in `requirements.txt`.
- ▶ Test from a clean clone before submitting.

▶ Starter repository

▶ Accept assignment A1

A1: DATES AND DELIVERABLES

Date	Milestone
8 October	Slurm introduction and technical support
15 October	Preliminary presentations; slides due 14 October at 23:59
22 October	Final presentations; slides and report due 21 October at 23:59
22 October, 05:00	Last accepted code updates

One presentation per group: preliminary *or* final.

Submit `slides.pdf` and `report.pdf` at the repository root. Late submissions will not be accepted; missing or incorrectly named files will be penalized.