Introduction to Recommender Systems

Fabio Petroni

SAPIENZA
UNIVERSITÀ DI ROMA

About me

Fabio Petroni

Sapienza University of Rome, Italy

Current position:

PhD Student in Engineering in Computer Science

Research Interests:

data mining, machine learning, big data

petroni@dis.uniroma1.it

- slides available at
  http://www.fabiopetroni.com/teaching

Materials

- **Xavier Amatriain** Lecture at *Machine Learning Summer School 2014*, Carnegie Mellon University
  - https://youtu.be/bLhq63ygoU8
  - https://youtu.be/mRToFXlNBpQ

- *Recommender Systems* course by **Rahul Sami** at Michigan's Open University
  - http://open.umich.edu/education/si/si583/winter2009

- *Data Mining and Matrices* Course by **Rainer Gemulla** at University of Mannheim
  - http://dws.informatik.uni-mannheim.de/en/teaching/courses-for-master-candidates/ie-673-data-mining-and-matrices/

Age of discovery

The Age of Search has come to an end

- **... long live the Age of Recommendation!**
- **Chris Anderson in “The Long Tail”**
  - *“We are leaving the age of information and entering the age of recommendation”*
- **CNN Money, “The race to create a 'smart' Google”:**
  - *“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.”*

Web Personalization & Recommender Systems

- Most of todays internet businesses deeply root their success in the ability to provide users with strongly personalized experiences.

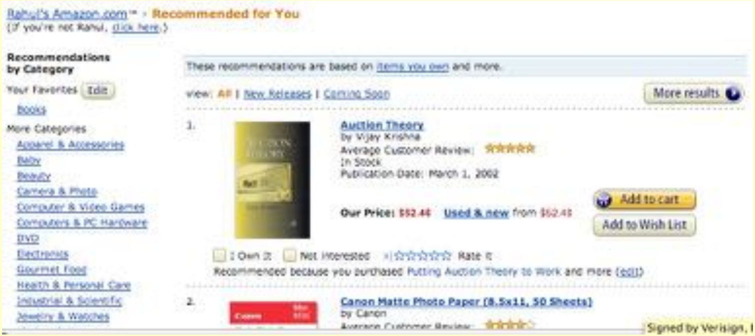
- Recommender Systems are a particular type of personalized Web-based applications that provide to users personalized recommendations about content they may be interested in.

Example 1


StumbleUpon
Hi.
We're StumbleUpon.
We help you explore new and interesting things from every corner of the Web.
Join for Free
Explore Anywhere
Discover the Web with our over 20 million members
Explore an Interest
Check out our Explore Box

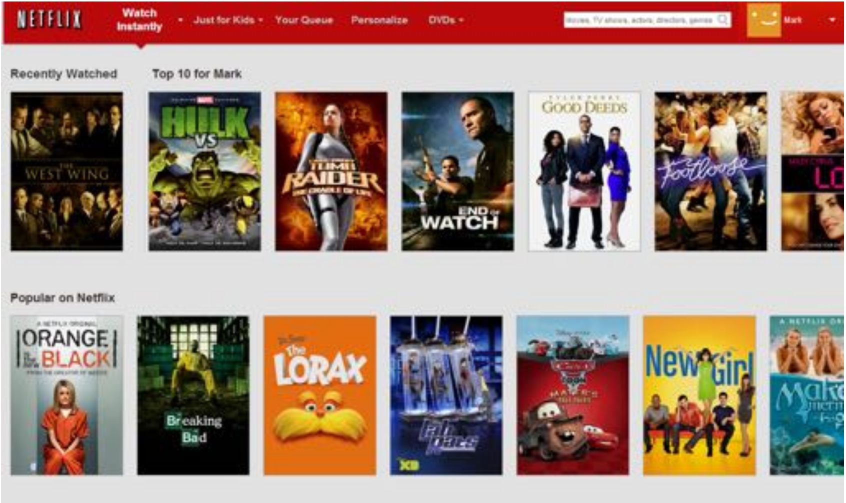
Example 2

Example: Amazon Recommendations

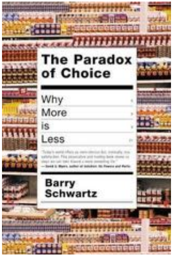


© FAIR USE http://www.amazon.com/

Example 3


NETFLIX
Watch Instantly
Just for Kids
Your Queue
Personalize
DVDs
Mark
Recently Watched
Top 10 for Mark
THE WEST WING
HULK VS
TOMB RAIDER
END of WATCH
GOOD DEEDS
Footloose
Popular on Netflix
ORANGE is the new BLACK
Breaking Bad
The LORAX
New Girl

Information overload

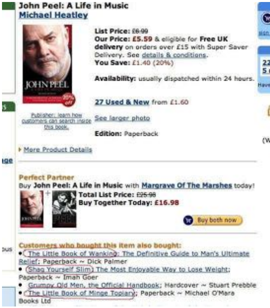


"People read around 10 MB worth of material a day, hear 400 MB a day, and see 1 MB of information every second" - The Economist, November 2006

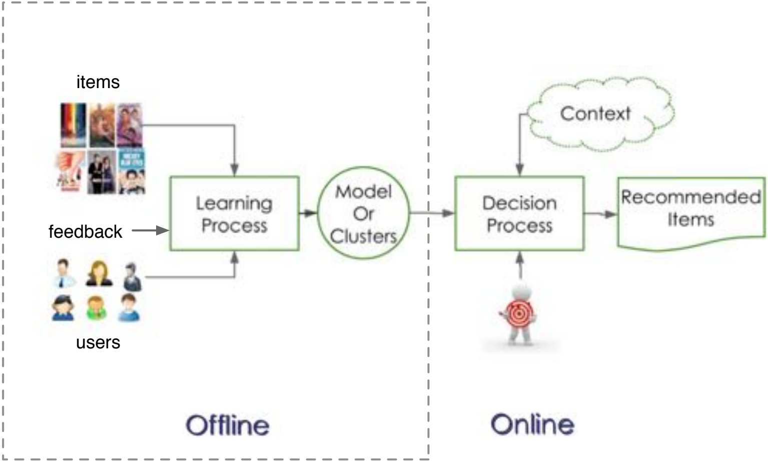
In 2015, consumption will raise to 74 GB a day - UCSD Study 2014

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.



Recommendation process



Input

Sources of information

- Explicit ratings on a numeric/ 5-star/3-star etc. scale
- Explicit binary ratings (thumbs up/thumbs down)
- Implicit information, e.g.,
  - who bookmarked/linked to the item?
  - how many times was it viewed?
  - how many units were sold?
  - how long did users read the page?
- Item descriptions/features
- User profiles/preferences

Methods of a aggregating inputs

- Content-based filtering
  - recommendations based on item descriptions/features, and profile or past behavior of the "target" user only.

- Collaborative filtering
  - look at the ratings of like-minded users to provide recommendations, with the idea that users who have expressed similar interests in the past will share common interests in the future.

Collaborative Filtering

- Collaborative Filtering (**CF**) represents today's a widely adopted strategy to build recommendation engines.

Collaborative Filtering: Lifeblood of The Social Web

- CF analyzes the known preferences of a group of users to make predictions of the unknown preferences for other users.

Collaborative filtering

- problem
  - set of users
  - set of items (movies, books, songs, ...)
  - feedback
    - explicit (ratings, ...)
    - implicit (purchase, click-through, ...)
- predict the preference of each user for each item
  - assumption: similar feedback $\leftrightarrow$ similar taste
- example (explicit feedback):

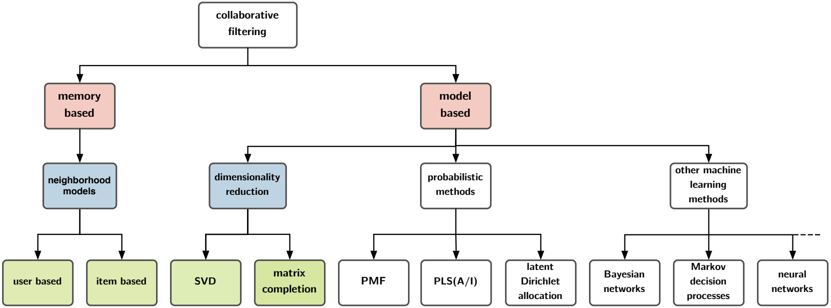
	Avatar	The Matrix	Up
Marco		4	2
Luca	3	2	
Anna	5		3

Collaborative filtering

- problem
  - set of users
  - set of items (movies, books, songs, ...)
  - feedback
    - explicit (ratings, ...)
    - implicit (purchase, click-through, ...)
- predict the preference of each user for each item
  - assumption: similar feedback $\leftrightarrow$ similar taste
- example (explicit feedback):

	Avatar	The Matrix	Up
Marco	?	4	2
Luca	3	2	?
Anna	5	?	3

Collaborative filtering taxonomy



- **Memory-based** use the ratings to compute similarities between users or items (the "memory" of the system) that are successively exploited to produce recommendations.
- **Model-based** use the ratings to estimate or learn a model and then apply this model to make rating predictions.

Memory based

neighborhood models

The CF Ingredients

- List of **m Users** and a list of **n Items**
- Each user has a **list of items** with associated **opinion**
  - **Explicit opinion** - a rating score
  - Sometime the rating is **implicitly** – purchase records or listen to tracks
- **Active user** for whom the CF prediction task is performed
- **Metric** for measuring **similarity between users**
- Method for selecting a subset of **neighbors**
- Method for **predicting a rating** for items not currently rated by the active user.

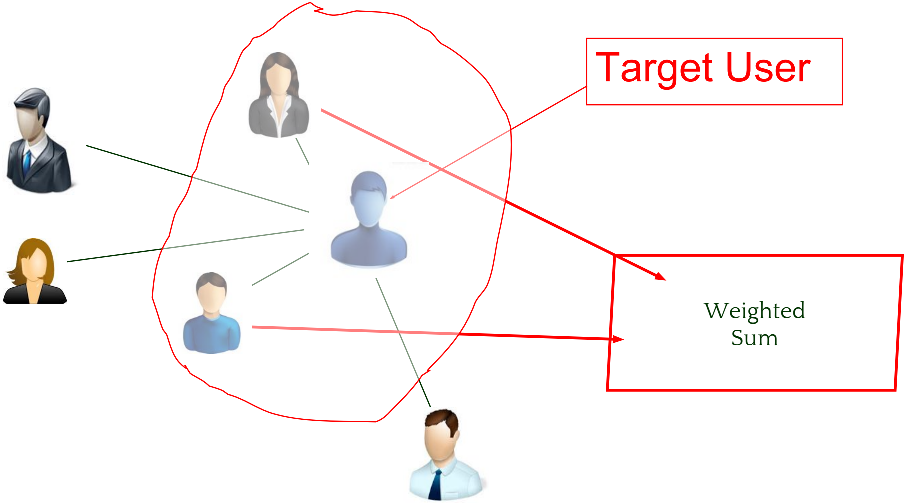
Collaborative Filtering

The basic steps:

1. Identify set of ratings for the **target/active user**
2. Identify set of users most similar to the target/active user according to a similarity function (**neighborhood** formation)
3. Identify the products these similar users liked
4. **Generate a prediction** - rating that would be given by the target user to the product - for each one of these products
5. Based on this predicted rating recommend a set of top N products

User-based Collaborative Filtering

User-User Collaborative Filtering



UB Collaborative Filtering

- A collection of user u_i, $i=1, \ldots n$ and a collection of products p_j, $j=1, \ldots, m$
- An $n \times m$ matrix of ratings v_{ij} , with $v_{ij} = ?$ if user i did not rate product j
- Prediction for user i and product j is computed

$$v_{ij}^* = K \sum_{v_{kj} \neq ?} u_{jk} v_{kj} \quad \text{or} \quad v_{ij}^* = v_i + K \sum_{v_{kj} \neq ?} u_{jk} (v_{kj} - v_k)$$

- Similarity can be computed by Pearson correlation

$$u_{ik} = \frac{\sum_j (v_{ij} - v_i)(v_{kj} - v_k)}{\sqrt{\sum_j (v_{ij} - v_i)^2 \sum_j (v_{kj} - v_k)^2}} \quad \text{or} \quad \cos(u_i, u_j) = \frac{\sum_{k=1}^{m} v_{ik} v_{jk}}{\sqrt{\sum_{k=1}^{m} v_{ik}^2 \sum_{k=1}^{m} v_{jk}^2}}$$

User-based CF Example

						sim(u,v)
2			4	5		NA
5		4			1	
		5		2		
	1		5		4	
		4			2	
4	5		1			NA

User-based CF Example

						sim(u,v)
2			4	5		NA
5		4			1	0.87
		5		2		
	1		5		4	
		4			2	
4	5		1			NA

User-based CF Example

						sim(u,v)
2			4	5		NA
5		4			1	0.87
		5		2		1
	1		5		4	
		4			2	
4	5		1			NA

User-based CF Example

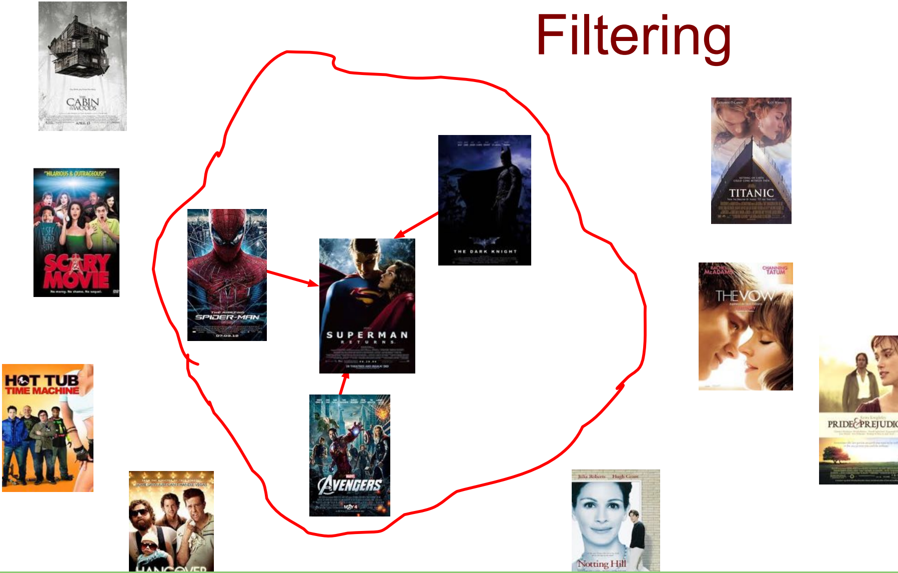
						sim(u,v)
2			4	5		NA
5		4			1	0.87
		5		2		1
	1		5		4	-1
		4			2	
4	5		1			NA

User-based CF Example

						sim(u,v)
2			4	5		NA
5		4			1	0.87
		5		2		1
	1		5		4	-1
3.51*	3.81*	4	2.42*	2.48*	2	
4	5		1			NA

Item-based Collaborative Filtering

Item-Item Collaborative Filtering



Item Based CF Algorithm

- Look into the items the target user has rated
- Compute how similar they are to the target item
  - Similarity **only using** past **ratings** from other users!
- Select k most similar items.
- Compute Prediction by taking weighted average on the target user's ratings on the most similar items.

Item Similarity Computation

- Similarity between items *i* & *j* computed by finding users who have rated them and then applying a similarity function to their ratings.
- <u>Cosine-based Similarity</u> – items are vectors in the m dimensional user space (difference in rating scale between users is not taken into account).

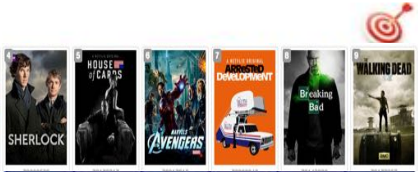
$$S(i,j)=\cos(\vec{i},\vec{j})=\frac{\vec{i}\cdot\vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

Prediction Computation

- Generating the prediction – look into the target users ratings and use techniques to obtain predictions.
- Weighted Sum – how the active user rates the similar items.

$$P_{u,i} = \frac{\sum_{all\ similar\ items,N} (S_{i,N} * R_{u,N})}{\sum_{all\ similar\ items,N} (|S_{i,N}|)}$$

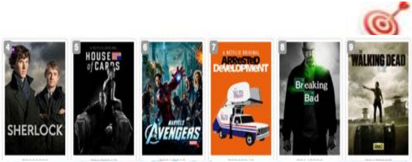
Item-based CF Example



2			4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

sim(i,j) 1

Item-based CF Example

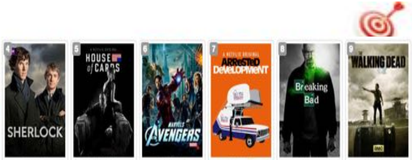


	2			4	5	
	5		4			1
			5		2	
		1		5		4
			4			2
	4	5		1		
sim(i,j)	-1	-1				

Item-based CF Example

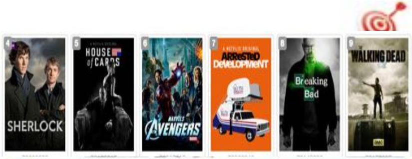
	2			4	5	
	5		4			1
			5		2	
		1		5		4
			4			2
	4	5		1		
sim(i,j)	-1	-1	0.86			

Item-based CF Example



	2			4	5	
	5		4			1
			5		2	
		1		5		4
			4			2
	4	5		1		
sim(i,j)	-1	-1	0.86	1		

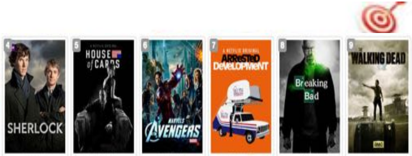
Item-based CF Example



	2			4	5	
	5		4			1
			5		2	
		1		5		4
			4			2
	4	5		1		
sim(i,j)	-1	-1	0.86	1	NA	

sim(6,5) cannot be calculated

Item-based CF Example



	2			4	5	2.94*
	5		4			1
			5		2	2.48*
		1		5		4
			4			2
	4	5		1		1.12*
sim(i,j)	-1	-1	0.86	1	NA	

Performance Implications

- Bottleneck - Similarity computation.
- Time complexity, highly time consuming with millions of users and items in the database.
  - Isolate the neighborhood generation and predication steps.
  - “off-line component” / “model” – similarity computation, done earlier & stored in memory.
  - “on-line component” – prediction generation process.

Challenges Of User-based CF Algorithms

- Sparsity – evaluation of large item sets, users purchases are under 1%.
- Difficult to make predictions based on nearest neighbor algorithms =>Accuracy of recommendation may be poor.
- Scalability - Nearest neighbor require computation that grows with both the number of users and the number of items.
- Poor relationship among like minded but sparse-rating users.
- Solution : usage of latent models to capture similarity between users & items in a reduced dimensional space.

Model based dimensionality reduction

Netflix Prize COMPLETED

What we were interested in:

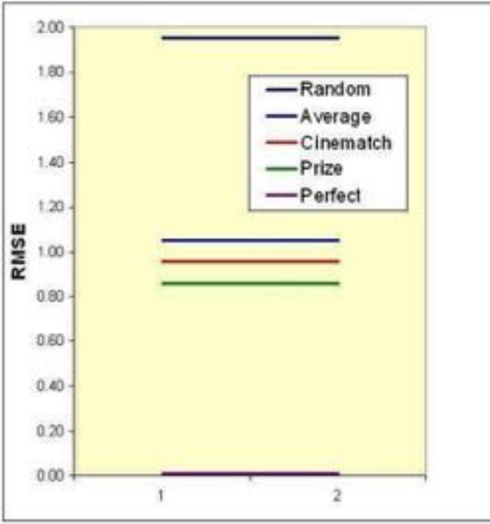
- High quality *recommendations*

Proxy question:

- Accuracy in predicted rating
- Improve by 10% = \$1million!

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

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



SVD/MF

$$X[n \times m] = U[n \times r]\ S\,[\,r \times r]\ (V[m \times r])^T$$

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

- **X**: *m x n* matrix (e.g., m users, n videos)
- **U**: *m x r* matrix (m users, r factors)
- **S**: *r x r* diagonal matrix (strength of each ‘factor’) (r: rank of the matrix)
- **V**: r x n matrix (n videos, r factor)

Recap: Singular Value Decomposition

- SVD is useful in data analysis
  $\rightarrow$ Noise removal, visualization, dimensionality reduction, ...
- Provides a mean to understand the hidden structure in the data

We may think of $\boldsymbol{A}_k$ and its factor matrices as a **low-rank model** of the data:

- Used to capture the important aspects of the data
  (cf. principal components)
- Ignores the rest
- Truncated SVD is best low-rank factorization of the data in terms of Frobenius norm
- Truncated SVD $\boldsymbol{A}_k = \boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{V}_k^T$ of $\boldsymbol{A}$ thus satisfies

$$\|\boldsymbol{A} - \boldsymbol{A}_k\|_F = \min_{\text{rank}(\boldsymbol{B})=k} \|\boldsymbol{A} - \boldsymbol{B}\|_F$$

SVD problems

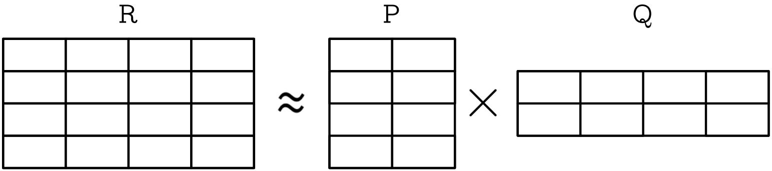
- complete input matrix: all entries available and considered
- large portion of missing values
- heuristics to pre-fill missing values
  - item's average rating
  - missing values as zeros

Matrix completion

- **Matrix completion** techniques avoid the necessity of pre-filling missing entries by reasoning only on the observed ratings.
- They can be seen as an estimate or an approximation of the SVD, computed using application specific optimization criteria.
- Such solutions are currently considered as the best single-model approach to collaborative filtering, as demonstrated, for instance, by the Netflix prize.

Matrix completion for collaborative filtering

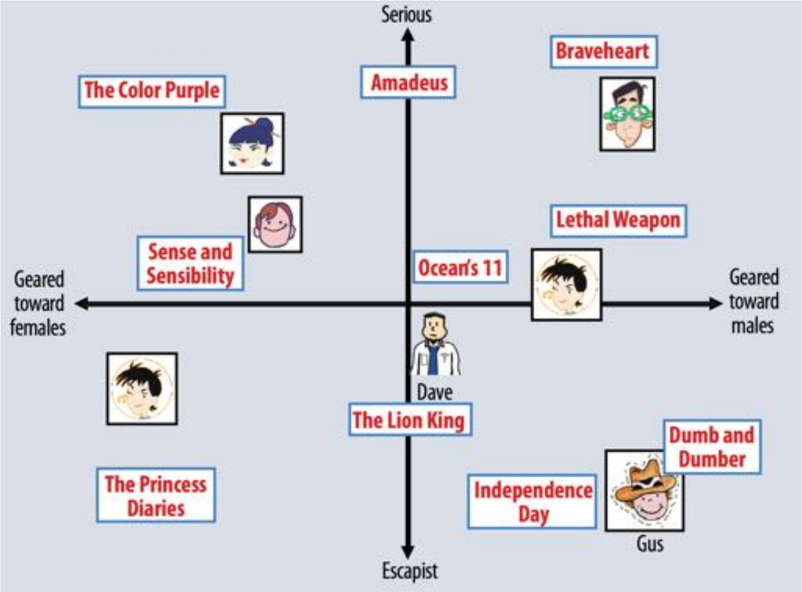
- the completion is driven by a factorization



- associate a latent factor vector with each user and each item
- missing entries are estimated through the dot product

$$r_{ij} \approx p_i q_j$$

Latent factor models (Koren et al., 2009)



Latent factor models

- Discover latent factors ($r = 1$)

	Avatar	**The Matrix**	**Up**
Anni		**4**	**2**
Bob	**3**	**2**	
Charlie	**5**		**3**

Latent factor models

- Discover latent factors ($r = 1$)

	Avatar (*2.24*)	**The Matrix** (*1.92*)	**Up** (*1.18*)
Anni (*1.98*)		**4**	**2**
Bob (*1.21*)	**3**	**2**	
Charlie (*2.30*)	**5**		**3**

Latent factor models

- Discover latent factors ($r = 1$)

	Avatar (*2.24*)	**The Matrix** (*1.92*)	**Up** (*1.18*)
Anni (*1.98*)		**4** (*3.8*)	**2** (*2.3*)
Bob (*1.21*)	**3** (*2.7*)	**2** (*2.3*)	
Charlie (*2.30*)	**5** (*5.2*)		**3** (*2.7*)

- Minimum loss

$$\min_{\boldsymbol{Q},\boldsymbol{P}} \sum_{(i,j)\in\Omega} (v_{ij} - [\boldsymbol{Q}^T \boldsymbol{P}]_{ij})^2$$

Latent factor models

- Discover latent factors ($r = 1$)

	Avatar (*2.24*)	**The Matrix** (*1.92*)	**Up** (*1.18*)
Anni (*1.98*)	? (*4.4*)	**4** (*3.8*)	**2** (*2.3*)
Bob (*1.21*)	**3** (*2.7*)	**2** (*2.3*)	? (*1.4*)
Charlie (*2.30*)	**5** (*5.2*)	? (*4.4*)	**3** (*2.7*)

- Minimum loss

$$\min_{\boldsymbol{Q},\boldsymbol{P}} \sum_{(i,j)\in\Omega} (v_{ij} - [\boldsymbol{Q}^T\boldsymbol{P}]_{ij})^2$$

Latent factor models

- Discover latent factors ($r = 1$)

	Avatar (*2.24*)	**The Matrix** (*1.92*)	**Up** (*1.18*)
Anni (*1.98*)	? (*4.4*)	**4** (*3.8*)	**2** (*2.3*)
Bob (*1.21*)	**3** (*2.7*)	**2** (*2.3*)	? (*1.4*)
Charlie (*2.30*)	**5** (*5.2*)	? (*4.4*)	**3** (*2.7*)

- Minimum loss

$$\min_{\boldsymbol{Q},\boldsymbol{P},\boldsymbol{u},\boldsymbol{m}} \sum_{(i,j)\in\Omega} (v_{ij} - \mu - u_i - m_j - [\boldsymbol{Q}^T\boldsymbol{P}]_{ij})^2$$

- Bias

Latent factor models

- Discover latent factors ($r = 1$)

	Avatar (*2.24*)	**The Matrix** (*1.92*)	**Up** (*1.18*)
Anni (*1.98*)	? (*4.4*)	**4** (*3.8*)	**2** (*2.3*)
Bob (*1.21*)	**3** (*2.7*)	**2** (*2.3*)	? (*1.4*)
Charlie (*2.30*)	**5** (*5.2*)	? (*4.4*)	**3** (*2.7*)

- Minimum loss

$$\min_{\boldsymbol{Q},\boldsymbol{P},\boldsymbol{u},\boldsymbol{m}} \sum_{(i,j)\in\Omega} (v_{ij} - \mu - u_i - m_j - [\boldsymbol{Q}^T\boldsymbol{P}]_{ij})^2$$
$$+ \lambda \left(\|\boldsymbol{Q}\| + \|\boldsymbol{P}\| + \|\boldsymbol{u}\| + \|\boldsymbol{m}\|\right)$$

- Bias, regularization

Latent factor models

- Discover latent factors ($r = 1$)

	Avatar (2.24)	**The Matrix** (1.92)	**Up** (1.18)
Anni (1.98)	? (4.4)	**4** (3.8)	**2** (2.3)
Bob (1.21)	**3** (2.7)	**2** (2.3)	? (1.4)
Charlie (2.30)	**5** (5.2)	? (4.4)	**3** (2.7)

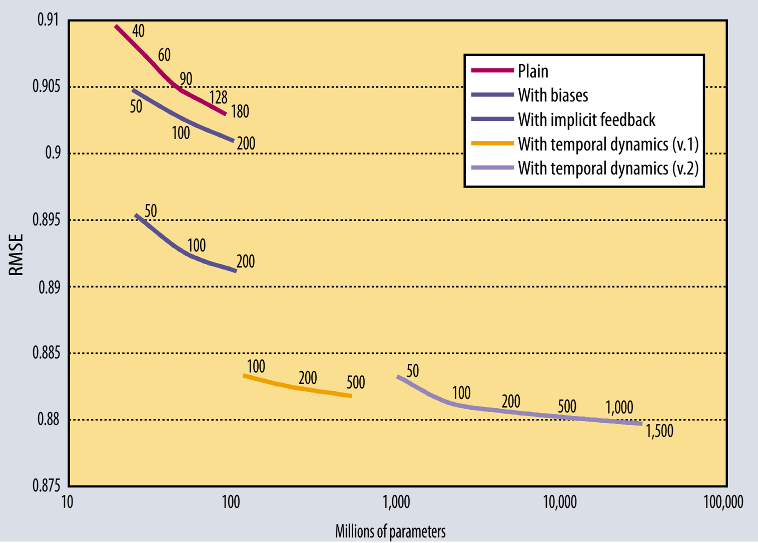
- Minimum loss

$$\min_{\boldsymbol{Q},\boldsymbol{P},\boldsymbol{u},\boldsymbol{m}} \sum_{(i,j,t)\in\Omega_t} \left(v_{ij} - \mu - u_i(t) - m_j(t) - [\boldsymbol{Q}^T(t)\boldsymbol{P}]_{ij}\right)^2 + \lambda\left(\|\boldsymbol{Q}(t)\| + \|\boldsymbol{P}\| + \|\boldsymbol{u}(t)\| + \|\boldsymbol{m}(t)\|\right)$$

- Bias, regularization, time, ...

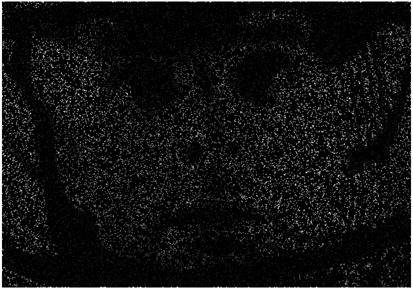
Example: Netflix prize data

Root mean square error of predictions

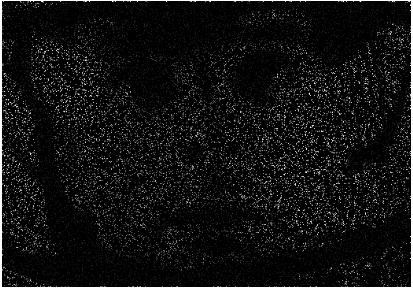


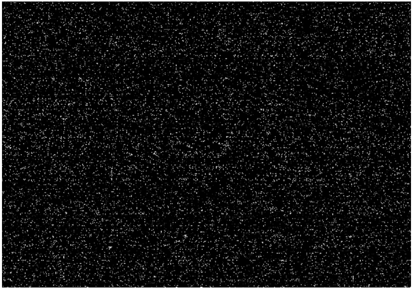
Another matrix

Matrix reconstruction (unregularized)

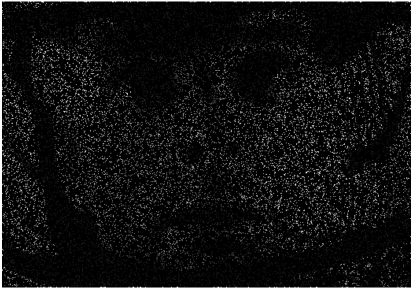


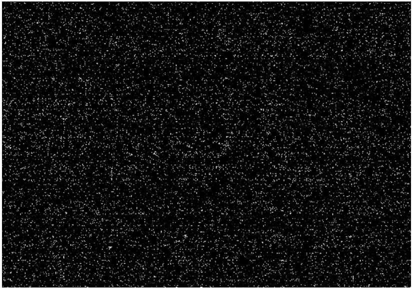
Matrix reconstruction (unregularized)

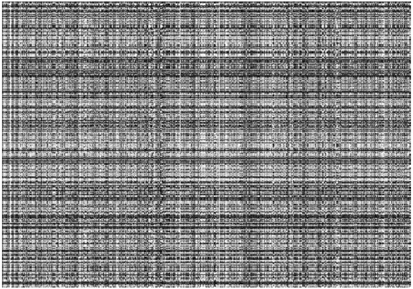




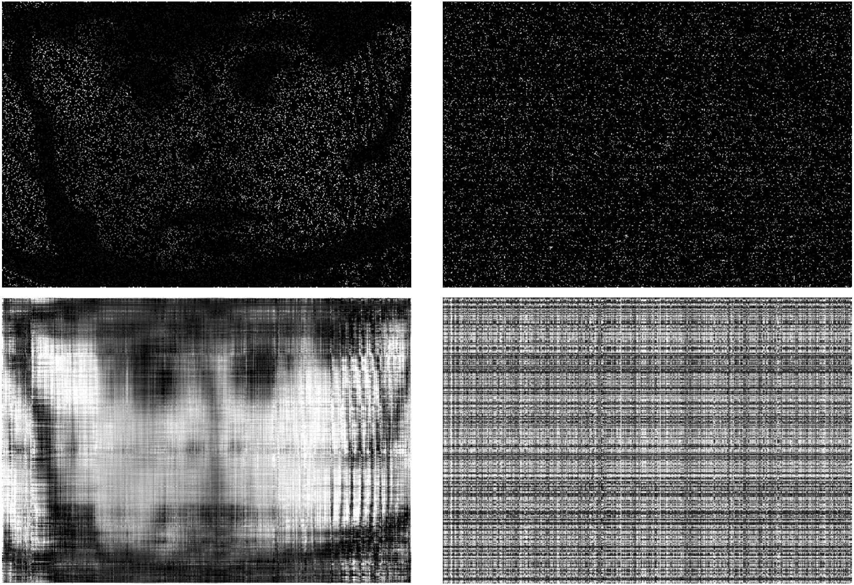
Matrix reconstruction (unregularized)





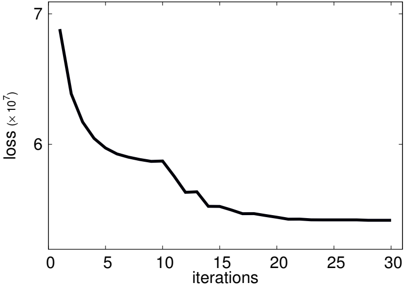


Matrix reconstruction (unregularized)



Stochastic gradient descent

- parameters $\Theta = \{P, Q\}$
- find minimum Θ^* of loss function L
- pick a starting point Θ^0
- iteratively update current estimations for Θ



$$\Theta_{n+1} \leftarrow \Theta_n - \eta \frac{\partial L}{\partial \Theta}$$

- learning rate η
- an update for each given training point

Stochastic updates

$$L_{ij}(P,Q) = (r_{ij} - p_i q_j)^2$$

- SGD to minimize the squared loss iteratively computes:

$$p_i \leftarrow p_i - \eta \frac{\partial L_{ij}(P,Q)}{\partial p_i} = p_i + \eta(\varepsilon_{ij} \cdot q_j)$$
$$q_j \leftarrow q_j - \eta \frac{\partial L_{ij}(P,Q)}{\partial q_j} = q_j + \eta(\varepsilon_{ij} \cdot p_i)$$

- where $\varepsilon_{ij} = r_{ij} - p_i q_j$

Suggested reading

- G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. Internet Computing, IEEE, 7(1):76–80, 2003.
- Y. Koren, R. Bell, and C. Volinsky. **Matrix factorization techniques for recommender systems.** Computer, 42(8):30–37, 2009.
- X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. Advances in Artificial Intelligence, 2009:4, 2009.
- F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. Springer, 2011.
- M. D. Ekstrand, J. T. Riedl, and J. A. Konstan. Collaborative filtering recommender systems. Foundations and Trends in Human-Computer Interaction, 4(2):81–173, 2011.
- J. A. Konstan and J. Riedl. Recommender systems: from algorithms to user experience. User Modeling and User-Adapted Interaction, 22(1-2):101–123, 2012.