Master on Foundations of Data Science

Recommender Systems

Collaborative Recommender Systems (II)

Santi Seguí | 2019-2020

Today

- RecSys Challenge Task
- How are we gonna survive to COVID-19
- Factorization Models

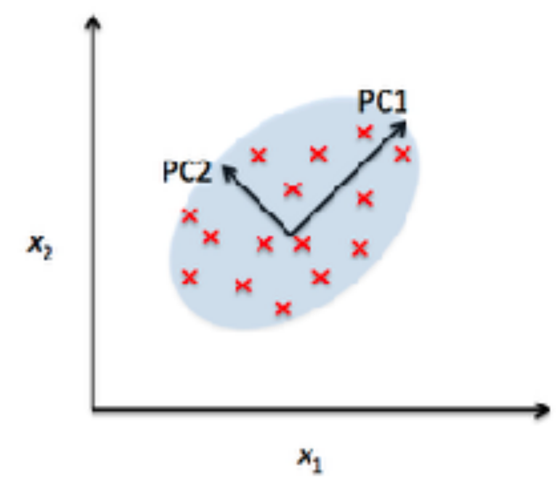
How we will survive to COVID-19

- Course Content will be provided by:
  - Weekly call (20-30 min) + questions
  - Slides + Papers
  - Notebooks
- Evaluation will be based on:
  - RecSys Challenge + 2 new **individual** assigments
  - NO EXAM

Dimensionality Reduction

- Pairwise similarities are hard to robustly be computed in sparse matrices.
- Dimensionality reduction can be used to **improve** neighborhood-based methods both in terms of **quality** and in terms of **efficiency**
- A reduced representation of the data can be created in terms of either row-wise latent factors or in terms of column-wise latent factors.

Dimensionality Reduction



Items

users

≈

Latent Factors

users

m x n **sparse matrix**

m x d **matrix**
where d<<n

Dimensionality Reduction

- The low-dimensional representation can be computed using **PCA** or **SVD-Like** methods.
- After the d-dimensional representation of each user is estimated, the similarity between users can be computed
- Cosine or dot product on the reduced vectors can be used in order to compute the similarity
- More robust since the feature vector is fully specified
- More efficient

Dimensionality Reduction

- How to **obtain** the **d-dimensional representation** on the sparse matrix?
- **SVD Method**. Steps:
  - Augment the *m x n* incomplete rating matrix R -> R_f
    - Mean-user rating or mean-item rating for each row/column
  - Lets define the similarity matrix S as $\mathbf{S = R_f^T R_f}$. S is a positive semi-definite of size *n x n*
  - Determine the dominat basis vectors of R_f by computing the **diagonalization** of the similarity matrix S.
    - $S = P \Lambda P^T$, where P is an *n x n* matrix, whose columns contain the orthonormal eigenvectors of S. Λ is a diagonal matrix containing the non-negative eigenvalues of S along its diagonal.
  - Let denote P_d the *n x d* matrix only containing the columns of P with the largest eigenvalues
  - The low representation of R is obtained by the multiplication of $\mathbf{R_f P_d}$

Dimensionality Reduction

- How to **obtain** the **d-dimensional representation** on the sparse matrix?
- **PCA Method**. Steps:
    - Augment the *m x n* incomplete rating matrix R -> R_f
        - Mean-user rating or mean-item rating for each row/column
    - Lets define the similarity matrix S as **the Covariance Matrix of Rf**
    - Determine the dominat basis vectors of R_f by computing the diagonalization of the similarity matrix S.
        - $S = P\Lambda P^T$, where P is an *n x n* matrix, whose columns contain the orthonormal eigenvectors of S. Λ is a diagonal matrix containing the non-negative eigenvalues of S along its diagonal.
    - Let denote P_d the *n x d* matrix only containing the columns of P with the largest eigenvalues
    - The low representation of R is obtained by the multiplication of $\mathbf{R_f\, P_d}$

Challenges on Factorization

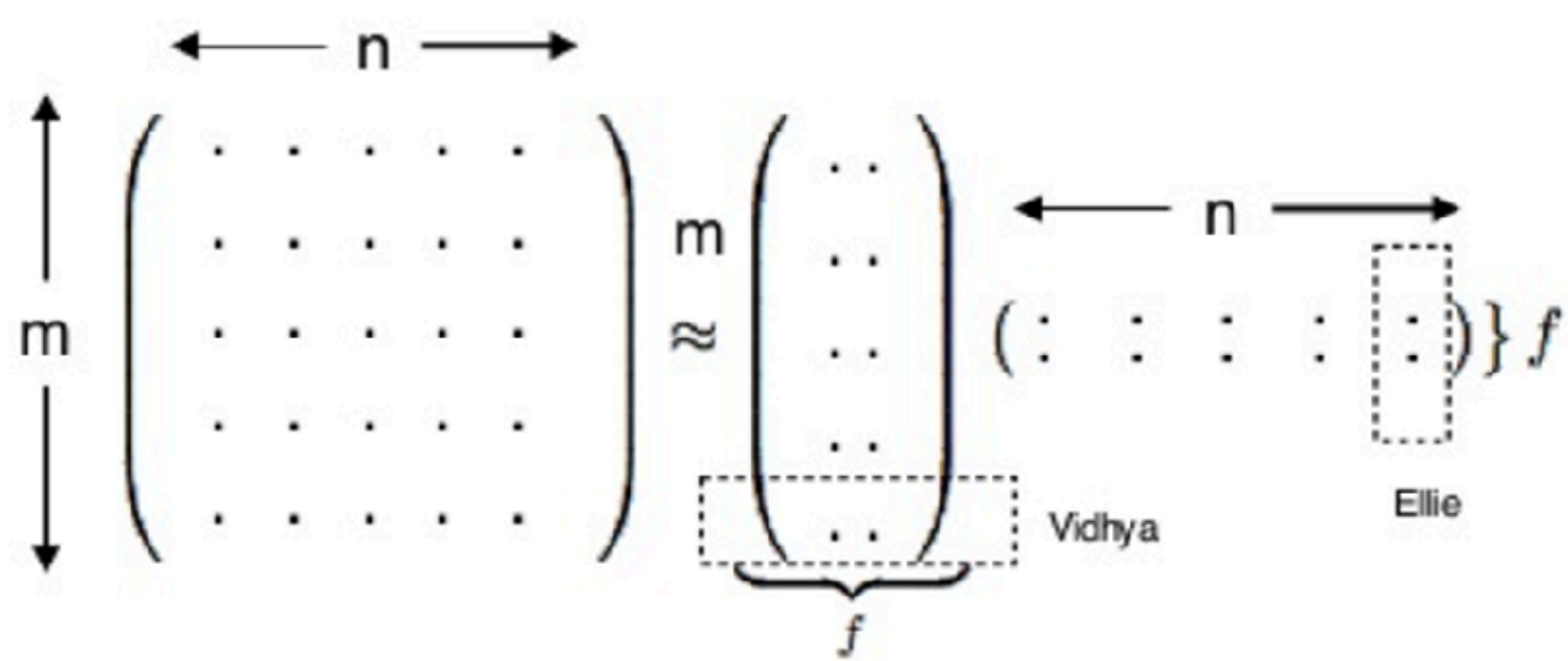
- Challenges:
  - Missing Values
    - Need a way to fill it
    - Several alternatives, including clever averages and predictions
  - Computational Complexity
  - Lack of transparency/explainability

Model-Based methods

Model-Based methods

- **Neighborhood-based** models are very popular because its **simplicity**, today, these methods are not necessarily the most accurate ones.
  - based on distance between user/items.
  - Knn-generalization
- Today, model-based methods and in particular **latent factor models** are some of the most **accurate** methods
  - learn best parameters for the task

Latent Factor Models



Latent Factor Models

$$R = UV^T$$

$$R \approx UV^T$$

$$r_{ij} \approx \bar{u_i} \cdot \bar{v_j}$$

$$= \sum_{s=1}^{k} (\text{Affinity of user I to concept s}) \cdot (\text{Affinity of item j to the concept s})$$

Latent Factor Models

	Nero	Julius Cesar	Cleopatra	Sleepless in Seattle	Pretty Woman	Casablanca
U1	1	1	1	0	0	0
U2	1	1	1	0	0	0
U3	1	1	1	0	0	0
U4	1	1	1	1	1	1
U5	-1	-1	-1	1	1	1
U6	-1	-1	1	1	1	1
U7	-1	-1	-1	1	1	1

R

=

History	Romance
1	0
1	0
1	0
1	1
-1	1
-1	1
-1	1

U

x

	Nero	Julius Cesar	Cleopatra	Sleepless in Seattle	Pretty Woman	Casablanca
History	1	1	1	0	0	0
Romance	0	0	1	1	1	1

V

Model-Based methods

- Advantages over neighborhood methods:
  - **Space-efficency.** Usually, the learned model is much smaller than the original rating matrix
  - **Training speed and prediction speed.** Neighborhood-based models takes quadratic complexity in either number of users or number of items
  - **Avoiding overfitting.** The summarized model use to help in avoiding overfitting. Regularization methods can be used to make these models robust

Latent Factor Models

Matrix factorization models map both users and items to a joint latent factor space of dimensionality, such that user-item interactions are modeled as inner products in that space

Matrix Factorization Methods

- Latent factor models approach tries to explain the ratings by characterizing both items and users to a small number of factors inferred from the rating patterns.

- Singular Value Decomposition (SVD) is a well established technique for identifying latent semantic factors. Done by factorizing the user-item rating matrix.

- **PROBLEM**: Hard to optimize due to the huge amount of missing values

Singular Value Decomposition (SVD)

- Columns of U and V are constrained to be mutually orthogonal.

- Mutual orthogonality has the advantage that the concepts can be completely independent of one another. Can be interpreted in scatterplots

$$\underset{m \times n}{\overset{\hat{X}}{\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix}}} \approx \underset{m \times r}{\overset{U}{\begin{pmatrix} u_{11} & \cdots & u_{1r} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{pmatrix}}} \underset{r \times r}{\overset{S}{\begin{pmatrix} s_{11} & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{pmatrix}}} \underset{r \times n}{\overset{V^{\top}}{\begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix}}}$$

Example of Singular Value Decomposition

$$R_f = \begin{pmatrix} 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & ? & -1 & -1 & -1 \\ ? & 1 & 1 & -1 & -1 & ? \\ 1 & -1 & -1 & 1 & 1 & 1 \\ 1 & ? & -1 & 1 & 1 & 1 \end{pmatrix}$$

The original Matrix

Example of Singular Value Decomposition

$$R_f = \begin{pmatrix} 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -0.2 & -1 & -1 & -1 \\ 0 & 1 & 1 & -1 & -1 & 0 \\ 1 & -1 & -1 & 1 & 1 & 1 \\ 1 & 0.2 & -1 & 1 & 1 & 1 \end{pmatrix}$$

Step 1: Fill missing values with the mean value of the row

Example of Singular Value Decomposition

$$R_f = \begin{pmatrix} 1.0592 & -1.1604 & 0.9716 & -0.8515 & 0.8040 & -1.0592 \\ 0.6636 & 0.9039 & 0.5881 & -0.9242 & -1.1244 & -0.6636 \\ 0.4300 & 0.9623 & 0.3746 & -0.6891 & -1.1045 & -0.4300 \\ -0.9425 & -0.8181 & -0.8412 & 1.2010 & 1.1320 & 0.9425 \\ -1.0290 & -0.2095 & -0.9270 & 1.1475 & 0.5535 & 1.0290 \end{pmatrix}$$

Step 2: Apply SVD to the matrix

Example of Singular Value Decomposition

$$R_f = \begin{pmatrix} 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & 0.5881 & -1 & -1 & -1 \\ 0.4300 & 1 & 1 & -1 & -1 & -0.43 \\ 1 & -1 & -1 & 1 & 1 & 1 \\ 1 & -0.2095 & -1 & 1 & 1 & 1 \end{pmatrix}$$

Step 3. Modify the target matrix setting in the missing values the learn values and iterate **until convergence**

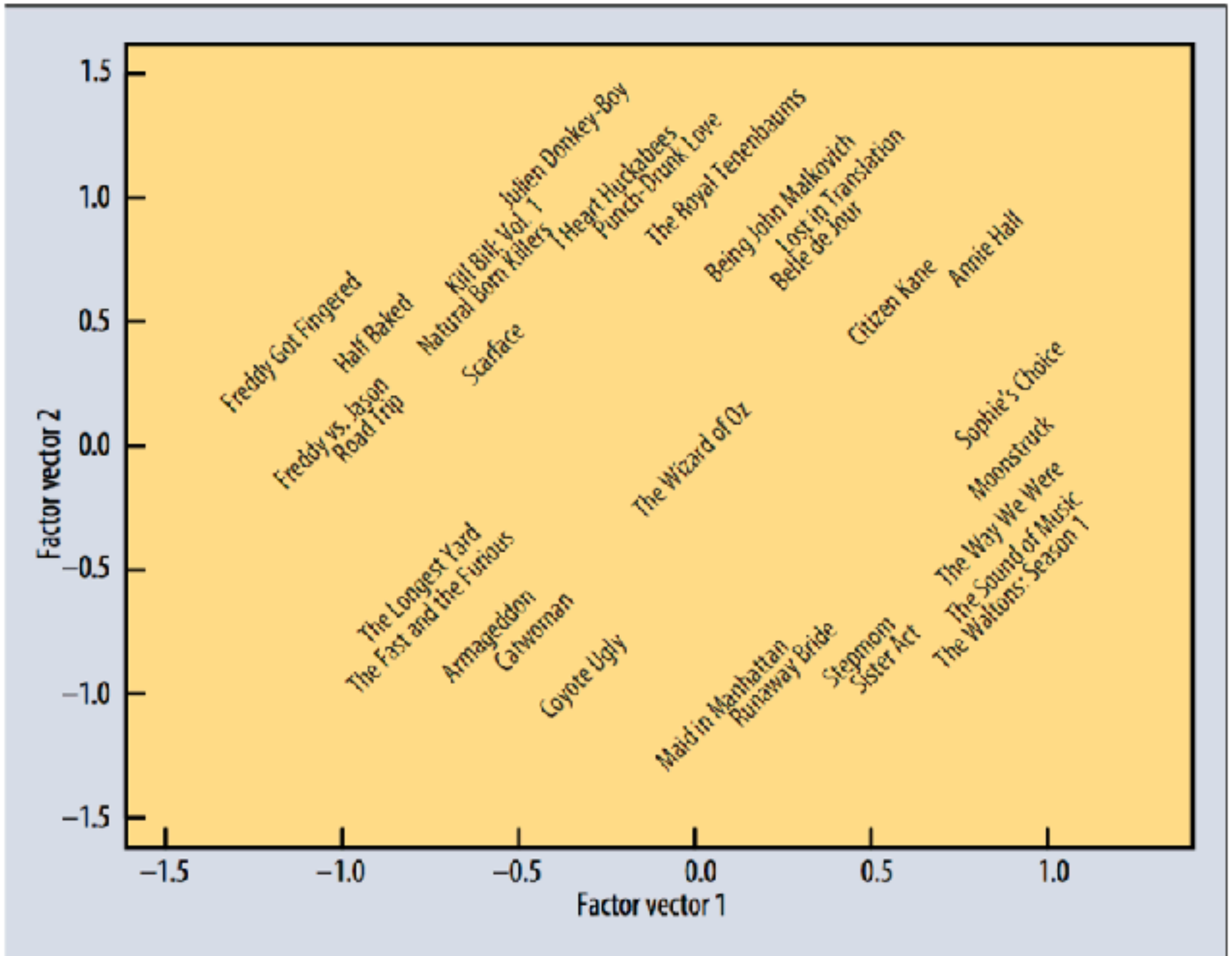
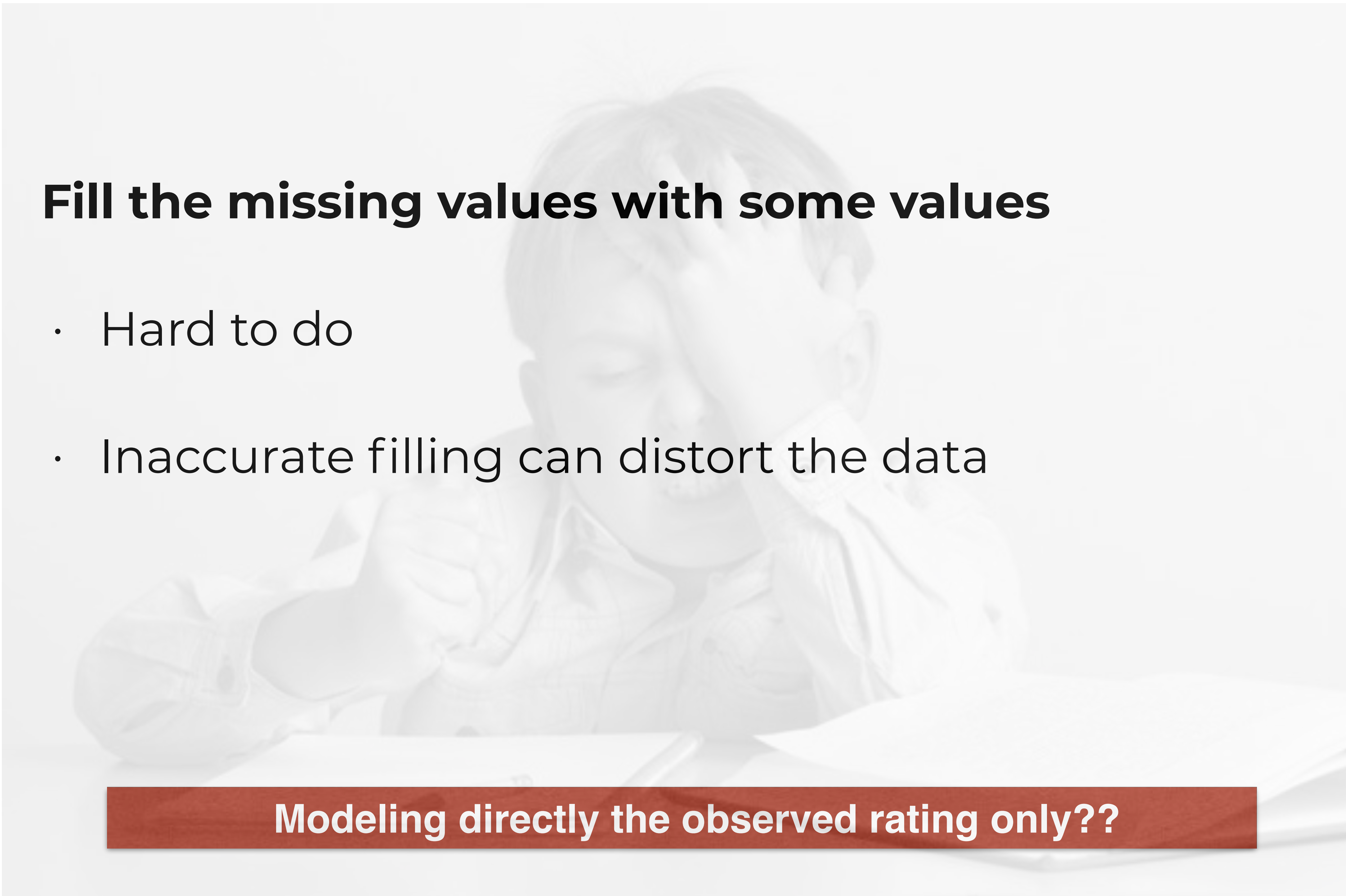


Figure 1: A mapping of movies based on two latent factors

Factorization Models



Fill the missing values with some values

- Hard to do
- Inaccurate filling can distort the data

Modeling directly the observed rating only??

Modeling directly the observed rating only??

- Stochastic gradient descent
  - For each given training set, the system predicts r_{ui}, and computes the associated prediction error

$$e_{ui} \stackrel{def}{=} r_{ui} - q_i^T p_u.$$

  - Then it modifies the parameters by a magnitude proportional to ɣ in the opposite direction of the gradient, yielding:

$$q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i)$$
$$p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u)$$

Regularized SVD

- Direct Matrix Factorization of **Incomplete Data**
  - **Modeling directly the observed rating only**
  - When data is sparse, covariance estimates will be statistically unreliable.
  - **Matrix factorization as a cost function**

$$\text{Min}_{p_*, q_*} \sum_{\text{known } r_{ui}} \left(r_{ui} - p_u^T q_i\right)^2 + \lambda\left(\|p_u\|^2 + \|q_i\|^2\right)$$

  - Optimize by either **stochastic gradient-descent** or **alternating least squares**
  - The constant **λ** controls the extent of regularization

Improved SVD

Adding **Bias**

$$P'_{ui} = b_u + b_i + (p_u^T q_i)$$

b_u is observed deviation of user u and
b_i is observed deviation of item i.

Improving regularized singular value decomposition for collaborative filtering A Paterek Proceedings of KDD Cup and Workshop 2007, 5-8	713	2007

SVD++

- Explicit + Implicit information
  - boolean implicit feedback, indicating if the item has been rated by the user or not.

$$\hat{r}_{ui} = b_{ui} + q_i^T \left(p_u + |\mathrm{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}(u)} y_j \right)$$, where $$b_{ui} = \mu + b_u + b_i$$

  - *N(u)* is the set of items for which the user *u* has implicit information, and q, p and y are latent factors

Factorization meets the neighborhood: a multifaceted collaborative filtering model Y Koren Proceedings of the 14th ACM SIGKDD international conference on Knowledge ...	2098	2008

Non-Negative Matrix Factorization

- Can be used for ratings matrices that are non negative

$$\text{Minimize} \quad J = \frac{1}{2} \| R - UV^T \|$$
$$\text{subject to:} \quad U \geq 0, V \geq 0$$

- The major advantage of this approach is not the accuracy, but that of the **high level of interpretability** it provides in understanding the user-item interactions.

Matrix Factorization Methods

Method	Constraints	Objective	Advantatges/Disadvatgaes
Unconstrained	No Constraints	Frobenius + regularizer	Highest quality solution Good for most matrices Regularization prevents overfitting Poor interpretability
SVD	Orthogonal Basis	Frobenius + regularizer	Good Visual Interpretability Out-of-sample recommendations Good for dense matrices Poor semantic interpretability Suboptimal in sapre matrices
Max Margin	No Constraints	Hinge Loss + margin regularizer	Highest quality solution Resists overfitting Similar to unconstrained Poor Interepretability Good for discrete ratings
NMF	Non Negativity	Frobenius + regularizer	Good quality solution High semantic interpretability Loses interpretability with like/dislike ratings Less overfitting in some cases Best for implicit feedback

Sparse Linear Models (SLIM)

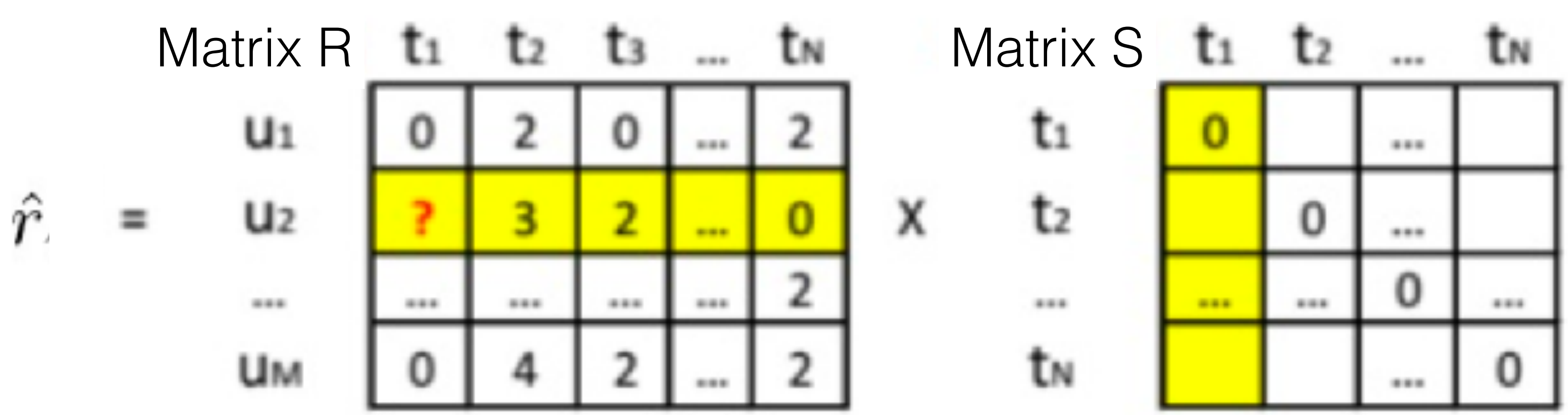
- Computes the item-item relations, by estimating an **item x item** sparse aggregation coefficient **matrix** S.
- The recommendation score of an unrated item **i** for a user **u** is:

$$\hat{r}_{ui} = \mathbf{r}_u^T \mathbf{s}_i.$$

$$\begin{aligned} \underset{S}{\text{minimize}} \quad & \tfrac{1}{2}\textstyle\sum_{u,i}(r_{ui} - \hat{r}_{ui})^2 + \tfrac{\beta}{2}||S||_F^2 + \lambda||S||_1, \\ \text{subject to} \quad & S \geq 0, \text{and} \\ & \text{diag}(S) = 0. \end{aligned}$$

SLIM: Sparse linear methods for top-n recommender systems X Ning, G Karypis Data Mining (ICDM), 2011 IEEE 11th International Conference on, 497-506	115	2011

Sparse Linear Models (SLIM)



$$\hat{r}_{ui} = \mathbf{r}_u^T \mathbf{s}_i.$$

$$\begin{array}{ll} \underset{S}{\text{minimize}} & \frac{1}{2}\sum_{u,i}(r_{ui} - \hat{r}_{ui})^2 + \frac{\beta}{2}||S||_F^2 + \lambda||S||_1, \\ \text{subject to} & S \geq 0, \text{and} \\ & \text{diag}(S) = 0. \end{array}$$

Sparse Linear Models (SLIM)

- Although the SLIM method proposes a prediction model for the rating, the final use of the ratings is for **ranking** the items
- Generally used for data sets with unary ratings (clicks, buy,..) from implicit feedback
- Since, the weights are restricted to be positive, the impact of each weight on the score are highly interpretable

Evaluation

Evaluation

	User	Item	Rating
Training	U1	T1	4
	U1	T2	3
	U1	T3	3
	U2	T2	4
	U2	T3	5
	U2	T4	5
	U3	T4	5
Test	U1	T4	? (3)
	U2	T1	? (2)
	U3	T1	? (3)
	U3	T2	? (3.5)
	U3	T3	? (4)

P(U1, T4) = Avg(T4) = (5+4)/2 = 4.5
P(U2, T1) = Avg(T1) = 4/1 = 4
P(U3, T1) = Avg(T1) = 4/1 = 4
P(U3, T2) = Avg(T2) = (3+4)/2 = 3.5
P(U3, T3) = Avg(T3) = (3+5)/2 = 4

3. Evaluation by Metrics

Mean Absolute Error (MAE) = $\frac{1}{n}\sum_{i=1}^{n}|e_i|$

ei = R(U, T) – P(U, T)

MAE = (|3 – 4.5| + |2 - 4| + |3 - 4| + |3 – 3.5| + |4 - 4|) / 5 = 1

Rank Accuracy

- **Mean Average Precision (MAP)** calculates the precision at the position of every corrected item in the ranked results list

$$\text{AveP} = \frac{\sum_{k=1}^{n}(P(k) \times \text{rel}(k))}{\text{number of relevant documents}} \qquad \text{MAP} = \frac{\sum_{q=1}^{Q} \text{AveP(q)}}{Q}$$

Query 1: AP=⅓(1/1+⅔+3/6)=0.72

Query 2: AP=⅓(1/1+2/2+¾)=0.917

MAP= (0.72+0.917)/2=0.8185

Top-N Evaluation

Task: Top-N Items to a user U3

	User	Item	Rating
Training	U1	T1	4
	U1	T2	3
	U1	T3	3
	U2	T2	4
	U2	T3	5
	U2	T4	5
	U3	T4	5
Test	U1	T4	? (3)
	U2	T1	? (2)
	U3	T1	? (3)
	U3	T2	? (3.5)
	U3	T3	? (4)

P(U3, T1) = Avg(T1) = 4/1 = 4
P(U3, T2) = Avg(T2) = (3+4)/2 = 3.5
P(U3, T3) = Avg(T3) = (3+5)/2 = 4

Real rank: I3,I2,I1
Predicted rank: I3,I1,I2

Precision@N = #of hits/N
Precision@1 = 1/1
Precision@2 = 1/2
Precision@3 = 3/3

Rank **Accuracy**

Precision/Recall

- We can use the strategy followed by P. Cremonesi et.al.
- In order to measure the precision recall, first the models is trained using the training data, and then, for each item ***i*** rated with 5 stars in the test data set:
  - A set of 100 random unseen movies for the user of the item ***i*** are selected. We assume that these random movies will not be at the same interest than the 5 star movie
  - We predict the rating of the movie of item ***i*** and 100 random unseen movies.
  - We form a rank list by ordering all the 101 item according to the predicted rating. Let denote ***p*** the rank of the test item ***i*** within the list. The best results corresponds to the case the test item ***i*** precedes all the random items (i.e., p=1).
  - A top-N recommendation list by piking the **N** top ranked items from the list. If ***p≤N*** we have a hit. Otherwise we have a miss. Chances of hit increases as **N** is higher.

Performance of recommender algorithms on top-n recommendation tasks
P Cremonesi, Y Koren, R Turrin
Proceedings of the fourth ACM conference on Recommender systems, 39-46
612 2010

Rank **Accuracy**

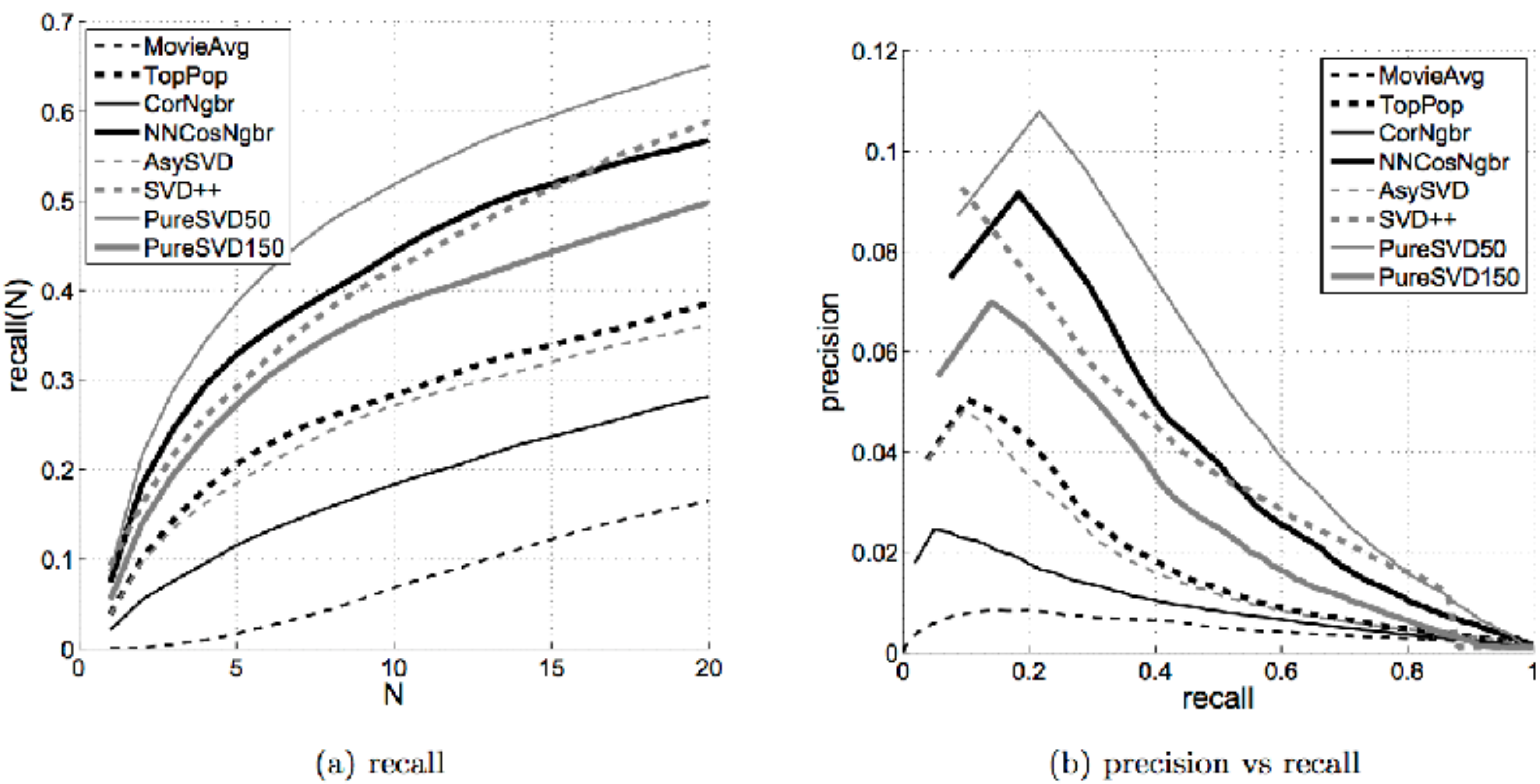
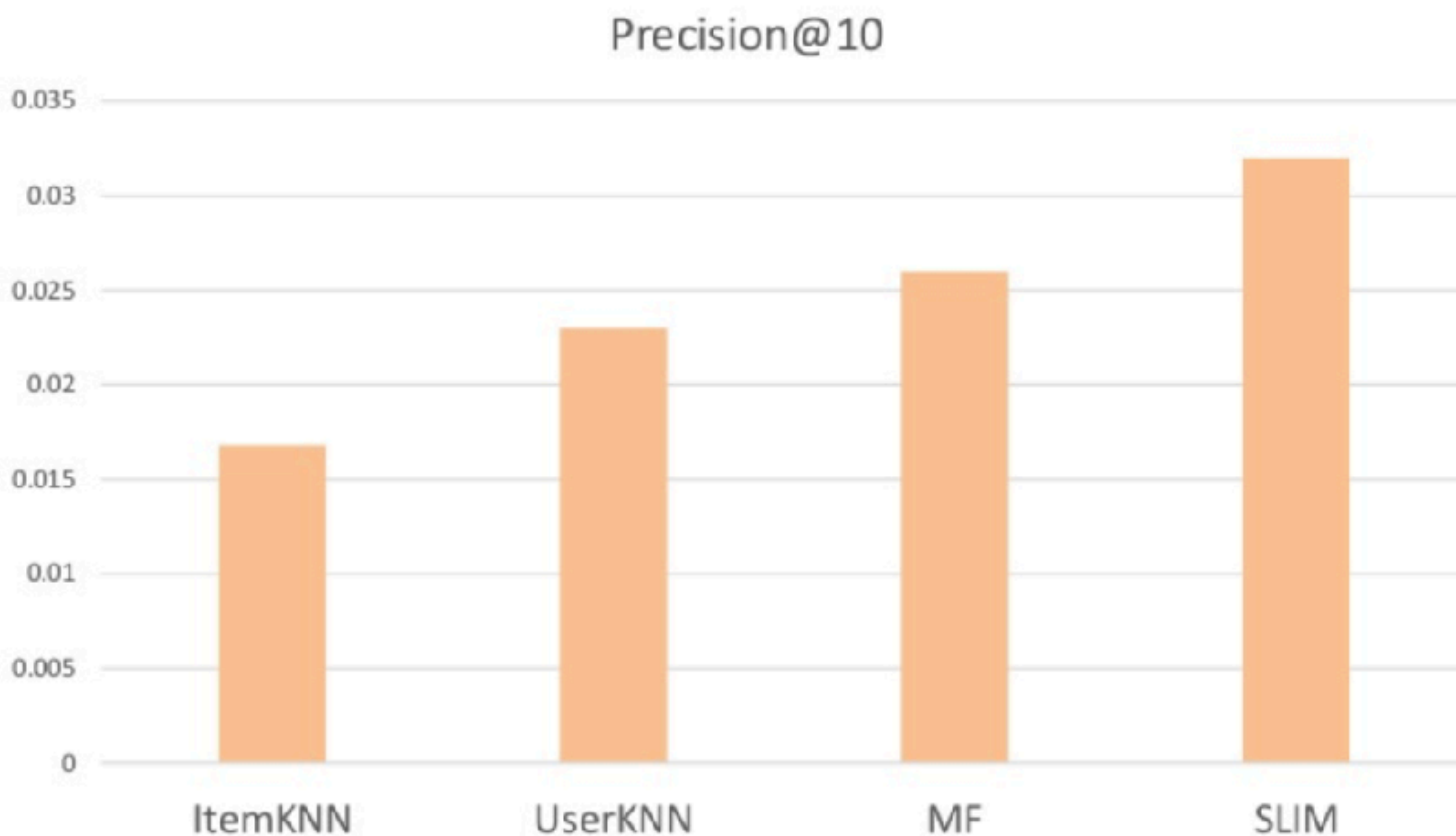


Figure 2: Movielens: (a) recall-at-N and (b) precision-versus-recall on all items.

Performance of recommender algorithms on top-n recommendation tasks
P Cremonesi, Y Koren, R Turrin
Proceedings of the fourth ACM conference on Recommender systems, 39-46

There are 100K ratings given by 943 users on 1,682 movies



Task #2
Movie Recommender

Create at least:

1) Item-Based Recommender System
2) Factorization Recommender system

What to deliver:

Write a paper/report (4-8 pages)
Explain: 1) how you see the problem, 2) the used methods, 3) the results and 4) your conclusions in terms of accuracy and complexity

Github repository of the task.

Deadline - May 13th