Assignment 1

Collaborative Filtering

Benjamin Negrevergne, Laurent Meunier

PSL University – Paris Dauphine

About this class

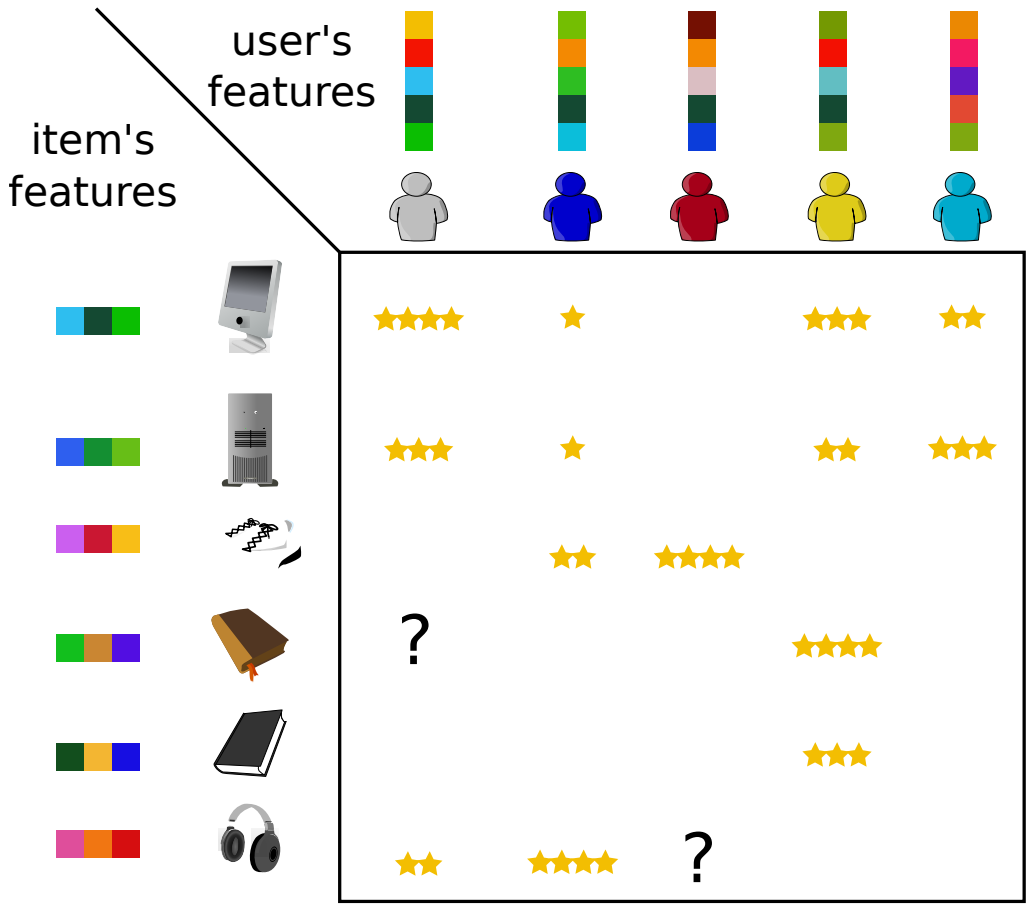
- Focus on selected topics, rather than an exhaustive program
- Mobilize knowledge from the other theoretical courses
- Provide an experience with (fairly) advanced research problems
- Support intuition

About this class

- Focus on selected topics, rather than an exhaustive program
- Mobilize knowledge from the other theoretical courses
- Provide an experience with (fairly) advanced research problems
- Support intuition

- 3 assignments
- 1 general problem
- several approaches
- one application / one theoretical concept

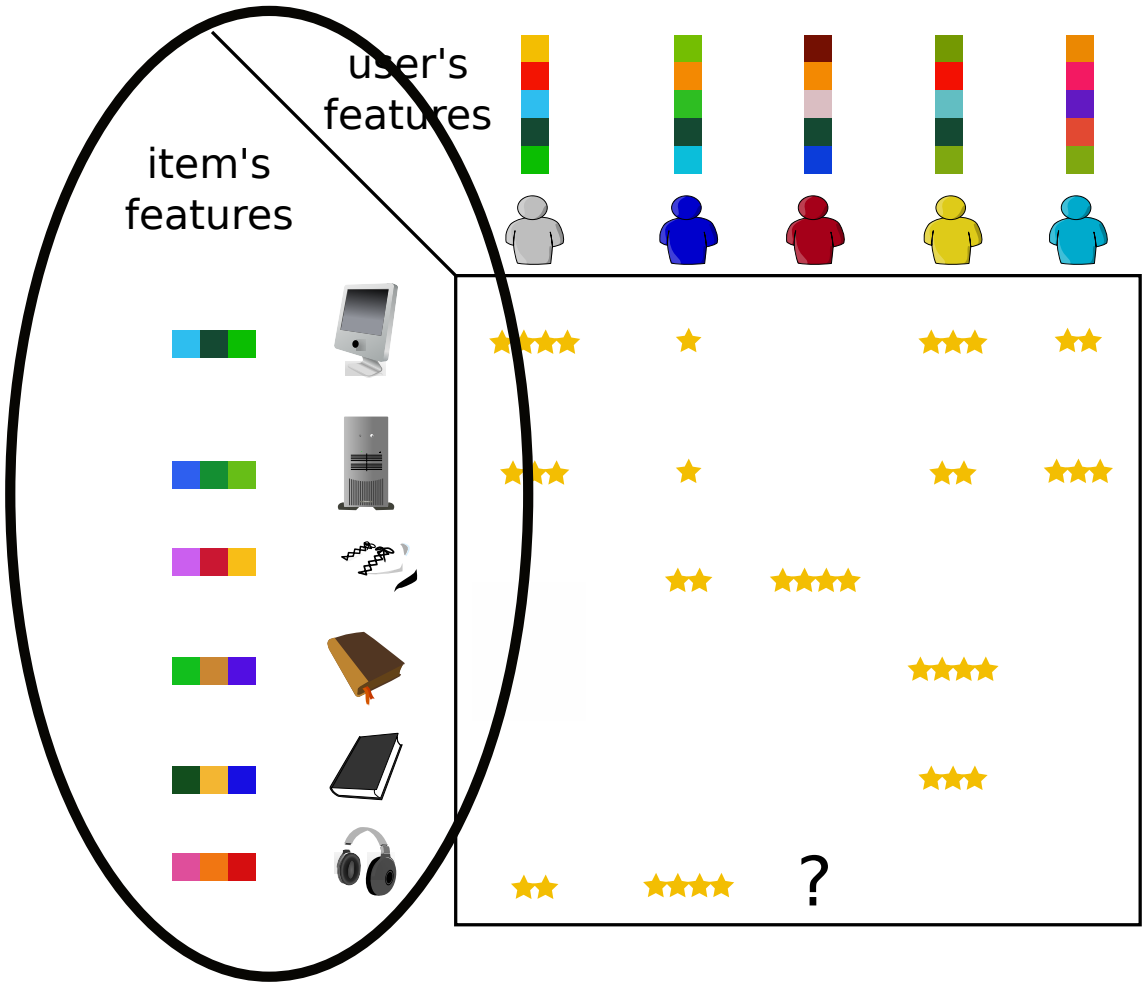
Recommendation: general setting



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

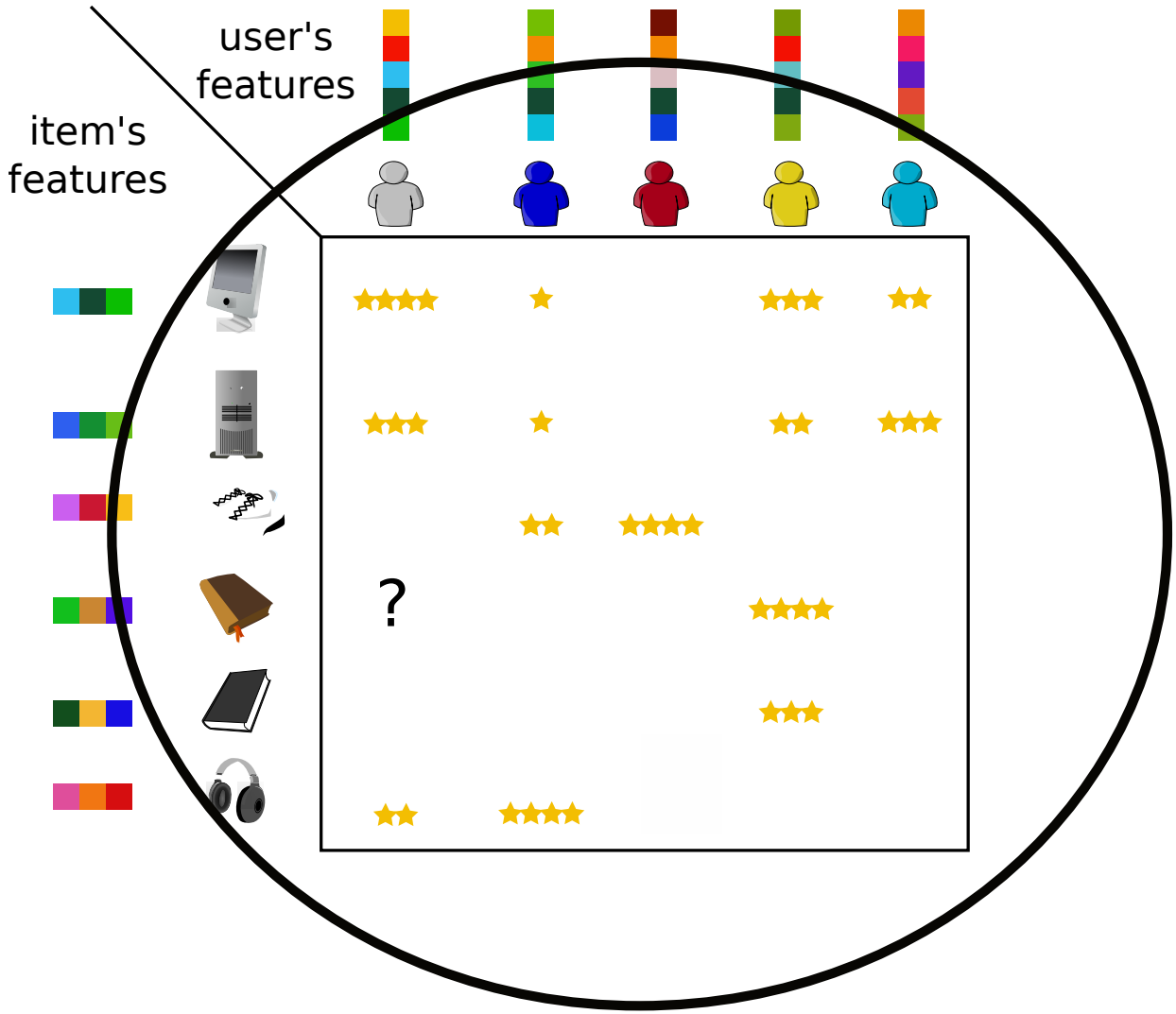
Content based filtering

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



Collaborative filtering

Recommendations based on **user-object relationship**



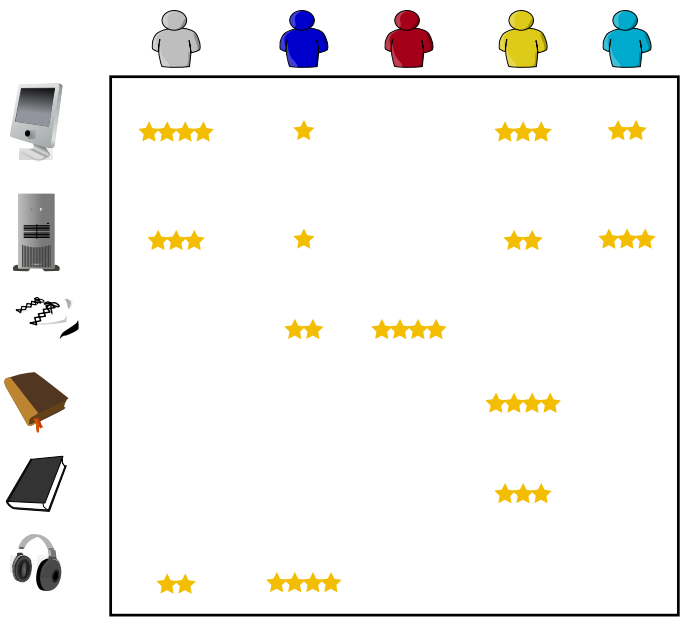
CF vs. CBF in recommender systems

- **Content-Based Filtering**
  - Able to deal with **cold start**
  - Disappointing performances

- **Collaborative filtering**
  - Good performance in practice
  - Unable to deal with cold start

▶ The first assignment will focus on collaborative filtering

Rating matrix



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

Remark

- Value 0 is ambiguous: not rated or rated as zero

  particularly relevant with unitary ratings (e.g. online store)

Outline

User / item -based CF

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

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

User-based CF

Basic algorithm

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

1. For all users v compute $Sim(u, v)$
2. Retain top-k nearest neighbors $v_1, \ldots, v_k$
3. set $\hat{R}_{iu} = \sum_{i=1}^{k} Sim(u, v_i) \cdot R_{iv}$

User-based CF

Basic algorithm

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

1. For all users v compute $Sim(u, v)$
2. Retain top-k nearest neighbors $v_1, \ldots, v_k$
3. set $\hat{R}_{iu} = \sum_{i=1}^{k} Sim(u, v_i) \cdot R_{iv}$

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

$$Sim(u, v) = Pearson(u, v)$$

$$=$$

$$\frac{\sum_{k \in M_u \cap M_v} (r_{ku} - \mu_u) \cdot (r_{kv} - \mu_v)}{\sqrt{\sum_{k \in M_u \cap M_v} (r_{ku} - \mu_u)^2} \cdot \sqrt{\sum_{k \in I_u \cap I_v} (r_{kv} - \mu_v)^2}}$$

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 $\approx$ 2 days to process.

Locality sensitive hashing functions

Properties:

- $h(u) = h(v) \Rightarrow sim(u, v) \geq \alpha$
- $h(u) \neq h(w) \Rightarrow sim(u, w) \leq \beta$

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

Outline

Matrix factorization in CF

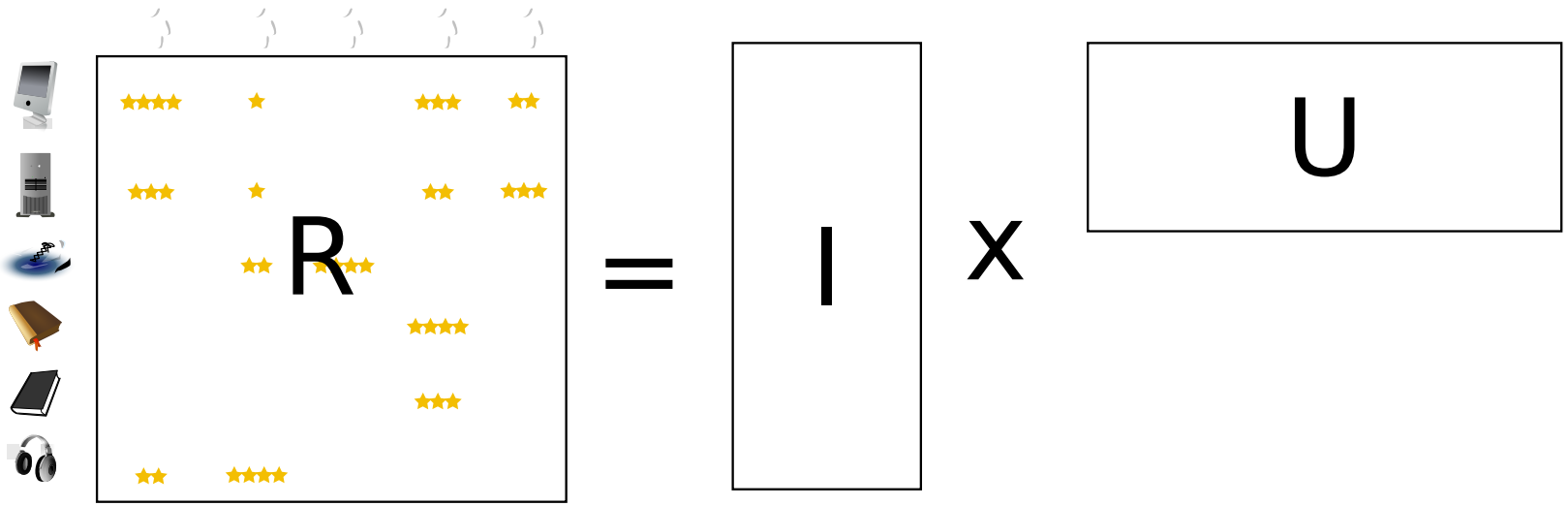
Idea: if ratings are correlated, then R can be approximated with a low rank matrix

Low rank matrix factorization

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

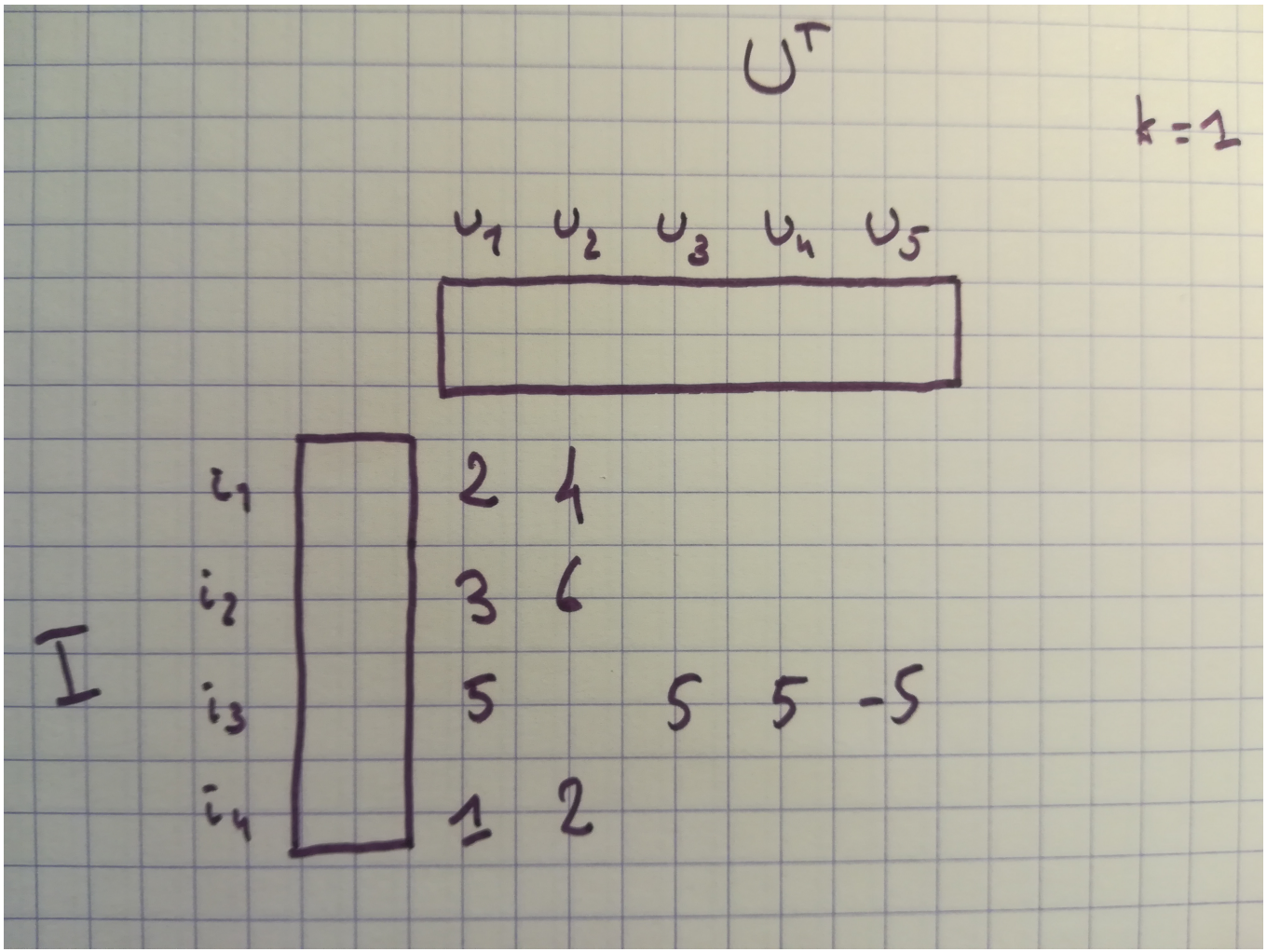
Where

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



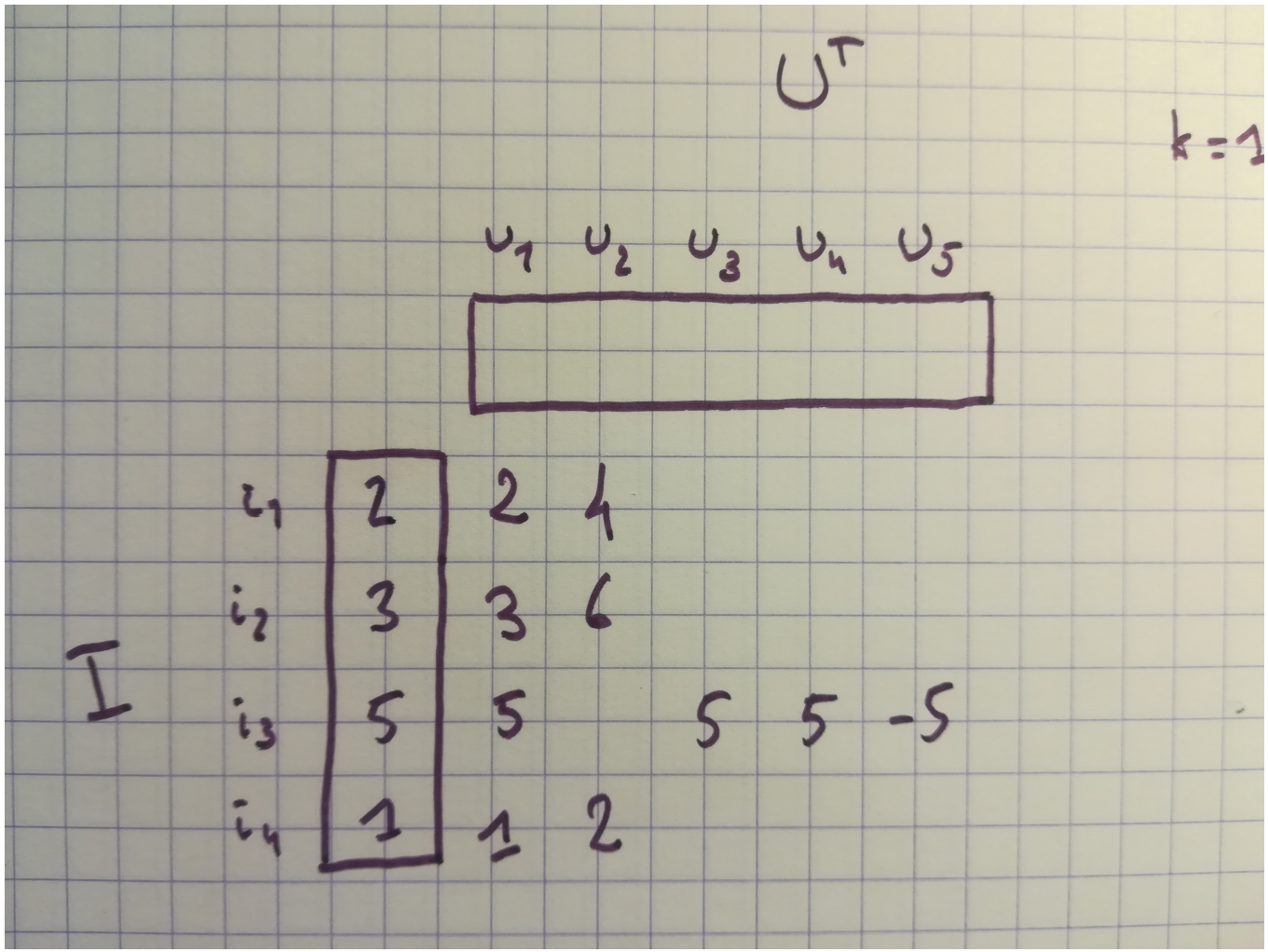
Matrix factorization demo

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



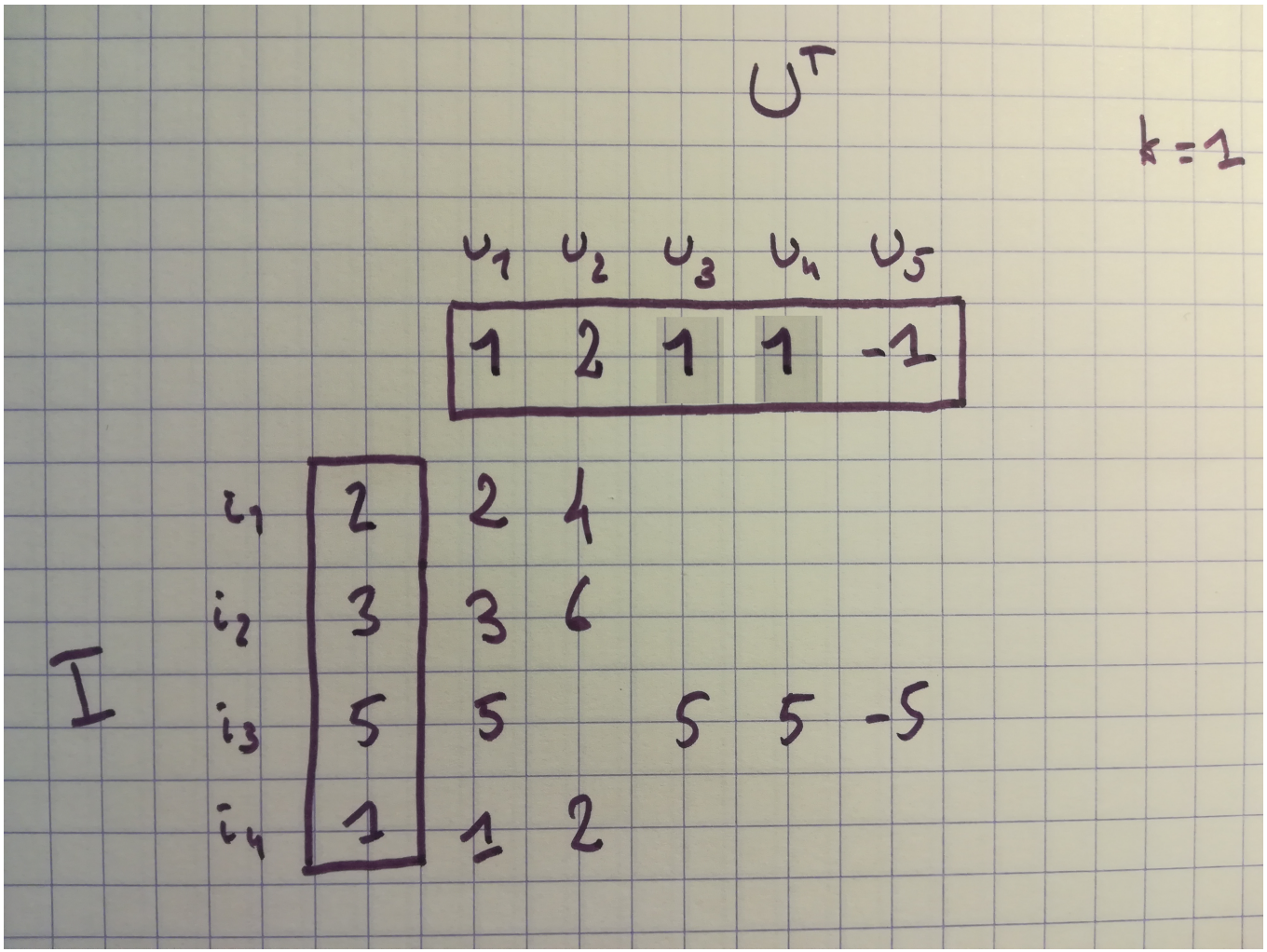
Matrix factorization demo

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



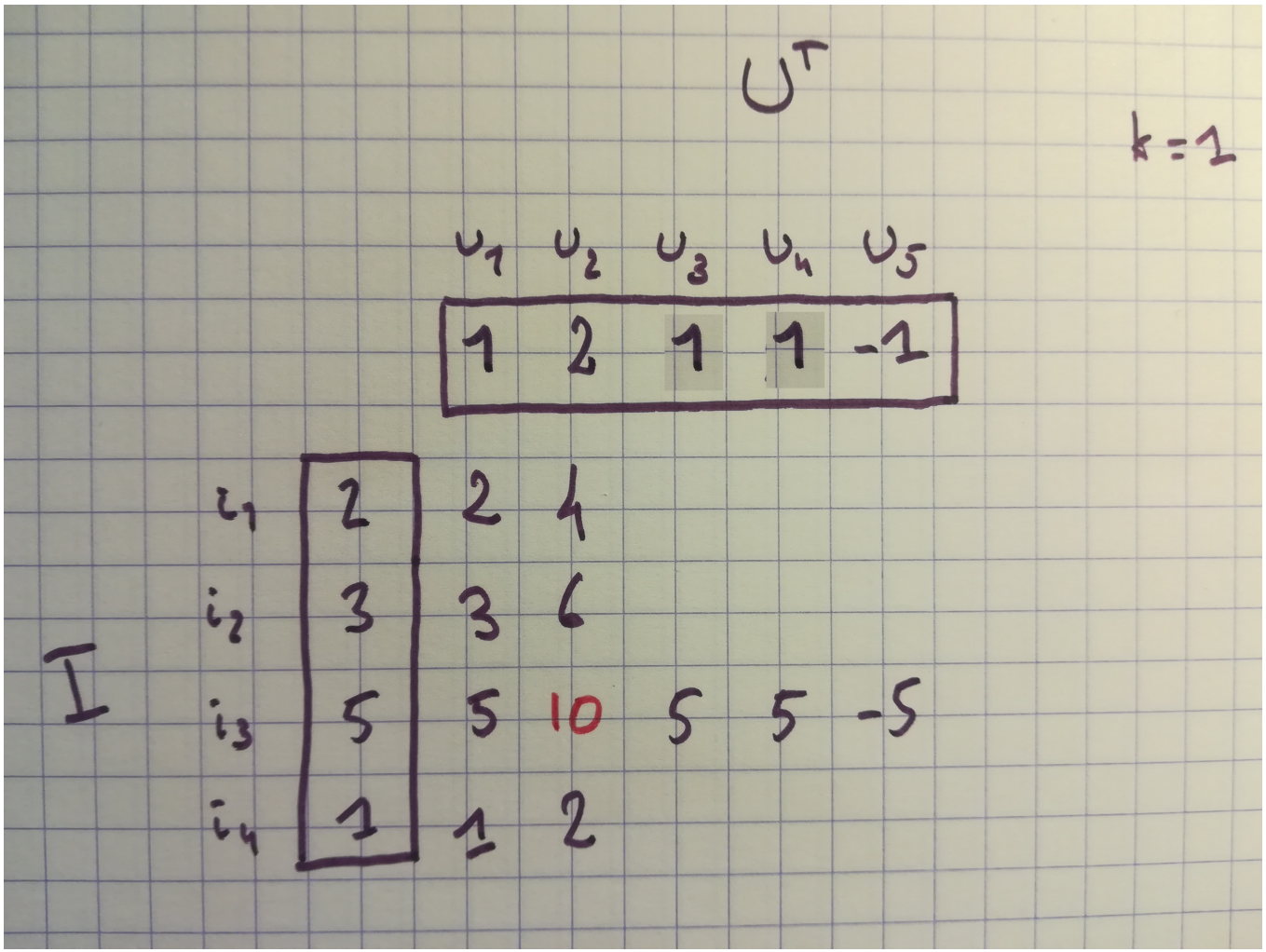
Matrix factorization demo

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



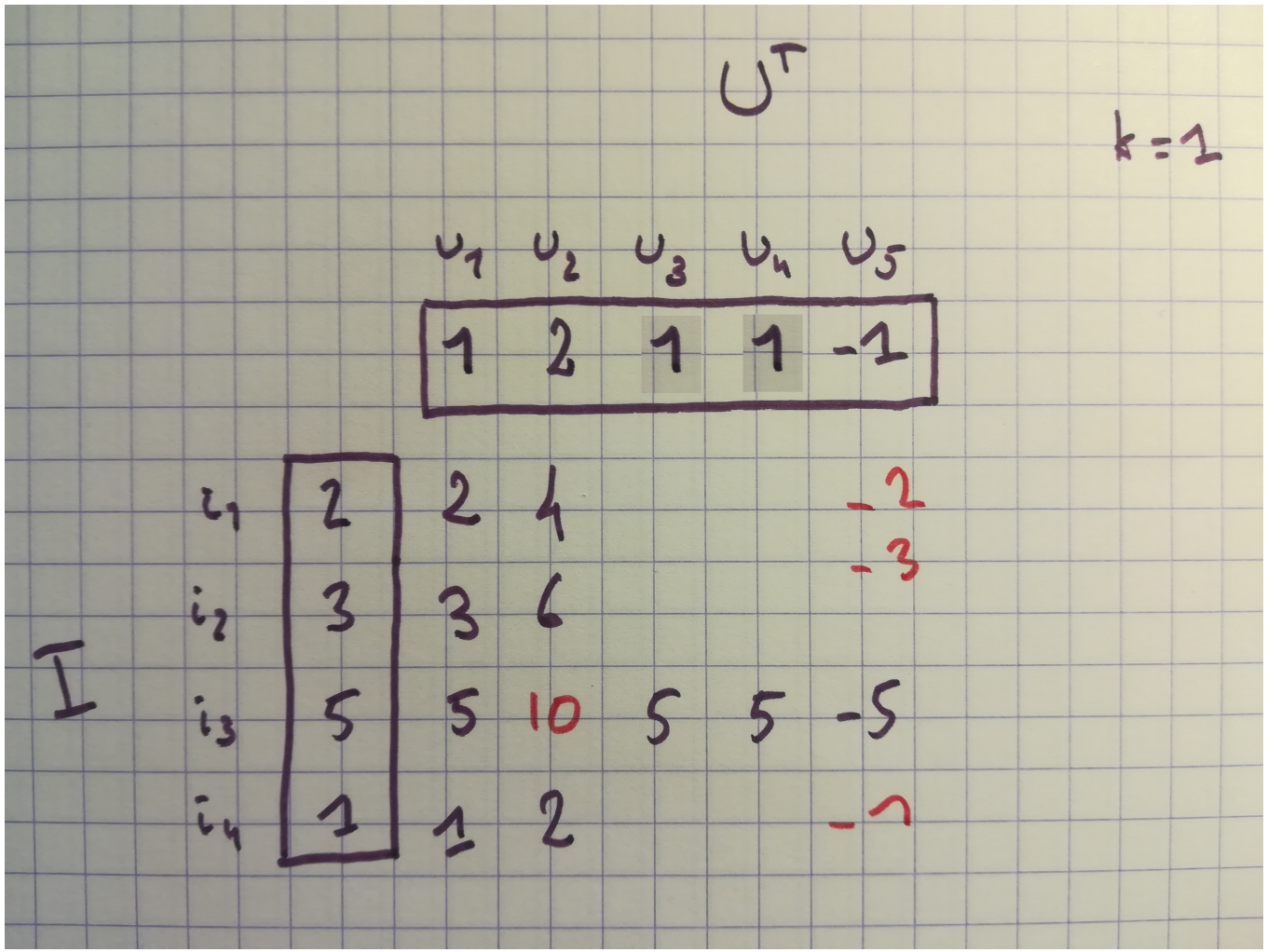
Matrix factorization demo

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



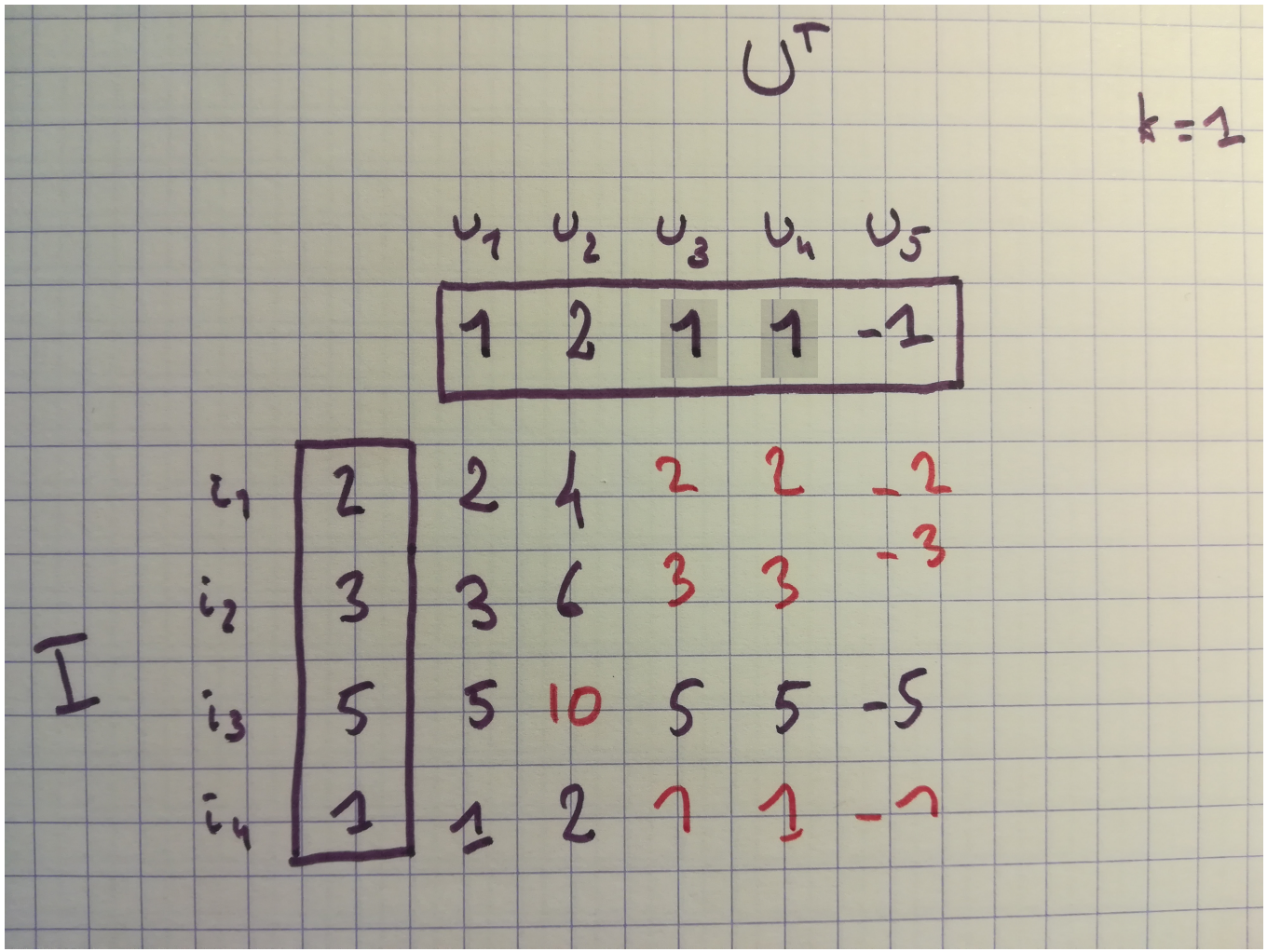
Matrix factorization demo

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



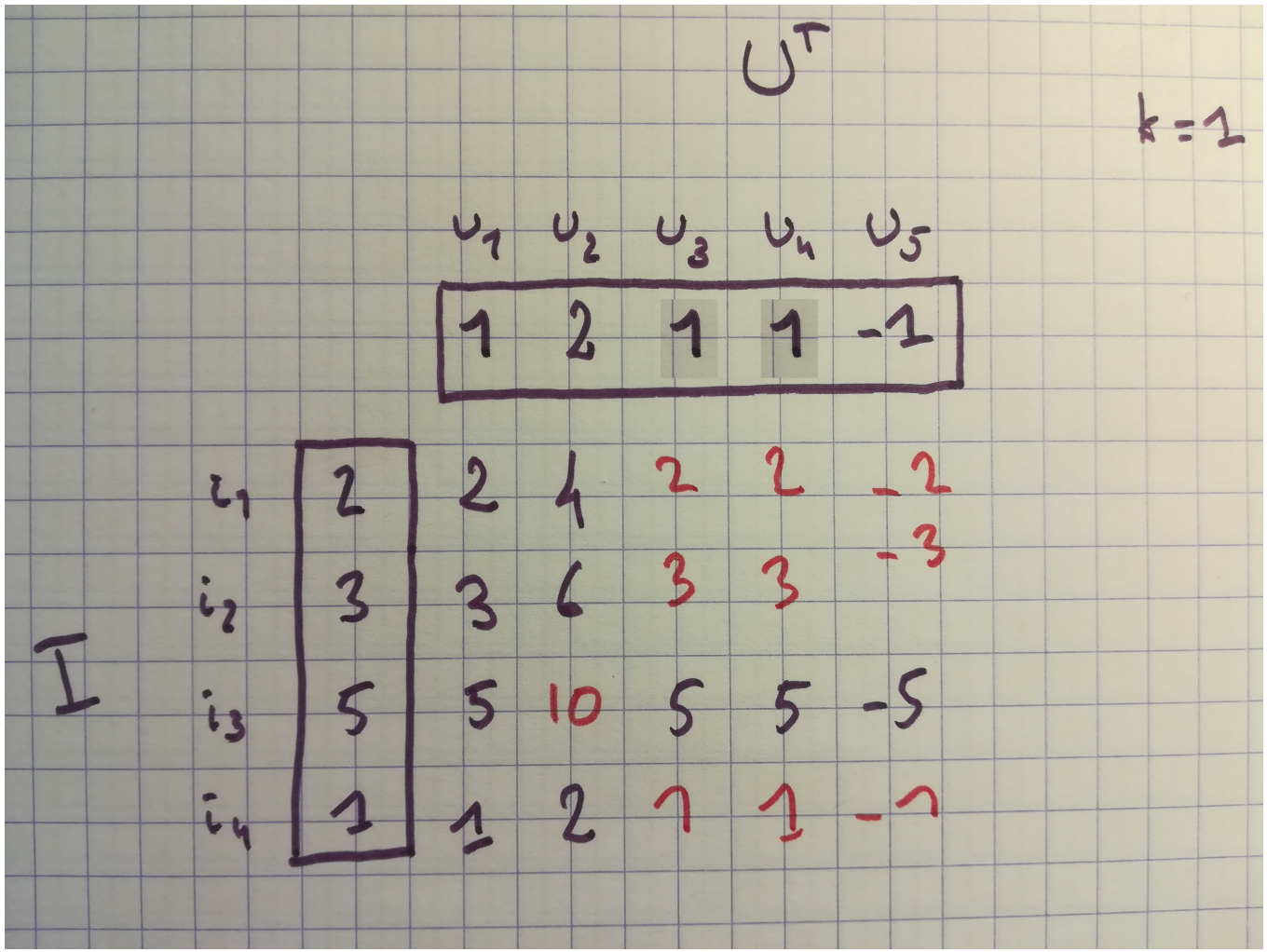
Matrix factorization demo

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



Matrix factorization demo

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



Remark: I and U can be interpreted.

Matrix factorization demo (k=2)

With $k = 2$

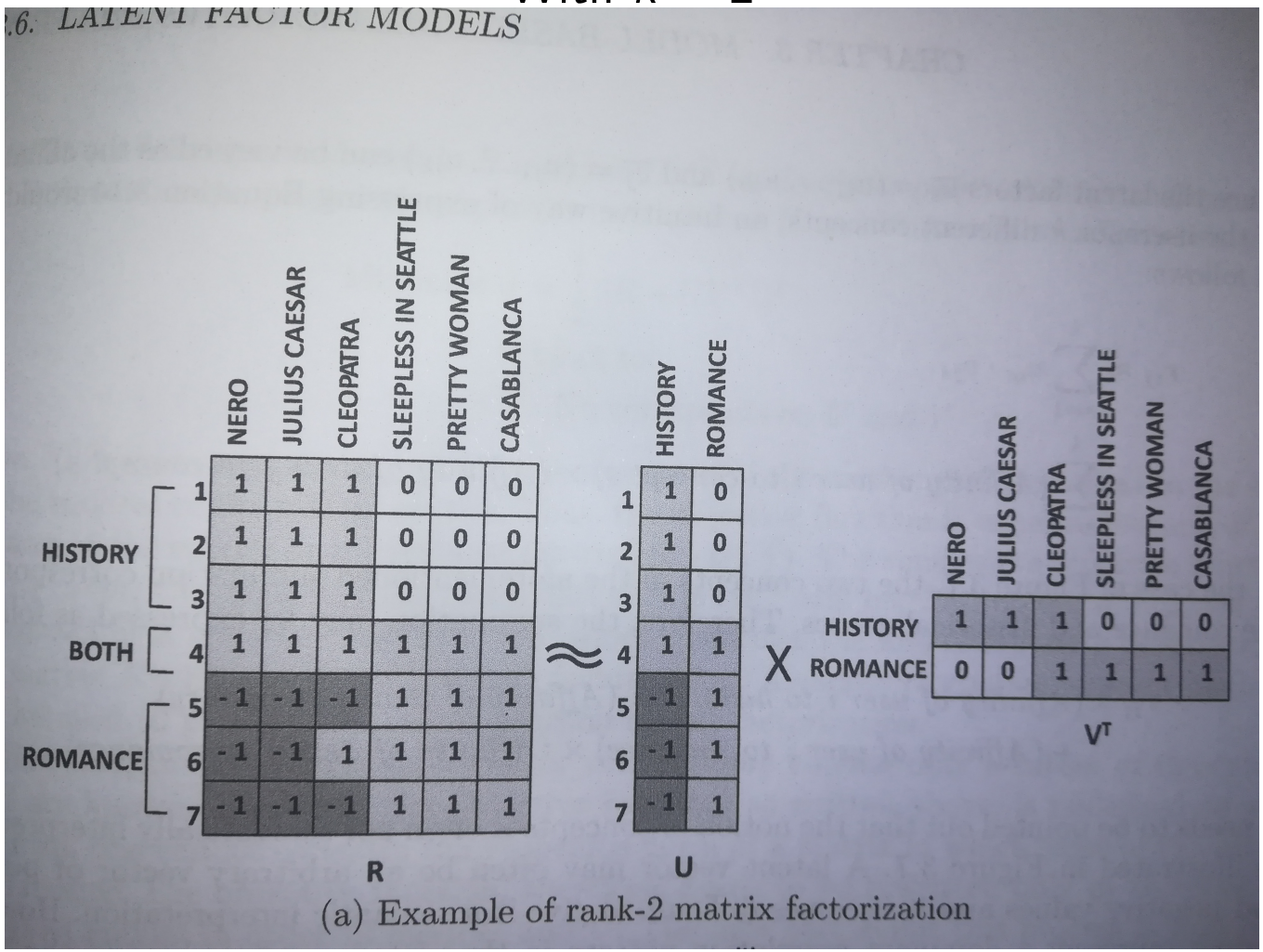


Image from Charu C. Aggarwal, Recommender Systems.

Machine Learning formulation

Regularized loss minimization on fixed-rank matrices
using the Frobenius matrix norm $\|\cdot\|_{\mathcal{F}}$

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

Where:

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

Machine Learning formulation

Regularized loss minimization on fixed-rank matrices

using the Frobenius matrix norm $\|\cdot\|_{\mathcal{F}}$

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

Where:

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

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

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:

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

Role of regularization

(same as always)

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

How to optimize

Cost function C

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

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

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

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

Partial derivatives

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

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: Alternated Least-Square (ALS)

Setting the partial derivatives to zero, we have :

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

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: Alternated Least-Square (ALS)

Setting the partial derivatives to zero, we have :

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

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: Alternated Least-Square (ALS)

Setting the partial derivatives to zero, we have :

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

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: Alternated Least-Square (ALS)

Setting the partial derivatives to zero, we have :

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

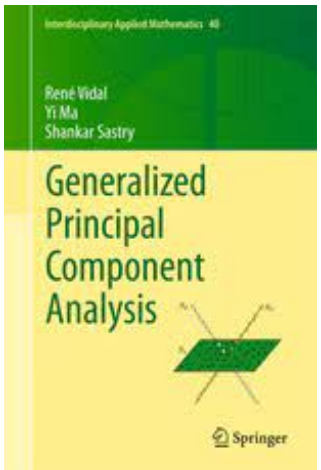
Missing values

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

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

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

Alternative methods



Generalized Principal Component Analysis by René Vidal Yi Ma and S.Shankar Sastry

See section *PCA with Robustness to Missing Entries.*

Deep Matrix factorization

Deep Matrix Factorization Models for Recommender Systems
`https://www.ijcai.org/Proceedings/2017/0447.pdf`

Learn f_θ^U and f_θ^I such that

$$R_{iu} \approx cosine(f_\theta^I(r_{i*}), f_\theta^U(r_{*u}))$$

- $r_{i*} = i^{th}$ row vector of R
- $r_{*u} = u^{th}$ column vector of R

Outline

Bandits

Multi-armed Bandits setting

In casino, you have K machines (armed bandits) and you want to learn which one gives the best reward (which is a random value).
For coherence with the bandit literature, we use the notation $i = 1, ..., K$ set of possible actions, i.e. the arms to pull.

- $t = 1, ..., n$ time
- t action selected at time t
- $X_{i,t}$ reward for action i at time t

Which strategy to adopt to find and exploit the best arm?

Linear Bandits

Cold start users

The users are new and you don't know their preferences yet. Which movie do you propose them?

Idea: Linear bandits

- 1st step: Compute on existing users a matrix $I_1, ...I_m \in \mathbb{R}^k$ of item features
- You have a new user u. We model to make him recommendations.
- Implement linear bandits to learn $\theta_1, ..., \theta_n$

Linear UCB algorithm

For $t = 1...n$:

1. The user u asks for a recommendation. You want to propose him an item I_j.
2. The learner choose an action $a_t \in \{1...m\}$ to recommend a movie and will get a reward $r_{t,a_t} = \theta_{a_t}^T I_{a_t} + \epsilon$ where ϵ in a centered sub-gaussian noise.
3. a_t is chosen via LinearUCB

Linear UCB policy

$$a_t \in \arg\max_a \hat{\theta}_a^T I_a + \alpha \sqrt{I_a^T (D_a^T D_a + I)^{-1} I_a}$$

where:

- $T_a = \{t : a_t = a\}$.
- $D_a \in \mathbb{R}^{|T_a| \times d}$ is the design matrix of all the contexts observed when action a has been taken.
- $c_a \in \mathbb{R}^{|T_a| \times d}$ is the reward vector of all the rewards observed when action a has been taken.
- $\hat{\theta}_a = (D_a^T D_a + I)^{-1} D_a^T c_a$

Outline

Optimal transport

Monge-Kantorovich problem: coupling between two probability distributions. Let suppose you have two space X and Y and two probability measures μ and ν respectively on X and Y. Let c be a positive functions defined on $X \times Y$.
The Wasserstein metric is defined as:

$$W_c(\mu, \nu) = \min_{\pi \in \Pi(\mu,\nu)} \mathbb{E}_{(x,y)\sim\pi}(c(x, y))$$

where $\Pi(\mu, \nu)$ is the space of probability distributions on $X \times Y$ with marginals μ and ν.
You want to find the best matching π between the two distributions μ and ν. It is then a matching problem

Cost learning

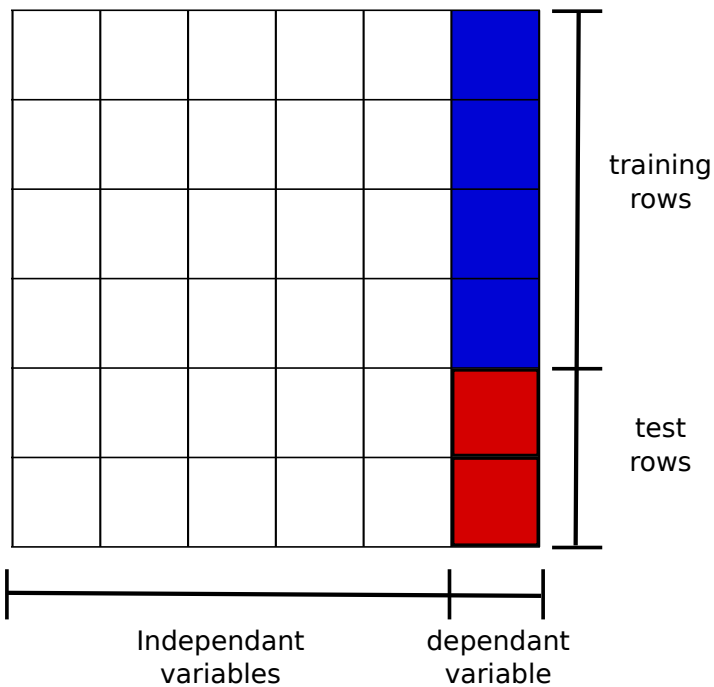
You have observed matchings (user-items), you aim to learn what cost c fit the best the observed matching.
Then you use this cost c to determine which item to recommend. To do so, suppose you observed a matching $\hat{\pi}$ for a distribution of users $\hat{\mu}$ and one of items $\hat{\nu}$. Then you may solve the following inverse problem:

$$\min_{c \text{ s.t. } \pi \in \arg\min_{\pi \in \Pi(\hat{\mu},\hat{\nu})} \mathbb{E}_{(x,y)\sim\pi}(c(x,y))} KL(\hat{\pi}||\pi)$$

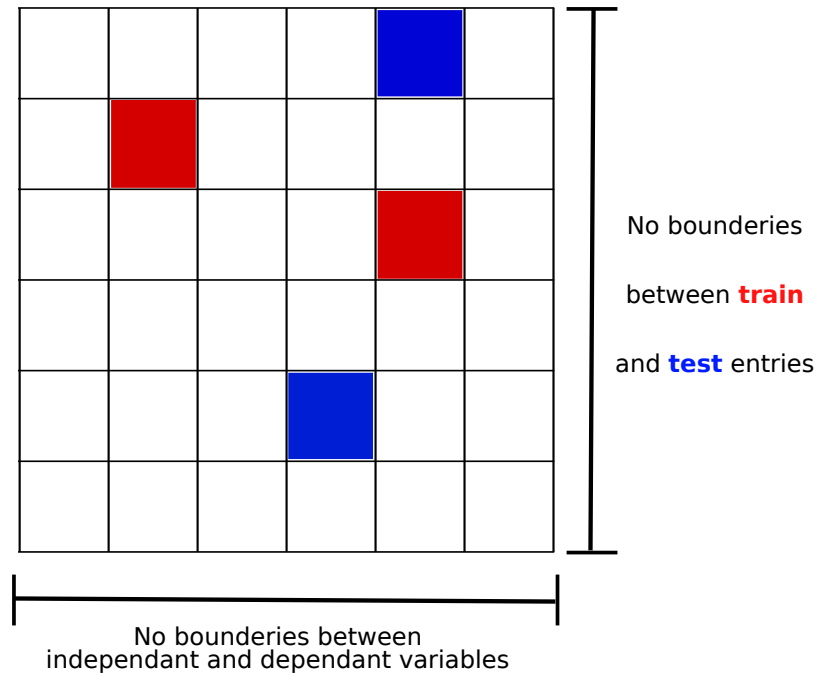
This project is more difficult, but a bit of investment inside will help you a lot in the future since Optimal transports is widely used in Machine Learning nowadays.

Outline

CF vs. the classification setting



Standard classification



Collaborative filtering

The MovieLens Dataset

GroupLens (U. Minnesota)
`https://grouplens.org/datasets/movielens/`

- **Small:** 9000 movies, 600 users, 100 000 ratings
- **Full:** 27M ratings, 58 000 movies, 280 000 users
- **1B:** synthetic dataset with 1B ratings, 855K movies, 2M users
  Expanded from the real dataset 20M.

Note: with $n = 58\,000$ and $m = 280\,000$, an $\mathcal{O}(n \cdot m)$ algorithm running at 10 000 step / second takes ≈ 19 days to process.

Evaluation metric

To evaluate the quality of your predictions, we will use the *Rooted 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 in the original rating matrix (sparse)
- $\hat{R}$ is the estimated rating matrix (dense)
- $T = \{(u, i) \mid R_{iu} \text{ is in the testing set } \}$

Planning

Please check the course website at:
`https://www.lamsade.dauphine.fr/~bnegrevergne/ens/ProjetDataScience/`

Expected work

For each assignment, you are expected to:

- Get familiar with the general problem, and all approaches discussed in class
- Conduct a bibliographic study, and choose possible solutions to study thoroughly
- Implement several solutions, and conduct a **comparative**, **interesting**, experimental study (with your own experimental results)
- Present your work and teach the rest something to the rest of the class
- Write a small report

Source code/results

Code, slides and reports must be uploaded on github.

Classroom

How to join the classroom and get a github repository

1. Create a github account (or use an existing one)
2. Join the git classroom for assignment 1 (see course website)
3. Click on your name in the list
   Warning: Do not click on someone else's name!
4. Create a group, or join an existing group
   Warning: Do not create a team without coordinating with your partners!