Web Intelligence
Recommender Systems

Claudio Lucchese claudio.lucchese@unive.it

Goal

- To recommend interesting stuff:
  - Songs, movies, web-pages, (queries…), …
- Popular items vs. the Long Tail
  - There is a lot of money in the tail
  - Recommending popular stuff may not be sufficient
  - Really understand what a user is looking for
- Several different approaches:

Quality Measures

- Efficiency in ***building the model***
  - How expensive is to process our data and build the recommender system?
- Efficiency in ***generating suggestions***
  - How expensive is to produce recommendations at run-time
- ***Serendipity*** of recommendations
  - Are recommendations novel and interesting?
- ***Cold-start*** problem
  - What when we know little about users?

Any Idea ?

- What about a song recommender system?

Content-based Recommender systems

- A user is the set of songs she listened
  - Define the ***song representation***
    - E.g., on the basis of the song lyrics
  - Define the ***user profile***
    - E.g., aggregate (Sum/Avg) descriptions of listened songs
- ***Similarity-based Recommendation***
  - Recommend the songs most similar to the user profile
- This can be generalized to:
  - Movies, e-commerce, Web pages, etc. etc.

A bit of terminology

- ***Users***:
  - "You and me"
  - denoted as the set U
- ***Items***:
  - The objects being recommended
  - denoted as the set I
- ***Rating***:
  - The feedback provided by the user
- ***Rating Matrix***:
  - A matrix $|U|x|I|$ where each entry stores the user feedback
  - Denoted as $R[u,i]$

Content-based Recommender systems

- We model text by using the ***vector space model***:

- Each ***item $x \in I$*** is a vector of size N
  - where *N* is the number of words in the lexicon
  - Stemming, lemmatization, stop-word removal applies here
- ***x[t] = tf(t,x) idf(t)***
  - $tf(t,x)$ is the frequency of term t in item x
  - $idf(t)$ is the inverse document frequency of t in I
- The ***profile of a user u*** is the average of the items I_u he interacted with:

$$u = \frac{1}{|I_u|} \sum_{x \in I_u} x$$

Content-based Recommender systems

- We model text by using the ***vector space model***:

- Given a user u and an item x,
  their similarity is as the ***cosine*** of the two vectors:

$$\cos(u, x) = \frac{u \cdot x}{\|u\|\|x\|} = \frac{\sum_t u[t] \cdot x[t]}{\sqrt{\sum_t u[t]^2}\sqrt{\sum_t x[t]^2}}$$

- The recommender system returns ***the top-k most similar items*** to the user profile

Content-based – Wrap up

- Efficiency in building the model
  - Easy processing for the corpus
  - Cheap model for the user
- Efficiency in generating suggestions
  - Not cheap Nearest neighbor search among the collection of documents
- Serendipity of recommendations
  - Small, tied to text similarity
- Cold-start problem
  - Partial, user should “touch” at list one item first
- Other:
  - Text representation has some non trivial issues: synonymy, polysemy, ….

Any idea?

- What information we are *not* using ?

User-based Collaborative Filtering

- Rather then finding similar items, ***find similar users***!
  - Greater serendipity !
- In many cases, users rate items
  - Explicit: stars
  - Implicit: time on a web page, clicks on a result page

- The algorithm:
  - ***Find a set neighbors*** $N(u)$ of users similar to u
  - ***Exploit the rating of users*** $N(u)$ to build recommendations for u

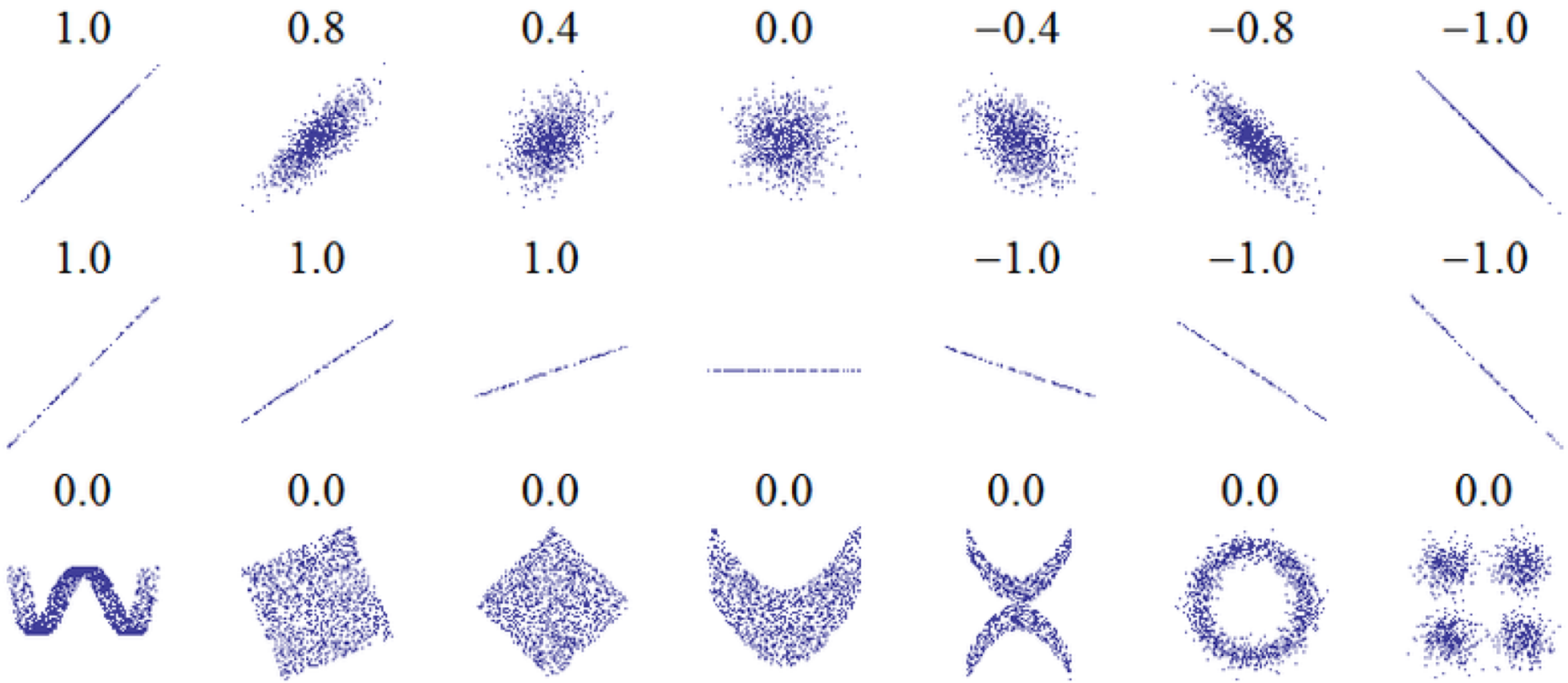
User-based Collaborative Filtering

- The ***profile of a user u*** is a vector of size $|I|$
- ***$u[i] = r$*** if the user U gave a rating of r to the item i.
- ***u*** is the ***row $R[u,:]$ of the rating matrix***
- ***Find similar users $N(u)$***:
  - Euclidean Distance?
  - Cosine similarity?

- We use ***Pearson Correlation***:
  - It is a measure of linear correlation
  - Informally, it is large if both u and v increase/decrease similarly above/below their mean $\bar{u}$ and $\bar{v}$

$$\rho(u,v) = \frac{\text{COV}(u,v)}{\sigma_u \sigma_v} = \frac{\sum_i (u[i]-\bar{u})(v[i]-\bar{v})}{\sqrt{\sum_i (u[i]-\bar{u})^2}\sqrt{\sum_i (v[i]-\bar{v})^2}}$$

Pearson Correlation



User-based Collaborative Filtering

- Build recommendations by ***ranking***:

- For each ***item $i \in I$*** compute a ranking ***score s_{ui}***:
  - By considering ***rating of users in $N(u)$***
  - By weighting those rating by the ***users' similarity with u***

$$s_{ui} = \frac{\sum_{v \in N(u)} (v[i] - \bar{v}) \cdot \rho(u, v)}{\sum_{v \in N(u)} |\rho(u, v)|}$$

- Eventually, s_{ui} is used to rank every item in I
- We *upgrade* the formula to ***predict the score by u***

$$s_{ui} = \bar{u} + \frac{\sum_{v \in N(u)} (v[i] - \bar{v}) \cdot \rho(u, v)}{\sum_{v \in N(u)} |\rho(u, v)|}$$

User-based Collaborative Filtering – Wrap up

- Efficiency in building the model
  - User similarity is expensive - $O(|U|^2)$
  - Very expensive even if done off-line
  - For every new rating by a user, similarities should be recomputed
- Efficiency in generating suggestions
  - Nearest neighbor search is not needed if pre-computed off-line
- Serendipity of recommendations
  - Great !
- Cold-start problem
  - We need a sufficiently large set of ratings for a new user/item
- Sparsity:
  - Few ratings and few shared ratings

Any idea?

- How to reduce the cost of user-based similarities?
- Do we have a different «point of view» ?

Item-based Collaborative Filtering

- Symmetric approach based on ***item similarity***!
  - An item i is represented by a vector of size $|U|$
  - $i[u] = r$ if the user U gave a rating of r to the item i.
  - i is the ***column $R[:,i]$ of the rating matrix***

- Rationale:
  - What is the score that user u gave to items similar to i ?

- The algorithm:
  - To estimate the score for an ***item i***
  - ***Find a set $N(i)$*** of other items ***rated by u*** and ***similar to i***
  - ***Exploit the ratings of user u for items in $N(i)$*** to compute a score for i

Item-based Collaborative Filtering

- ***Item Similarity***:
  - Cosine?
  - Person correlation?
    - Does the mean of a column make sense ?
  - We need to adjust by the user mean rating $\bar{u}$ ("severity")
  - ***Adjusted Cosine Similarity***:

$$\text{a}-\cos(i,j) = \frac{\sum_u (i[u]-\bar{u})(j[u]-\bar{u})}{\sqrt{\sum_u (i[u]-\bar{u})^2}\sqrt{\sum_u (j[u]-\bar{u})^2}}$$

- ***Score computation***:
  - Given the set of items $N(i)$, the ***score for the user u*** is:

$$s_{ui} = \frac{\sum_{j\in N(i)} R[j,u]\cdot \text{a}-\cos(i,j)}{\sum_{j\in N(i)} |\text{a}-\cos(i,j)|}$$

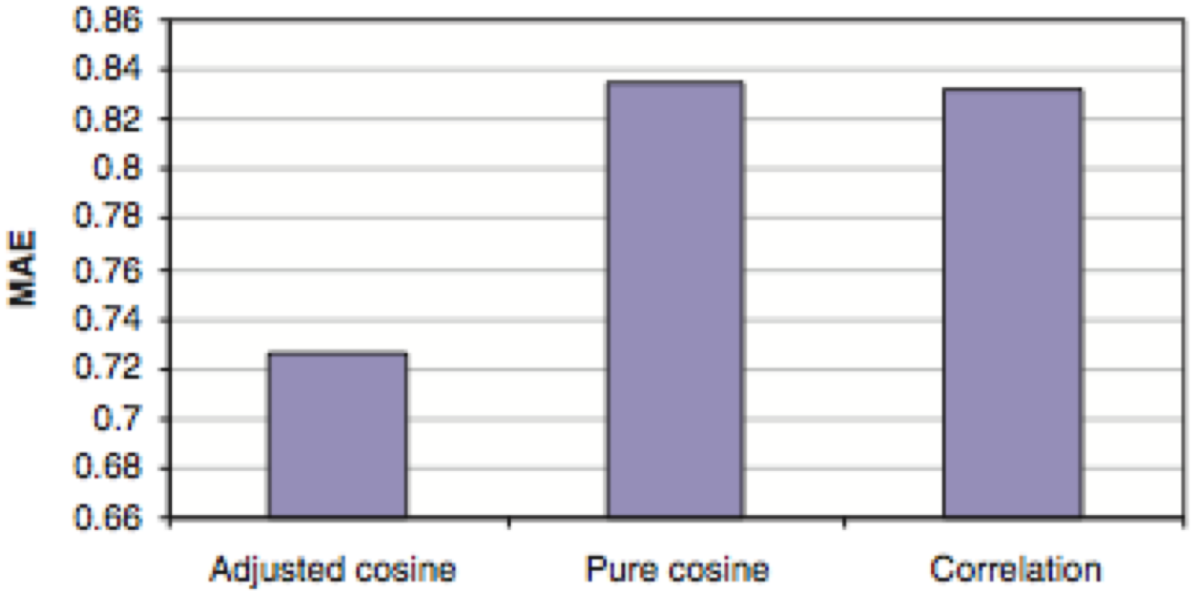
Experiments on

- Movie lens dataset:
  - 3500 movies
  - 43000 users

- Quality Measure:
  - On a subset of N ratings
  - Mean Absolute Error:

$$\text{MAE} = \frac{\sum_{i=1}^{N} |R[u,i] - s_{ui}|}{N}$$

Does it make any difference ?



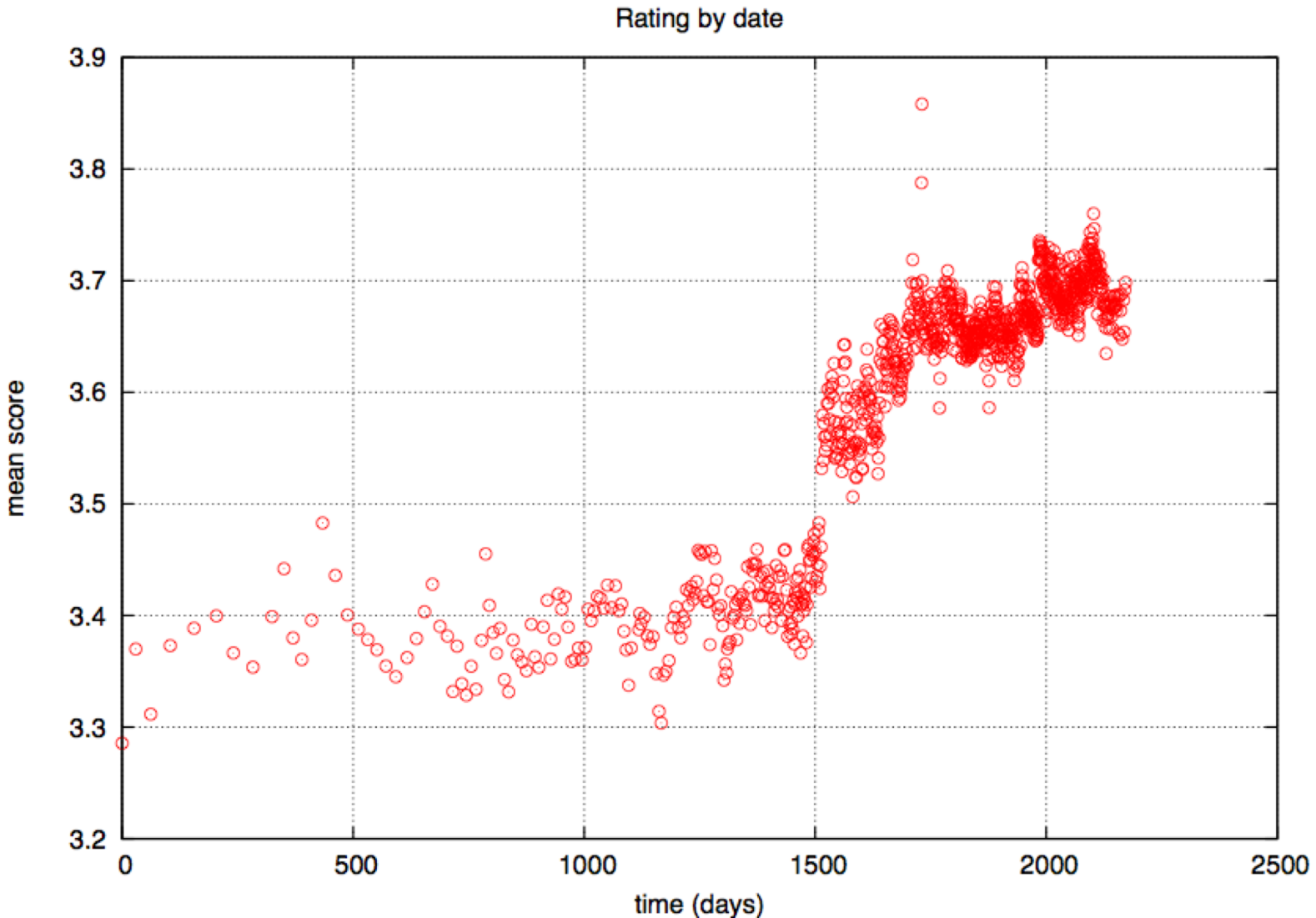
- *Lower is better*

Item-based Collaborative Filtering – Wrap up

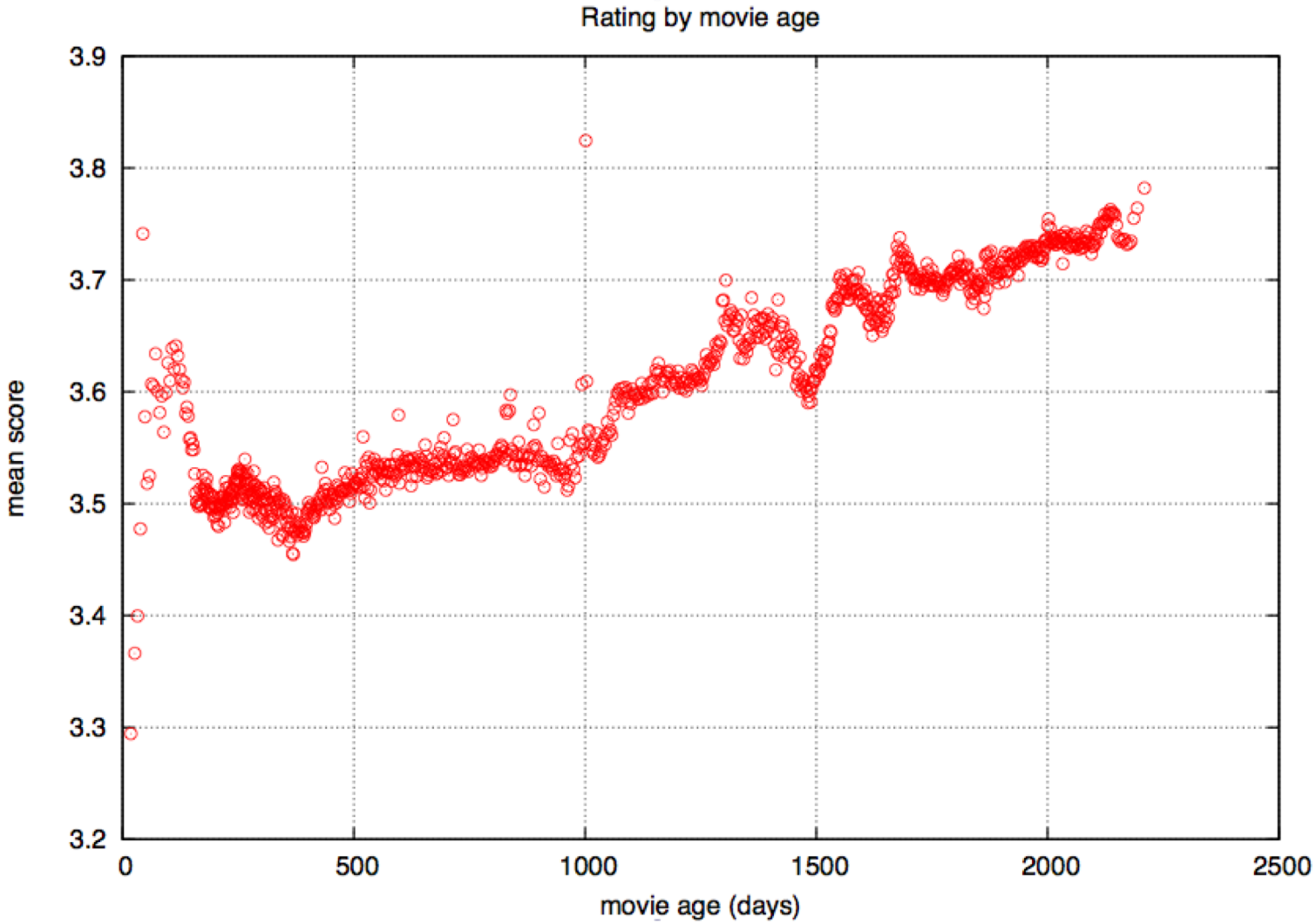
- Efficiency in building the model
  - $|I|$ is usually smaller than $|U|$, this makes the computation cheaper than user-based
  - Computation is offline
- Efficiency in generating suggestions
  - Nearest neighbor search is not needed if pre-computed off-line
- Serendipity of recommendations
  - Less than user-based, as final ***ranking depends on ratings by the current user***
- Cold-start problem
  - We need a sufficiently large set of ratings for a new user/item
- Sparsity:
  - ***Items have larger probability of sharing ratings***
- Used by Amazon !

Is this enough to win the netflix prize?

Effects of time



Effects of Time



Baseline Predictors

$$b_{ui} = \mu + b_u + b_i$$

- μ is the average rating over every item and every user
- b_u models how much the user u is different from the average
- b_i models how much item i is different from the average

- All of the above can be computed by "mining" the available movie ratings
  - It is a minimization problem, where the error of the prediction is minimized
  - Gradient descent is one was to find the minimum of a function

Incorporating Time in Item predictors

$$b_{ui}(t) = \mu + b_u(t) + b_i(t)$$

- b_u and b_i are a function of time

- Discretize time into bins
  - E.g. each bin corresponds to 10 weeks

$$b_i(t) = b_i + b_{i,\mathrm{Bin}}(t)$$

Incorporating Time in Users predictors

- Modeling of user drift in time

$$b_u(t) = b_u + \alpha_u \mathrm{dev}_u(t)$$

$$\mathrm{dev}_u(t) = \mathrm{sign}(t - t_u)|t - t_u|^{\beta}$$

- t_u is the mean day of rating
- β controls the "speed" of the linear drift
- α_u is a *per user* weight of the time drift impact

- Finally, to model single day effects $b_{u,t}$ is added:

$$b_u(t) = b_u + \alpha_u \mathrm{dev}_u(t) + b_{u,t}$$

Not yet there… Periodicity

- Items may be more appealing in different times:
  - E.g., {Spring, Summer, Fall, Winter}

$$b_i(t) = b_i + b_{i,\mathrm{Bin}(t)} + b_{i,period(t)}$$

- User may change behavior over time:
  - (e.g. Monday, Tuesday, …, Sunday)

$$b_u(t) = b_u + \alpha_u \cdot \mathrm{dev}_u(t) + b_{u,t} + b_{u,\mathrm{period}(t)}$$

Baseline item predictor is nor completely user independent

- Users have different rating scales:

$$b_{ui}(t) = \mu + b_u + \alpha_u \cdot \mathrm{dev}_u(t) + b_{u,t} + (b_i + b_{i,\mathrm{Bin}(t)}) \cdot c_u(t)$$

Putting it all together

- We predict the score for an item by merging the baseline predictors with the item-based CF

$$s_{ui}(t) = \frac{\sum_{j \in N(i)} R[i,u] \cdot \text{a}{-}\cos(i,j)}{\sum_{j \in N(i)} \text{a}{-}\cos(i,j)} + b_{ui}(t)$$

$$b_{ui}(t) = \mu + b_u + \alpha_u \cdot \text{dev}_u(t) + b_{u,t} + (b_i + b_{i,\text{Bin}(t)}) \cdot c_u(t)$$

- Additionally, temporal dynamics are added to CF
  - contribution of item *j* is discounted "exponentially" by the following factor:

$$e^{-\beta_u |t - t_j|}$$

So What ??

- Was it worthwhile ??
  - Yes!

- Anything else?
  - Yes: Factor analysis, Singular Value Decomposition, …
    - Yehuda Koren. Collaborative Filtering with Temporal Dynamics. KDD 2009

Try Yourself

- https://grouplens.org/datasets/movielens/

References

- **Recommender Systems Handbook.** Francesco Ricci, Lior Rokach, Bracha Shapira, Paul B. Kantor, Springer.
  - http://www.cs.ubbcluj.ro/~gabis/DocDiplome/SistemeDeRecomandare/Recommender_systems_handbook.pdf
  - Chapter 1: Introduction
  - Chapter 4: A Comprehensive Survey of Neighborhood-based Recommendation Methods
    - Excluding Sections 4.2.2 e 4.2.3
    - Until Section 4.3.2.1 (included)