RECOMMENDER SYSTEMS :

INTRODUCTION, DEFINITION, APPROACHES & ISSUES

Vincent Guigue (et Benjamin Piwowarski)

SCIENCES SORBONNE UNIVERSITÉ
LIP6
MLIA Machine Learning & Deep Learning for Information Access

Introduction

Recommender Systems in several dates

1998 Amazon item-to-item recommendation

2004-Now Special sessions in recommender system in several important conferences & journals:

AI Communications ; IEEE Intelligent Systems; International Journal of Electronic Commerce; International Journal of Computer Science and Applications; ACM Transactions on Computer-Human Interaction; ACM Transactions on Information Systems

2007 First ACM RecSys conference

2008 Netflix online services (& innovative HMI)

2008-09 Netflix RS prize

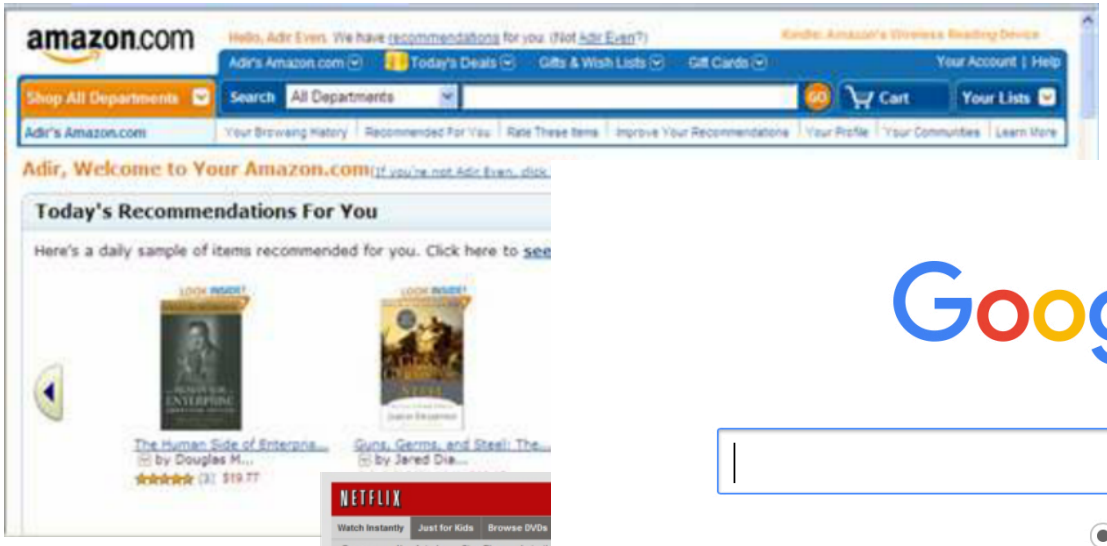
2010-Now RS become essential : `YouTube`, `Netflix`, `Tripadvisor`, `Last.fm`, `IMDb`, etc...

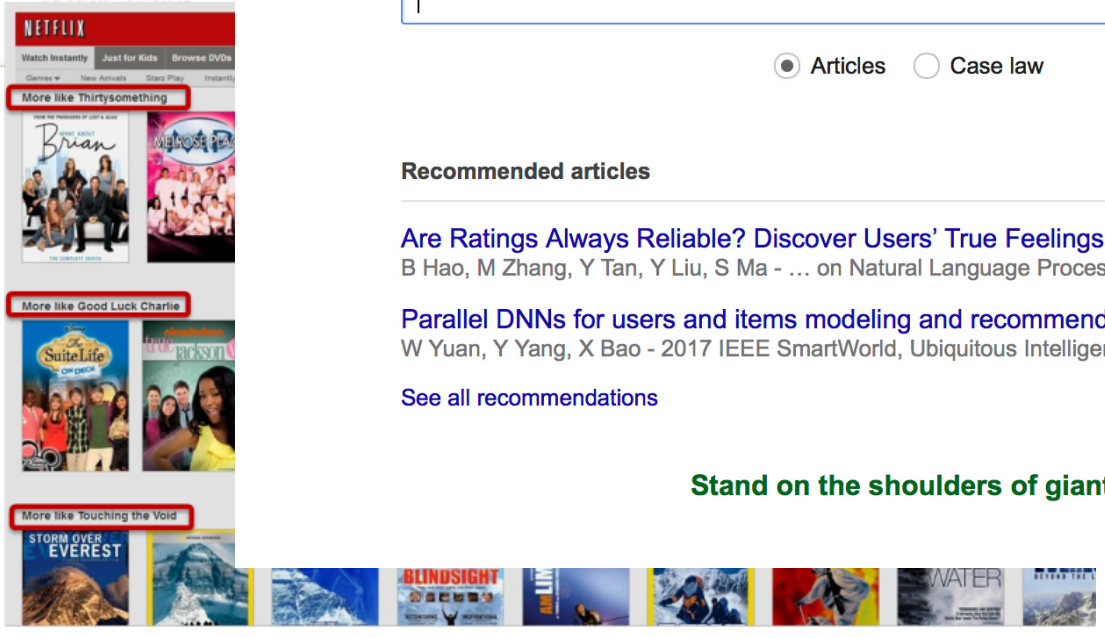

Ricci, F., Rokach, L., Shapira, B., Kantor, Springer 2011
Recommender Systems Handbook


X. Amatriain, MLSS 2014
Collaborative Filtering and other approaches

Illustrated examples

Illustrated examples





Google Scholar

Articles Case law

Recommended articles

Are Ratings Always Reliable? Discover Users' True Feelings with Textual Reviews
B Hao, M Zhang, Y Tan, Y Liu, S Ma - ... on Natural Language Processing and Chinese ..., 2018

Parallel DNNs for users and items modeling and recommendation using comments
W Yuan, Y Yang, X Bao - 2017 IEEE SmartWorld, Ubiquitous Intelligence & ..., 2017

See all recommendations

Stand on the shoulders of giants

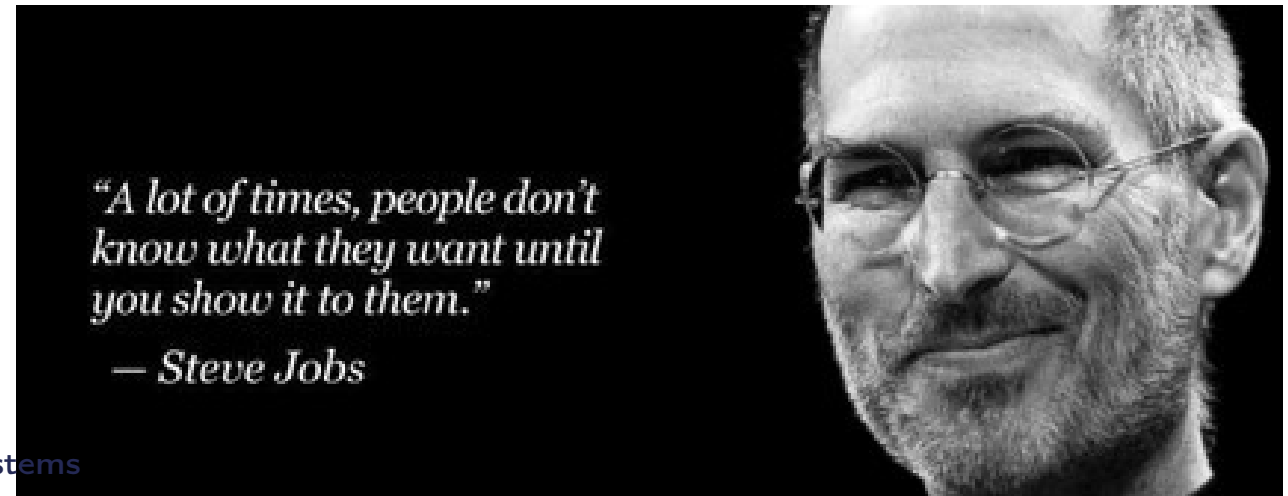
Recommended famous citations

J. O'Brien, *Fortune*, 2006

The Web, they say, is leaving the era of **search** and entering one of **discovery**.
What's the difference? **Search** is what you do when you're looking for something. **Discovery** is when something wonderful that you didn't know existed, or didn't know how to ask for, finds you.

Chris Anderson in *The Long Tail*

We are leaving the age of information and entering the age of recommendation



Why developing a RS? [Seller's perspective]

- Increase the number of items sold
- Sell more diverse items
- Increase the user satisfaction / Increase user fidelity
  - virtuous loop $\Rightarrow$ improving the profiles & exploiting them
- Understand better what the user wants

Why using collaborative tools? [User's perspective]

- Find Some Good Items *[precision issue]*
  - quickly
  - and/or in a huge catalog
- Find all good items *[recall issue]*
  - Layers Information Retrieval tasks
- Being recommended a sequence / a bundle
- Just browsing
- Help others (forum profiles)

Why do most people in the world use Google?

Because the best IR algorithm in the world is:

People who wrote this request clicked on this link

The value of recommendations

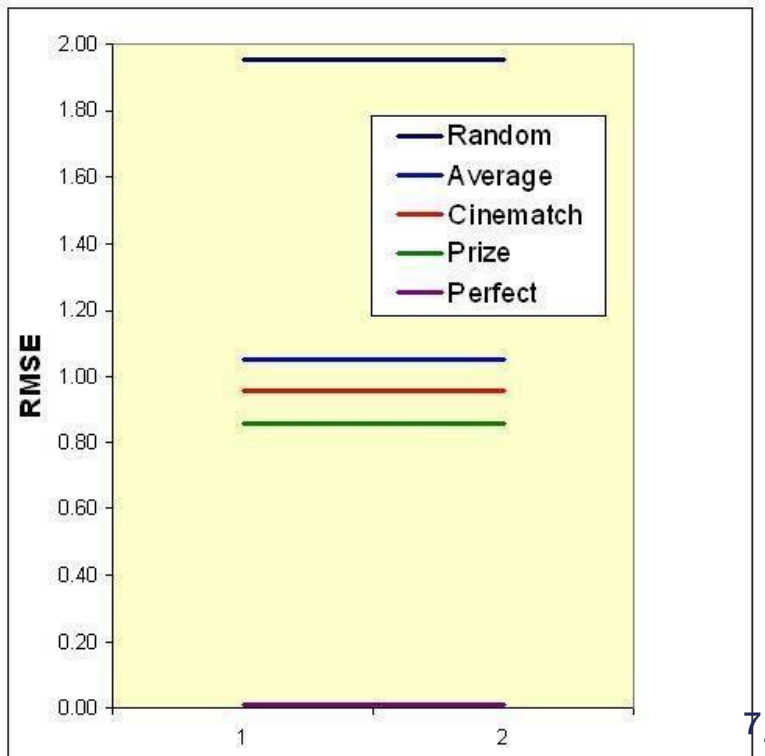
- Netflix: 2/3 of the movies watched are recommended
- Google News: recommendations generate 38% more clickthrough
- Amazon: 35% sales from recommendations
- Choicestream: 28% of the people would buy more music if they found what they liked.

The value of recommendations

- Netflix: 2/3 of the movies watched are recommended
- Google News: recommendations generate 38% more clickthrough
- Amazon: 35% sales from recommendations
- Choicestream: 28% of the people would buy more music if they found what they liked.

However...

Netflix Prize's first conclusion: it is really extremely simple to produce "reasonable" recommendations and extremely difficult to improve them.



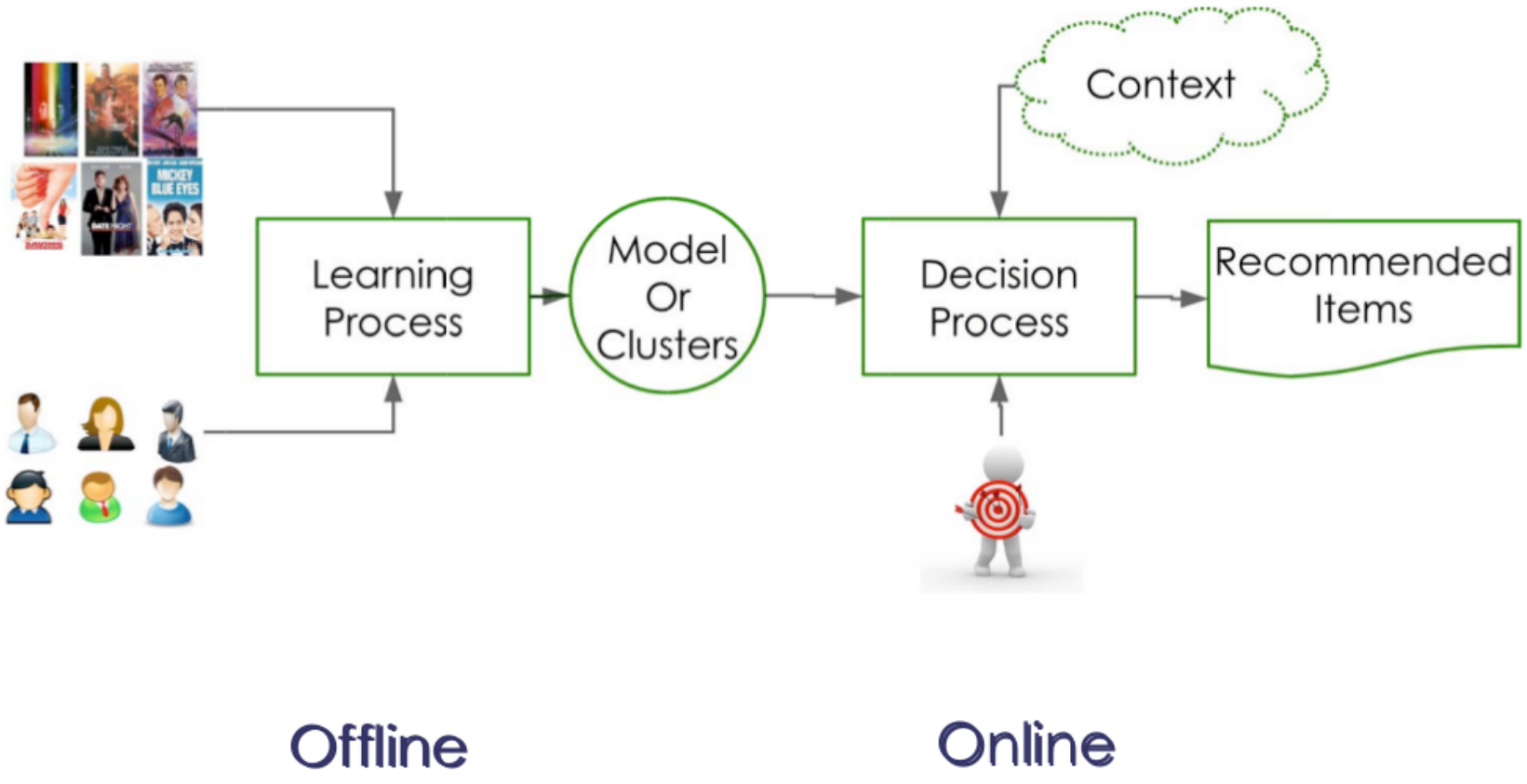
Main steps in RS design

1. Understanding the application field
   - Business expertise
   - ROI
2. Modeling behavior
   - Technical architecture
   - Features engineering
   - Metric definition
3. HMI Integration

In most applications, a RS is a specific IR tool
Request = user u at time t in context c (e.g. visiting item i)

Computational cost

Distinguishing offline & online costs



The Recommender problem

Estimate a *utility function* that automatically predicts how a user **will like** an item.

Based on:

- Past behavior
- Relations to other users
- Item similarity
- Context
  - Time,
  - Sequence,
  - Item description,
  - User categorization:

    age, socio-professional category, ...

Technical architecture: main families of RS

Content based RS

Focus on item **descriptions**:

- Data Base format:

Do the items share several common point?

- Raw text description

Do the texts share several common words/topics?

+ Easy to implement
+ Fast (offline similarity graph)
+ Very stable results
− Poor qualitative results

Collaborative Filtering

Focus on user **interactions**

- User side:

Who have the same behavior?

- Item side:

Which item are visited by the the same users?

+ More relevant output
+ Not so difficult to implement
+ Can be fast (item-to-item reco)
− Can be more expensive

Content Based Recommender Systems

Automation of editorial choices?

Understanding the product description to...

- Know which items are similar

  [global description]

- Focus on common points between various items

  [part of the description]

Paradigm of browsing:
you liked A, did you already consider B,C &D that are close?

Nature of the description & associated metrics

- **Tabular description :**
  - Inside a specific domain (e.g. camera)
  - Descriptive features
    (e.g. definition, zoom, storage capacity, brand, ...)

  $\Rightarrow$ mostly an engineering job + domain expert knowledge
  (understanding & weighting the features)
- **Textual description :**

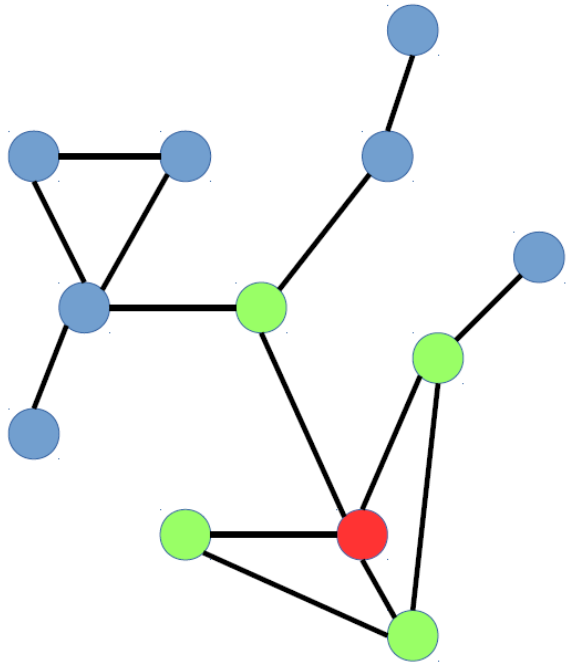
  special session in Information Retrieval (IR)/NLP domain
  - Matching **raw texts**:
    preprocessing issues (stop words, basic language structure, ...)
  - **Keyword**-based Vector Space Model (TF-IDF, etc...)
  - **Topic** modeling: matching in the latent space
    - Internal or external topic modeling
  - **Ontology** / domain specific reference + mapping

  $\Rightarrow$ Choosing a metrics adapted to the representation (cosine for raw texts, KL for topic distribution, ...)

Static implementation & scaling up

Learning step :

1. Feature engineering:
   Item description $\Rightarrow$ relevant vector
2. k-Nearest Neighbors graph
3. Product description update



Inference

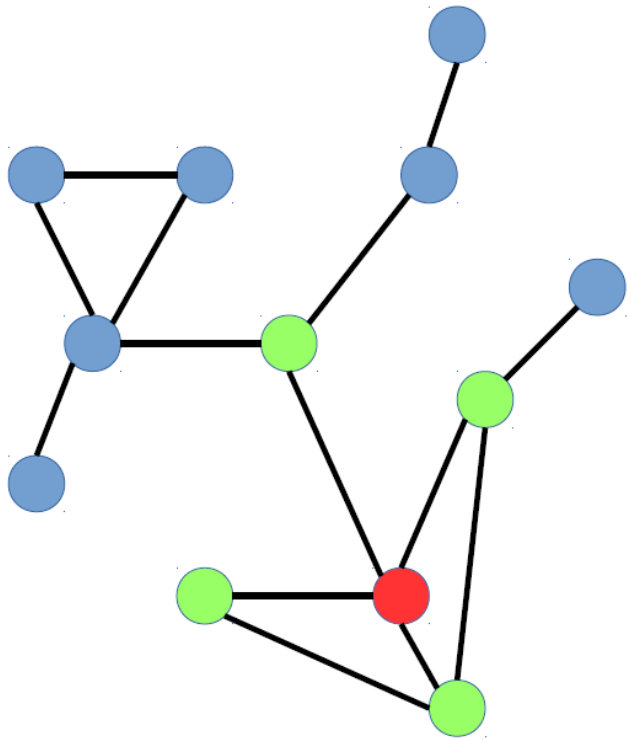
- Presenting new informations within the product description

Issue: How to make the representation relevant?

- Basic description...
- Hierarchy of item (e.g. Cultural goods $\rightarrow$ DVD $\rightarrow$ Western Movies)

Updating a Content Based RS

- Taking into account user feedback:
  - Which links are used or not?
  - Integrating users reviews in the item description
  - Extracting aspects from users reviews + exploiting ratings



Main idea = integrating user feedback...

$\Rightarrow$ sliding to the collaborative filtering paradigm

User profiles

- Case 1 : explicit user profile
  - Textual description of the user...
    User = query, Item = document... An IR task : $p(u|i)$
  - Rocchio's relevance feedback
    1. Query $\Rightarrow$ set of responses
    2. first responses = query enrichments
    3. last (or other documents) $\approx$ negative query

$$\overrightarrow{Q_m} = (a \cdot \overrightarrow{Q_o}) + \left(b \cdot \frac{1}{|D_r|} \cdot \sum_{\overrightarrow{D_j} \in D_r} \overrightarrow{D_j} \right) - \left(c \cdot \frac{1}{|D_{nr}|} \cdot \sum_{\overrightarrow{D_k} \in D_{nr}} \overrightarrow{D_k} \right)$$

- Case 2: no user profile
  - Query = stack of visited items

Pros & Cons

+ Explainable
+ Easy to implement
+ Scale up
+ Can be updated to increase relevance

± (Often) not personalized...
but intrinsically robust to new users !

– Lack of an authority score (as in PageRank)
– Require an item description
– Not adapted to User Generated Contents (intrinsically)

⇒ Mostly an NLP engineering game to obtain baselines that will be combined to CF approaches...

Collaborative filtering Recommender Systems

From k-NN to matrix factorization

General idea

Behavior modeling depending on users' traces:

- User Generated Contents
- Infered information
  - Hyp: you liked what you purchase
  - Hyp: you liked what you visit/rate
  - Hyp: you don't like video you close less that 3 seconds after they started

Interaction data are valuable:

The best information filter is human...

Collaborative filtering = modeling humans from their traces

History: frequent item set

Idea:

Extracting logical rules from (frequent) co-occurences

1. Frequent item set.
   e.g. receipt mining in a supermarket : *Milk*, *Beer*, *Diaper*
2. Extraction of the support. e.g. $(Milk, Diaper) \Rightarrow Beer$

\+ Easy to understand

– Costly (combinatorial search)

– Not very robust to noise

Neighborhood based approaches

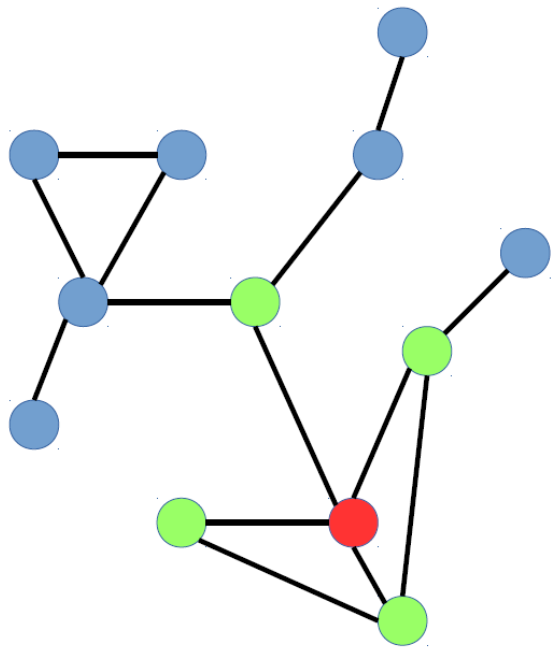
In collaborative filtering...

User domain: if you behave as user u, then you might be interested by u's choices

Item domain: item i is often associated to item i' in users' traces; if you visit i, you might be interested by i'

Same approach than Content Based...
Based on another behavior sensor!

$\Rightarrow$ In the item domain = very **light inference**
... But such RS is **not personalized** !



k-nearest neighbors in the user domain

Easy way to perform a personalized recommendation...

Credit: X. Amatrian

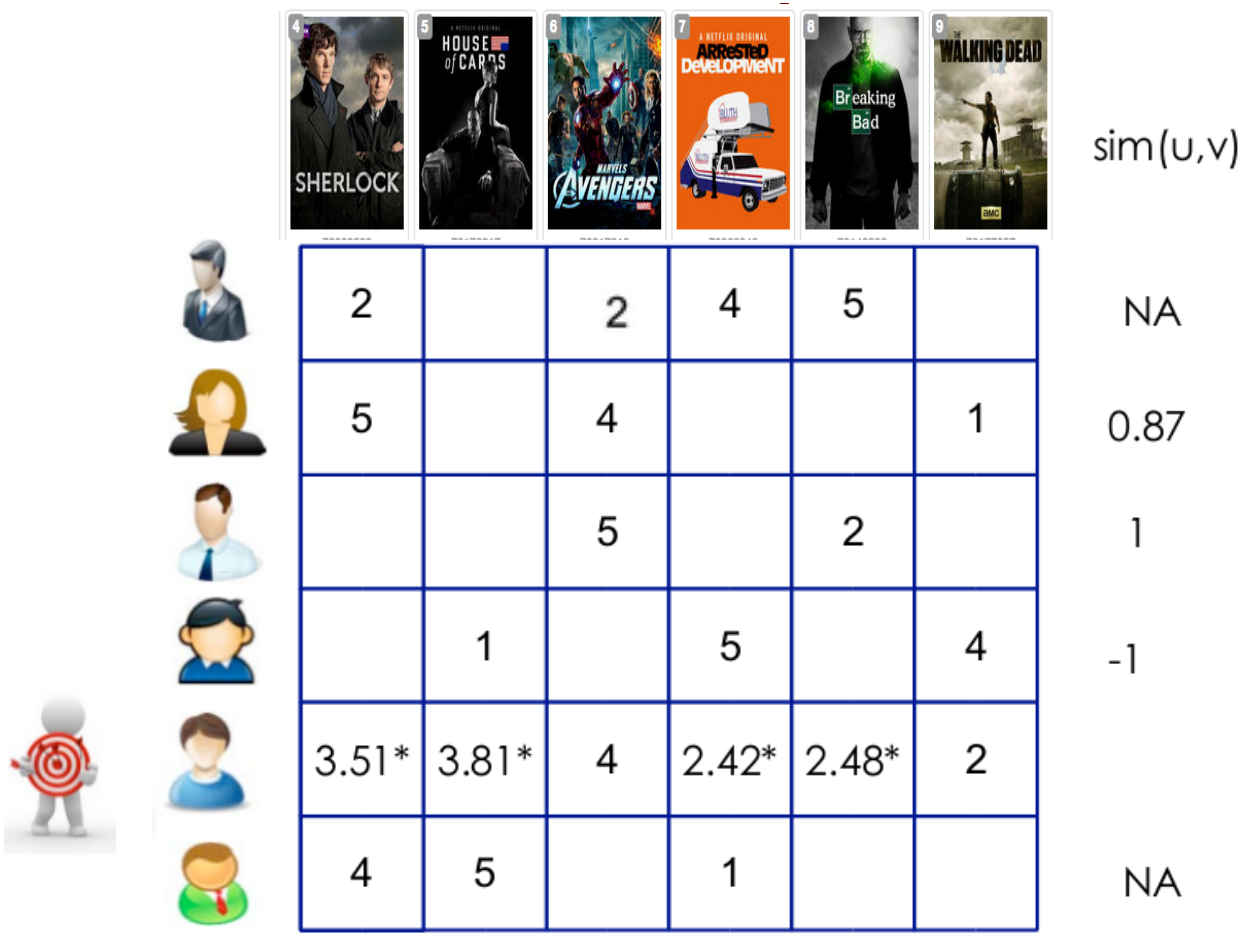
k-nearest neighbors in the user domain

Easy way to perform a personalized recommendation...

Credit: X. Amatrian

k-nearest neighbors in the user domain

Easy way to perform a personalized recommendation...



Credit: X. Amatrian

k-nearest neighbors in the user domain

Easy way to perform a personalized recommendation...

...But very **expensive** in the inference step !

- Bottleneck = Similarity computation + sort
- Complexity is $\mathcal{O}(n_u n_i + k n_u)$
  - Possible approximation (partitioning/hashing space) : LSH
- Possible implementation:
  - Isolate the neighborhood generation and predication steps.
  - "off-line component" / "model" – similarity computation, done earlier & stored in memory.
  - "on-line component" – prediction generation process.

Bi-partite graph approach

Two ways to use neighbors

1. User-based

$$\hat{r}_{ui} = \frac{1}{\#N_u(i)} \sum_{v \in N_u(i)} sim(u, v) r_{uv}$$

2. Item-based

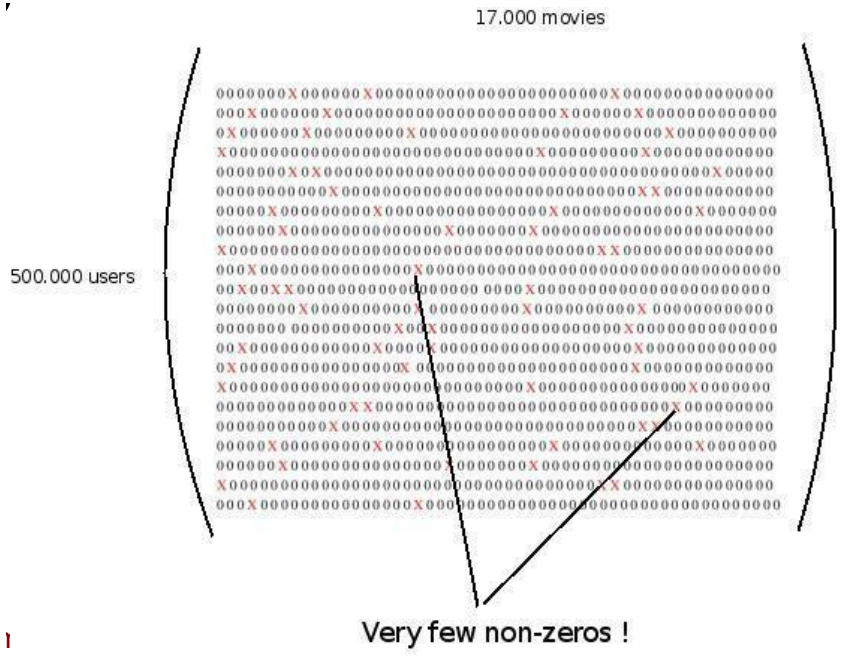
$$\hat{r}_{ui} = \frac{1}{\#N_i(u)} \sum_{j \in N_i(u)} sim(i, j) r_{uv}$$

Missing value paradigm

Netflix Prize rating matrix

If you represent the Netflix Prize rating data in a User/Movie matrix you get...

- 500,000 x 17,000 = 8,500 M positions
- Out of which only 100M are not 0's!



Data Set	users	items	total	density
Jester	48483	100	3519449	0,725
MovieLens	6040	3952	1000209	0,041
EachMovie	74424	1649	2811718	0,022

Matrix factorization

Idea:

Compressing the representation of the matrix based on observed values is a strong way to reconstruct missing values.

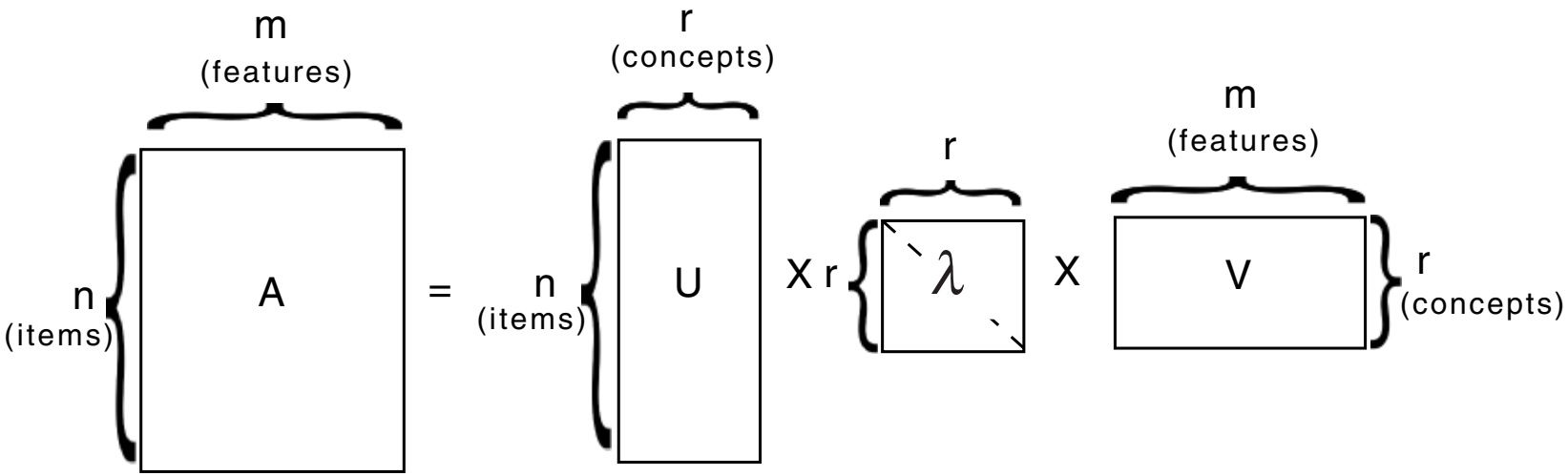
- Singular Value Decomposition (SVD)
- Non Negative Matrix Factorization (NMF)
- ... & many variations

Link with *Minimum Description Length* paradigm:

What is the smallest modeling that can explain observed ratings?

Singular Value Decomposition

Framework of matrix factorization over non square matrix = SVD



D. Billsus and M. J. Pazzani, AAAI 1998
Learning Collaborative Information Filters

- Not adapted to missing values... ⇒ turn into 0.
- Weak reconstruction performance...

⇒ SVD for recommender systems... **Is not an SVD** !

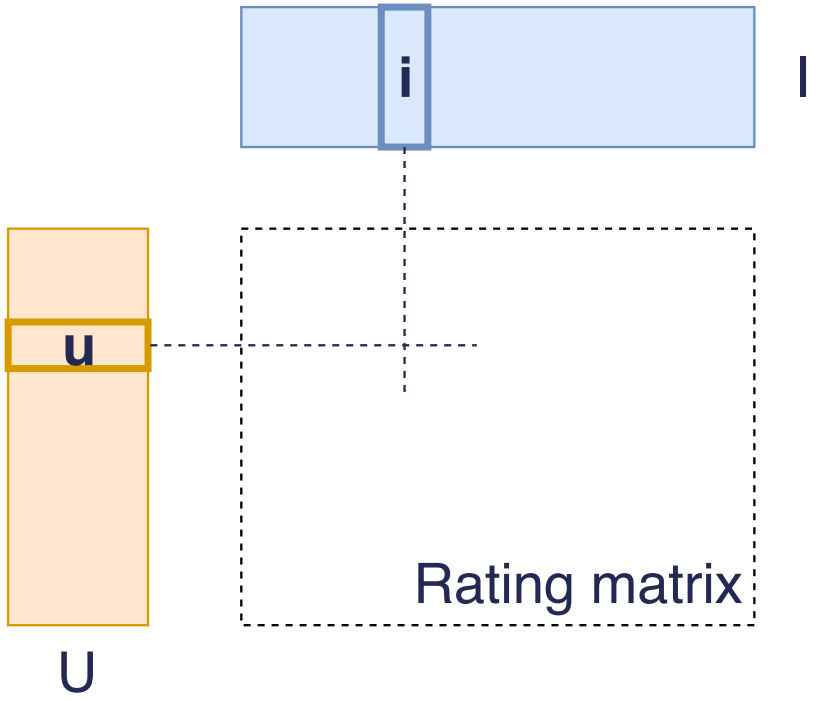
SVD for recommender systems

$$U = \{\mathbf{u}_1, \ldots, \mathbf{u}_{n_u}\}, I = \{\mathbf{i}_1, \ldots, \mathbf{i}_{n_i}\}$$

$$\mathbf{u} \in \mathbb{R}^z, \mathbf{i} \in \mathbb{R}^z$$

$$R = \{(u, i, r_{ui})\}$$

Estimator : $\hat{r}_{ui} = \mathbf{u}_u \cdot \mathbf{i}_i$



$$\mathcal{C} = \sum_{(u,i)\in R} (r_{ui} - \mathbf{u}_u \cdot \mathbf{i}_i)^2$$

Focus on missing values + Mean Square Error (MSE)

$$U^\star, I^\star = \arg\min_{U,I} \sum_{(u,i)\in R} (r_{ui} - \mathbf{u}_u \cdot \mathbf{i}_i)^2$$

SVD issues

1. **Optimization**: (stochastic) gradient descent

$$\nabla_{\mathbf{u}}\mathcal{C} = -\sum_{i|(i,u)\in R} 2\mathbf{u}(r_{ui} - \mathbf{u}_u \cdot \mathbf{i}_i), \qquad \mathbf{u} \leftarrow \mathbf{u} - \varepsilon\nabla_{\mathbf{u}}\mathcal{C}$$

   - Fast convergence ...
   - ... but non convex formulation

2. **Overfitting**:
   Even with $z = 20$: $\#param = 20 \times (n_u + n_i) \geq |R|$
   $\Rightarrow$ Regularization:

$$U^\star, I^\star = \arg\min_{U,I} \sum_{(u,i)\in R} (r_{ui} - \mathbf{u}_u \cdot \mathbf{i}_i)^2 + \lambda_u \|U\|_F^2 + \lambda \|I\|_F^2$$

   - Implementation : penalizing weights every λ iterations

Introducing the bias

Let's go back to the basics... & the baselines

- General bias: $b = \bar{r}$
- User bias: $b_u = \frac{1}{|\{i | r_{ui} \neq \emptyset\}|} \sum_{i | r_{ui} \neq \emptyset} r_{ui}$

Hyp: one user always gives the same rate

- Item bias: $b_i = \frac{1}{|\{u | r_{ui} \neq \emptyset\}|} \sum_{u | r_{ui} \neq \emptyset} r_{ui}$

Strong Hyp: one item is always evaluated with the same rate

We obtain **three baselines**... And an advanced formulation:

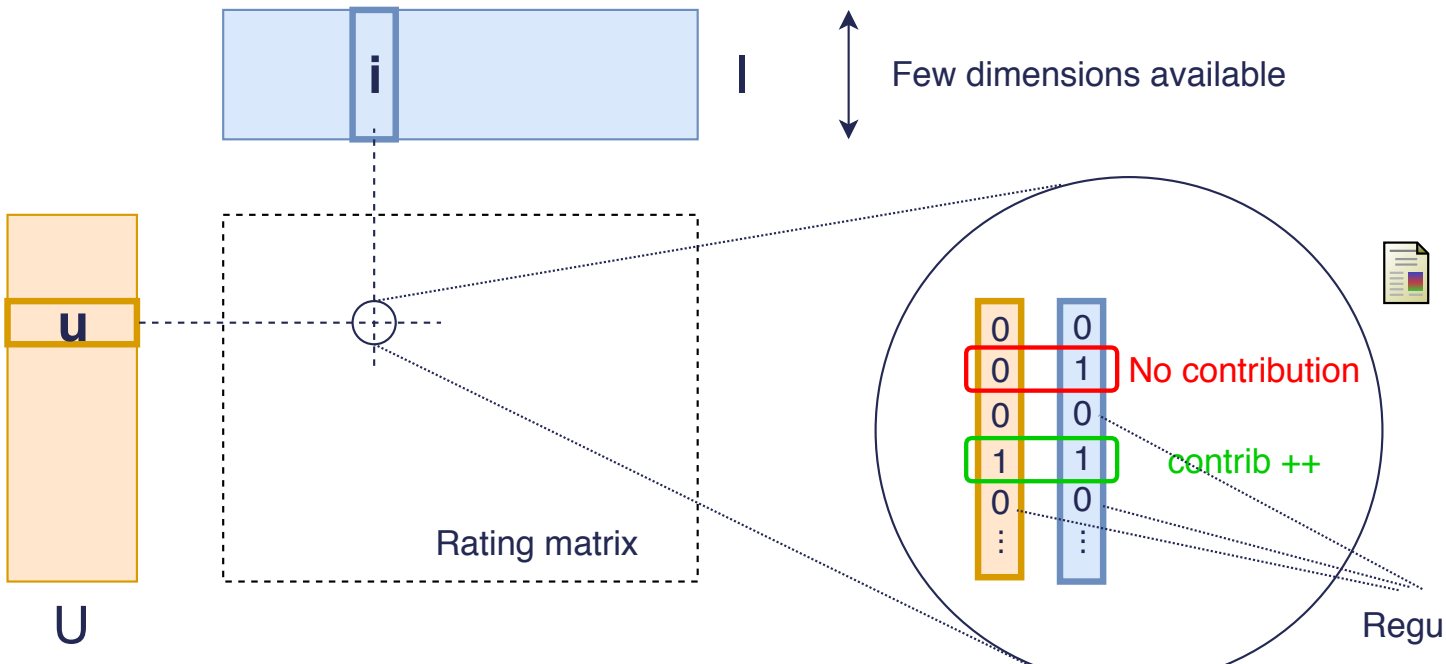
$$\hat{r}_{ui} = b + b_u + b_i + \mathbf{u}_u \cdot \mathbf{i}_i$$

$\Rightarrow$ $\mathbf{u}_u, \mathbf{i}_i$ profiles encode the deviation wrt basic predictions

NMF: the promise of understandable aspects

NMF: Non-negative Matrix Factorization = SVD + $\mathbf{u} \geq 0$ + $\mathbf{i} \geq 0$

$$U^\star, I^\star = \arg\min_{U,I} \sum_{(u,i)\in R} (r_{ui} - \mathbf{u}_u \cdot \mathbf{i}_i)^2 + \lambda_u \|U\|_F^2 + \lambda \|I\|_F^2$$

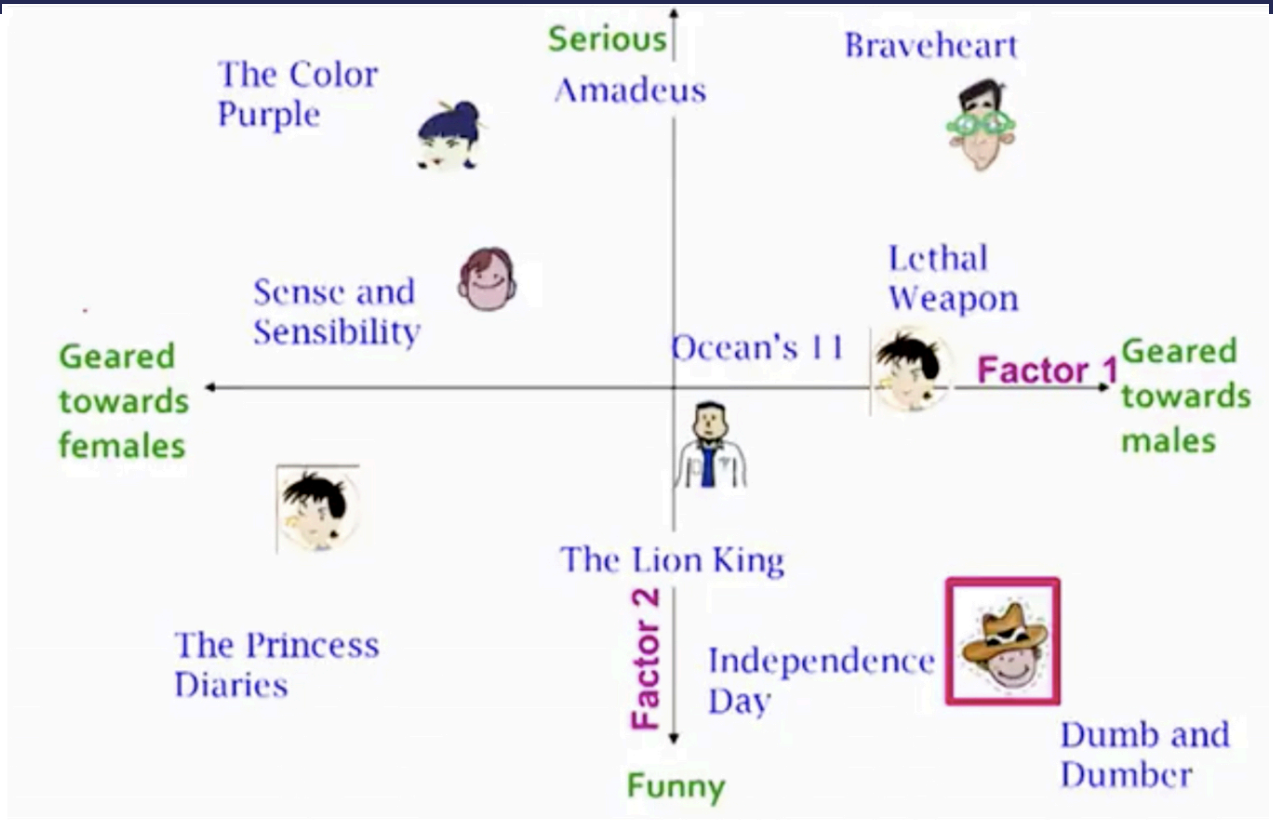


P. Hoyer, JMLR 2004
Non-negative matrix factorization with sparseness constraints

- Regularization brings rating to 0
- Issue = finding aspects shared by many users

NMF/SVD variation

What we expect:

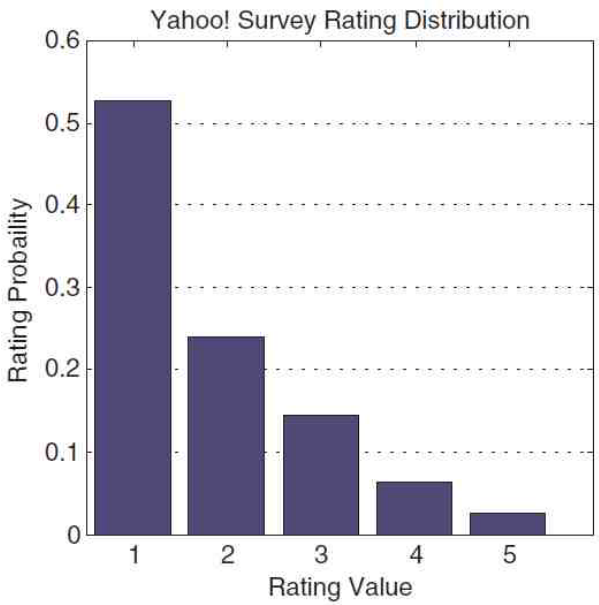


- Iterative procedure + simple SGD
- Easy to enforce constraint:
  - Orthogonality
  - Specific initialization
  - Modeling of negative agreements
  - ...
- Reasonable learning cost
- Fast inference

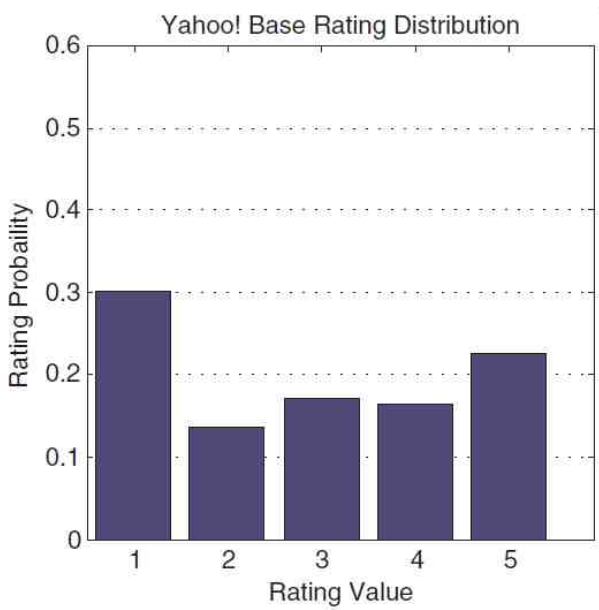
Weakness of MCAR hypothesis

Data are not Missing Completely At Random...

Graphs from [Marlin & Zemel '09]:



Survey: ask users to rate a random list of items: approximates **complete** data



Typical Data: users are free to choose which items to rate -> available data are **MNAR** :

instead of giving low ratings, users tend to not give a rating at all.

Weakness of MCAR hypothesis

Data are not Missing Completely At Random...

Predicting profile behavior on this kind of data:

H. Steck, KDD, 2010
Training and Testing of Recommender Systems on Data Missing Not at Random

Table 1: Simplistic Example for ratings missing not at random (MNAR): test data where users rated only what they liked or knew.

		users							
		horror fans				romance lovers			
	h	5		5	5				
m	o	5	5						
o	r		5		5				
v	.	5		5	5				
i	r					5	5		5
e	o						5	5	5
s	m					5		5	
	.						5	5	5

Credit: H. Steck

Weakness of MCAR hypothesis

Data are not Missing Completely At Random...

Predicting profile behavior on this kind of data:

H. Steck, KDD, 2010
Training and Testing of Recommender Systems on Data Missing Not at Random

Table 1: Simplistic Example for ratings missing not at random (MNAR): test data where users rated only what they liked or knew.

		users							
		horror fans				romance lovers			
	h	5		5	5				
m	o	5	5						
o	r		5		5				
v	.	5		5	5				
i	r					5	5		5
e	o						5	5	5
s	m					5		5	
	.						5	5	5

Credit: H. Steck

Several outcomes:

- Changing the error function
  - ranking criteria
- Changing the task
  - predicting item rank (not the rating)

Bayesian Personalized Ranking

Learning :

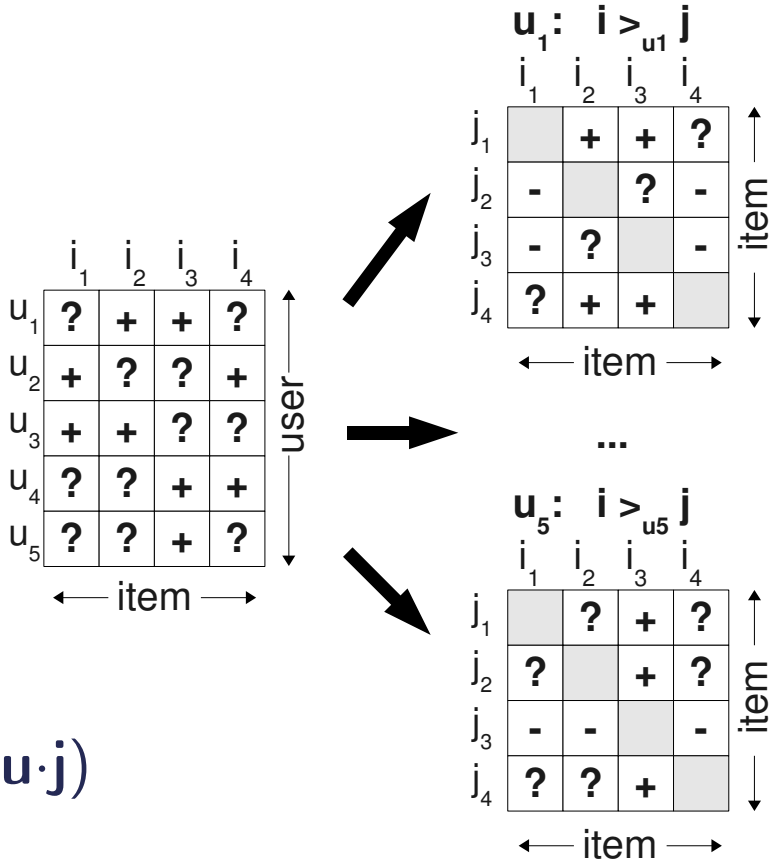
$$p(i >_u j|\theta) = \frac{1}{1+\exp(-f_\theta(u,i,j))}$$

For instance (inspired from MF):

$$f_\theta(u,i,j) = \mathbf{u}\cdot\mathbf{i} - \mathbf{u}\cdot\mathbf{j}$$

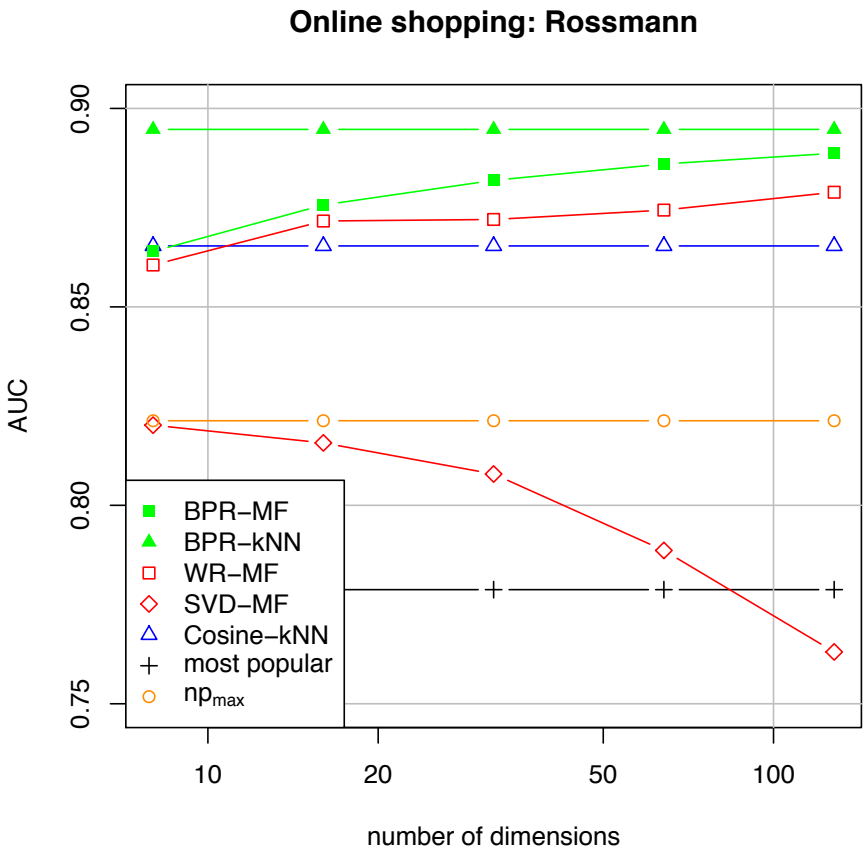
Evaluation =

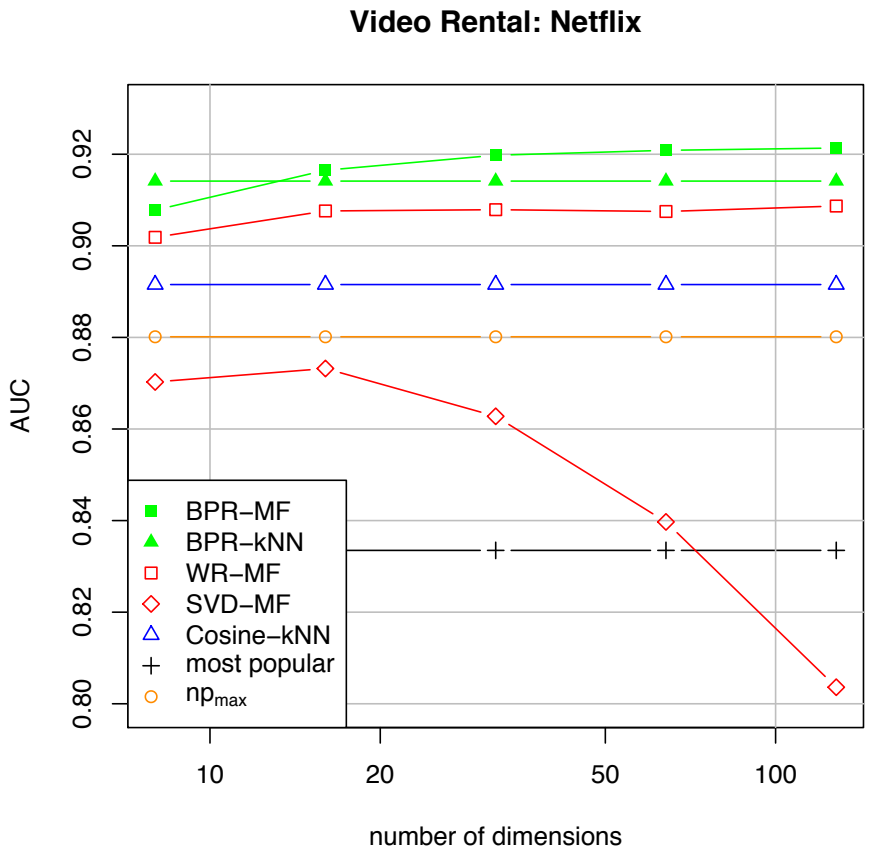
$$AUC = \frac{1}{n_u}\sum_u \frac{1}{|E(u)|}\sum_{(i,j)\in E(u)} \delta(\mathbf{u}\cdot\mathbf{i} > \mathbf{u}\cdot\mathbf{j})$$



S. Rendle et al. UAI 2009
BPR: Bayesian Personalized Ranking from Implicit Feedback,

Bayesian Personalized Ranking





S. Rendle et al. UAI 2009
BPR: Bayesian Personalized Ranking from Implicit Feedback,

Implicit Feedback (SVD++)

For example, a dataset shows that users that rate "Lord of the Rings 3" high also gave high ratings to "Lord of the Rings 1–2".
$\Rightarrow$ establish high weights from "Lord of the Rings 1–2" to "Lord of the Rings 3".
Now, if a user did not rate "Lord of the Rings 1–2" at all, his predicted rating for "Lord of the Rings 3" will be penalized.

- Binary coding (cf BPR) + predictor f
- $R(u)$: set of items rated by u + f

$\Rightarrow$ $N(u)$: set of items implicitly rated by u

Yehuda Koren, KDD 2008
Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model

SVD++

General idea:

$$\hat{r}_{ui} = \underbrace{b + b_u + b_i}_{b_{ui}} + \frac{1}{\sqrt{|R(u)|}} \sum_{j \in R(u)} (r_{uj} - b_{uj}) w_{ij} + \frac{1}{\sqrt{|N(u)|}} \sum_{j \in N(u)} c_{ij}$$

Overweighting deviation for prolific users ($\frac{1}{\sqrt{|R(u)|}}$ instead of $\frac{1}{|R(u)|}$)

w_{ij} Learning deviation meaning wrt b_{uj}

c_{ij} Learning the meaning of j absence wrt i

- ... Too expensive (too many coefficients to learn)

Factorized formulation = SVD++:

$$\hat{r}_{ui} = b_{ui} + \mathbf{i} \cdot \left(\mathbf{u} + \frac{1}{\sqrt{|N(u)|}} \sum_{j \in N(u)} \mathbf{y}_j \right)$$

Factorization machine

Back to linear model !

Feature vector **x**

	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...	Time	TI	NH	SW	ST	...	Target y	
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	13	0	0	0	0	...	5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...	3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...	1	$y^{(2)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...	4	$y^{(3)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...	5	$y^{(4)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...	1	$y^{(5)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...	5	$y^{(6)}$
	User				Movie					Other Movies rated					Time	Last Movie rated						

$U = \{\text{Alice (A)}, \text{Bob (B)}, \text{Charlie (C)}, \ldots\}$
$I = \{\text{Titanic (TI)}, \text{Notting Hill (NH)}, \text{Star Wars (SW)}, \text{Star Trek (ST)}, \ldots\}$

$S = \{(\text{A}, \text{TI}, \text{2010-1}, 5), (\text{A}, \text{NH}, \text{2010-2}, 3), (\text{A}, \text{SW}, \text{2010-4}, 1), (\text{B}, \text{SW}, \text{2009-5}, 4), (\text{B}, \text{ST}, \text{2009-8}, 5), (\text{C}, \text{TI}, \text{2009-9}, 1), (\text{C}, \text{SW}, \text{2009-12}, 5)\}$

S. Rendle, ICDM 2010
Factorization machines

Pros & Cons

+ Requires minimal knowledge engineering efforts
+ Users and products are symbols without any internal structure or characteristics
+ Produces good-enough results in most cases
– Collaborative filtering is very sensitive to cold start issues (both in the item & user domains)
– Requires a large number of reliable "user feedback data points" to bootstrap
– Requires products to be standardized (users should have bought exactly the same product)

Evaluation:
evaluation metrics
vs
learning metrics

How to evaluate RS performance?

Warning

We should not confuse **evaluation metrics** & **learning metrics**

$\Rightarrow$ MSE is a convenient learning metrics

(easily differentiable + convex ...)

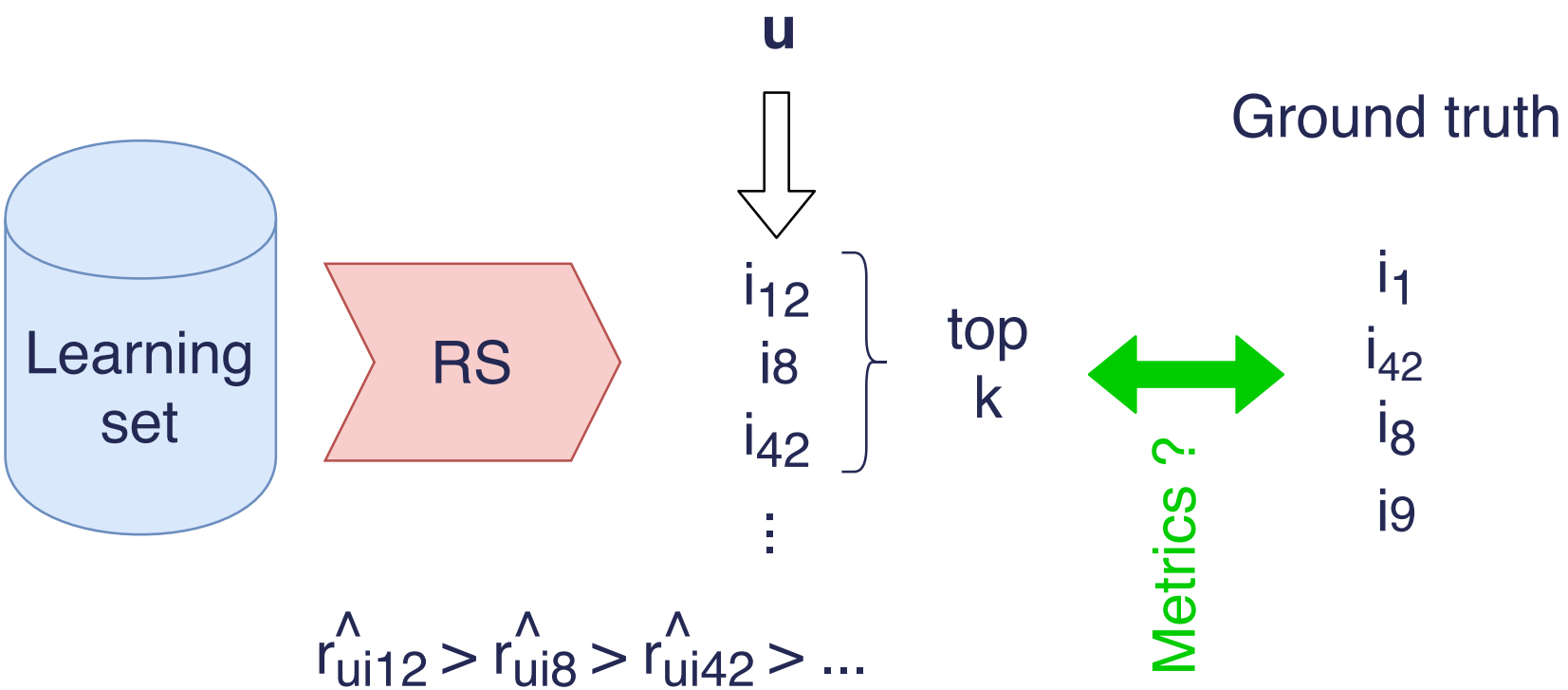
... but it is a poor evaluation metrics

... cf Netflix Challenge feedbacks

It do not tell us if we provide relevant suggestions

- What are the other available metrics?
- Can we use those metrics during the learning step?

Precision / Recall



- **Precision :** Among our k prediction, how many are in the ground truth?
- **Recall :** Among our k prediction, what is the GT coverage ?

1/0 labeling, AUC metrics

- Rendle popularize both 1/0 prediction & AUC metrics
- AUC = tradeoff between precision & recall
  - Percentage of correct binary ranking

$$AUC = \frac{1}{n_u} \sum_u \frac{1}{|E(u)|} \sum_{(i,j) \in E(u)} \delta(\mathbf{u} \cdot \mathbf{i} > \mathbf{u} \cdot \mathbf{j})$$

\+ k not required

\− **top** of the list = same impact as **bottom** of the list

Mean Average Precision (from the IR domain)

RS aim at proposing an ordered list of suggestion...

Which **head** **is far more important** than the rest.

For a user u with 4 liked items to discover:

$$query = \mathbf{u} \Rightarrow RS_1 \Rightarrow \begin{bmatrix} i_{12} \\ i_8 \\ i_{42} \\ i_1 \end{bmatrix} \quad \Leftrightarrow \quad \begin{bmatrix} i_1 \\ i_{42} \\ i_8 \\ i_9 \end{bmatrix} = GT$$

- Average precision :

$$\frac{1}{4}\sum_{k=1}^{4} precision@k = \frac{1}{4}(0 + \frac{1}{2} + \frac{2}{3} + \frac{3}{4}) = 0.478$$

- Mean Average Precision = averaging over the whole population

Mean Average Precision (from the IR domain)

RS aim at proposing an ordered list of suggestion...

Which **head** **is far more important** than the rest.

For a user u with 4 liked items to discover:

$$query = \mathbf{u} \Rightarrow RS_2 \Rightarrow \begin{bmatrix} i_1 \\ i_8 \\ i_{42} \\ i_{12} \end{bmatrix} \quad \Leftrightarrow \quad \begin{bmatrix} i_1 \\ i_{42} \\ i_8 \\ i_9 \end{bmatrix} = GT$$

- Average precision :

$$\frac{1}{4}\sum_{k=1}^{4} precision@k = \frac{1}{4}(1+1+1+\frac{3}{4}) = 0.9375$$

- Mean Average Precision =

averaging over the whole population

nDCG : Normalized Discounted Cumulative Gain

We assume that we have a relevance score for each item...

$$query = \mathbf{u} \Rightarrow RS \Rightarrow \left[\begin{array}{ll} i_{12} & ind = 1 \\ i_8 & ind = 2 \\ i_{42} & ind = 3 \\ i_1 & ind = 4 \end{array}\right] \Leftrightarrow \left[\begin{array}{l} 0 \\ 2 \\ 3 \\ 3 \end{array}\right] = relevance$$

$$\mathrm{DCG_p} = \sum_{ind=1}^{p} \frac{rel_{ind}}{\log_2(ind+1)} = 0 + 1.26 + 1.5 + 1.29 = 4.05$$

$$\mathrm{nDCG} = \frac{\mathrm{DCG}}{\mathrm{IdealDCG}} = \frac{4.05}{3 + 1.89 + 1 + 0/0.86} = 0.69/0.6$$

Relative ideal *vs* Absolute ideal

Learning to rank

- Pointwise :
  - Ranking score based on regression or classification
- Pairwise :
  - Loss function is defined on pair-wise preferences
  - RankSVM, RankBoost, RankNet, FRank...
- Listwise :
  - Gradient descent on smoothed version of objective function (e.g. CLiMF presented at Recsys 2012 or TFMAP at SIGIR 2012)
  - SVM-MAP relaxes the MAP metric by adding it to the SVM constraints
  - AdaRank $\Rightarrow$ optimize NDCG

A/B testing & production launch

In a real situation

Designing an online Recommender System offers new performance indicators

- Online click, purchase, etc

A/B **testing:**

1. Defining some performance indicator with expert
2. Re-direct a small part of the customers to the new system B
   - make sure that the redirection is random (not biased)
3. Compare indicators from A and B

$\Rightarrow$ **Best evaluation**...

But only available **online** & with **access to the backoffice**

Conclusion

Main issues

- Data quality
- Scaling up
- Cold start
- Time modeling
- Context modeling

Conclusions

- For many applications such as Recommender Systems (but also Search, Advertising, and even Networks) understanding data and users is vital
- Algorithms can only be as good as the data they use as input
- But the inverse is also true: you need a good algorithm to leverage your data
- Importance of User/Data Mining is going to be a growing trend in many areas in the coming years
- RS have the potential to become as important as Search is now

$\Rightarrow$ there are still many open questions and a lot of interesting research to do!