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/purchese/browsing/etc. data

- Examples:
  - Amazon.com: recommend books
  - Netflix: recommend movies
  - iTunes: recommend CD's
  - Google News: recommend news
  - booking.com, trip-advisor, ….

Common Scenario: estimate R(User, Item)

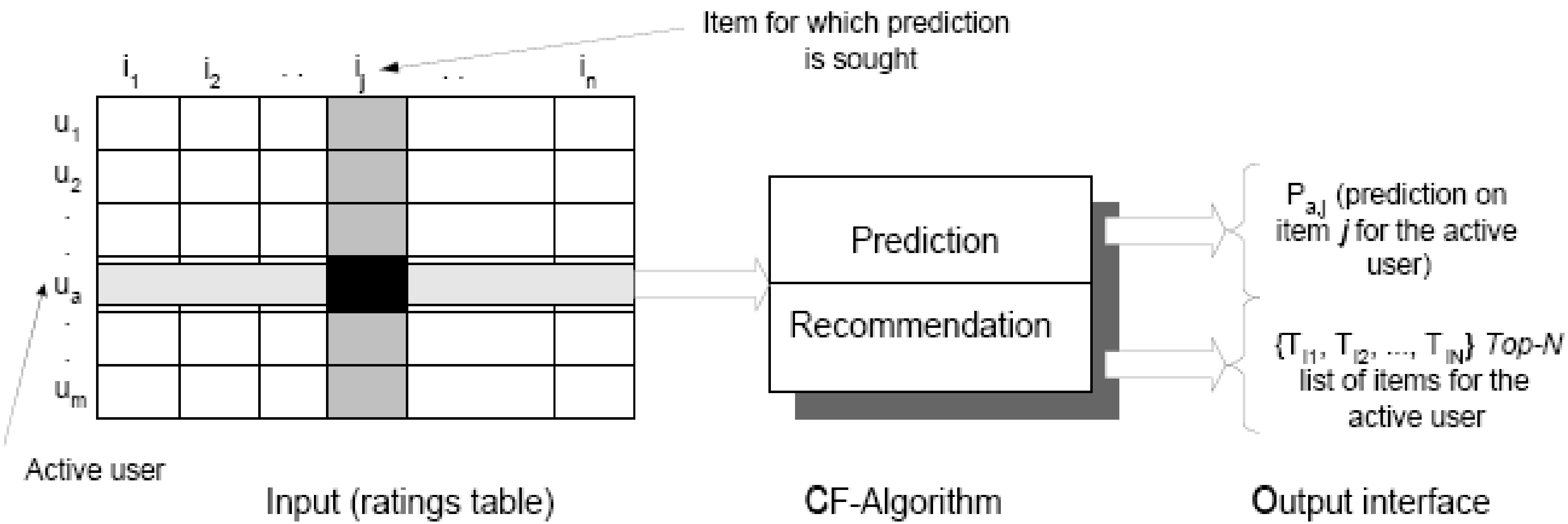


Figure 1: The Collaborative Filtering Process.

For a given "active user" estimate the rating (s)he would give to the given item, R(User, 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.

New Releases (see more)

Ghost Rider

Breach

The Messengers

Music and Lyrics

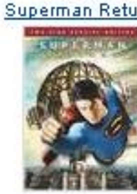
Blood Diamond

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

SEARCH FOR MOVIES

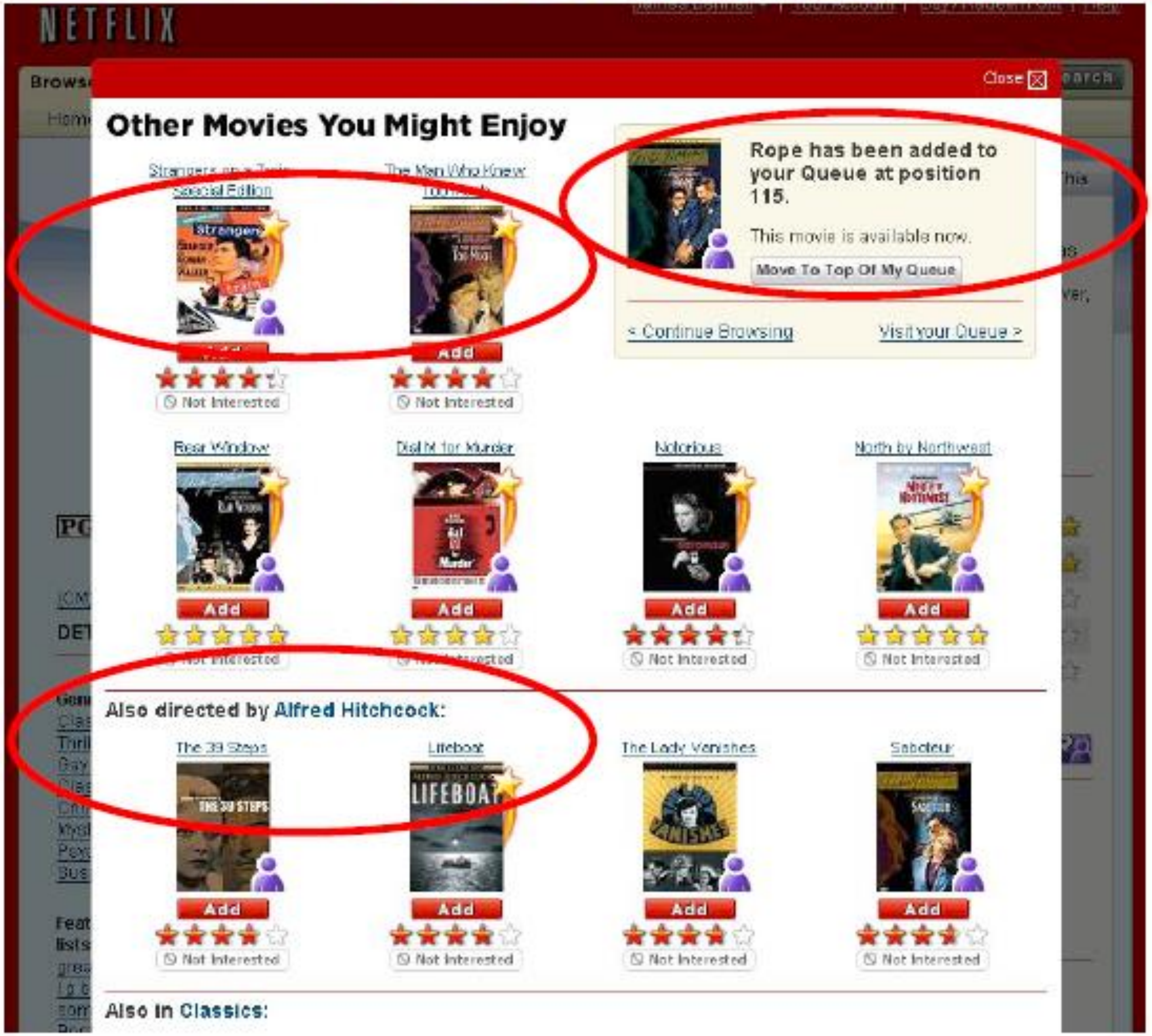
GO

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

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(User, Item)

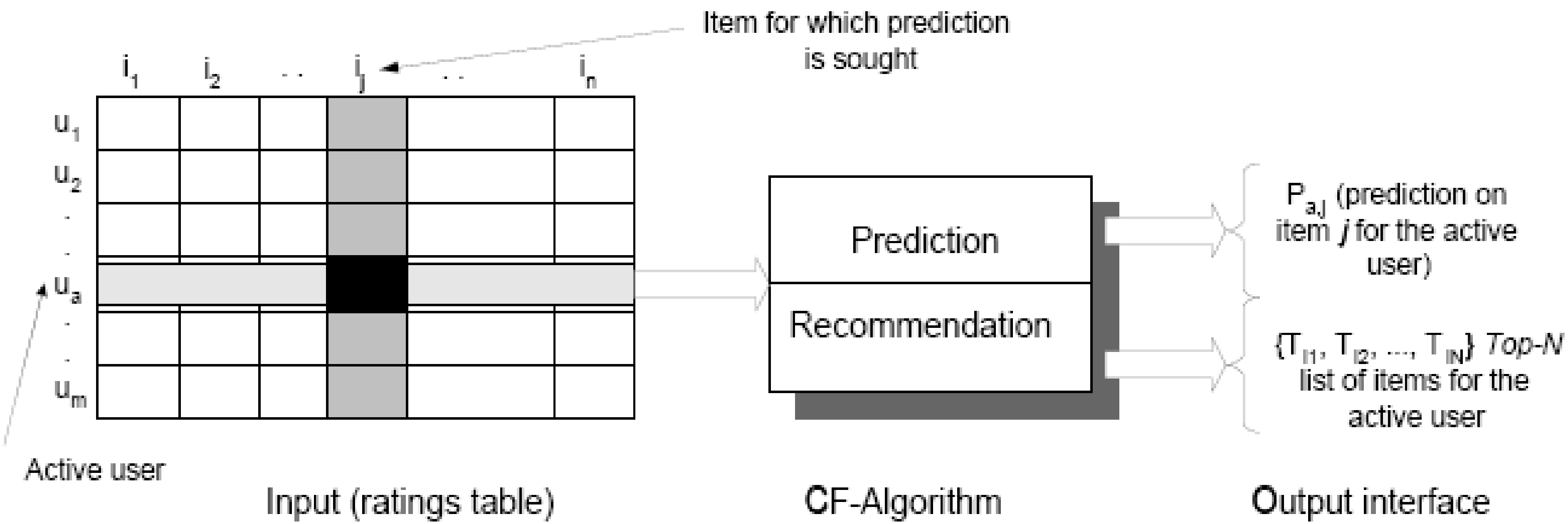


Figure 1: The Collaborative Filtering Process.

For a given “active user” estimate the rating (s)he would give to the given item, R(User, 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_{global}(User, Item)=mean(all ratings)
- R_{item}(User, Item)=mean(all ratings for Item)
- R_{user}(User, Item)=mean(all ratings for User)
- $R_{user\text{-}item}$(User, Item)=α*R_{user}(User, Item) + β*R_{item}(User, Item) + γ

(parameters α, β, γ estimated with Linear Regression)

- $R_{user^*\text{-}item}$(User, Item)=α*R_{user}(User, Item) + β_{user}*R_{item}(User, Item)
- ...

Naive Approaches

- Naive Approaches work surprisingly well
- Models very easy to calculate and to maintain *(how?)*
- 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

$$corr_{ab} = \frac{\sum_i (r_{ai} - \bar{r_a})(r_{bi} - \bar{r_b})}{\sqrt{\sum_i (r_{ai} - \bar{r_a})^2 \sum_i (r_{bi} - \bar{r_b})^2}}$$

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

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

			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

$$\frac{(4x0.9+5x0.85+5x0.64)}{(0.9+0.85+0.64)} = 4.6$$

			u$_3$	u$_4$	u$_5$	U$_6$	
			5	4		3	0.62
			1	1	5	1	0.44
item$_3$		1	**4.6**	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

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)
u **v**

• Assume that each rating can be approximated by a product of corresponding vectors:
rating=sum(“user vector” x “item vector”)
u * v’

• 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:

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

- 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(….., x, ….)** is a function of one variable
   - actually, **f(…, x, …)** is a <u>quadratic</u> function of **x** !
   - find the optimal value for **x** *(for quadratic functions it's easy!)*
   - freeze **x**, 'defreeze' another (randomly chosen?) variable **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:

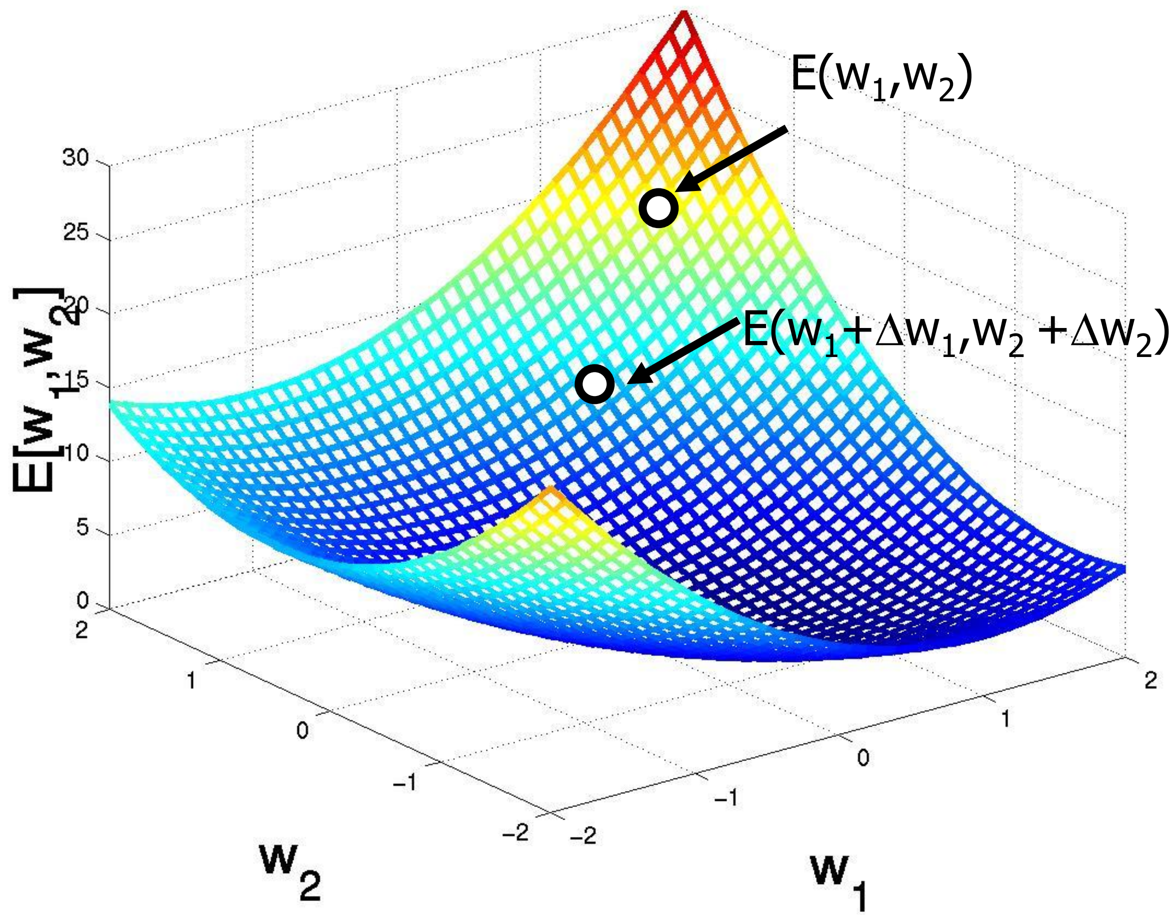
$$-\left[\frac{\partial f(x_0, y_0)}{\partial x};\frac{\partial f(x_0, y_0)}{\partial y}\right]$$

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 descend 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:

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

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

$u_{user} = u_{user} + lrate * (err * v_{item} - lambda * u_{user})$

$v_{item} = v_{item} + lrate * (err * u_{user} - lambda * v_{item})$

until convergence.

Typical values: lrate=0.001; 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:

```
sum((ratingPredicted[user][movie]-ratingTrue[user][movie])^2)
```

- Find the miniumum 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.

https://sifter.org/~simon/journal/20061211.html

Monday, December 11, 2006

Netflix Update: Try This at Home

[Followup to this]

Ok, so here's where I tell all about how I (now we) got to be tied for third place on the netflix prize. And I don't mean a sordid tale of computing in the jungle, but rather the actual math and methods. So yes, after reading this post, you too should be able to rank in the top ten or so.

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

Dimensionality Reduction and Visualization

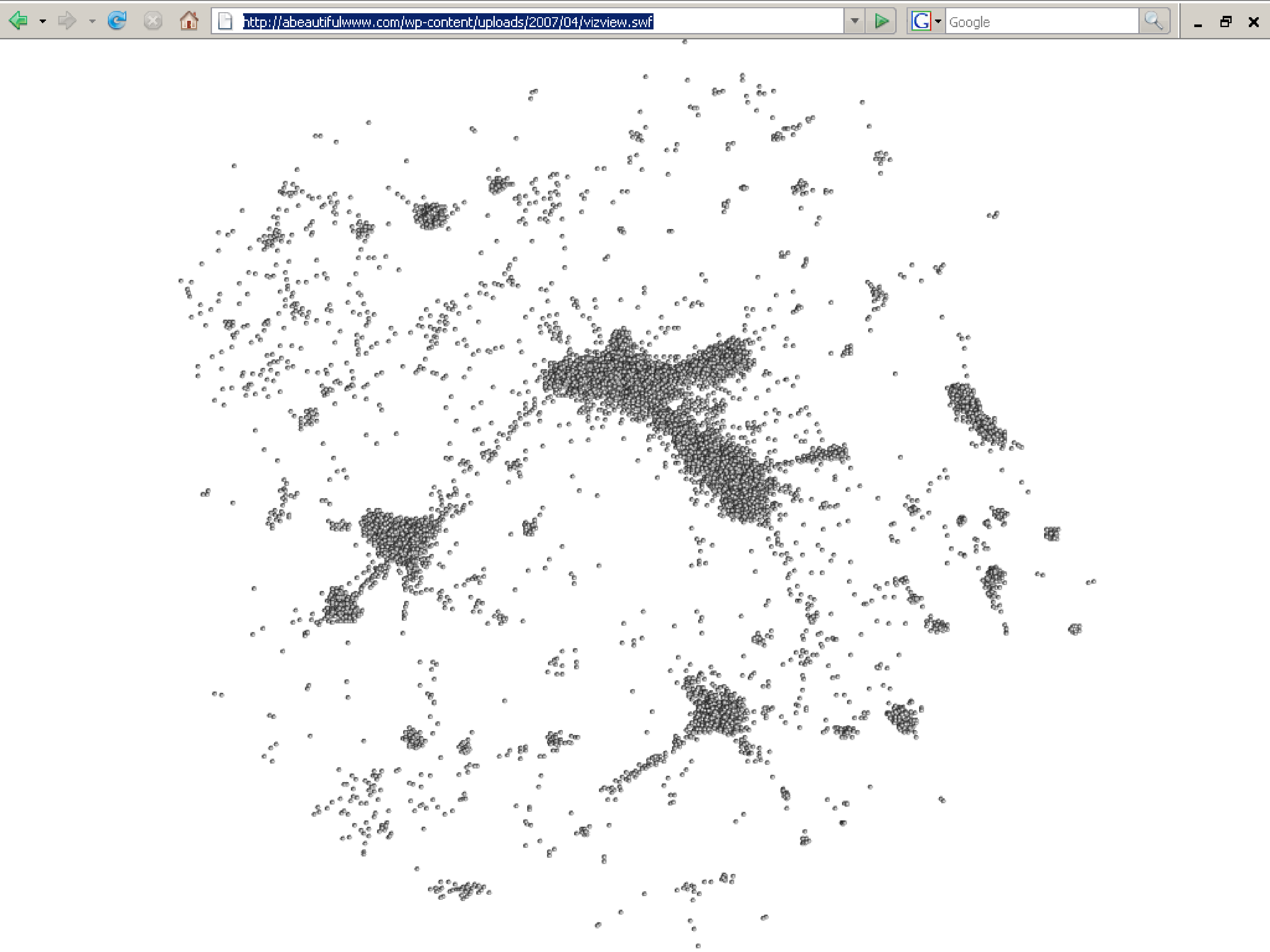
Main idea:

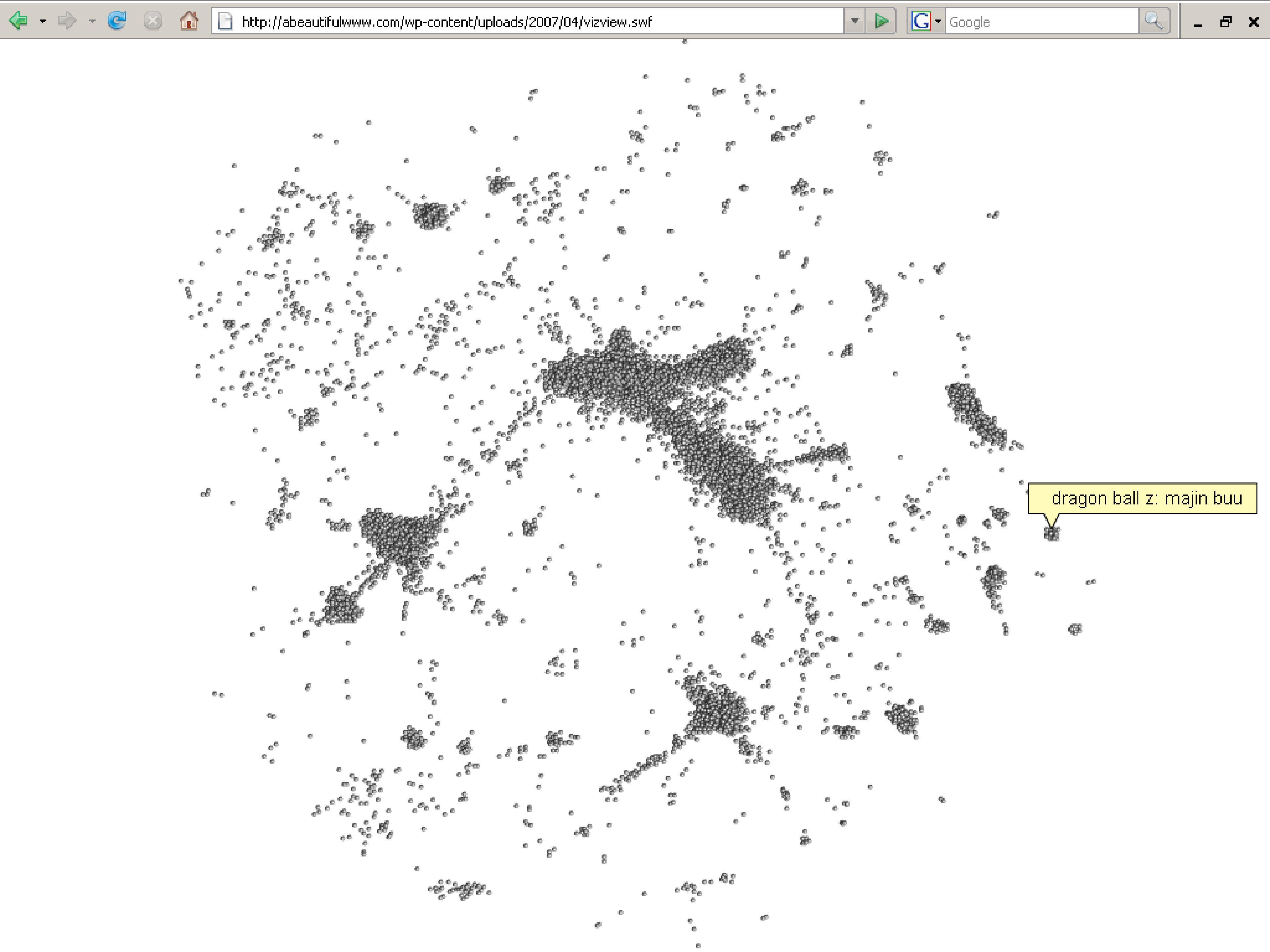
- movies are points in 100 dimensional space

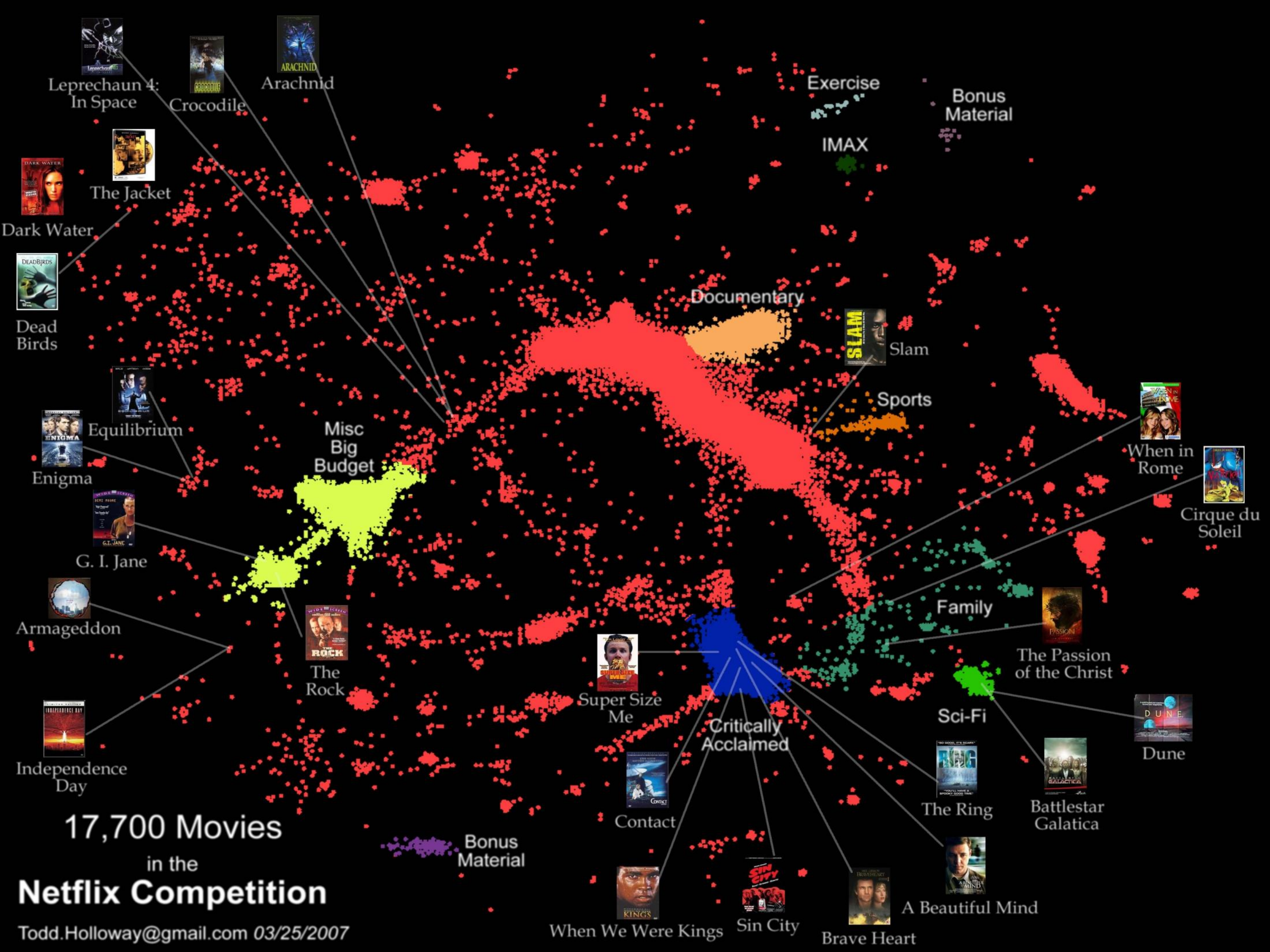
- project these points into two dimensional space preserving as much information as possible

- there are many algorithms for that:
    - –Principal Components Analysis
    - – Multi-Dimensional Scaling
    - – Self-Organizing Maps (Kohonen networks)
    - – t-SNE (will be covered later!)

•Visualize the results !


http://abeautifulwww.com/wp-content/uploads/2007/04/vizview.swf
Google


http://abeautifulwww.com/wp-content/uploads/2007/04/vizview.swf
Google
dragon ball z: majin buu


Leprechaun 4:
In Space
Crocodile
Arachnid
Exercise
Bonus
Material
IMAX
The Jacket
Dark Water
Dead
Birds
Documentary
Slam
Sports
Equilibrium
Enigma
Misc
Big
Budget
When in
Rome
Cirque du
Soleil
G. I. Jane
Family
Armageddon
The Rock
The Passion
of the Christ
Super Size
Me
Sci-Fi
Critically
Acclaimed
Dune
Independence
Day
Contact
The Ring
Battlestar
Galatica
17,700 Movies
in the
Netflix Competition
Bonus
Material
A Beautiful Mind
When We Were Kings
Sin City
Brave Heart
Todd.Holloway@gmail.com 03/25/2007

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

Simon Funk approach:
http://sifter.org/~simon/journal/20061211.html

Gravity Recommendation System (SVD):
gravity-Tikk.pdf

Netflix Workshop KDD2007:
http://www.cs.uic.edu/~liub/KDD-cup-2007/proceedings.html

More references:
https://en.wikipedia.org/wiki/Netflix_Prize