

Recommender Systems

Wojtek Kowalczyk

Contents

- Recommender Systems
- Motivation: Netflix Challenge
- Recommender Systems: how do they work?
 - Content-based methods
 - Collaborative Filtering
 - Item-to-item
 - User-to-user
 - Matrix Factorization

Recommender Systems

- Systems that can estimate, for any user u and item i the degree of interest/satisfaction/importance of u in i .
- The estimation is based on historical rating/purchase/browsing/etc. data
- Examples:
 - Amazon.com: recommend books
 - Netflix: recommend movies
 - iTunes: recommend CD's
 - Google News: recommend news
 - bookings.com, trip-advisors,

Common Scenario: estimate $R(\text{User}, \text{Item})$

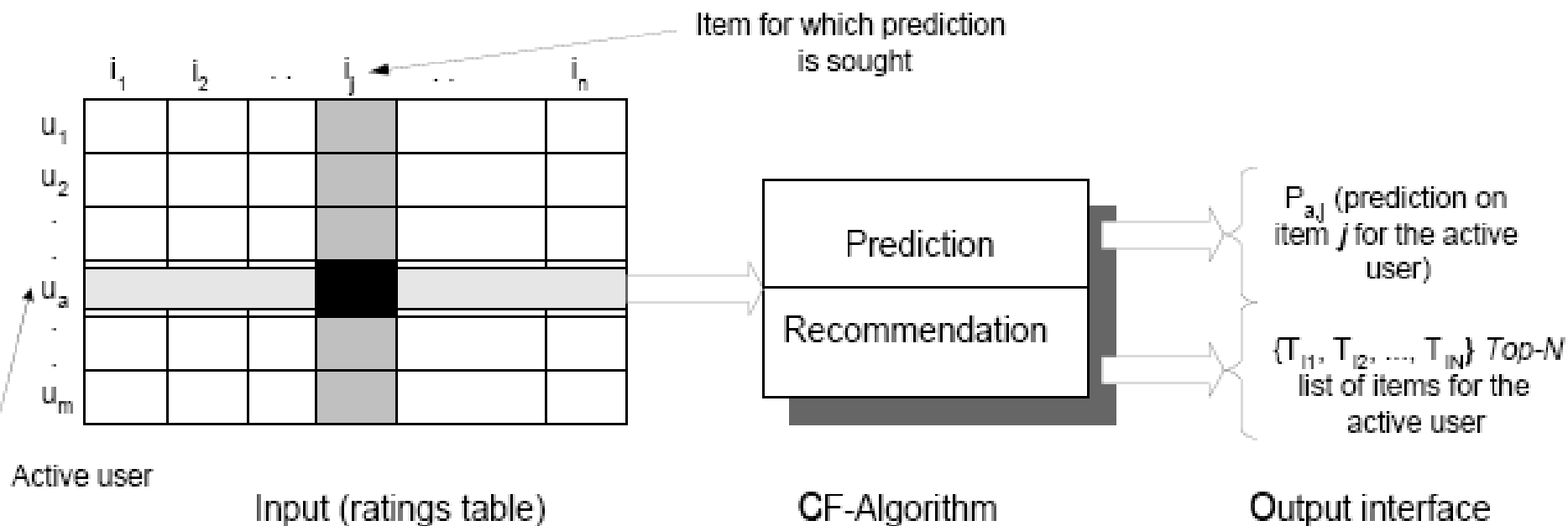


Figure 1: The Collaborative Filtering Process.

For a given “active user” estimate the rating (s)he would give to the given item, $R(\text{User}, \text{Item})$.

What is NETFLIX (in 2006)?

- Biggest on-line DVD movie rental company:
 - 5 million active customers
 - 80.000 movies to choose from
 - ship 2 million disks per day
- How people choose from 80.000 movies???
 - Get feedback: 3 million ratings per day
 - Analyze data and predict preferences:
1 billion predictions per day
 - The CINEMATCH system: state-of-the-art (in 2006)

Browse Selection

We have virtually every DVD published - from classics and new releases to TV and cable series. You'll be able to choose from over 80,000 DVD titles. In addition, you'll be able to select from over 2,000 instant viewing movies (such as the Matrix, Super Size Me, and Zoolander) to watch instantly on your PC.

**Start your
FREE TRIAL**

SEARCH FOR MOVIES

 GO

New Releases [\(see more\)](#)

[Ghost Rider](#)



[Breach](#)



[The Messengers](#)



[Music and Lyrics](#)



[Blood Diamond](#)



Action & Adventure [\(see more\)](#)

[Casino Royale](#)



[Pirates of the
Caribbean: Dead
Man's Chest](#)



[Superman Returns](#)



[Mission: Impossible
III](#)



[Inside Man](#)



Drama [\(see more\)](#)

[The Pursuit of
Happyness](#)



[Babel](#)



[The Departed](#)



[The Guardian](#)



[The Illusionist](#)



**80,000+ Titles
200+ Genres**

Browse Our Selection

[New Releases](#)
[Action & Adventure](#)
[Anime & Animation](#)
[Blu-ray](#)
[Children & Family](#)
[Classics](#)
[Comedy](#)
[Documentary](#)
[Drama](#)
[Faith & Spirituality](#)
[Foreign](#)
[Gay & Lesbian](#)
[HD DVD](#)
[Horror](#)
[Independent](#)
[Music & Musicals](#)
[Romance](#)
[Sci-Fi & Fantasy](#)
[Special Interest](#)
[Sports & Fitness](#)
[Television](#)
[Thrillers](#)

Show Interest

Get Recommendations

NETFLIX

Browse Home

Other Movies You Might Enjoy

Strangers on a Train
Special Edition

The Man Who Knew Too Much

Rope has been added to your Queue at position 115.
This movie is available now.
[Move To Top Of My Queue](#)

< [Continue Browsing](#) [Visit your Queue](#) >

Rear Window

Dial M for Murder

Notorious

North by Northwest

Also directed by Alfred Hitchcock:

The 39 Steps

Lifboat

The Lady Vanishes

Saboteur

Also In Classics:

NETFLIX

NETFLIX (2006) expected that in 2010-2012:

- ☐ 20 million subscribers
- ☐ receive 10 million ratings a day
- ☐ generate 5 billion predictions per day
- ☐ Movies distributed via Internet
- ☐ Accuracy of predictions and speed of the system is crucial for maintaining competitive advantage!

➔ Announce a \$1.000.000 CHALLENGE !!!

www.netflixprize.com

- ❑ **\$ 1.000.000 Grand Prize** for a data miner who will improve the accuracy of Netflix recommendation system **by 10% !!!**
- ❑ **Each year \$ 50.000 Progress Prize** for a best submission (**> 5%**) !!!
- ❑ Started in **October 2006**
- ❑ To be finished in or before **October 2011**

The Netflix Challenge

- ❑ **100.000.000** rating **records** collected over 1997-2005
- ❑ **rating record:**
 <customer_id, movie_id, date, rating>
- ❑ **500.000 customers**
- ❑ **18.000 movies**
- ❑ **rating** = an integer: **1, 2, 3, 4 or 5**
- ❑ Additionally, **3.000.000 test records:**
 <customer_id, movie_id, date, ? >

**GOAL: fill in “?’s” with numbers,
so the error is minimized!**

RMSE and percents

- **RMSE=Root Mean Squared Error**

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (predicted - true)^2}$$

- **Netflix baseline: RMSE=0.9514 (Cinematch)**
- **1% improvement: RMSE=0.9419**
- **5% improvement: RMSE=0.9038**
- **10% improvement: RMSE=0.8563**

www.netflixprize.com

- 45.000+ participants
- 37.000+ teams
- 36.000+ valid submissions from 4200 teams (!)
- about 50-100 submissions per day
- World-wide press coverage
- Netflix Forum, Conferences, Workshops, Special Issues of Journals, etc.

➔ **Immense boost of research on Recommender Systems**

The winner is ...

- ❑ **October, 2006: Start of the Competition**
- ❑ **After 7 days:** Netflix base-line reached !!!
(0% improvement)
- ❑ **After 42 days:** 5% improvement
- ❑ **January 2007:** 6% improvement
- ❑ **May 2007:** 7% improvement
- ❑ **October 2007:** 8.46% (\$50.000 Prize, AT&T Team)
- ❑ **October 2008:** 9.44% (AT&T + BigChaos)
- ❑ **July 2009:** 10.06% (*several teams of teams*)

Recommender Systems: how do they work?

- ☐ Content-based recommenders

- ☐ Memory-based
- ☐ Model-based

- ☐ Collaborative Filtering

- ☐ K-Nearest-Neighbours
 - ☐ Item-to-item
 - ☐ User-to-user
- ☐ Matrix Factorization (SVD)

Common Scenario: estimate $R(\text{User}, \text{Item})$

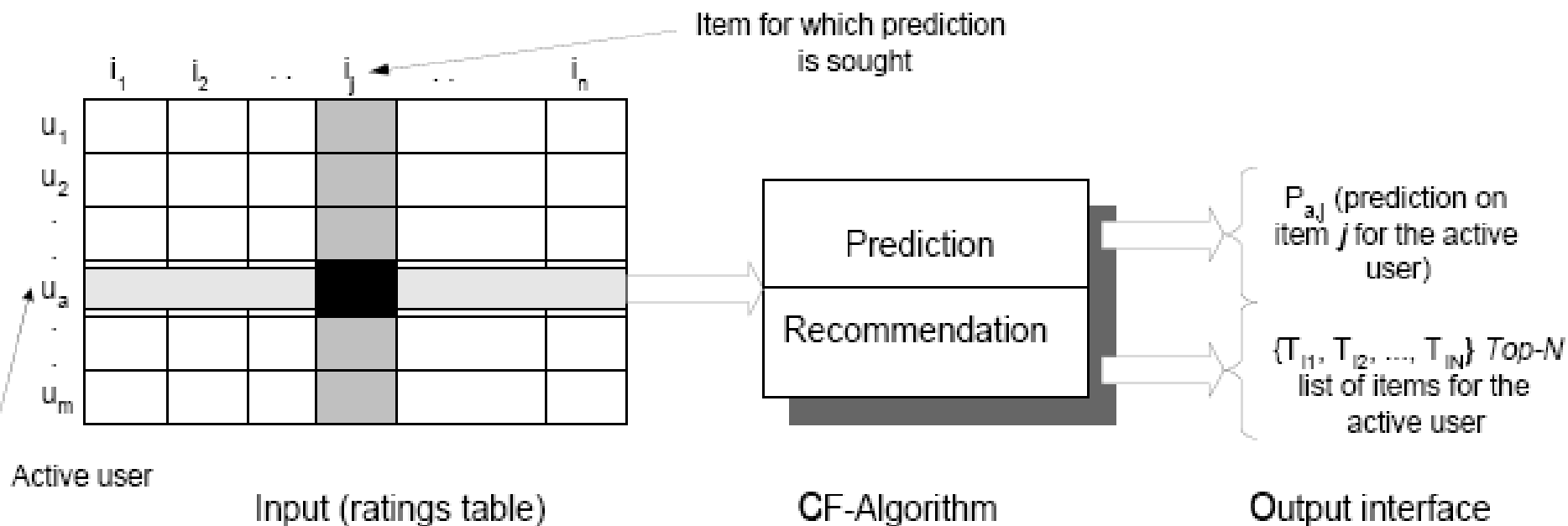


Figure 1: The Collaborative Filtering Process.

For a given “active user” estimate the rating (s)he would give to the given item, $R(\text{User}, \text{Item})$.

Rating/Utility Matrix

- A matrix: Users x Items partially filled with
 - ratings (how users rated items)
 - interest (how much browsing time)
 - purchase history (0 or 1)
 - ...
- Usually sparse: most entries missing
- Usually huge: millions of users x thousands of items

Naive Approaches

- $R_{\text{global}}(\text{User}, \text{Item}) = \text{mean}(\text{all ratings})$
- $R_{\text{item}}(\text{User}, \text{Item}) = \text{mean}(\text{all ratings for Item})$
- $R_{\text{user}}(\text{User}, \text{Item}) = \text{mean}(\text{all ratings for User})$
- $R_{\text{user-item}}(\text{User}, \text{Item}) = \alpha * R_{\text{user}}(\text{User}, \text{Item}) + \beta * R_{\text{item}}(\text{User}, \text{Item}) + \gamma$
(parameters α , β , γ estimated with Linear Regression)
- $R_{\text{user*item}}(\text{User}, \text{Item}) = \alpha * R_{\text{user}}(\text{User}, \text{Item}) + \beta_{\text{user}} * R_{\text{item}}(\text{User}, \text{Item})$
- ...

Naive Approaches

- ❑ Naive Approaches work surprisingly well
- ❑ Models very easy to calculate and to maintain
- ❑ Simple interpretation of the models:
 - "good movies"
 - "harsh users"
 - "crowd followers"
 - ...

Content-based approach / kNN /Memory-based

- ❑ Construct for every item its *profile* – a vector (or set) of features that characterize the item
- ❑ Construct for every user his/her *profile* – an “average” profile of the items he/she likes
- ❑ Recommend *items that are closest to the user profile*
- ❑ Closeness: Jaccard, cosine, Pearson, ... distance

Example Item Profiles

☐ Movies:

- Genre
- Director
- Stars
- Production Year

☐ Scientific Articles:

- Important words (high TF.IDF values)
- Newspaper/Journal title
- Author(s)
- Institution

☐ Music:

- Instruments
- ...

Content/Memory-based approach

□ Advantages:

- Item-profiles constructed up front (without historical data)
- Natural way of item clustering
- Intuitively simple

□ Disadvantages:

- Memory & cpu-time expensive ($O(N^2)$)
- Low accuracy

How do we measure accuracy? RMSE on the test data!

Model-based approach

- Once we have item profiles we can train, for every user, a classification (or regression) model which predicts rating from the item profile:
 - Model: a decision tree, neural network, SVM,
 - Input: item profile
 - Output: user ratings
- Expensive to build and maintain; low accuracy; what about new users (cold-start problem)?

Collaborative filtering

- Recommends items to the user based on what other **similar users** have liked

similar users: users that rate items in a similar way

- **How to find other user's preferences? From data!**
 - **Explicit methods** (ratings: what they like?)
 - **Implicit methods** (observations: what they buy?)

User-User recommendations: key idea

- Each user is represented by a *vector* that contains *ratings for each product he bought*.
- Users with “*similar*” *rating vectors* are *similar*.
- *How do we measure similarity of vectors? (later)*
- Which items should be recommended to X?
=> **find users similar to X and recommend items they liked that X hasn't rated yet.**

Step 1: Represent input data

Ranking matrix

	u_1	u_2	u_3	u_4	u_5	u_6
item ₁	5	1	5	4		3
item ₂	3	3	1	1	5	1
item ₃		1	?	2	1	4
item ₄	1	1	4	1	1	2
item ₅	3	2	5			3
item ₆	4	3			4	
item ₇		1	5	1	1	1

Step 2: Find nearest neighbours

Step 2.1. Calculate similarity between vectors


Cosine formula

$$\cos(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|_2 * \|\vec{b}\|_2}$$

Pearson correlation

$$\text{corr}_{ab} = \frac{\sum_i (r_{ai} - \bar{r}_a)(r_{bi} - \bar{r}_b)}{\sqrt{\sum_i (r_{ai} - \bar{r}_a)^2 \sum_i (r_{bi} - \bar{r}_b)^2}}$$

User-User Collaborative filtering



	u_1	u_2	u_3	u_4	u_5	u_6
item ₁	5	1	5	4		3
item ₂	3	3	1	1	5	1
item ₃		1	?	2	1	4
item ₄	1	1	4	1	1	2
item ₅	3	2	5			3
item ₆	4	3			4	
item ₇		1	5	1	1	1

Similarity measure

0.63

0.76

0.71

0.22

0.93



Step 2: Find nearest neighbours

Step 2.2. Define neighbourhood (size L)

- *sort and take first L*
- *aggregate neighbourhood* (e.g., take a weighted average rating)

Weighted sum

$$\frac{(1 \times 0.76 + 2 \times 0.71 + 4 \times 0.93)}{(0.76 + 0.71 + 0.93)} = 2.5$$

			u_3	u_4	u_5	u_6
			5	4		3
			1	1	5	1
item ₃		1	2.5	2	1	4
item ₄	1	1	4	1	1	2
item ₅	3	2	5			3
item ₆	4	3			4	
item ₇		1	5	1	1	1

Similarity measure

0.63

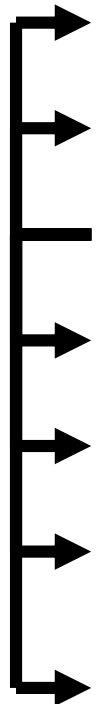
0.76

0.71

0.22

0.93

Item-Item Collaborative filtering



	u_1	u_2	u_3	u_4	u_5	u_6	
item ₁	5	1	5	4		3	0.62
item ₂	3	3	1	1	5	1	0.44
item ₃		1	?	2	1	4	
item ₄	1	1	4	1	1	2	0.9
item ₅	3	2	5			3	0.64
item ₆	4	3			4		0.23
item ₇		1	5	1	1	1	0.85

Similarity measure

Weighted sum

$$(4 \times 0.9 + 5 \times 0.85 + 5 \times 0.64)$$

$$(0.9 + 0.85 + 0.64)$$

$$= 4.6$$

			u_3	u_4	u_5	u_6	
			5	4		3	0.62
				1	5	1	0.44
	item ₃	1	4.6	2	1	4	
→	item ₄	1	4	1	1	2	0.9
→	item ₅	3	5			3	0.64
→	item ₆	4			4		0.23
→	item ₇	1	5	1	1	1	0.85

Similarity measure

Collaborative filtering: summary

□ **Three phases:**

- Represent data: rating/utility matrix
- Define neighbourhood: cosine, Pearson correlation, Jaccard, ...
- Make predictions or recommendations: weighting scheme

□ **Advantages:**

- can recommend items that are not linked to the user's earlier choices (useful for promotions)
- considers the opinions of a wide spectrum of users

□ **Limitations:**

- Sparsity of data
- Individual characteristics are not taken into account
- Tends to recommend popular items (convergence around few items)
- Computationally very expensive (time & memory)

Matrix Factorization (Simon Funk)

Main ideas:

- Look at the data from both perspectives (users and items) at the same time!
- Assume that each user and each movie can be represented by a fixed-length vectors of “features” that characterize them:
“user vector” and “item vector” (e.g., of length 20)
- Assume that each rating can be approximated by a product of corresponding vectors:
$$\text{rating} = \text{sum}(\text{“user vector”} \times \text{“item vector”})$$
- Find optimal values of all feature vectors that minimize SSE

How do we find the minimum of the error function?

- Error function (SSE) is a polynomial of many (50 million?) unknowns:

$$f(u_1, \dots, u_m, v_1, \dots, v_n) = \text{sum}((r_{ij} - u_i v_j')^2)$$

- Find a minimum of this function with help of any optimization method:
 - *“simple line search”*
 - *“gradient descent”*
 - *“Alternating Least Squares”*

Simple Line Search (Chapter 9 of the MMDS book)

1. Initialize all variables at random
2. Continue till convergence:
 - “freeze” all but one parameters at some values
 - then $f(\dots, \mathbf{x}, \dots)$ is a function of one variable
 - actually, $f(\dots, \mathbf{x}, \dots)$ is a quadratic function of \mathbf{x} !
 - find the optimal value for \mathbf{x} (*for quadratic functions it's easy!*)
 - freeze \mathbf{x} , ‘defreeze’ another (randomly chosen?) variable \mathbf{y} and repeat the trick ...

Gradient Descent Algorithm

How to find a minimum of a function $f(x,y)$?

1. Start with an arbitrary point (x_0, y_0)
2. Find a direction in which f is decreasing most rapidly:

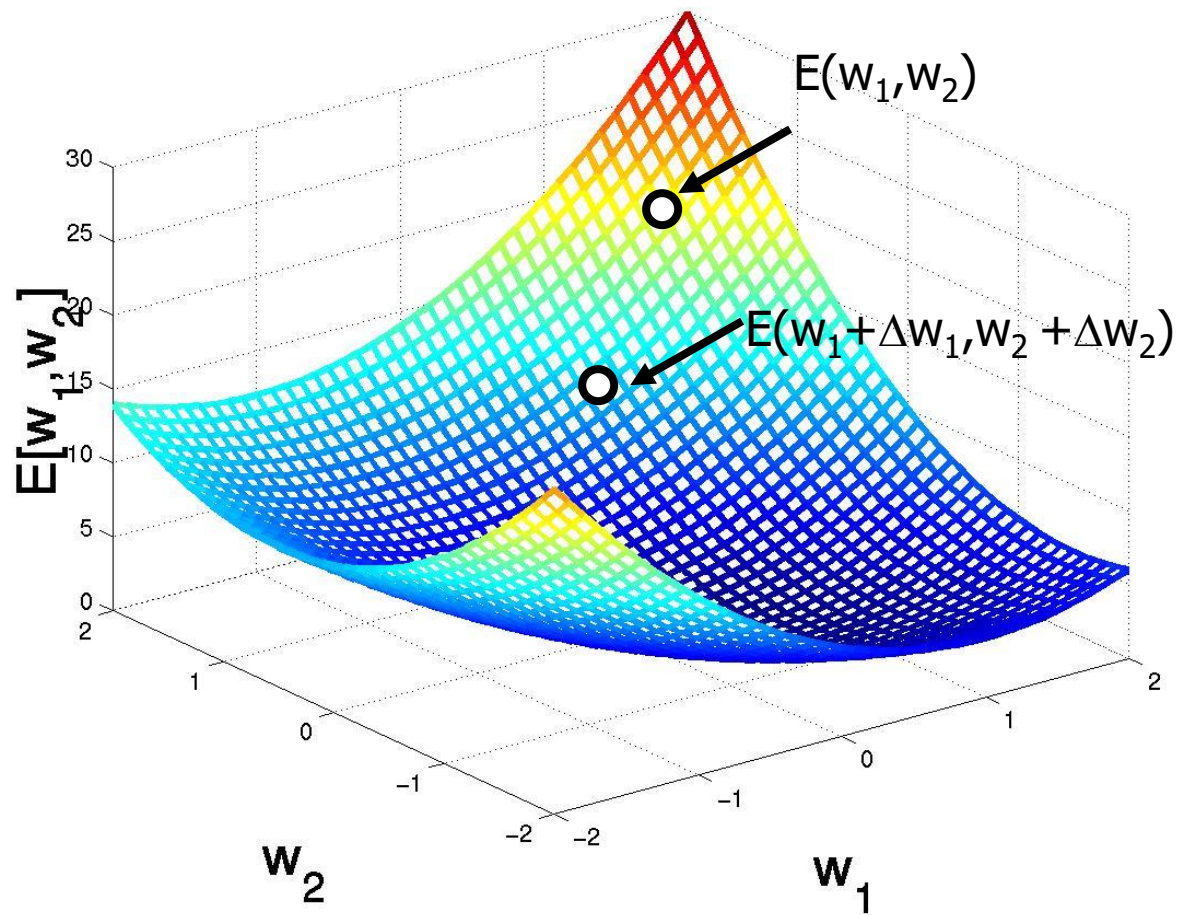
$$-\left[\frac{\partial f(x_0, y_0)}{\partial x}; \frac{\partial f(x_0, y_0)}{\partial y} \right]$$

3. Make a small step in this direction:

$$(x_0, y_0) = (x_0, y_0) - \eta \left[\frac{\partial f(x_0, y_0)}{\partial x}; \frac{\partial f(x_0, y_0)}{\partial y} \right]$$

4. Repeat the whole process

Gradient Descent



Matrix Factorization: Gradient Descent

1. Initialize all variables at random
2. Iterate over all records (user, item, rating):
 - *calculate the direction of steepest descent of function $f()$ (i.e., find a vector of partial derivatives)*
 - *make a step of size l_{rate} in this direction*till convergence or a stop criterion is met.

Calculation of partial derivatives of our error function (slide 34) leads to the following algorithm:

Iterate:

$$err := rating - u_{user} * v_{item}$$

$$u_{user} := u_{user} + l_{rate} * err * v_{item}$$

$$v_{item} := v_{item} + l_{rate} * err * u_{user}$$

until convergence

Matrix Factorization: Gradient Descent with Regularization (prevents overfitting)

[page 29, gravity-Tikk.pdf]

1. Initialize all variables at random
2. Iterate over all records (user, item, rating):

calculate the error:

$$\text{err} = \text{rating} - u_{\text{user}} * v_{\text{item}}$$

update parameters u_{user} and v_{item} :

$$u_{\text{user}} = u_{\text{user}} + \text{lr} * (\text{err} * v_{\text{item}} - \text{lambda} * u_{\text{user}})$$

$$v_{\text{item}} = v_{\text{item}} + \text{lr} * (\text{err} * u_{\text{user}} - \text{lambda} * v_{\text{item}})$$

until convergence.

Typical values: $\text{lr} = 0.001$; $\text{lambda} = 0.01$

Matrix Factorization: Alternating Least Squares

[ALS.pdf]

1. Initialize all variables at random

2. Repeat:

Freeze the user features and treat all item features as variables;
Solve the resulting Least Squares Problem.

Freeze the item features and unfreeze the user features;
Solve the resulting Least Squares Problem.

until done.

Simon Funk's trick: details

- ❑ Introduce about 50M unknowns (200MB):
 - 100 unknowns for each user: `userFeature[f][user]`
 - 100 unknowns for each movie: `movieFeature[f][movie]`
- ❑ Assume that predictions can be approximated by the dot products of these unknowns:

```
ratingPredicted[user][movie] =  
sum(userFeature[f][user]*movieFeature[f][movie])
```
- ❑ Define the error (err) as a function of all unknowns:
$$\text{sum}((\text{ratingPredicted}[\text{user}][\text{movie}] - \text{ratingTrue}[\text{user}][\text{movie}])^2)$$
- ❑ Find the minimum with gradient descent “delta-rule” :

```
userValue[user] += lrate * err * movieValue[movie];  
movieValue[movie] += lrate * err * userValue[user];
```

Simon Funk's trick

- ❑ Amazingly short code (essential part: 2 lines!)
- ❑ Can be run on a laptop with 1GB in a few hours
- ❑ Very good results (around 4-5% better than Netflix)
- ❑ Easy to extend: regularization, non-linear scoring function, initialization, etc.
- ❑ Vector representations can be used for **clustering, visualization, interpretation**, etc.

Interpretation of dimensions

Dimension 1 (f1)

Offbeat / Dark-Comedy	Mass-Market / 'Beniffer' Movies
Lost in Translation	Pearl Harbor
The Royal Tenenbaums	Armageddon
Dogville	The Wedding Planner
Eternal Sunshine of the Spotless Mind	Coyote Ugly
Punch-Drunk Love	Miss Congeniality

Dimension 2 (f2)

What a 10 year old boy would watch	What a liberal woman would watch
Dragon Ball Z: Vol. 17: Super Saiyan	Fahrenheit 9/11
Battle Athletes Victory: Vol. 4: Spaceward Ho!	The Hours
Battle Athletes Victory: Vol. 5: No Looking Back	Going Upriver: The Long War of John Kerry
Battle Athletes Victory: Vol. 7: The Last Dance	Sex and the City: Season 2
Battle Athletes Victory: Vol. 2: Doubt and Conflic	Bowling for Columbine

Why is it called “Matrix Factorization”?

Singular Value Decomposition

For an $m \times n$ matrix \mathbf{A} of rank r there exists a factorization (Singular Value Decomposition = **SVD**) as follows:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

$m \times m$ $m \times n$ V is $n \times n$

The columns of \mathbf{U} are orthogonal eigenvectors of $\mathbf{A}\mathbf{A}^T$.

The columns of \mathbf{V} are orthogonal eigenvectors of $\mathbf{A}^T\mathbf{A}$.

Eigenvalues $\lambda_1 \dots \lambda_r$ of $\mathbf{A}\mathbf{A}^T$ are the eigenvalues of $\mathbf{A}^T\mathbf{A}$.

$$\sigma_i = \sqrt{\lambda_i}$$

$$\mathbf{\Sigma} = \text{diag}(\sigma_1 \dots \sigma_r) \leftarrow \text{Singular values.}$$

SVD as dimensionality reduction

$$\underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}}_A = \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_U \underbrace{\begin{bmatrix} \bullet & & & \\ & \bullet & & \\ & & \bullet & \\ & & & \text{yellow box} \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}}_{V^T}$$

$$\underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}}_{A_k} = \underbrace{\begin{bmatrix} * & * & \text{blue box} \\ * & * & \text{blue box} \\ * & * & \text{blue box} \end{bmatrix}}_U \underbrace{\begin{bmatrix} \bullet & & \text{blue box} & \text{yellow box} \\ & \bullet & \text{blue box} & \text{yellow box} \\ & & \bullet & \text{yellow box} \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ \text{blue box} & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}}_{V^T}$$

Low(er) rank approximations might have better generalization power!

SVD and data reduction

- ❑ User-Movie matrix A has $m=500.000$, $n=18.000$ (and rank close to 18.000)
- ❑ Construct an approximation A_{100} with rank 100 (!?)
- ❑ A_{100} approximates A as good as possible (???)
- ❑ In practice it means that we represent every movie and every user by a vector of 100 numbers
- ❑ Rating: multiplying 2 vectors !

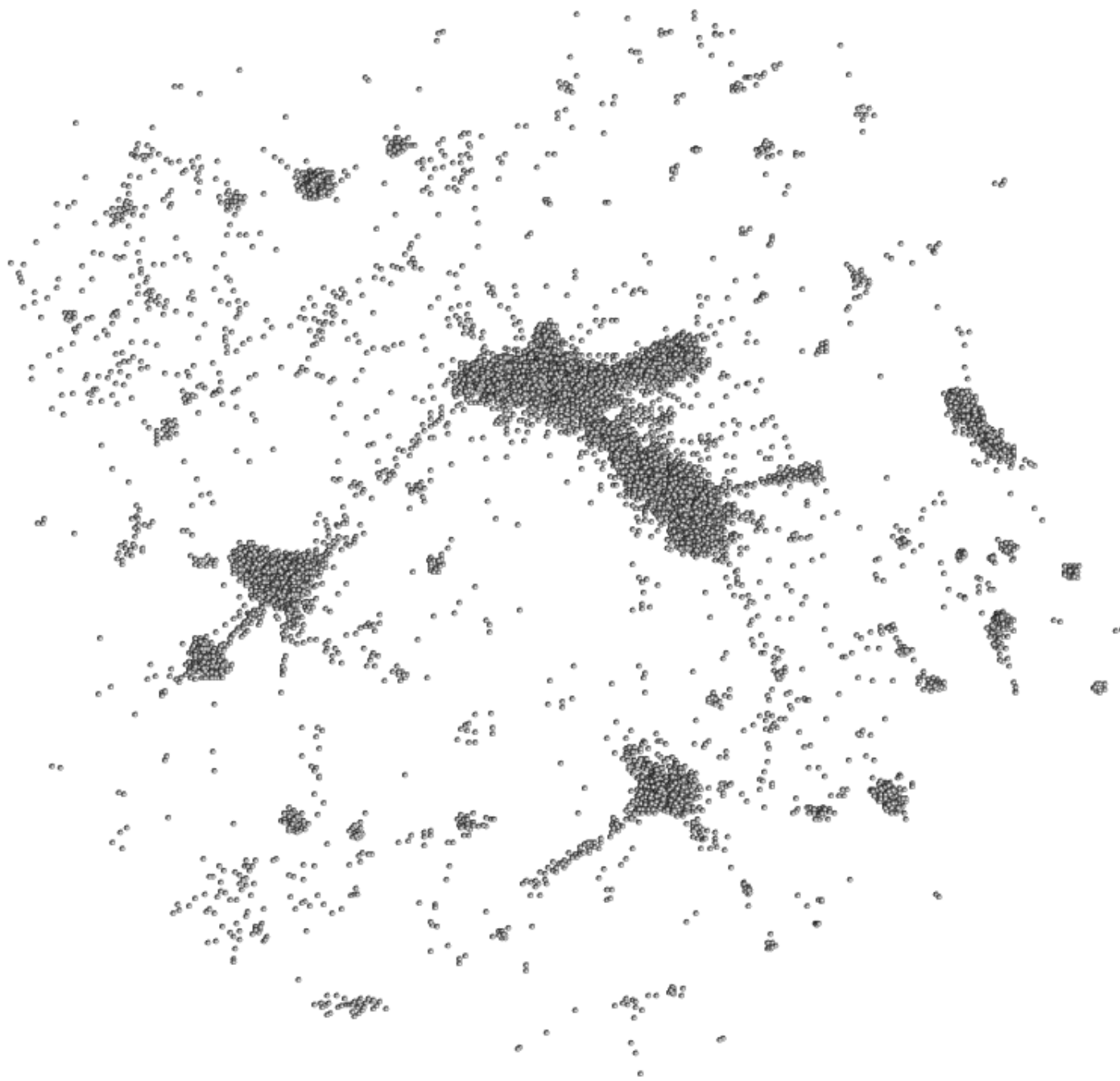
In practice, the SVD algorithm wouldn't work!

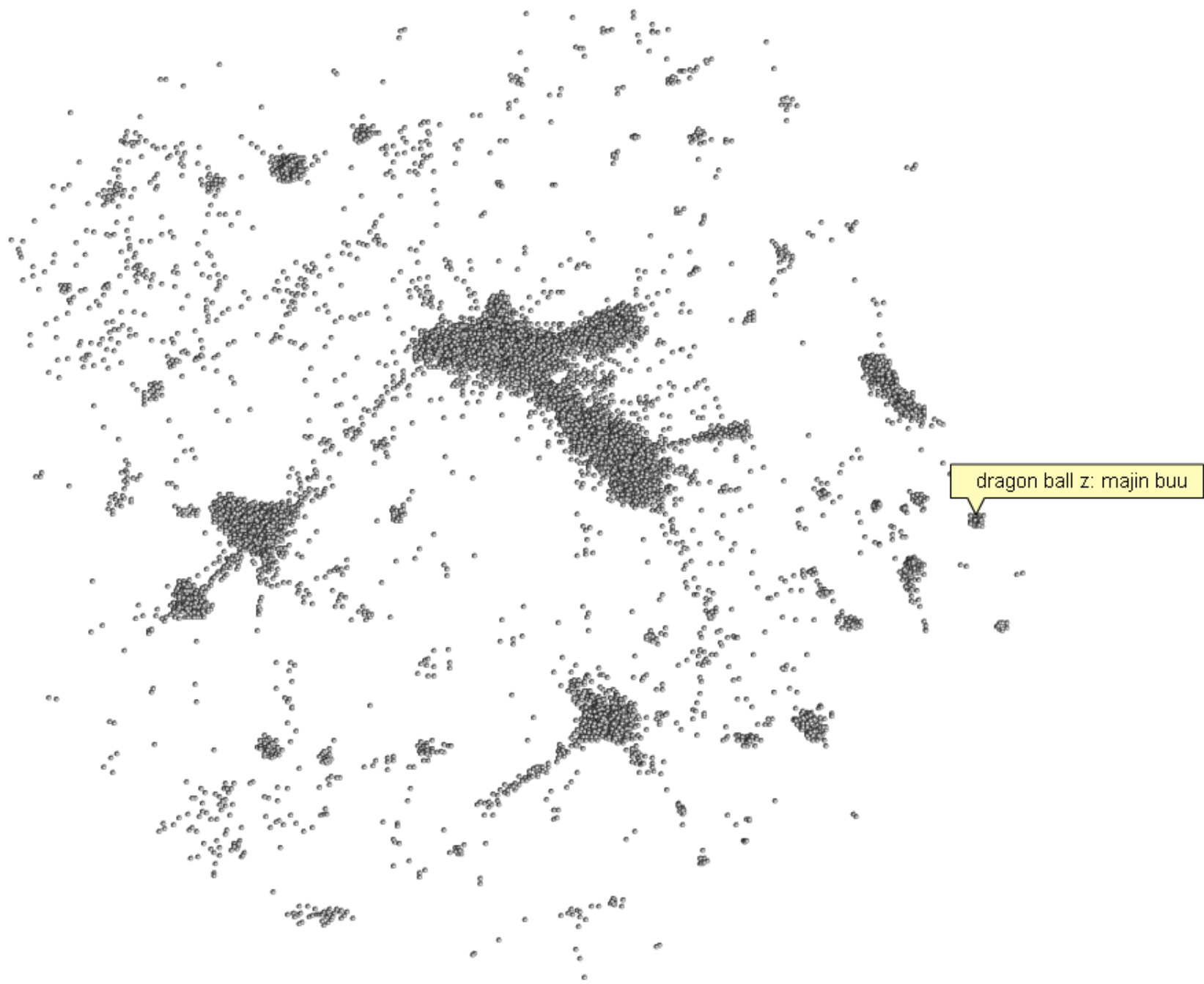
- ❑ The 500.000x18.000 matrix A is huge!
(it has about 8.5 billion entries = a few Terabytes of memory needed)
- ❑ The matrix A is sparse – only 1% of entries are non-zeros
(it's good - we can save memory), but: “missing” are assumed to be 0's!
- ❑ SVD focuses on minimizing the “global” error (Frobenius Norm),
where zero's entries also play a role: not good - we don't care about 0's!
- ❑ Simon Funk: ignore all “missing” values and directly optimize
the Mean Squared Error(MSE) over all non-missing ratings!
- ❑ Simon Funk's blog contains more details:
<http://sifter.org/~simon/journal/20061211.html>

Dimensionality Reduction and Visualization

Main idea:

- movies are points in 100 dimensional space
- project these points into two dimensional space preserving as much information as possible
- there are many algorithms for that:
 - Principal Components Analysis
 - Multi-Dimensional Scaling
 - Self-Organizing Maps (Kohonen networks)
- Visualize the results !





Boosting accuracy: blending models!

Main idea:

- build many models with help of several techniques, on various sample of data and BLEND the predictions
- BLENDING: applying linear regression to find optimal coefficients of a linear combination of models
- winning submissions used combinations of 100-200 models, build with 3-5 "main algorithms"
- combining models of different types (e.g., SVD, kNN, RBM) is very beneficial!

Links ...

Netflix competition site:

<http://www.netflixprize.com/leaderboard>

Simon Funk approach:

<http://sifter.org/~simon/journal/20061211.html>

Netflix Workshop KDD2007:

<http://www.cs.uic.edu/~liub/KDD-cup-2007/proceedings.html>

Winner 2007:

<http://www.netflixprize.com/community/viewtopic.php?id=799>

Winner 2008:

<http://www.netflixprize.com/community/viewtopic.php?id=1193>



Assignment 1

- ❑ MovieLens data
(<http://www.grouplens.org/node/73>)
 - 1M (or 10M) ratings
 - About 6000 users, 4000 movies
- ❑ Additionally:
 - User data: <UserID::Gender::Age::Occupation::Zip-code>
 - Movie data: <MovieID::Title (Year)::Genres>
- ❑ Well-documented/researched data set!

The Task

- ❑ Implement from scratch (in Python) the key algorithms discussed during the course:
 - naïve recommenders (4 variants)
 - Gradient descent with regularization
 - *(optionally-if you want a bonus) Implement the ALS algorithm*
- ❑ Estimate their accuracy (RMSE and MAE) with 5-fold cross validation
- ❑ Document your findings, providing performance analysis of the implemented algorithms (accuracy, speed, scalability, ...)

Key objectives

- ❑ Understand in depth the algorithms and practical issues related to their deployment on real data
- ❑ Master your Python/Numpy skills (e.g., sparse matrices, linear algebra)
- ❑ Systematic evaluation of accuracy of various algorithms (5-fold cross validation, standard errors)
- ❑ Gaining insights into computational costs
- ❑ Experience with reporting results of experiments

Extra help

- ☐ Linear Regression (how to find α, β, γ ?)
- ☐ Alternating Least Squares
- ☐ ALS for Recommender System: parallel version

Linear Regression

$\mathbf{x}_1, \dots, \mathbf{x}_k$ are input vectors in n-dimensional space,
 d_1, \dots, d_k are desired/target outputs

Assume a linear regression model: *y is a linear combination of x's*:

$$y = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n$$

and let its predictions on $\mathbf{x}_1, \dots, \mathbf{x}_k$ are y_1, \dots, y_k

Find values for w_0, \dots, w_n for which the squared error:

$$error = (d_1 - y_1)^2 + (d_2 - y_2)^2 + \dots + (d_k - y_k)^2$$

is minimized.

Finding Linear Regression Model: Least Squares Method

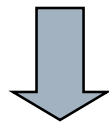
- Error function is a quadratic function of $n+1$ variables:

$$y = w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n$$

$$\text{error} = (d_1 - y_1)^2 + (d_2 - y_2)^2 + \dots + (d_k - y_k)^2$$

$$\text{error}(\mathbf{w}) = (d_1 - y_1(\mathbf{w}))^2 + (d_2 - y_2(\mathbf{w}))^2 + \dots + (d_k - y_k(\mathbf{w}))^2$$

- all partial derivatives = 0 \Rightarrow error is minimal
- partial derivatives are linear functions of **\mathbf{w}**



finding optimal **\mathbf{w}** reduces to the problem of solving
a system of $(n+1)$ linear equations with $(n+1)$ variables
(no matter how big the training set) [$n=3 \Rightarrow 4$ linear eqs.]

Matrix Factorization: Alternating Least Squares (optional)

1. Initialize all variables at random

2. Repeat:

Freeze the user features and treat all item features as variables;
Solve the resulting Least Squares Problem.

Freeze the item features and unfreeze the user features;
Solve the resulting Least Squares Problem.

until done.

In Matlab solving Least Squares problems is easy:

If A is an M -by- N matrix and B is a column vector with M components, then $X = A \backslash B$ is the solution in the least squares sense to the under- or overdetermined system of equations $A * X = B$.

Alternating Least Squares: some details

Large-scale Parallel Collaborative Filtering for the Netflix Prize

Yunhong Zhou, Dennis Wilkinson, Robert Schreiber and Rong Pan

$$f(U, M) = \sum_{(i,j) \in I} (r_{ij} - \mathbf{u}_i^T \mathbf{m}_j)^2 + \lambda \left(\sum_i n_{u_i} \|\mathbf{u}_i\|^2 + \sum_j n_{m_j} \|\mathbf{m}_j\|^2 \right)$$

Step 1 Initialize matrix M by assigning the average rating for that movie as the first row, and small random numbers for the remaining entries.

Step 2 Fix M , Solve U by minimizing the objective function (the sum of squared errors);

Step 3 Fix U , solve M by minimizing the objective function similarly;

Step 4 Repeat Steps 2 and 3 until a stopping criterion is satisfied.

Alternating Least Squares: some details

$$\Rightarrow \mathbf{u}_i = A_i^{-1}V_i, \quad \forall i$$

where $A_i = M_{I_i}M_{I_i}^T + \lambda n_{u_i}E$, $V_i = M_{I_i} R^T(i, I_i)$, and E is the $n_f \times n_f$ identity matrix. M_{I_i} denotes the sub-matrix of M where columns $j \in I_i$ are selected, and $R(i, I_i)$ is the row vector where columns $j \in I_i$ of the i -th row of R is taken.

Similarly, when M is updated, we can compute individual m_j 's via a regularized linear least squares solution, using the feature vectors of users who rated movie j , and their ratings of it, as follows:

$$\mathbf{m}_j = A_j^{-1}V_j, \quad \forall j,$$

where $A_j = U_{I_j}U_{I_j}^T + \lambda n_{m_j}E$ and $V_j = U_{I_j}R(I_j, j)$. U_{I_j} denotes the sub-matrix of U where columns $i \in I_j$ are selected, and $R(I_j, j)$ is the column vector where rows $i \in I_j$ of the j -th column of R is taken.

The Circle: a case study

- ☐ What are recommender systems and what can you do with them?
- ☐ Are they really so great? Two cases:

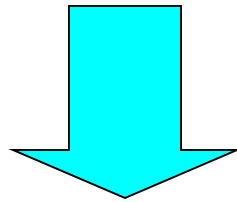


- ☐ Conclusions and Recommendations

Recommender systems:

□ **Input:** 10.000.000 purchase records:

- 500.000 customers
- 50.000 products
- purchase record: <customer, product>



Recommender Algorithm

- **Output:** a matrix 500.000 x 50.000 numbers:
 - for every <customer, product> combination
the *probability* that the customer will buy the product

- "best chance of purchase"
or
"best monetary value"

Customer	Product	Prob
1	345	0.32
1	2364	0.15
1	743	0.07
2	354	0.12
2	6443	0.03
2	12334	0.01
3	432	0.44
3	74321	0.21
3	864	0.18
4	2345	0.16
4	532	0.04
4	2656	0.01
500000	345	0.31
500000	43378	0.3
500000	2350	0.26

Problem: Sell 5 Thrillers via Newsletter

☐ Select clients that will buy any of the 5 thrillers:

- TOT HET VOORBIJ IS, FRENCH, NICCI
- ONAANTASTBAAR, SLAUGHTER, KARIN
- GENIAAL GEHEIM, BALDACCI, DAVID
- DE ELFDE PLAAG, SMITH, WILBUR
- DOOD DOOR SCHULD, BEISHUIZEN, TINEKE

☐ Three selections of about 20.000 clients:

- Random
- ECI
- Recommender System

☐ Evaluation:

- two weeks after mailing, ECI analyzes response

Results

Click Through Rates					
Group	Delivered	Opened	Opened %	Clicks	Clicks %
Random Selection	16,744	5,884	35.14%	1,752	29.8%
ECI Selection	16,890	6,144	36.38%	1,829	29.8%
Recommender	17,154	7,024	40.95%	2,346	33.4%
Sales Internet					
Group	Mailed	Responded	Response	Turnover	T per M
Random Selection	8,124	1,221	15.0%	100	12.31
ECI Selection	4,969	905	18.2%	120	24.15
Recommender	3,166	604	19.1%	132	41.69
Sales from Newsletter					
Group	Mailed	Responded	Response	Turnover	T per M
Random Selection	8,124	42	0.5%	100	12.31
ECI Selection	4,969	192	3.9%	103	20.73
Recommender	3,166	96	3.0%	112	35.38

Conclusion: The Big Winner!

- ❑ Recommender System gives better results:
 - ❑ Click Through Rate:
12% (relative), 3.6% (absolute) better than ECI
 - ❑ Return per Mail:
3.4 times better than random
1.7 times better than ECI
- ❑ No background knowledge, no thinking, just data
- ❑ Fast (real-time predictions)
- ❑ Unlimited Scalability (billions of records)

True power still unleashed: 1-2-1 selections !!!

Recommenders and Psychology

Is Seeing Believing? How Recommender Interfaces Affect Users' Opinions

Dan Cosley, Shyong K. Lam, Istvan Albert, Joseph A. Konstan, John Riedl
Department of Computer Science and Engineering
University of Minnesota, Minnesota, MN 55455 USA
+1 612 624-8372
{cosley, lam, ialbert, konstan, riedl}@cs.umn.edu

Psychological experiments with 536 users (2003)
Several setups, questions, results ...

Questions and Answers

Q: Can the system make a user rate
a “bad” movie “good”?

A: YES (!!!)

Q: Do users notice when predictions
are manipulated?

A: YES: they don't like the system!

=> users can be really manipulated by recommenders

A BIG QUESTION:

HOW TO PROTECT RECOMMENDER SYSTEMS AGAINST
INTRUDERS?

New research area: fighting for trust!

Some titles from Bamshad Mobasher's group:

- ❑ Towards Trustworthy Recommender Systems: An Analysis of Attack Models and Algorithm Robustness
- ❑ Robustness of Collaborative Recommendation Based On Association Rule Mining
- ❑ Defending Recommender Systems: Detection of Profile Injection Attacks
- ❑ Effectiveness of Crawling Attacks Against Web-based Recommender Systems
- ❑ Attacks and Remedies in Collaborative Recommendation

Recommenders and Business

Recommender Systems and their Impact on Sales Diversity

Daniel Fleder and Kartik Hosanagar

The Wharton School, University of Pennsylvania

Some believe recommenders help consumers discover new products and thus *increase sales diversity*.

Others believe recommenders only *reinforce the popularity of already popular products*.

What is the case???

Recommenders and Business

Approach:

a mathematical model of market, users, recommenders;
numerous simulations for various scenarios

(Preliminary) Results:

1. Common recommenders lead to
reduction in average sales diversity!!!
2. The same recommender *can increase or decrease diversity*,
depending on the setup.
3. *Better understanding* of how choices affect the outcome.

Recommenders and Business

Empirical Analysis of the Business Value of Recommender Systems

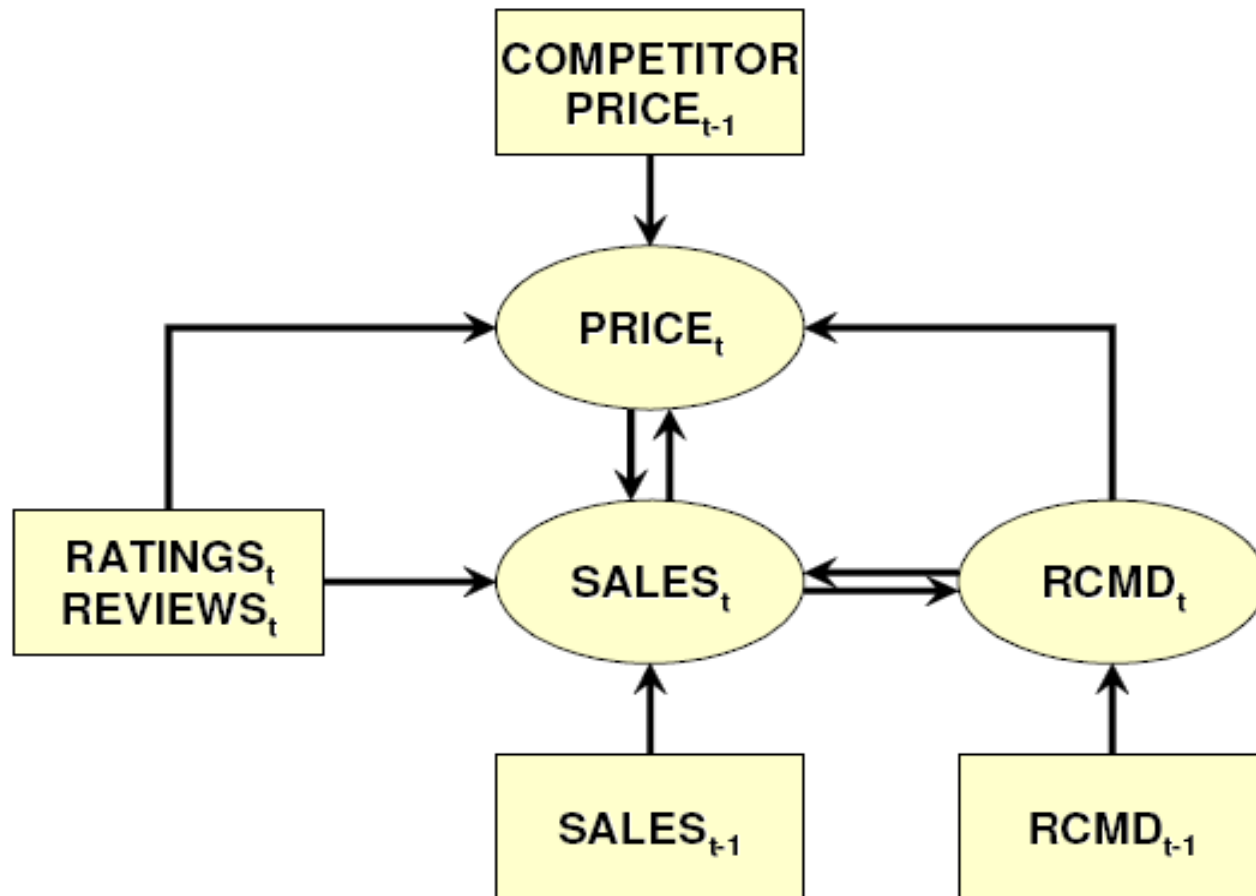
Garfinkel, Gopal, Pathak, Venkatesan, Yin

University of Connecticut

The purpose of the current work is to develop a robust empirical method to evaluate the relationship between recommendations and sales.

Model validated on data from Amazon.com and BarnesAndNobles.com (90% market share)

The Empirical Model



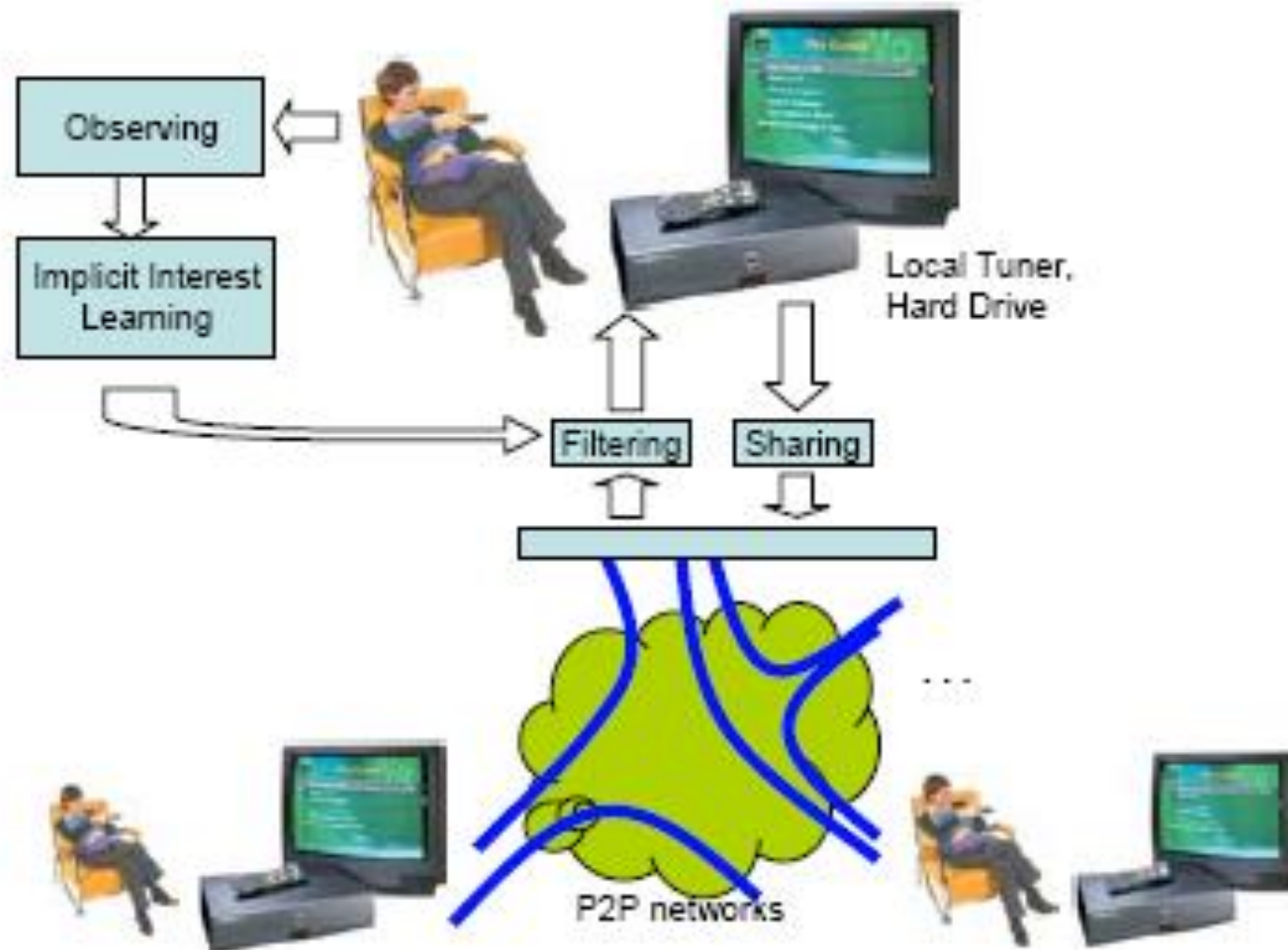
Results

- A strong impact on sales of:
 - paired recommendations
(clients who bought A also bought B)
 - reviews

- adding a paired recommendation =>
increase of sales rank by 3%

- adding an extra review =>
increase sales rank by 1%

The not so far Future: Recommenders and Social Networks



Recommenders and Social Networks

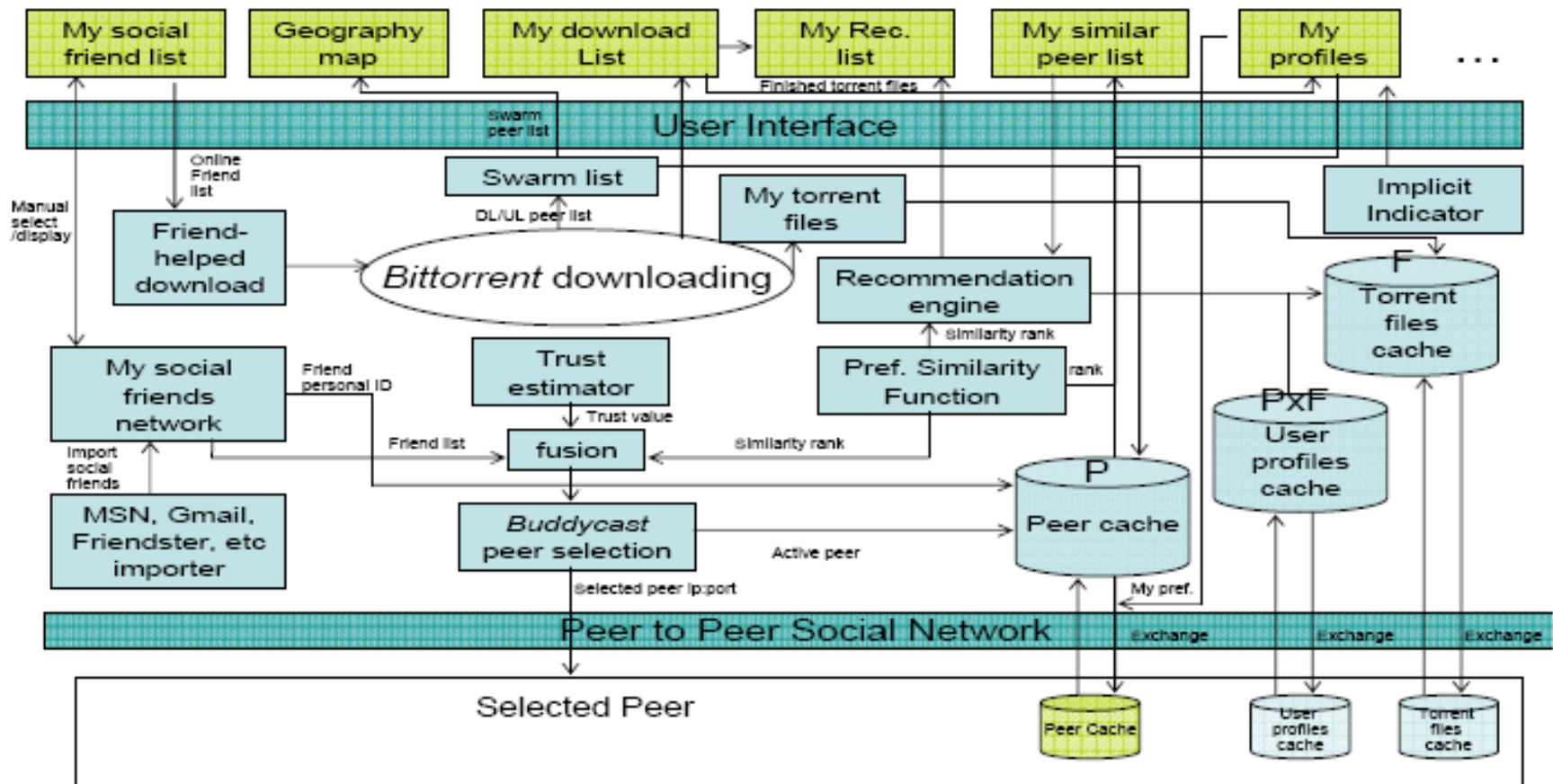


Fig. 2. The system architecture of Tribler.

Conclusions

- ❑ Efficient algorithms for recommender systems are very important, but not sufficient
- ❑ Design of interfaces, interactions, objectives is of key importance
- ❑ The impact on business is more complicated than one could expect
- ❑ Distributed recommender systems for peer-to-peer social-networks might revolutionize the business!

Theory in practice ...

“In theory,
there should be no difference between
theory and practice.

In practice, however,
there is always a difference.”

Jan L. A. van de Snepscheut/Yogi Berra