The Data Science Lab

Alexandre Vérine, Constant Bourdrez
(Benjamin Negrevergne)

PSL University – ENS PSL

Data Science Lab

What?

- 3 group assignments, focused on a series of selected problems
  - Recommendation with Collaborative filtering
  - Quality vs. diversity in Generative Models
  - Adversarial attacks in Classification Models
- For each problem, several alternative approaches are presented / discussed
- Students implement one approach of their choice, and compare with baselines
- Ideas & results are discusses during oral presentations

Data Science Lab

What?

- 3 group assignments, focused on a series of selected problems
  - Recommendation with Collaborative filtering
  - Quality vs. diversity in Generative Models
  - Adversarial attacks in Classification Models
- For each problem, several alternative approaches are presented / discussed
- Students implement one approach of their choice, and compare with baselines
- Ideas & results are discusses during oral presentations

Educational goals

- Apply theoretical knowledge acquired during other classes & support intuition
- Practice reading scientific publications and giving oral presentations
- Explore diverse research problems
- Go beyond the concept of traditional scholar evaluation (i.e. focus producing insights)

Course's website

Planning & info at:

`https://www.lamsade.dauphine.fr/~averine/Datalab/`

Or at :
`https://www.alexverine.com` - Teaching - Datalab

Assignment 1

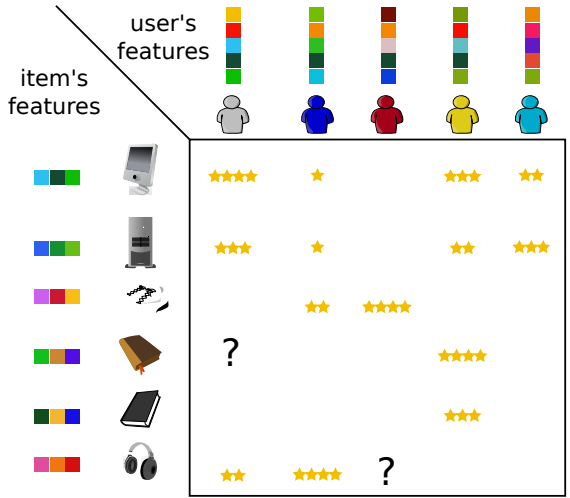
Collaborative Filtering

Alexandre Vérine, Constant Bourdrez
(Benjamin Negrevergne)

PSL University – ENS PSL

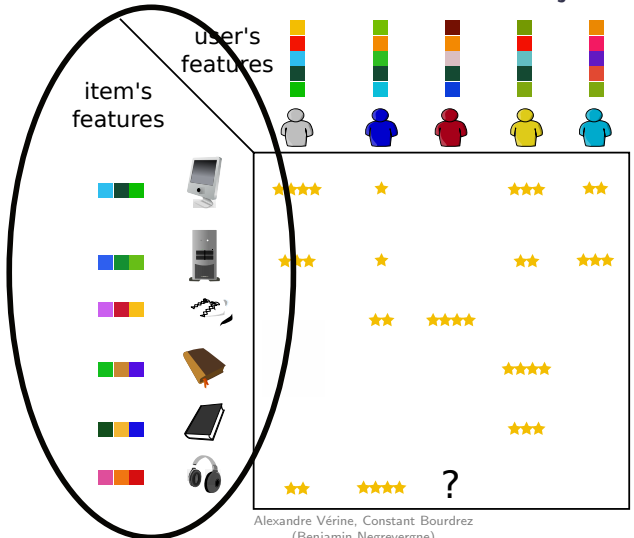
MILES
Machine Intelligence and Learning Systems

Recommendation: general setting



Content based filtering

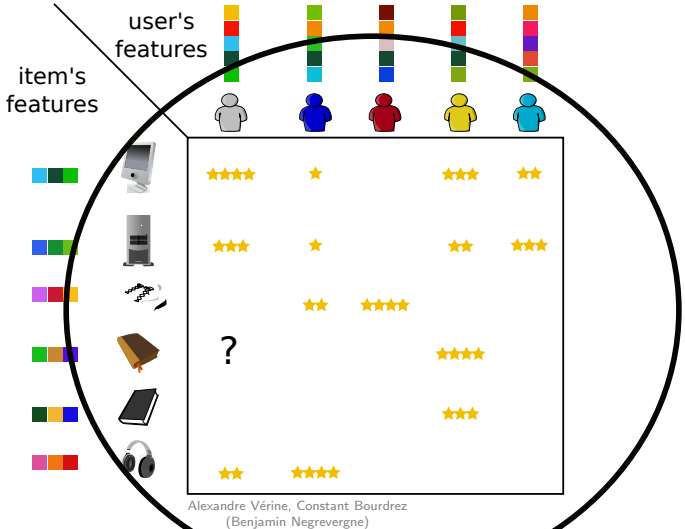
Recommendations based on **intra-user or intra-object relationships**



Alexandre Vérine, Constant Bourdrez
(Benjamin Negrevergne)

Collaborative filtering

Recommendations based on **user-object relationship**



Alexandre Vérine, Constant Bourdrez
(Benjamin Negrevergne)

CF vs. CBF in recommender systems

Content-Based Filtering

- Able to deal with **cold start**
- Requires user/item features
- Disappointing performances

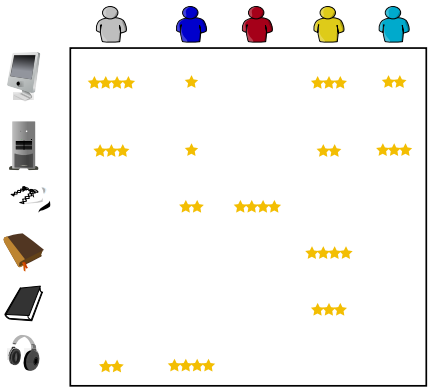
Collaborative filtering

- Works with ratings only
- Good performance in practice
- Unable to deal with cold start

▶ The first assignment will focus on collaborative filtering

Outline

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)

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

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 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 for CF

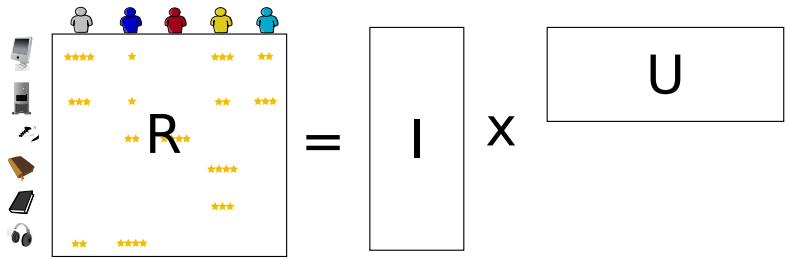
Idea: if ratings are correlated, then R can be approximated with a low rank matrix

Low rank matrix factorization

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

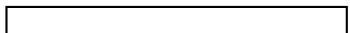
Where

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



Matrix factorization demo

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



$U^\top$

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

1	2	1	1	−1

$U^\top$

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

1	2	1	1	−1

$U^\top$

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

1	2	1	1	−1

$U^\top$

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

Matrix factorization demo

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

1	2	1	1	−1

$U^\top$

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

Matrix factorization demo

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

1	2	1	1	−1

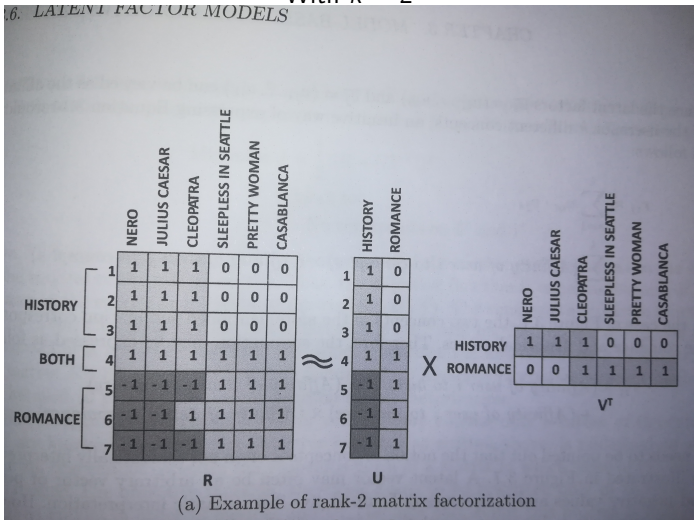
$U^\top$

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 (k=2)

With $k = 2$



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 :

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

Missing values

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

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

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

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.*

- Lapacian Embedding - Dimension reduction with a local only approach

- Deep Matrix Factorization

- Sparse Knn

- Optimal Transport based methods

Incroporating background knowledge

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

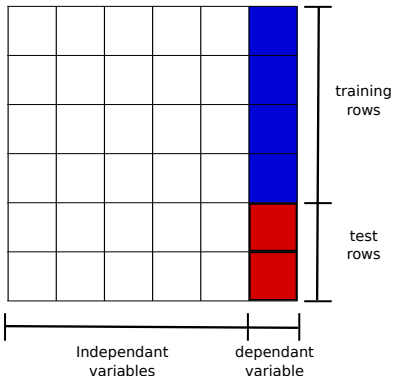
- Train multiple independent models
- Incorporate features into embedding
- ...

Outline

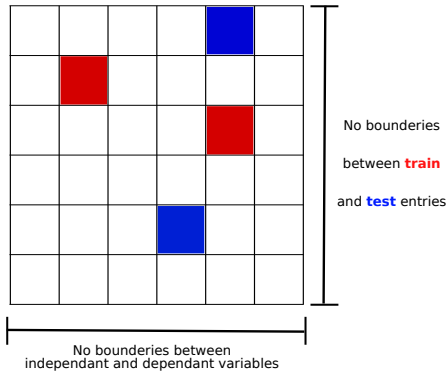
A Copy of the Movie Lens Dataset

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

Classification vs. Collaborative Filtering

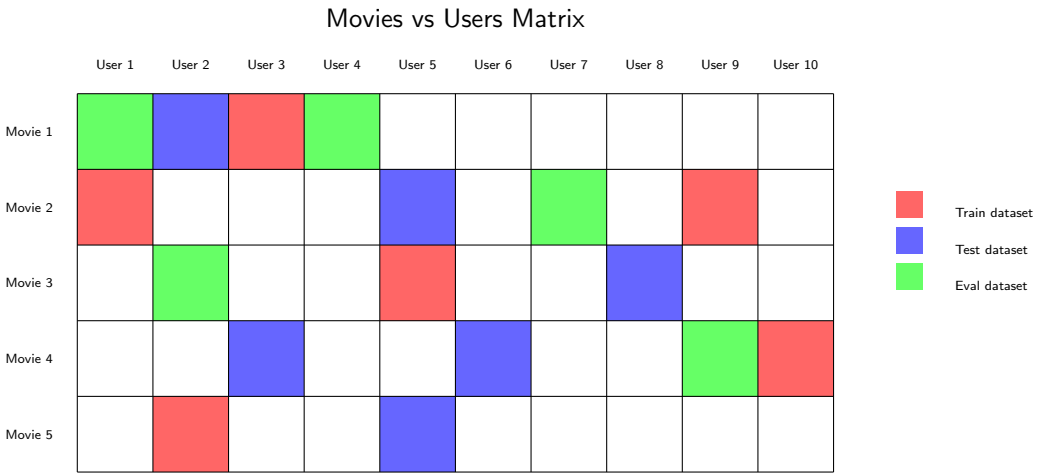


Standard classification



Collaborative filtering

Our setting


Movies vs Users Matrix
User 1
User 2
User 3
User 4
User 5
User 6
User 7
User 8
User 9
User 10
Movie 1
Movie 2
Movie 3
Movie 4
Movie 5
Train dataset
Test dataset
Eval dataset

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

The Evaluation Platform

You will be evaluated on a platform based on:

- RMSE
- Accuracy for exact ratings
- Time

Outline

Expected work

- Implement at least MF with ALS or Gradient Descent
- Implement one/several alternative method of your choice
  - LSH, PCA, Optimal transport, etc.
- Write a small report & present your work to the class
  - Discuss hyperparameter tuning (e.g. how to set k)
  - Discuss how to incorporate genre (or other features)
  - Theoretical comparison vs. baseline
  - Experimental comparison vs. baseline

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!

Typical errors to avoid.

- Don't copy paste the lecture in your slides and report.
- Don't use screenshot of equation
- Don't use screenshot of your code.
- Go check what the PALM method is and don't mention it in your slides or your report.