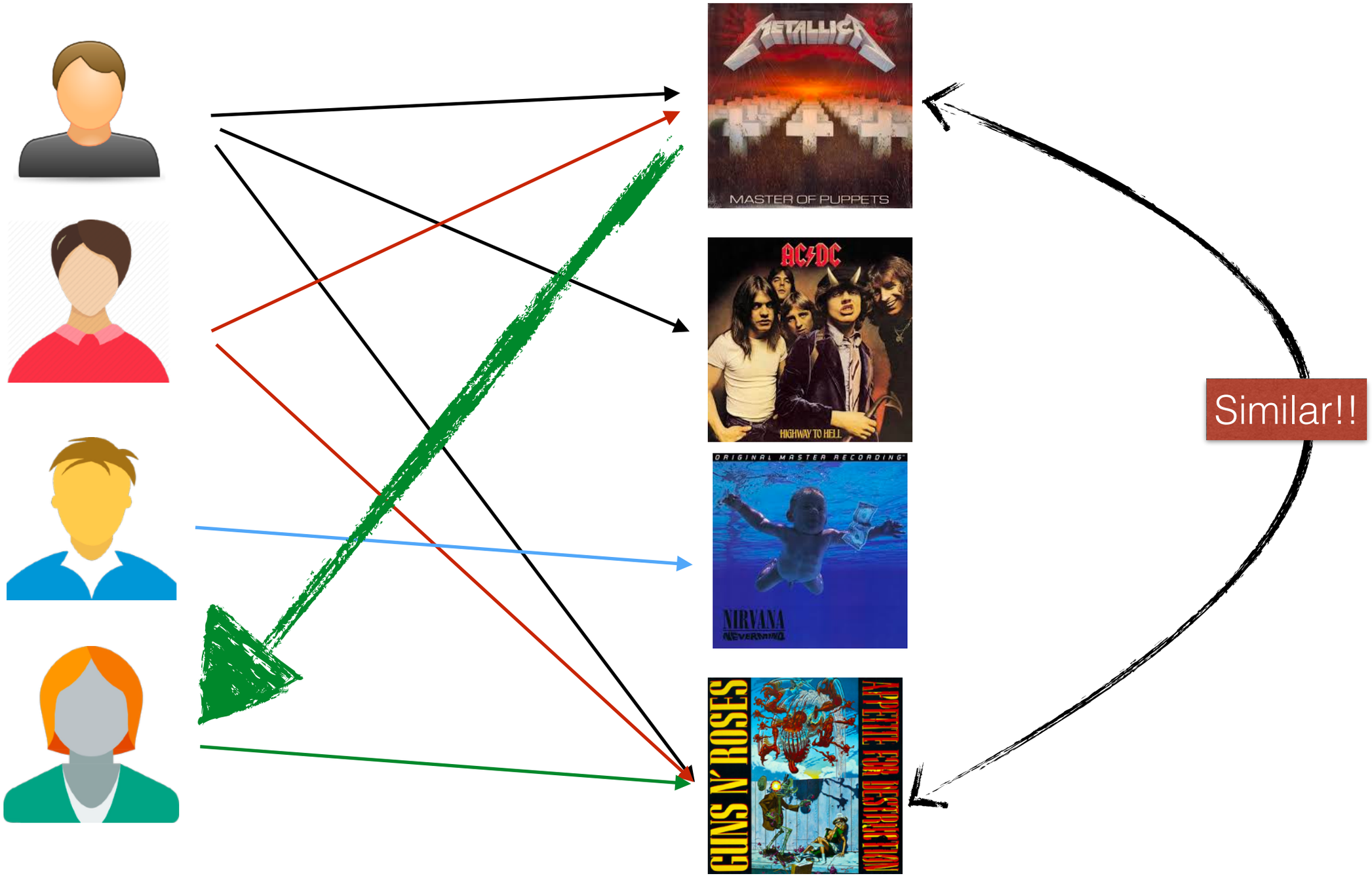
Master on Foundations of Data Science

Recommender Systems

Collaborative Recommender Systems (II)

Santi Seguí | 2017-2018

Item-Based Recommender



Let's see how we can create a **Item-Based CF** for Movie recommendations.

Item-Based Recommender

- Instead on relying on the user similarity, prediction can rely on **item similarities**.

- Item similarity used to be **more stable** than user-similarity. So, the the update frequency of the items similarity is not as critical than user-similarity

- Item-similarities are more static, while user-similarities are more dynamic

Item-based collaborative filtering recommendation algorithms B Sarwar, G Karypis, J Konstan, J Riedl Proceedings of the 10th international conference on World Wide Web, 285-295	5944	2001

Similarity Measures
What happens with item-based systems?

- Pearson Correlation

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}}$$

- Cosine distance

$$sim(a,b) = \frac{\vec{a} \cdot b}{|\vec{a}| * |\vec{b}|}$$

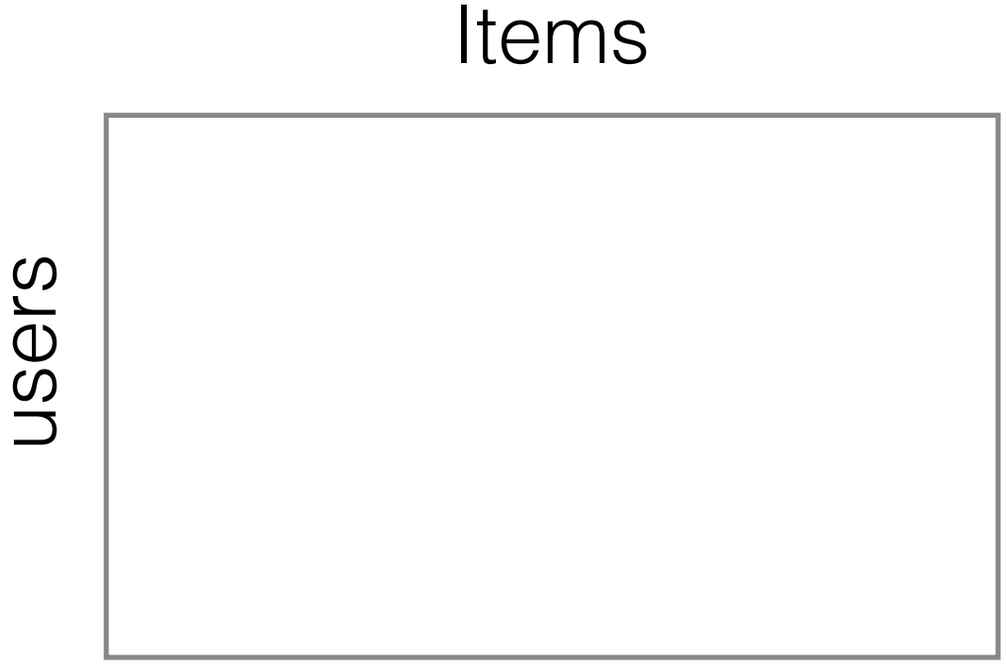
Where:

- $sim(a,b)$ is the similarity between user "a" and user "b"
- P is the set of common rated movies by user "a" and "b"
- $r_{a,p}$ is the rating of movie "p" by user "a"
- $\bar{r}_a$ is the mean rating given by user "a"

Are these measures valid?

Item Based CF

- Similarities are computed between items.



- Before computing the similarities between columns, each row of the rating matrix is centered to a mean zero.

Adjusted Cosine Similarity

- Computing similarity using basic cosine measure in item-based case has one important drawback: **The differences in rating scale between different users are not taken into account.**

- The Adjusted Cosine Similarity offsets this drawback by subtracting the corresponding user average from each co-rated pair:

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_u)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_u)^2}}$$

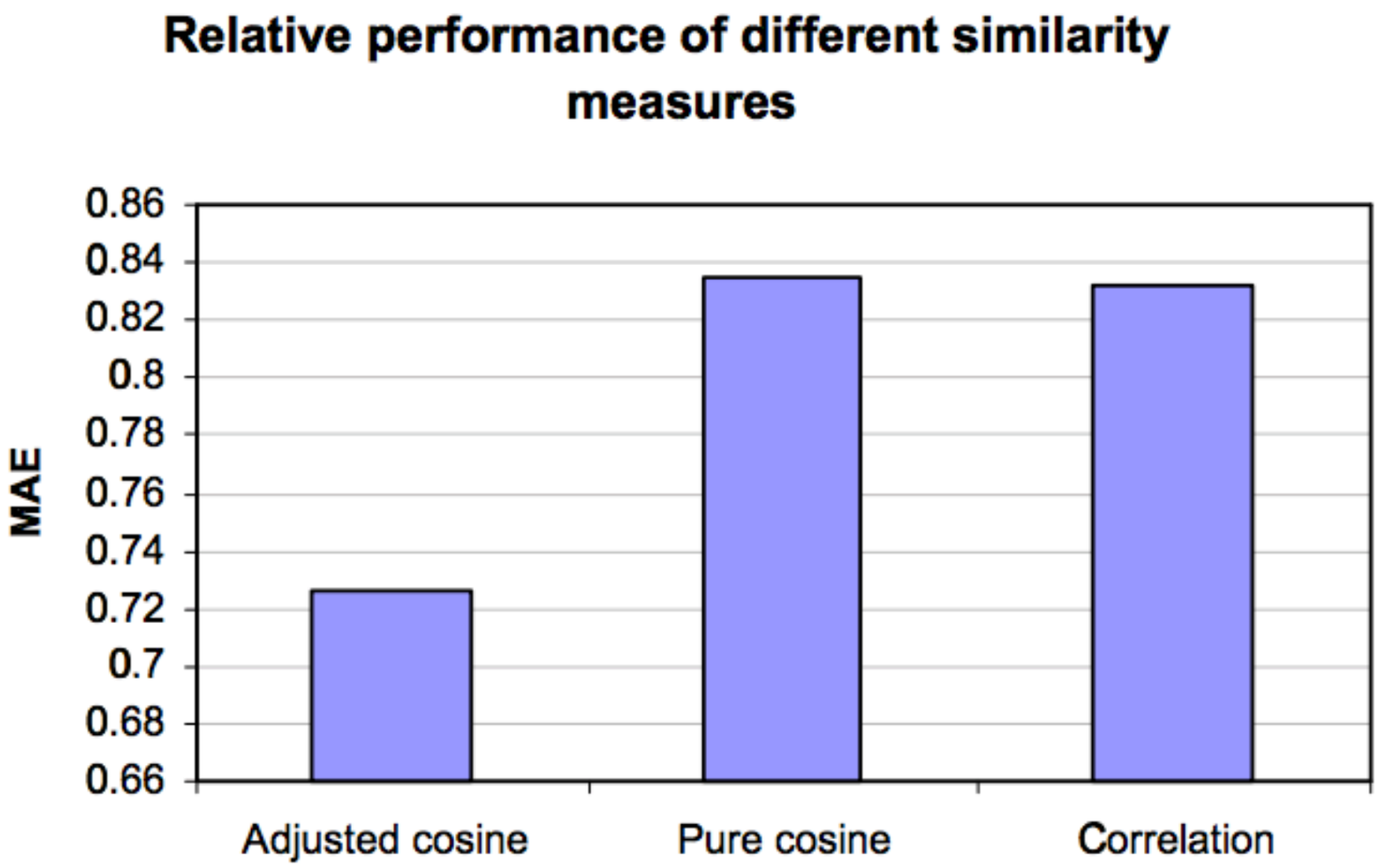


Figure 4: Impact of the similarity computation measure on item-based collaborative filtering algorithm.

Item-based collaborative filtering recommendation algorithms
B Sarwar, G Karypis, J Konstan, J Riedl
Proceedings of the 10th international conference on World Wide Web, 285-295 — 5944 — 2001

Item-Based
How do we generate a prediction?

$$\hat{r}_{u,j} = \bar{r}_u + \frac{\sum_{v \in P_u(j)} sim(u,v) \times (r_{v,j} - \bar{r}_v)}{\sum_{v \in P_u(j)} sim(u,v)}$$

Why not another equation?

Exercice:

I want to create a Recommender System for **NETFLIX** using MovieLens dataset.

I have to decide if my system is:
a) Non-Personalized
b) User-Based CF
c) Item-Based CF

Plan an implementation plan, think about which is the best under all possibles scenarios you can find.
What should we do in order to say which is best?

User-Based vs. Item-Based

- m = #users ; n = #items
- Normally, the number of users is much bigger than the number of items.

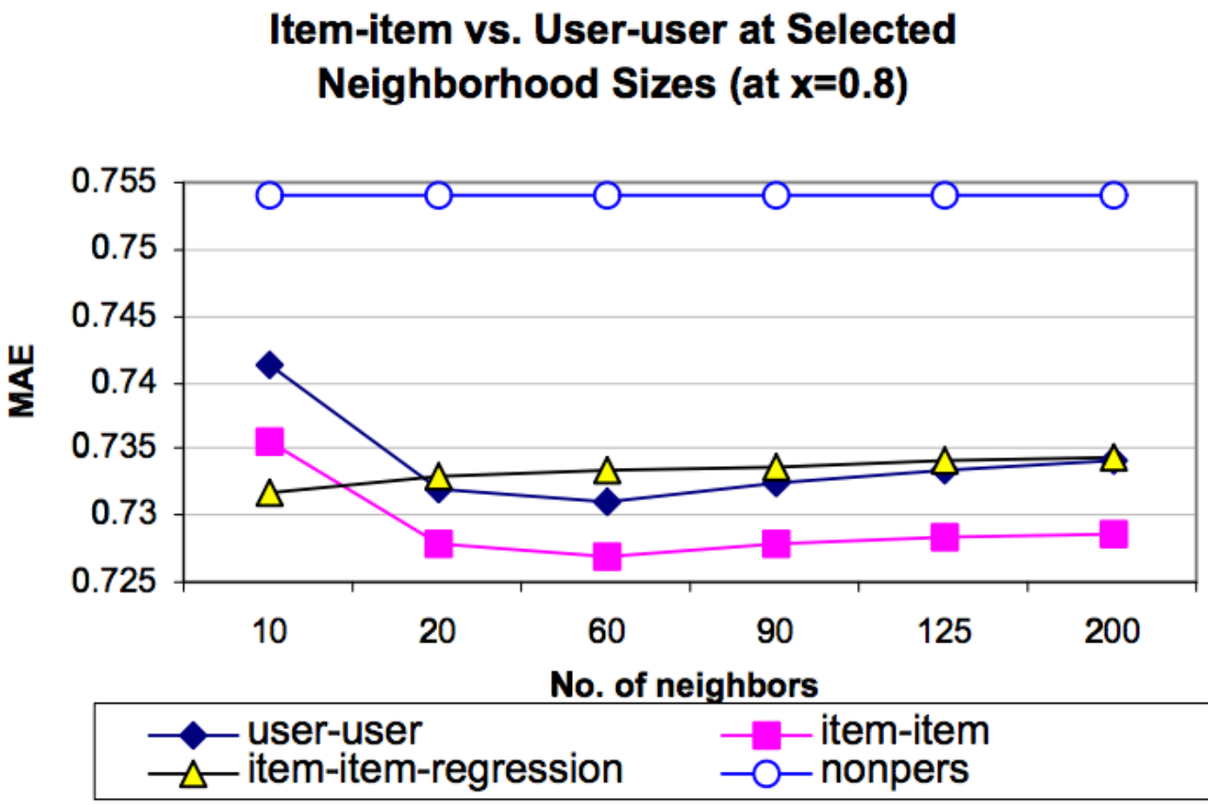
Computational time:

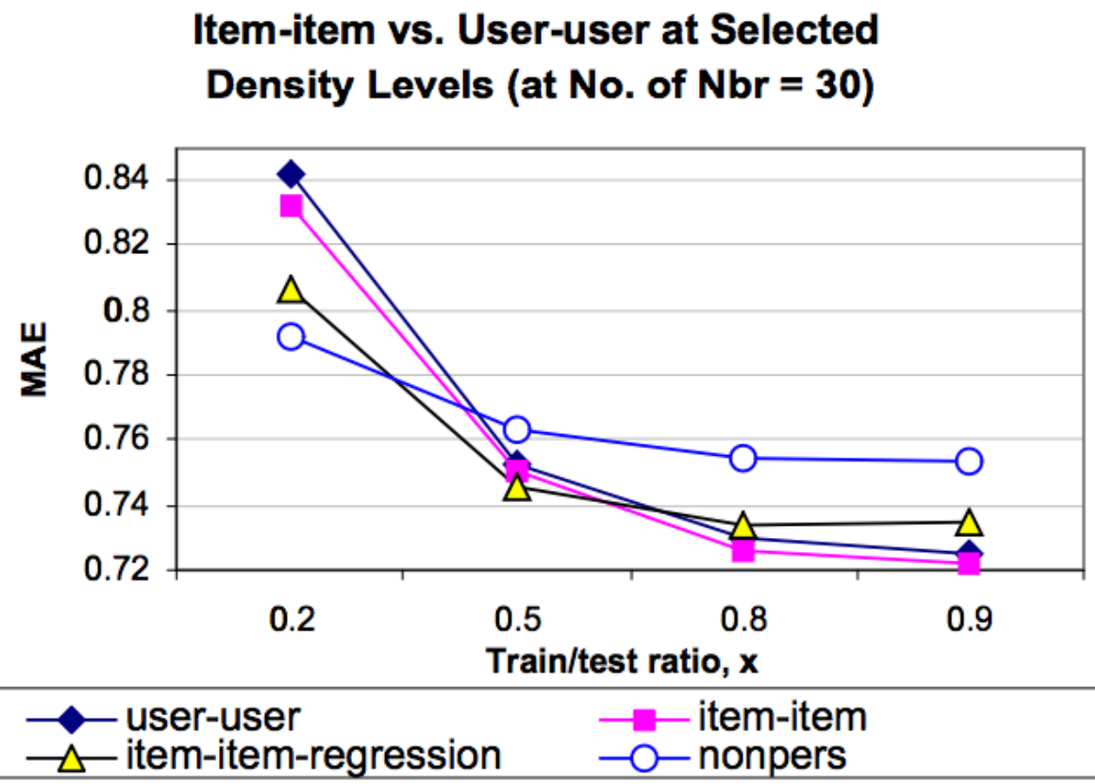
O(m^2 n) O(n^2 m)

Memory Requirements:

O(m^2) O(n^2)

User-Based vs. Item-Based





Item-based collaborative filtering recommendation algorithms
B Sarwar, G Karypis, J Konstan, J Riedl
Proceedings of the 10th international conference on World Wide Web, 285-295
5944 2001

User-Based vs. Item-Based

- **Pros User-based**
  - Tend to provide higher diversity (more serendipity)
- **Pros Item-based**
  - Better results (in terms of RMSE)
  - More stable to changes

User-Based vs. Item-Based

	User-Based	Item-Based
Scalability		
Explanation		
Novelty		
Coverage		
Cold start		
Performance		

User-Based vs. Item-Based

	User-Based	Item-Based
Scalability	Bad when users ar	Bad when item size is large
Explanation	Bad	Good
Novelty	Bad	Good
Coverage	Bad	Good
Cold start	Bad for new users	Bad for new items
Performance	Need to get many users history	Only need to get current users's history

Item-Based Nearest Neighbor Regression

- We can replace the (normalized) similarity coefficient AdjustedCosine(j,i) with a unknown parameter w_{ji}^{item} to model the rating prediction of a user u for target item i.

$$\hat{r}_{ui} = \sum_{j \in Q_i(u)} w_{ji}^{item} \cdot r_{uj}$$

The nearest items in Qi(u) can be determined using the adjusted cosine

Item-Based Nearest Neighbor Regression

$$Minimize\ J_t = \sum_{u \in U_t} (r_{ut} - \hat{r}_{ut})^2$$

$$= \sum_{u \in U_t} \left(r_{ut} - \sum_{j \in Q_t(u)} w_{jt}^{item} \cdot r_{uj} \right)^2$$

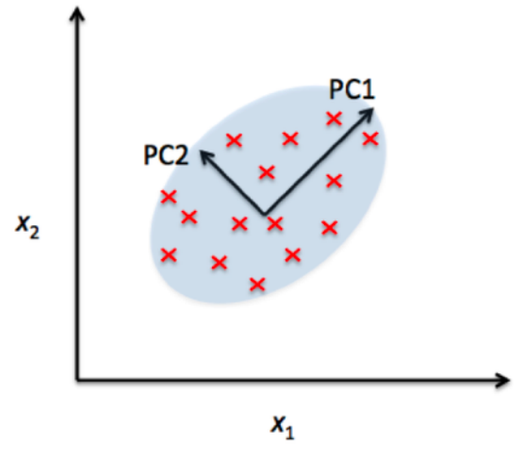
Dimensionality Reduction

- 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**

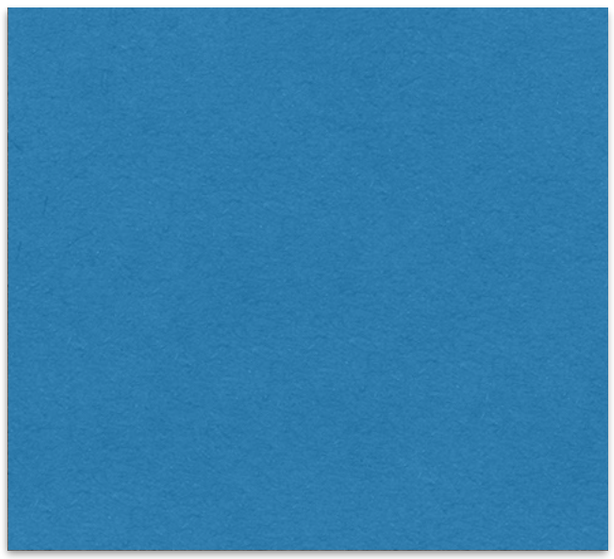
- A reduced representation of the data can be created in terms of either row-wise latent factors or in terms of column-wise latent factors.

Dimensionality Reduction



Items

users

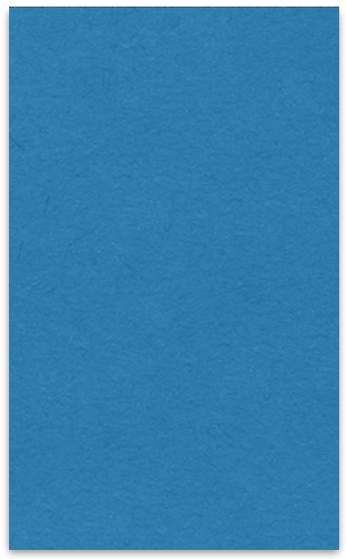


m x n **sparse matrix**

≈

Latent Factors

users



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 raring for each row/column
    - $S = R_f^T R_f$. S is a positive semi-definite of size *n x n*
    - $S = P\Lambda P^T$, 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 raring for each row/column
    - S = Covariacne Matrix
    - $S = P\Lambda P^T$, 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 $R_f P_d$

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.

- Today, model-based methods and in particular **latent factor models** are some of the most **accurate** methods

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 f, such that user-item interactions are modeled as inner products in that space

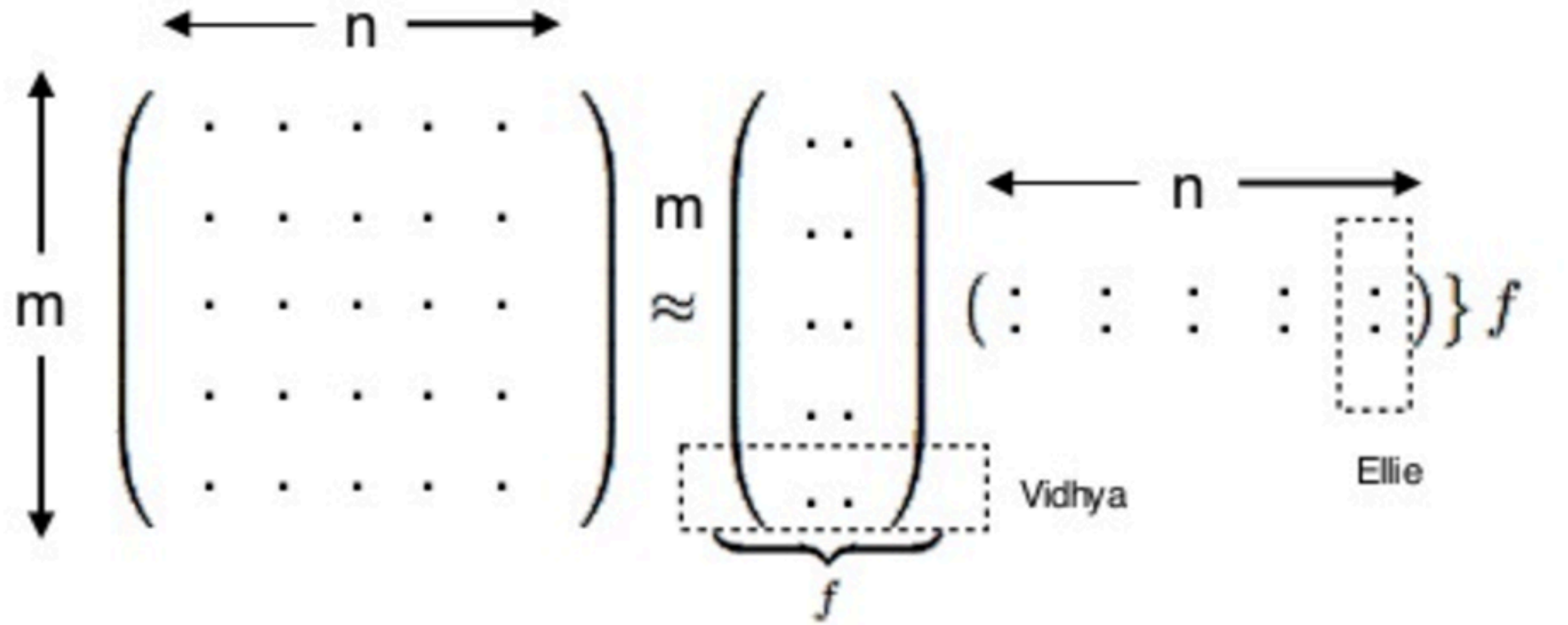
Matrix Factorization Methods

- 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

Latent Factor Models



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
U1	1	0
U2	1	0
U3	1	0
U4	1	1
U5	-1	1
U6	-1	1
U7	-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

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

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

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 column

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

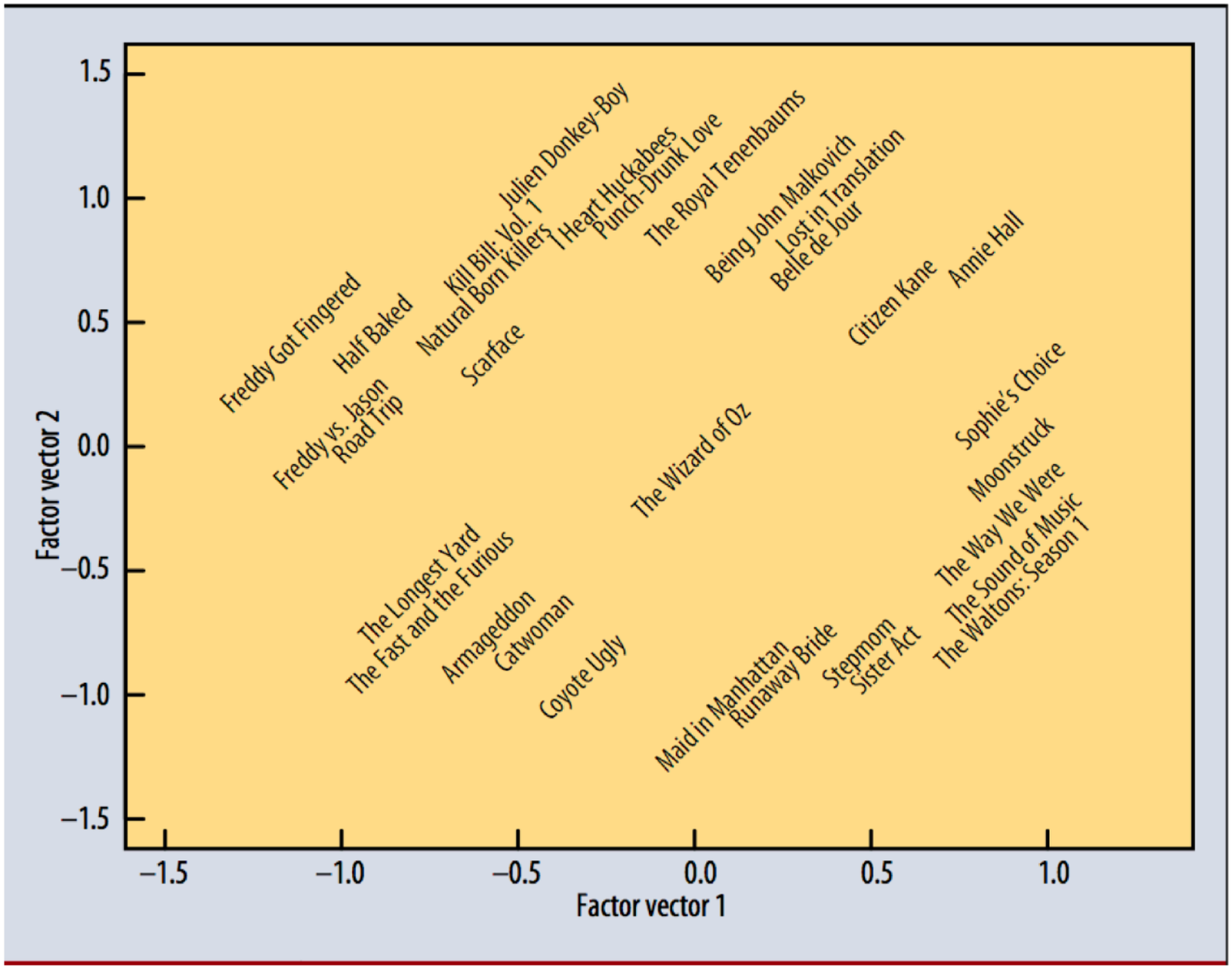


Figure 1: A mapping of movies based on two latent factors

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

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

- Then it modifies the parameters by a magnitude proportional to γ 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 Regularized 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.

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

- *N(u)* is the set of items for which the user *u* has implicit information

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

Non-Negative Matrix Factorization

- Can be used for ratings matrices that are non negative

$$\text{Minimize} \quad J = \frac{1}{2}\left\|R - UV^T\right\|$$

$$\text{subject to:} \quad U \geq 0, V \geq 0$$

- 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
PLSA	Non Negativity	Maxiumun likelihood + regularizer	Similar to NMF + Probabilistic interpretation

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:

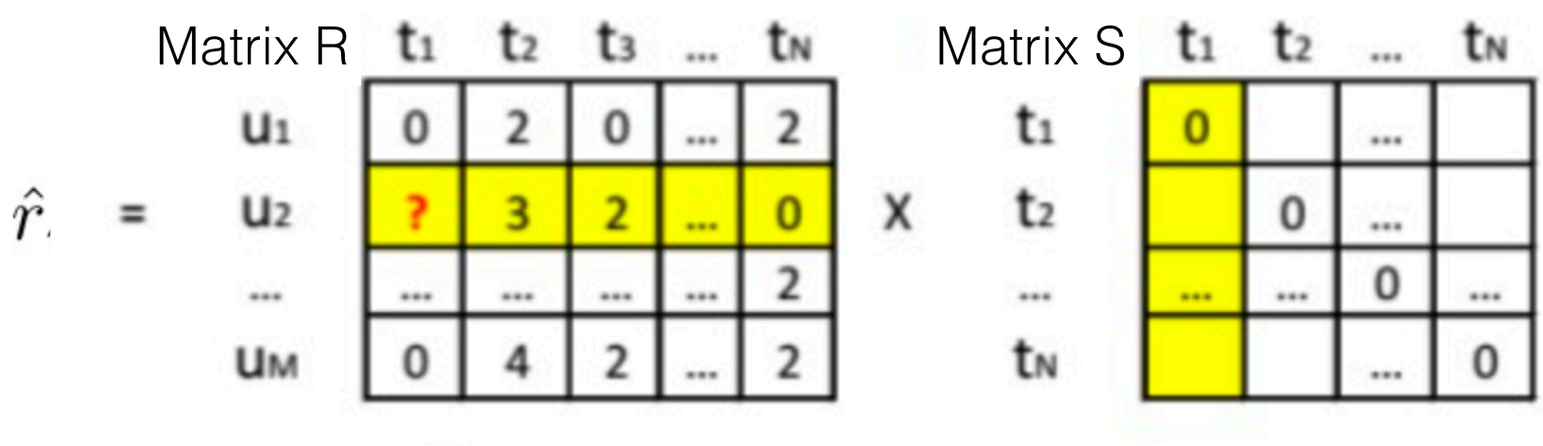
$$\hat{r}_{ui} = \mathbf{r}_u^T \mathbf{s}_i.$$

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

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

TASK 1
Movie Recommender

Create:

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

Explain your conclusions in terms of results and complexity

Deadline: 20 April (23.55)

TASK 2

Paper presentations. 8 -10 minutes

Student	Date	Paper
1	Apr 5th	
2	Apr 5th	
3	Apr 12th	
4	Apr 12th	
5	Apr 19th	
6	Apr 19th	
7	Apr 26th	
8	Apr 26th	
9	May 10th	
10	May 10th	
11	May 17th	
12	May 17th	
13	May 24th	
14	May 24th	

Paper Suggestions I

Restricted Boltzmann Machines for Collaborative Filtering

Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
Andriy Mnih AMNIH@CS.TORONTO.EDU
Geoffrey Hinton HINTON@CS.TORONTO.EDU
University of Toronto, 6 King's College Rd., Toronto, Ontario M5S 3G4, Canada

Abstract

Most of the existing approaches to collaborative filtering cannot handle very large data sets. In this paper we show how a class of two-layer undirected graphical models, called Restricted Boltzmann Machines (RBM's), can be used to model tabular data, such as user's ratings of movies. We present efficient learning and inference procedures for this class of models and demonstrate that RBM's can be successfully applied to the Netflix data set, containing over 100 million user/movie ratings. We also show that RBM's slightly outperform carefully-tuned SVD models. When the predictions of multiple RBM models and multiple SVD models are linearly combined, we achieve an error rate that is well over 6% better than the score of Netflix's own system.

Low-rank approximations based on minimizing the sum-squared distance can be found using Singular Value Decomposition (SVD). In the collaborative filtering domain, however, most of the data sets are sparse, and as shown by Srebro and Jaakkola (2003), this creates a difficult non-convex problem, so a naive solution is not going work.[1]

In this paper we describe a class of two-layer undirected graphical models that generalize Restricted Boltzmann Machines to modeling tabular or count data (Welling et al., 2005). Maximum likelihood learning is intractable in these models, but we show that learning can be performed efficiently by following an approximation to the gradient of a different objective function called "Contrastive Divergence" (Hinton, 2002).

2. Restricted Boltzmann Machines (RBM's)

Paper Suggestions II

This summer, I'm interning at Spotify in New York City, where I'm working on content-based music recommendation using convolutional neural networks. In this post, I'll explain my approach and show some preliminary results.

Overview

This is going to be a long post, so here's an overview of the different sections. If you want to skip ahead, just click the section title to go there.

http://benanne.github.io/2014/08/05/spotify-cnns.html

Paper Suggestions III

Deep content-based music recommendation

Aäron van den Oord, Sander Dieleman, Benjamin Schrauwen
Electronics and Information Systems department (ELIS), Ghent University
{aaron.vandenoord, sander.dieleman, benjamin.schrauwen}@ugent.be

Abstract

Automatic music recommendation has become an increasingly relevant problem in recent years, since a lot of music is now sold and consumed digitally. Most recommender systems rely on collaborative filtering. However, this approach suffers from the cold start problem: it fails when no usage data is available, so it is not effective for recommending new and unpopular songs. In this paper, we propose to use a latent factor model for recommendation, and predict the latent factors from music audio when they cannot be obtained from usage data. We compare a traditional approach using a bag-of-words representation of the audio signals with deep convolutional neural networks, and evaluate the predictions quantitatively and qualitatively on the Million Song Dataset. We show that using predicted latent factors produces sensible recommendations, despite the fact that there is a large semantic gap between the characteristics of a song that affect user preference and the corresponding audio signal. We also show that recent advances in deep learning translate very well to the music recommendation setting, with deep convolutional neural networks significantly outperforming the traditional approach.

Paper Suggestions IV

Deep Neural Networks for YouTube Recommendations

Paul Covington, Jay Adams, Emre Sargin
Google
Mountain View, CA
{pcovington, jka, msargin}@google.com

ABSTRACT

YouTube represents one of the largest scale and most sophisticated industrial recommendation systems in existence. In this paper, we describe the system at a high level and focus on the dramatic performance improvements brought by deep learning. The paper is split according to the classic two-stage information retrieval dichotomy: first, we detail a deep candidate generation model and then describe a separate deep ranking model. We also provide practical lessons and insights derived from designing, iterating and maintaining a massive recommendation system with enormous user-facing impact.

Keywords

recommender system; deep learning; scalability

1. INTRODUCTION

YouTube is the world's largest platform for creating, sharing and discovering video content. YouTube recommendations are responsible for helping more than a billion users discover personalized content from an ever-growing corpus of videos. In this paper we will focus on the immense impact deep learning has recently had on the YouTube video recommendations system. Figure 1 illustrates the recommendations on the YouTube mobile app home.

Recommending YouTube videos is extremely challenging

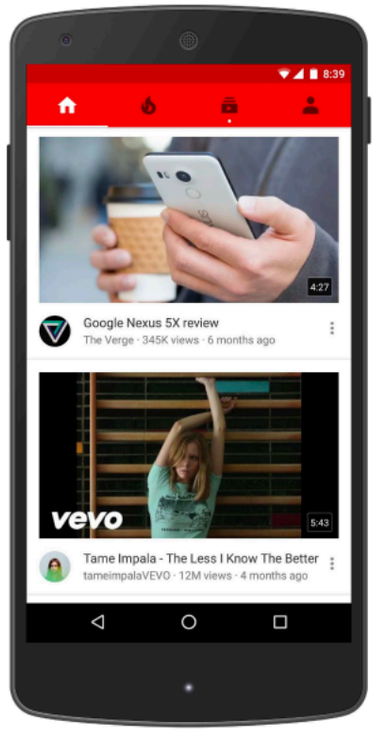


Figure 1: Recommendations displayed on YouTube mobile app home.

with well-established videos can be understood from an exploration/exploitation perspective.

Paper Suggestions V

Wide & Deep Learning for Recommender Systems

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, Hemal Shah
*
Google Inc.

Paper Suggestions VI

Collaborative Knowledge Base Embedding for Recommender Systems

Fuzheng Zhang†, Nicholas Jing Yuan†, Defu Lian‡, Xing Xie†,Wei-Ying Ma†
†Microsoft Research
‡Big Data Research Center, University of Electronic Science and Technology of China
{fuzzhang,nicholas.yuan,xingx,wyma}@microsoft.com,
dove.ustc@gmail.com

ABSTRACT

Among different recommendation techniques, collaborative filtering usually suffer from limited performance due to the sparsity of user-item interactions. To address the issues, auxiliary information is usually used to boost the performance. Due to the rapid collection of information on the web, the knowledge base provides heterogeneous information including both structured and unstructured data with different semantics, which can be consumed by various applications. In this paper, we investigate how to leverage the heterogeneous information in a knowledge base to improve the quality of recommender systems. First, by exploiting the knowledge base, we design three components to extract items' semantic representations from structural content, textual content and visual content, respectively. To be specific, we adopt a heterogeneous network embedding method, termed as TransR, to extract items' structural representations by considering the heterogeneity of both nodes and relationships. We apply stacked denoising auto-encoders and stacked convolutional auto-encoders, which are two types of deep learning based embedding techniques, to extract items' textual representations and visual representations, respectively. Finally, we propose our final integrated framework, which is termed as **C**ollaborative **K**nowledge Base **E**mbedding (**CKE**), to jointly learn the latent representations in collaborative filtering as well as items' semantic representations from the knowledge base. To evaluate the performance of each embedding component as well as the whole system, we conduct extensive experiments with two real-

filtering (CF) based methods, which make use of historical interactions or preferences, have made significant success [23]. However, CF methods usually suffer from limited performance when user-item interactions are very sparse, which is very common for scenarios such as online shopping where the item set is extremely large. In addition, CF methods can not recommend new items since these items have never received any feedbacks from users in the past. To tackle these problems, hybrid recommender systems, which combine collaborative filtering and auxiliary information such as item content, can usually achieve better recommendation results and have gained increasing popularity in recent years [2].

Over the past years, more and more semantic data are published following the Linked Data principles[1], by connecting various information from different topic domains such as people, books, musics, movies and geographical locations in a unified global data space. These heterogeneous data, interlinked with each other, forms a huge information resource repository called knowledge base. Several typical knowledge bases have been constructed, including academic projects such as YAGO[2], NELL[3], DBpedia[4], and DeepDive[5], as well as commercial projects, such as Microsoft's Satori[6] and Google's Knowledge Graph[7]. Using the heterogeneous connected information from the knowledge base can help to develop insights on problems which are difficult to uncover with data from a single domain [6]. To date, information retrieval [9], community detection [25], sentiment analysis [4] - to name a few - are the noteworthy applications that successfully leverage the knowledge

Paper Suggestions VII

SESSION-BASED RECOMMENDATIONS WITH RECURRENT NEURAL NETWORKS

Balázs Hidasi *
Gravity R&D Inc.
Budapest, Hungary
balazs.hidasi@gravityrd.com

Alexandros Karatzoglou
Telefonica Research
Barcelona, Spain
alexk@tid.es

Linas Baltrunas †
Netflix
Los Gatos, CA, USA
lbaltrunas@netflix.com

Domonkos Tikk
Gravity R&D Inc.
Budapest, Hungary
domonkos.tikk@gravityrd.com

ABSTRACT

We apply recurrent neural networks (RNN) on a new domain, namely recommender systems. Real-life recommender systems often face the problem of having to base recommendations only on short session-based data (e.g. a small sportsware website) instead of long user histories (as in the case of Netflix). In this situation the frequently praised matrix factorization approaches are not accurate. This problem is usually overcome in practice by resorting to item-to-item recommendations, i.e. recommending similar items. We argue that by modeling the whole session, more accurate recommendations can be provided. We therefore propose an RNN-based approach for session-based recommendations. Our approach also considers practical aspects of the task and introduces several modifications to classic RNNs such as a ranking loss function that make it more viable for this specific problem. Experimental results on two data-sets show marked improvements over widely used approaches.

https://arxiv.org/abs/1511.06939

Paper Suggestions VIII

Graph Convolutional Matrix Completion

Rianne van den Berg
University of Amsterdam

Thomas N. Kipf
University of Amsterdam

Max Welling
University of Amsterdam, CIFAR[1]

Abstract

We consider matrix completion for recommender systems from the point of view of link prediction on graphs. Interaction data such as movie ratings can be represented by a bipartite user-item graph with labeled edges denoting observed ratings. Building on recent progress in deep learning on graph-structured data, we propose a graph auto-encoder framework based on differentiable message passing on the bipartite interaction graph. Our model shows competitive performance on standard

in the form of node features. Predicting ratings then reduces to predicting labeled links in the bipartite user-item graph.

We propose graph convolutional matrix completion (GC-MC): a graph-based auto-encoder framework for matrix completion, which builds on recent progress in deep learning on graphs [2, 6, 19, 5, 15, 30, 14]. The auto-encoder produces latent features of user and item nodes through a form of message passing on the bipartite interaction graph. These latent user and item representations are used to reconstruct the rating links through a bilinear decoder.

Paper Suggestions IX

Deep Models of Interactions Across Sets

Jason Hartford [*1] **Devon R Graham** [*1] **Kevin Leyton-Brown** [1] **Siamak Ravanbakhsh** [1]

Abstract

We use deep learning to model interactions across two or more sets of objects, such as user–movie ratings or protein–drug bindings. The canonical representation of such interactions is a matrix (or tensor) with an exchangeability property: the encoding's meaning is not changed by permuting rows or columns. We argue that

$\langle n, m, x \rangle \in \mathbb{X}$. Learning our function corresponds to *matrix completion*: using patterns in $\mathbb{X}$ to predict values for the remaining elements of X.

X is what we will call an *exchangeable matrix*: any row- and column-wise permutation of X represents the same set of ratings and hence the same matrix completion problem. Exchangeability has a long history in machine learning and statistics. For example, the common iid assumption implies