Collaborative Filtering (Cont'd)

Harnessing quality judgments of other users

Thanks for source slides and material to: J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets
http://www.mmds.org

Collaborative Filtering: Overview

- CF works by **collecting user feedback**: e.g., **ratings for items**
  - Exploit similarities in rating behavior among users in determining recommendations
- **Two classes of CF algorithms:**

1. **Neighborhood-based or Memory-based approaches**
   - User-based CF
   - Item-based CF
2. **Model-based approaches**
   - Estimate parameters of statistical models for user ratings
   - Latent factor and matrix factorization models

Making User-based CF Predictions with Pearson: Weighted Sum of Neighbors Ratings

- **Find neighbors *U* of an active user *a* based co-rated items *I***

$$w_{u,v} = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}}$$

- **Making predictions**
  - **Summarize the neighbor *U's* weighted ratings for item *i***

$$P_{a,i} = \bar{r}_a + \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_u) \cdot w_{a,u}}{\sum_{u \in U} | w_{a,u} |}$$

Continued Example: User-Based CF Prediction with Pearson Correlation Coefficient

	I_1	I_2	I_3	I_4	"average other than I2"
U_1	4	?	5	5	(4+5+5)/3
U_2	4	2	1		(4+1)/2
U_3	3		2	4	
U_4	4	4			(4)/1
U_5	2	1	3	5	(2+3+5)/3

- **Want to predict rating for user U1 on item I2:** users U2, U4 and U5 rated I2
- **Similarity of U1 to these users:** $w_{1,5} = 0.756$, $w_{1,4} = 0$, $w_{1,2} = -1$

$$P_{1,2} = \bar{r}_1 + \frac{\sum_u (r_{u,2} - \bar{r}_u) \cdot w_{1,u}}{\sum_u |w_{1,u}|}$$

14/3 = 4.67

5/2=2.5

$$= \bar{r}_1 + \frac{(r_{2,2} - \bar{r}_2) w_{1,2} + (r_{4,2} - \bar{r}_4) w_{1,4} + (r_{5,2} - \bar{r}_5) w_{1,5}}{|w_{1,2}| + |w_{1,4}| + |w_{1,5}|}$$

4/1=4

$$= 4.67 + \frac{(2 - 2.5)(-1) + (4 - 4)0 + (1 - 3.33)0.756}{1 + 0 + 0.756}$$

10/3=3.33

$$= 3.95.$$

ITEM-BASED CF

Item-based Collaborative Filtering

- Neighborhood-based CF algorithms do not scale well when applied to millions of users & items
  - Due to computational complexity of search for similar users (possible solutions?)
- **Item-to-item collaborative filtering**
  - Rather than matching similar users
  - **Match user's rated items to similar items**
- In practice, often leads to faster online systems and better recommendations
- **Similarities between pairs of items i and j are computed off-line**
- Predict rating of user **a** on item ***i*** with a simple weighted average

Pearson Correlation between items i, j

For the item-based algorithm, denote the set of users $u \in U$ who rated both items i and j, then the *Pearson Correlation* will be

$$w_{i,j} = \frac{\sum_{u \in U} \left(r_{u,i} - \overline{r}_i\right)\left(r_{u,j} - \overline{r}_j\right)}{\sqrt{\sum_{u \in U} \left(r_{u,i} - \overline{r}_i\right)^2} \sqrt{\sum_{u \in U} \left(r_{u,j} - \overline{r}_j\right)^2}}, \qquad (2)$$

where $r_{u,i}$ is the rating of user u on item i, $\overline{r}_i$ is the average rating of the ith item by those users, see Figure 2 [40].

- **Note: Sum over set of users *U* who rated both items *i, j***
- $r_{u,i}$ is rating of user *u* on item *i*
- $\overline{r_i}$ is average rating of i^{th} item by those users

Pearson Correlation between items i, j

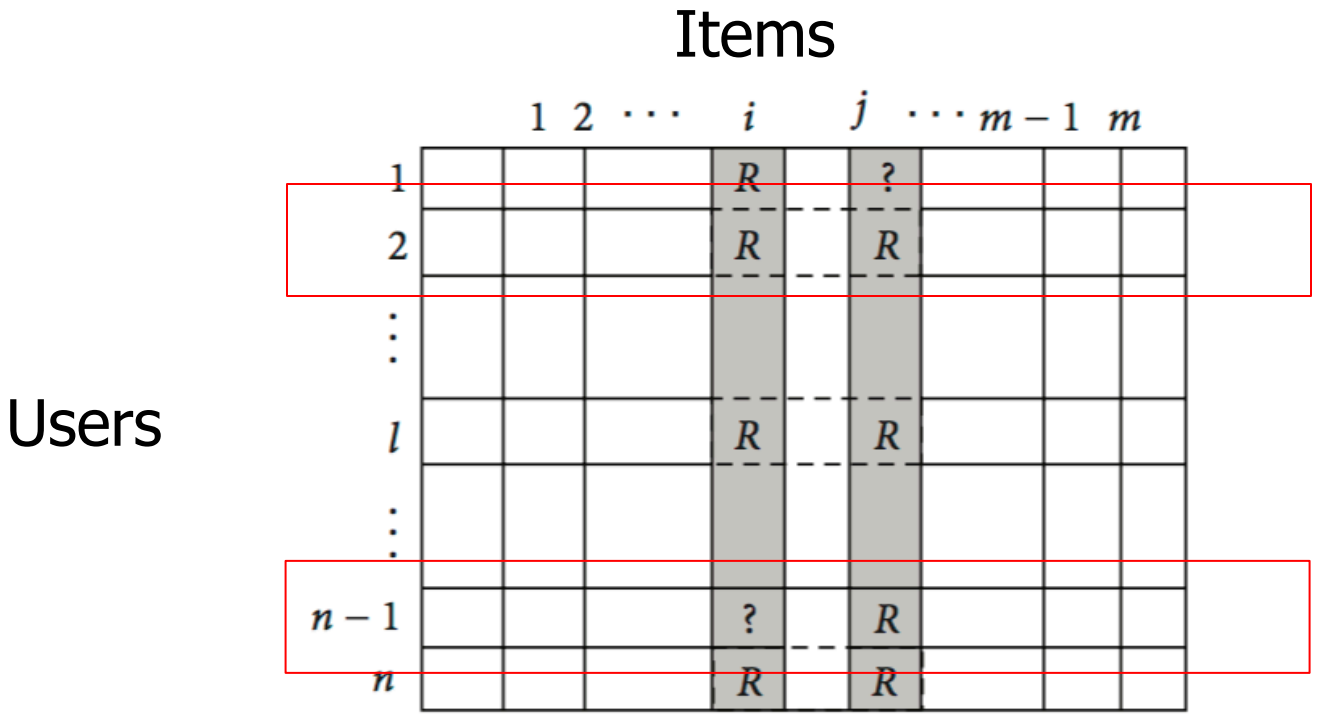


FIGURE 2: item-based similarity ($w_{i,j}$) calculation based on the co-rated items i and j from users 2, l and n.

Source: Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in artificial intelligence*, *2009*, 4.

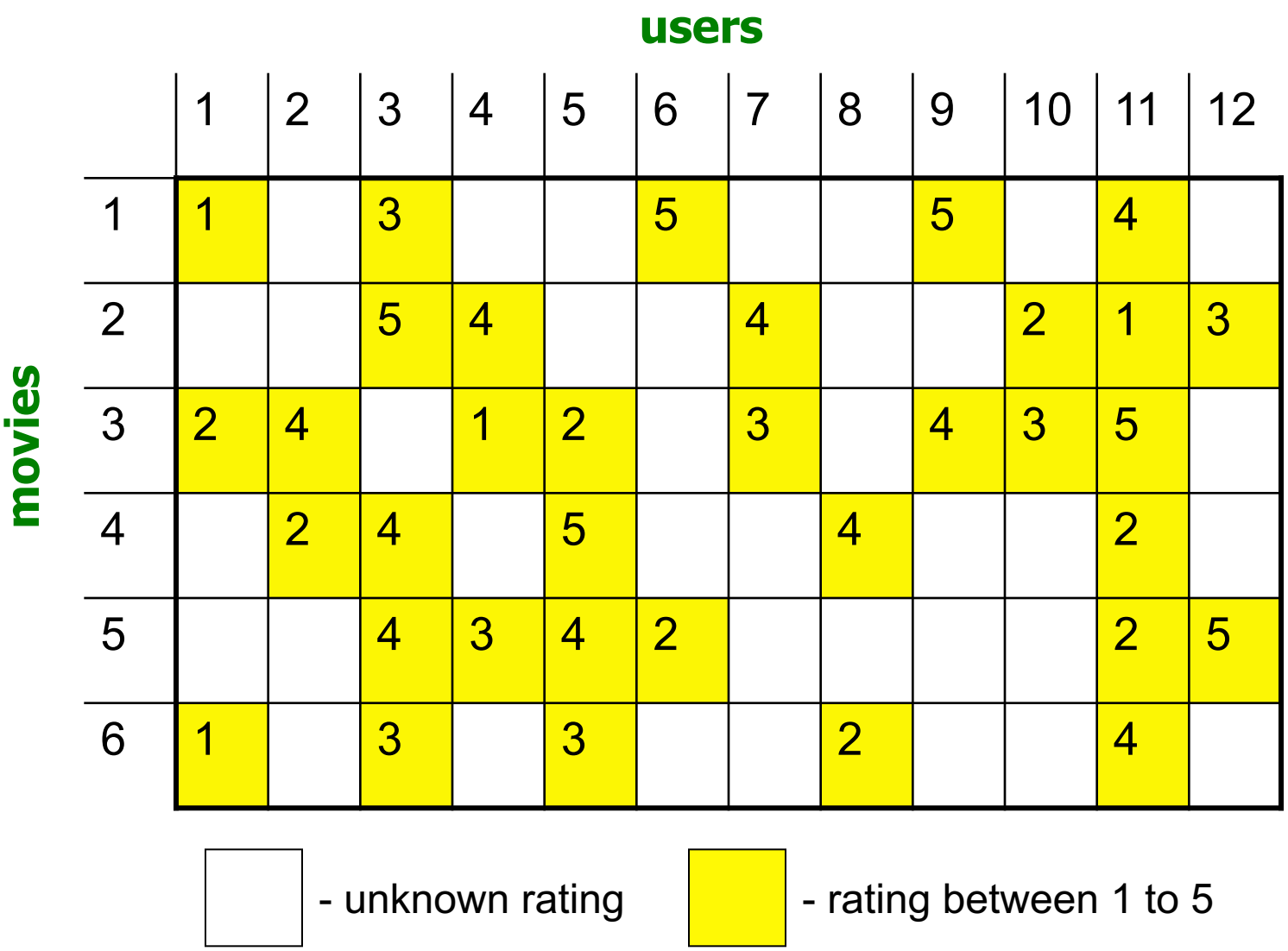
Make Item-Based Predictions Using a Simple Weighted Average

- Predict rating for user *u* on item *i*
- $w_{i,n}$ is weight between items *i and n*
- $r_{u,n}$ is rating for user *u* on item *n*
- Summation over **neighborhood set *N* of items** rated by *u* that are most similar to *i*

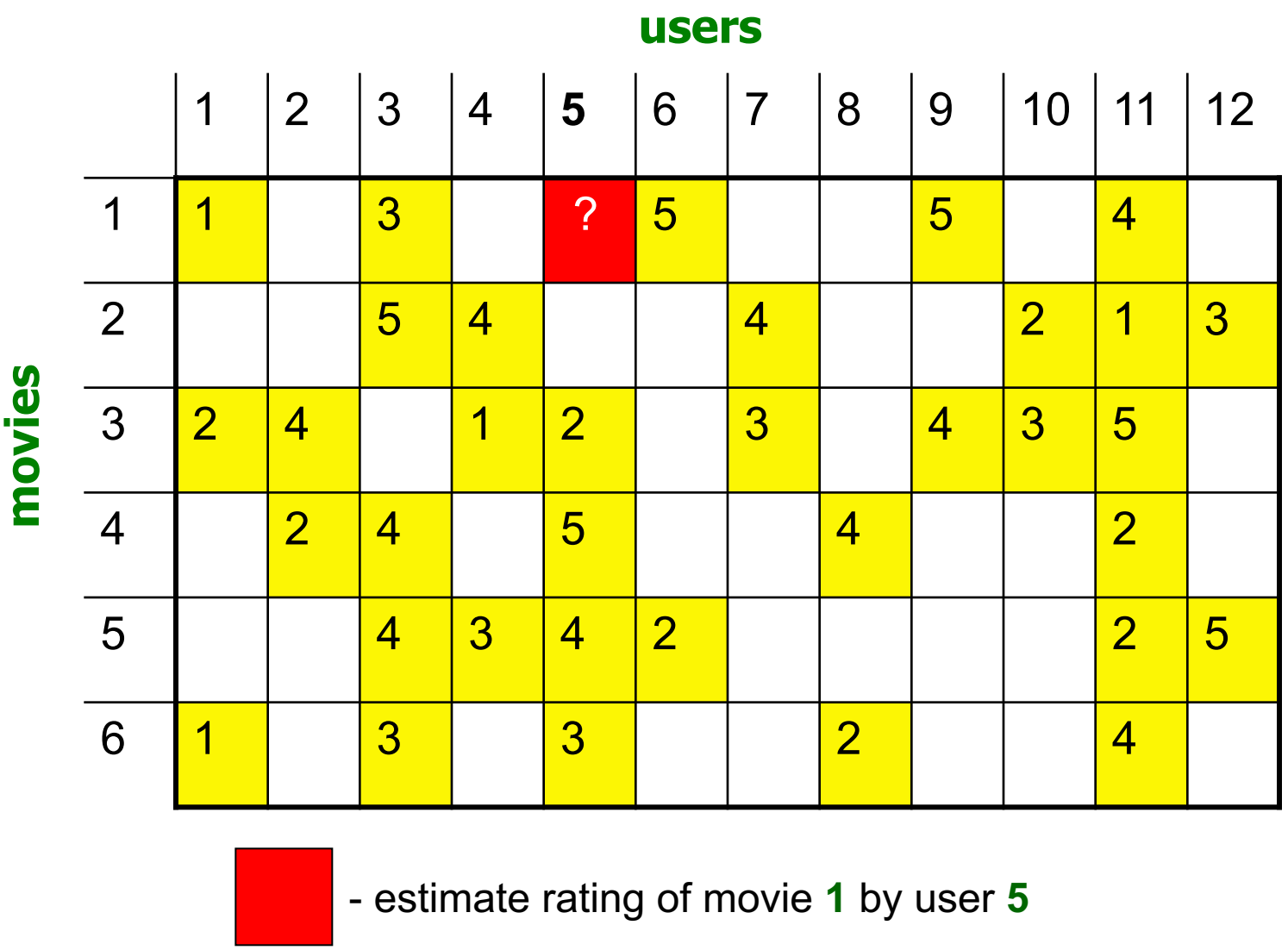
$$P_{u,i} = \frac{\sum_{n \in N} r_{u,n} w_{i,n}}{\sum_{n \in N} | w_{i,n} |}$$

where the summations are over all other rated items $n \in N$ for user u, $w_{i,n}$ is the weight between items i and n, $r_{u,n}$ is the rating for user u on item n.

Item-Item CF (|N|=2)



Item-Item CF (|N|=2)



Item-Item CF (|N|=2)

First: what is similarity between items?

users

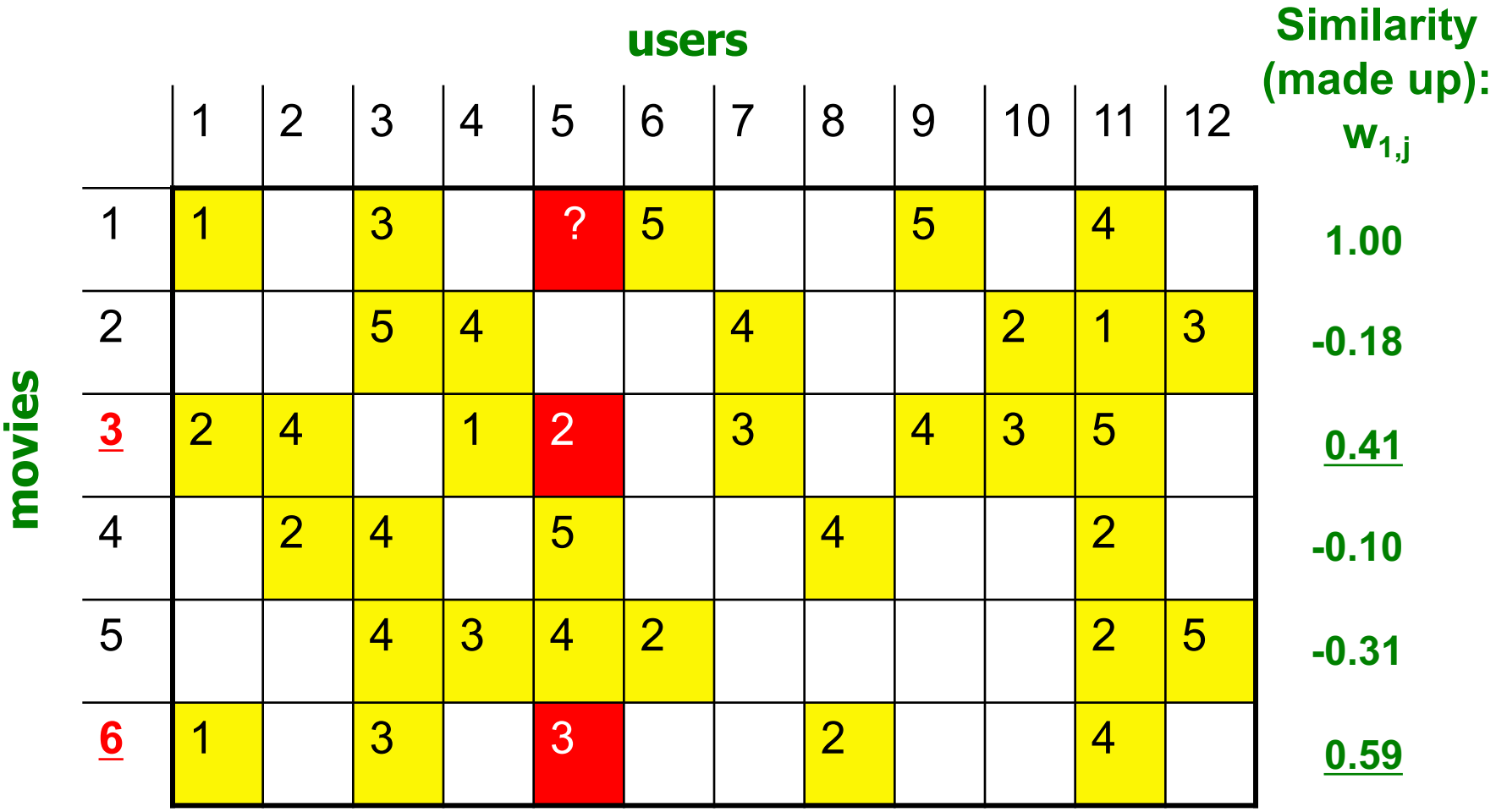
movies

	1	2	3	4	**5**	6	7	8	9	10	11	12	Similarity (made up for example): $w_{1,j}$
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		**0.41**
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		**0.59**

Neighbor selection: Identify movies **most similar to movie 1** and **rated by user 5**

Neighborhood size is 2: pick movies 3 and 6

Item-Item CF (|N|=2)



Similarity weights:

$w_{1,3}$=0.41, $w_{1,6}$=0.59

Item-Item CF (|N|=2)

users

movies

	1	2	3	4	5	6	7	8	9	10	11	12	Similarity: $w_{1,j}$
1	1		3		**2.6**	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		0.41
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		0.59

Predict by taking weighted average:

$$P_{u,i} = \frac{\sum_{n \in N} r_{u,n} w_{i,n}}{\sum_{n \in N} |w_{i,n}|}$$

$P_{1.5}$ = (0.41*2 + 0.59*3) / (0.41+0.59) = 2.6

Example: Item-to-Item Collaborative Filtering with Pearson Similarity

	I1	I2	I3	I4
U1	2	1		3
U2	3	?	5	2
U3		4	2	3
U4	5	3	1	

$$w_{i,j} = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_j)^2}}$$

- What does set U represent?
  - *Set of users U who rated both items i, j*
- What are the members of the set U for $w_{1,2}$?
  - *U1 and U4 rated items I1 and I2*
- When calculating average ratings for item i, which ratings do we use? All ratings or just for co-rated items?
  - *We can use all ratings: based on all ratings: avg(r_1) = (2+3+5)/3, avg(r_2) = (1+4+3)/3*
  - *We can also use* co-rated items (to be shown later)

Example: Item-to-Item Collaborative Filtering with Pearson Similarity

	I1	I2	I3	I4
U1	2	1		3
U2	3	?	5	2
U3		4	2	3
U4	5	3	1	

$$w_{i,j} = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_j)^2}}$$

- *Based on all ratings: average ratings for items I1, I2: $\overline{r_1} = 10/3$, $\overline{r_2} = 8/3$*
- Pearson correlation for $w_{1,2}$: *Similarity between items 1 and 2*
  - *Set U includes U1 and U4*
  - $w_{1,2}$ = (($r_{U1,I1} - 10/3)(r_{U1,I2} - 8/3) + (r_{U4,I1} - 10/3)(r_{U4,I2} - 8/3)$) /
    (sqrt(($r_{U1,I1} - 10/3)$^2 + ($r_{U4,I1} - 10/3)$^2) * sqrt(($r_{U1,I2} - 8/3)$^2) + ($r_{U4,I2} - 8/3)$^2))
  - $w_{1,2}$ = ((2-10/3)(1-8/3) + (5-10/3)(3-8/3)) /
    (sqrt((2-10/3)^2 + (5-10/3)^2) * sqrt((1-8/3)^2 + (3-8/3)^2))
    =2.778/3.628=0.765

Example: Item-to-Item Collaborative Filtering with Pearson Similarity

	I1	I2	I3	I4
U1	2	1		3
U2	3	?	5	2
U3		4	2	3
U4	5	3	1	

$$w_{i,j} = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_j)^2}}$$

co-rated items only

- *Based on co-ratings: average ratings for items I1, I2: $\overline{r_1}$ = (2+5)/2, $\overline{r_2}$ = (1+3)/2*
- Pearson correlation for $w_{1,2}$: *Similarity between items 1 and 2*
  - *Set U includes U1 and U4*
  - $w_{1,2}$ = (($r_{U1,I1}$ – 7/2)($r_{U1,I2}$ – 4/2) + ($r_{U4,I1}$ – 7/2)($r_{U4,I2}$ – 4/2)) /
    (sqrt(($r_{U1,I1}$ – 7/2)^2 + ($r_{U4,I1}$ – 7/2)^2) * sqrt(($r_{U1,I2}$ – 4/2)^2) + ($r_{U4,I2}$ – 4/2)^2))
  - $w_{1,2}$ = ((2-7/2)(1-4/2) + (5-7/2)(3-4/2)) /
    (sqrt((2-7/2)^2 + (5-7/2)^2) * sqrt((1-4/2)^2 + (3-4/2)^2))
    =3/3=1

Item-Based Prediction

	I1	I2	I3	I4
U1	2	1		3
U2	3	?	5	2
U3		4	2	3
U4	5	3	1	

$$P_{u,i} = \frac{\sum_{n \in N} r_{u,n} w_{i,n}}{\sum_{n \in N} | w_{i,n} |}$$

- Assume that item similarities are: $w_{2,1} = 0.5$, $w_{2,3} = 0.2$, $w_{2,4} = 0.3$
- Which items are in the neighborhood N for item 2 if |N| = 2?
  - Items I1 and I4
- **Predict the rating of user U2 on I2:** user = 2, item = 2, |N| = items 1,4

$$P_{2,2} = \frac{(r_{2,1} * w_{2,1}) + (r_{2,4} * w_{2,4})}{0.5 + 0.3} = \frac{3*0.5 + 2*0.3}{0.8} = 2.625$$

EXTENSIONS TO MEMORY-BASED ALGORITHMS

Extensions to Memory-Based Algorithms

- A variety of approaches/extensions have been studied to improve the performance of CF predictions
- Typically involve **modifying the similarity weights**
  or the **ratings** used in predictions
  or **guessing missing ratings**
- **User-based CF:**

$$w_{u,v} = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}} \qquad P_{a,i} = \bar{r}_a + \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_u) \cdot w_{a,u}}{\sum_{u \in U} |w_{a,u}|}$$

- **Item-Based CF:**

$$w_{i,j} = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_j)^2}} \qquad P_{u,i} = \frac{\sum_{n \in N} r_{u,n} w_{i,n}}{\sum_{n \in N} |w_{i,n}|}$$

Extensions to Memory-Based Algorithms: Default Voting

- In many collaborative filters, **pairwise similarity is computed only from the ratings in the intersection of the items both users have rated ("co-rated items")**
  - **Not reliable when there are too few votes** to generate similarity values (U is small)
  - Focusing on co-rated items ("intersection set similarity") also **neglects global rating behavior reflected in a user's entire rating history**
- Assuming some default voting values for the missing ratings: **can improve CF prediction performance**

Extensions to Memory-Based Algorithms: Default Voting (cont.)

Approaches to default voting values:

- Herlocker et al. accounts for small intersection sets (small number of co-rated items) by **reducing the weight of users that have fewer than 50 items in common**

$$P_{u,i} = \frac{\sum_{n \in N} r_{u,n} w_{i,n}}{\sum_{n \in N} | w_{i,n} |}$$

- Chee et al. **use average of the clique (small group of co-rated items) as a default voting** to extend a user's rating history
- Breese et al. **use a neutral or somewhat negative preference for the unobserved ratings** and then computes similarity between users on the resulting ratings data.

Extensions to Memory-Based Algorithms: Inverse User Frequency

- **Universally liked items are not as useful in capturing similarity as less common items**
- **Inverse frequency**
  - $f_j = log\ (n/n_j)$
  - **n_j is number of users who have rated item *j***
  - **n is total number of users**
- If everyone has rated item *j*, then f_j is zero
  - (Note: looks a lot like Inverse Document Frequency (IDF))
- **Approach: transform the ratings**
  - For vector similarity-based CF: **new rating = original rating multiplied by f_j**

$$P_{a,i} = \bar{r}_a + \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_u) \cdot w_{a,u}}{\sum_{u \in U} |w_{a,u}|}$$

  - **For very popular items, ratings $r_{u,i}$ will be greatly reduced**
  - **Less popular items will have greater effect on prediction**

Extensions to Memory-Based Algorithms: Case Amplification

- **Transform applied to weights used in CF prediction**
- Emphasizes high weights and punishes low weights

$$w'_{i,j} = w_{i,j} \cdot |w_{i,j}|^{\rho-1}, \quad (8)$$

where ρ is the *case amplification* power, $\rho \geq 1$, and a typical choice of ρ is 2.5 [65].

- **Reduces noise in the data**
- **Favors high weights**

$$P_{u,i} = \frac{\sum_{n \in N} r_{u,n} w_{i,n}}{\sum_{n \in N} |w_{i,n}|}$$

- Small values raised to a power become negligible
- Example:
  - For $w_{i,j} = 0.9$, weight it remains high ($0.9^{2.5} \approx 0.8$)
  - For $w_{i,j} = 0.1$, weight becomes negligible ($0.1^{2.5} \approx 0.003$)

Extensions to Memory-Based Algorithms: Imputation-Boosted CF

- **When the rating data for CF tasks are extremely sparse: hard to produce accurate predictions using the Pearson correlation-based CF**
- Su et al. uses imputation-boosted collaborative filtering (IBCF)
- **First uses an imputation technique to fill in missing data**
- **Then use traditional Pearson correlation-based CF algorithm** on this completed data to predict a user rating for a specified item
  - **Example imputation techniques:** mean imputation, linear regression imputation, predictive mean matching imputation, Bayesian multiple imputation, and machine learning classifiers (including naıve Bayes, SVM, neural network, decision tree, lazy Bayesian rules)

Extensions to Memory-Based Algorithms: Imputation-Boosted CF (Cont'd)

Simple mean imputation

The data on the left below has one missing observation on variable 2, unit 10.

We replace this with the arithmetic average of the observed data *for that variable*. This value is shown in red in the table below.

Unit	Variables 1	2
1	3.4	5.67
2	3.9	4.81
3	2.6	4.93
4	1.9	6.21
5	2.2	6.83
6	3.3	5.61
7	1.7	5.45
8	2.4	4.94
9	2.8	5.73
10	3.6	5.58

- This approach is clearly inappropriate for categorical variables.
- It does not lead to proper estimates of measures of association or regression coefficients. Rather, associations tend to be diluted.
- In addition, variances will be wrongly estimated (typically under estimated) if the imputed values are treated as real. Thus inferences will be wrong too.

Imputation Example

Source: http://missingdata.lshtm.ac.uk/index.php?option=com_content&view=article&id=68:simple-mean-imputation&catid=39:simple-ad-hoc-methods-for-coping-with-missing-data&Itemid=96

MODEL-BASED CF

Collaborative Filtering Overview

CF works by **collecting user feedback**: **ratings for items**

- Exploit similarities in rating behavior among users in determining recommendations

Two classes of CF algorithms:

1. **Neighborhood-based or Memory-based approaches**
   - User-based CF
   - Item-based CF
2. **Model-based approaches**
   - Estimate parameters for statistical models for user ratings
   - Latent factor and matrix factorization models

Model-based Collaborative Filtering

- Provide recommendations by estimating parameters of statistical models for user ratings
- Design and development of models can allow system to learn to recognize complex patterns
  - Based on training set – supervised learning
- Then make intelligent predictions for CF tasks based on the **learned models**
- Example models:
  - Bayesian models
  - **Clustering models**
  - Dependency networks
  - Classification algorithms (if users ratings are in categories)
  - Regression models and SVD (singular value decomposition) methods for numerical ratings

CLUSTERING CF

Clustering CF Algorithms

- **Cluster = collection of data objects that are:**
  - Similar to one another within the same cluster
  - Dissimilar to objects in other clusters
- **Measurements of similarity between objects include**
  - Pearson correlation
  - Cosine similarity
    - (it's important to study your data! E.g., http://grouplens.org/blog/similarity-functions-for-user-user-collaborative-filtering/)
  - Minkowski distance
    - Two objects $X = (x_1, x_2, \ldots, x_n)$, $Y = (y_1, y_2, \ldots, y_n)$
    - Where q is a positive integer
    - If q=2: Euclidean distance

$$d(X, Y) = \sqrt[q]{\sum_{i=1}^{n} |x_i - y_i|^q},$$

Clustering Algorithms

- Common clustering method
  - K-Means
  - Hierarchical Clustering

Clustering Algorithms (Cont'd)

- **Key operation: Repeatedly combine two nearest clusters**
- **(1) How to represent a cluster of many points?**
  - **Key problem:** As you merge clusters, how do you represent the "location" of each cluster, to tell which pair of clusters is closest?
  - **Euclidean case:** each cluster has a *centroid* = average of its (data)points
- **(2) How to determine "nearness" of clusters?**
  - Measure cluster distances by distances of centroids

K–Means Algorithm(s)

- Assumes Euclidean space/distance
- Start by picking ***k***, the number of clusters
- Initialize clusters by picking one point per cluster
  - **Example:** Pick one point at random, then ***k*-1** other points, each as far away as possible from the previous points

Populating Clusters

1) For each point, place it in the cluster whose current centroid it is nearest

2) After all points are assigned, update the locations of centroids of the ***k*** clusters

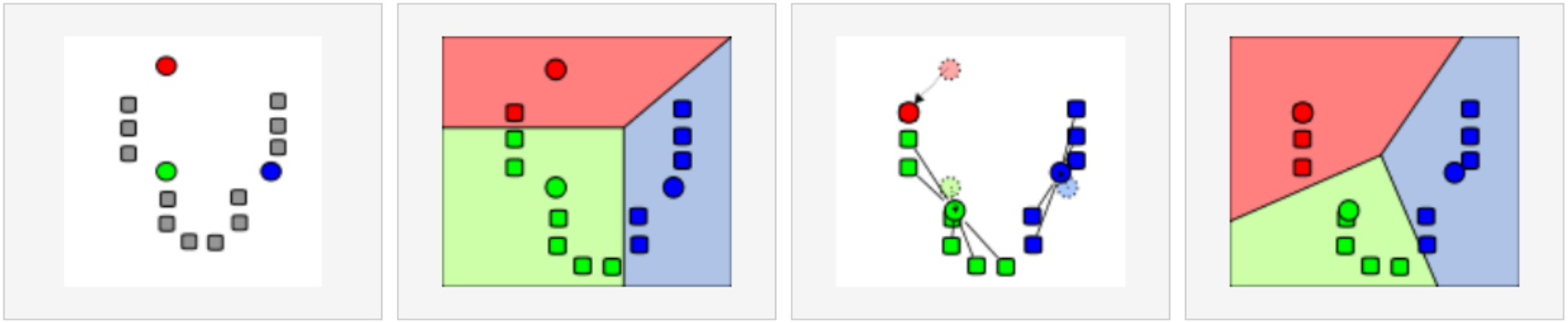
3) Reassign all points to their closest centroid

Sometimes moves points between clusters

- **Repeat 2 and 3 until convergence**
  - **Convergence:** Points don't move between clusters and centroids stabilize

Example: Standard K-Means

Demonstration of the standard algorithm



1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

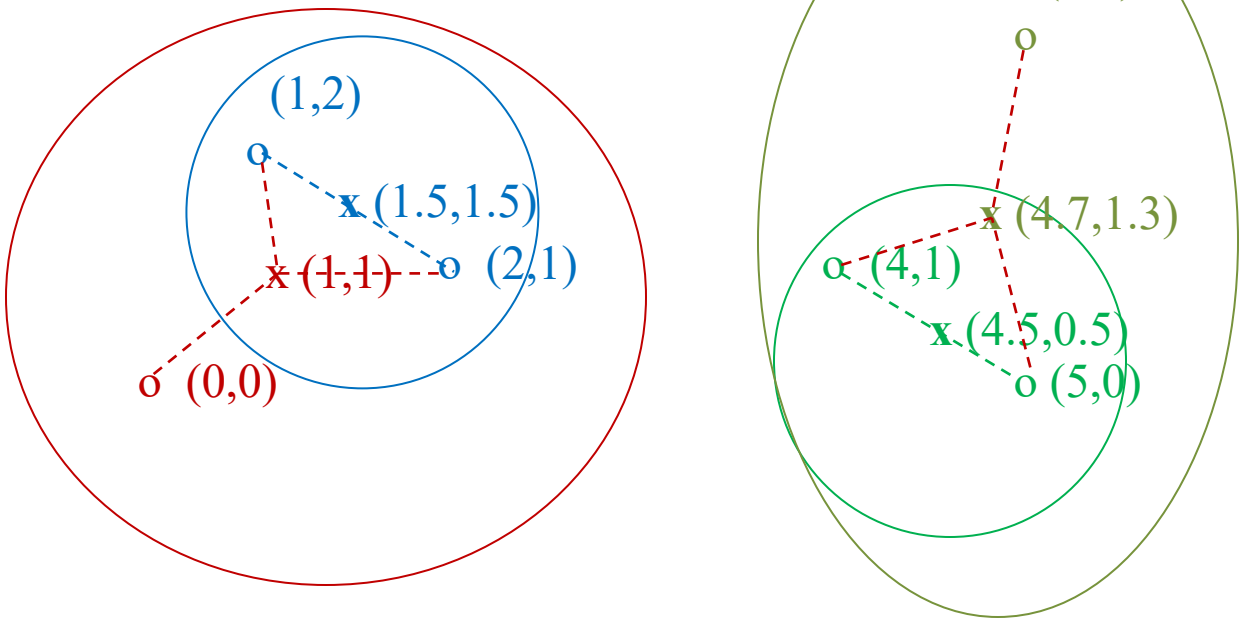
2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

3. The centroid of each of the *k* clusters becomes the new mean.

4. Steps 2 and 3 are repeated until convergence has been reached.

Source: https://en.wikipedia.org/wiki/K-means_clustering

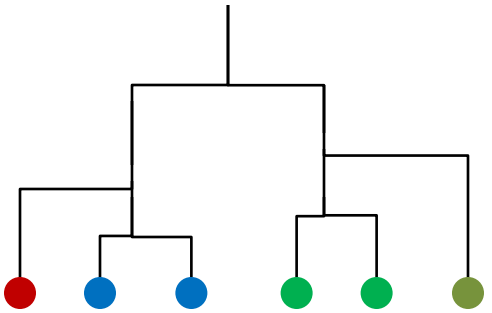
Example: Hierarchical clustering



Data:

o … data point

x … centroid



Dendrogram

Clustering CF Algorithms

- **Clustering is an intermediate step**
- Resulting clusters used for further analysis or processing
  - For classification and other tasks
  - Example: **partition data into clusters; then use memory-based CF algorithm like Pearson correlation to make predictions within each cluster**
- Clustering algorithms have **better scalability than typical CF methods** because they **make predictions on smaller clusters rather than whole customer base**
- **Complex and expensive clustering computation run offline**
- **Recommendation quality is generally low**
- Optimal clustering over large data sets is impractical
  - Most applications use greedy cluster generation techniques

REGRESSION-BASED CF

Regression-based CF Algorithms

- Numerical ratings are common in real recommender systems
- **Regression methods: good at making predictions for numerical values**
- **Uses an approximation of the ratings to make predictions based on a regression model**

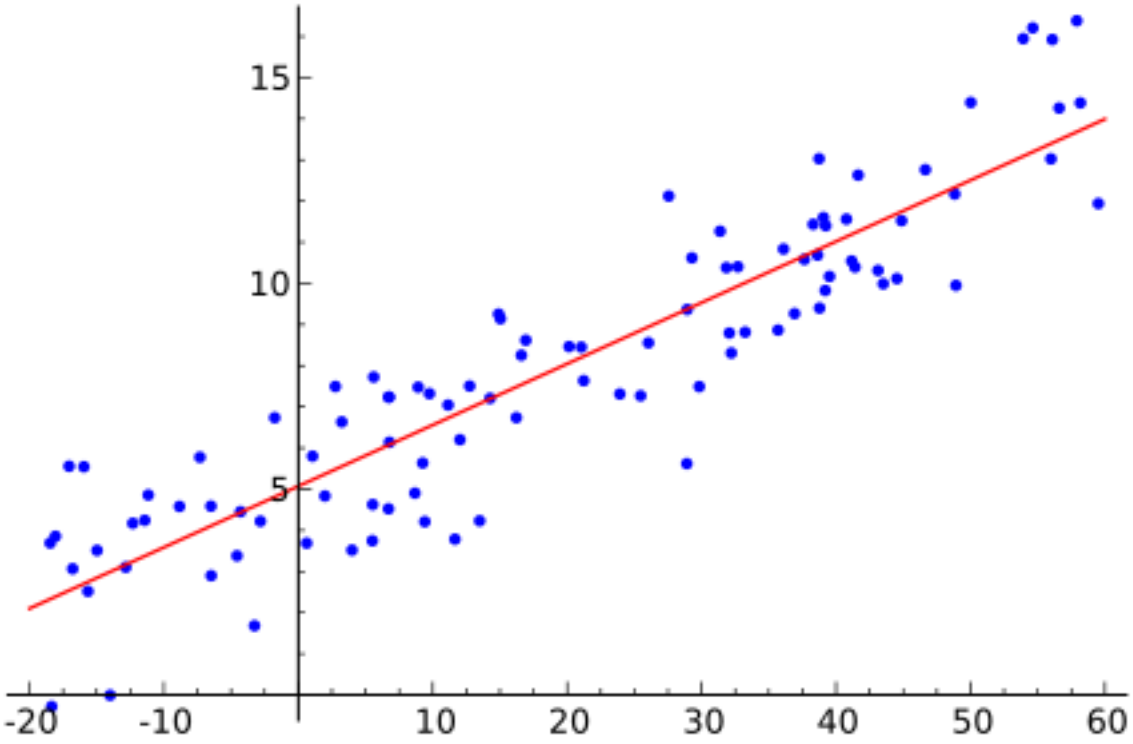
Linear Regression

- Data are modeled using linear predictor functions, and unknown model parameters are estimated from the data
- Such models are called *linear models*
- If the goal is prediction, forecasting, or reduction, linear regression can be used to **fit a predictive model to an observed data set of *y* and *X* values**
- After developing such a model, if an additional value of *X* is then given without its accompanying value of *y*
  - the fitted model can be used to **make a prediction of the value of *y***

Regression method

- **Let X = (X1, X2, …, XN) be a random variable representing user's preference on different items**
- Linear regression method:
  - **Y = MX + N**
  - Where **M is an n x k matrix**
  - **N = (N_1, …., N_n) is a random variable representing noise** in user choices
  - **Y is an n x m matrix where Y_{ij} is rating of user i on item** j (typically very sparse)
  - **X is a k x m matrix with each column as estimate of the value of the random variable X** (user's rating in k-dimensional rating space for one user)

Example: Linear Regression



Source: https://en.wikipedia.org/wiki/Linear_regression

CHARACTERISTICS AND CHALLENGES OF COLLABORATIVE FILTERING

Challenges of CF

- Data Sparsity
- Cold Start Problem
- Synonyms
- Scalability
- Gray Sheep and Black Sheep
- Shilling attacks

Characteristics and Challenges of Collaborative Filtering

- **Data Sparsity**
  - Many commercial recommender systems are used with very large product sets
  - Most users do not rate most items: User-item matrix is extremely sparse
  - For CF: reduces probability of finding set of users with similar ratings
- **Approaches:**
  - **Dimensionality reduction techniques**
    - **Singular Value Decomposition (SVD):** remove unrepresentative or insignificant users or items to reduce size of user-item matrix
    - **Latent semantic Indexing:** similarity between users is determined by representation of users in reduced space
    - **Principle Component Analysis**

Characteristics and Challenges of Collaborative Filtering

- **Data Sparsity (cont.)**
  - **Dimensionality reduction techniques**
  - **When users or items are discarded:**
    - **Useful information for recommendations may be lost**
    - **Recommendation quality may be degraded**

Challenges (cont.)

- **Cold start problem**
  - **When a new user or item has just entered the system**
    - Hard to find similarities: not enough information to make good recommendations
  - **New item problem:** can't be recommended until some users rate it
    - Also applies to obscure items
    - Also **called "first-rater problem"**
  - **New users**: not given good recommendations because of lack of rating or purchase history
- **Approaches:**
  - **Content-based systems** do not rely on ratings from other users
  - **Hybrid CF (content-boosted CF)**: external content information can be used to produce predictions for new users or new items
  - Research on **effectively selecting items to be rated by a user to rapidly improve recommendation performance**

Amazon Vine

How To Become An Amazon Vine Reviewer & Get Free Stuff

Written by Bakari Chavanu
August 4, 2010

If you're an Amazon.com customer who regularly reads product reviews by other customers, you may have noticed that some reviewers have a little badge (one of several) under their name identifying them as a ï»¿"Vine Voice" reviewer.

These Amazon Vine reviewers were invited by Amazon to receive a monthly newsletter of pre- and recently-released books and other products that Vine members can select from and review. Selected products are sent to Amazon Vine reviewers for free. So how do you become a Vine reviewer?

I'm not a prolific reviewer on Amazon, but about six months ago I was invited to become a member of its "exclusive club of influential Amazon voices."

Getting free items such as a 1 TB Western Digital External Hard Drive, a Duracell iPhone charger, a very expensive photography bag, computer software, and numerous books in exchange for honest reviews of these products has been a pretty good deal in my view.

Challenges (cont.)

- **Synonyms**
  - **Same or very similar items that have different names or entries**
  - Most recommender systems are unable to discover this latent association
  - Treat these products differently
  - **Synonyms decrease recommendation performance of CF systems**
- Approaches
  - **Automatic term expansion** or construction of thesaurus
    - Some added terms may have different meanings than intended
  - **SVD techniques: Latent Semantic Indexing (LSI)**: construct a semantic space where terms and documents that are closely associated are placed close to each other

Example: Latent Semantic Indexing (LSI)

Searches related to usc

usc **next coach**

university of southern california notable alumni

usc **address**

usc **clothing**

usc **tuition**

usc **ranking**

usc **jobs**

usc **athletics**

Searches related to ucla

ucla **vs stanford predictions**

ucla **football**

ucla **requirements**

ucla **tuition**

ucla **ranking**

ucla **admissions**

ucla **majors**

ucla **medical center**

Challenges (cont.)

- **Scalability**
  - **Traditional CF systems suffer scalability problems at very large scale**
  - With tens of millions of customers (M), millions of catalog items (N): and ***O(n)* algorithm is too large**
- Approaches
  - **Dimensionality reduction (SVD)** can scale and quickly produce good recommendations
    - But have to do expensive matrix factorization
  - Memory-based CF algorithms (e.g., **item-based Pearson correlation CF algorithm**) have good scalability
    - Instead of calculating similarities between all pairs of items, **calculate similarity only between pairs of co-rated items by a user**

Challenges (cont.)

- **Gray Sheep**
  - Users whose opinions **do not consistently agree or disagree with any group of people**
  - Do not benefit from collaborative filtering
- Approaches:
  - **Hybrid approach combining content-based and CF recommendations**
  - Base prediction on **weighted average of content-based prediction and CF prediction**
- **Black sheep**
  - **Idiosyncratic tastes make recommendations nearly impossible: considered an acceptable failure**

Challenges (cont.)

- **Shilling attacks**
- Shill definition
  - **Noun:** an accomplice of a hawker, gambler, or swindler who acts as an enthusiastic customer to entice or encourage others
  - **Verb:** act or work as a shill
- **In systems where anyone can provide ratings (Yelp, Amazon):**
  - People may give many positive ratings for their own materials
  - Negative recommendations for competitors
- CF systems want to discourage this phenomenon
- Approaches (research)
  - **Item-based less affected than user-based**
  - **Hybrid CF systems provide partial solutions**

Pros/Cons of Collaborative Filtering

+ Works for any kind of item

No feature selection needed

- Cold Start:

Need enough users in the system to find a match

- Sparsity:

The user/ratings matrix is sparse

Hard to find users that have rated the same items

- First rater:

Cannot recommend an item that has not been previously rated

New items, Esoteric items

- Popularity bias:

Cannot recommend items to someone with unique taste

Tends to recommend popular items