

# **Collaborative Filtering (Cont'd)**

**Harnessing quality judgments of other  
users**

# Three Approaches to Recommendation Systems

## ◆ 1) Content-based

- Use characteristics of an item
- Recommend items that
  - have similar content to items user liked in the past
  - match pre-defined attributes of the user

## ◆ 2) Collaborative filtering

- Build a model from
    - a user's past behavior (e.g., items previously purchased or rated) and
    - similar decisions made by other users
  - Use the model to predict items that the user may like
  - Collaborative: suggestions made to a user utilizing information across the entire user base
1. Neighborhood-based or Memory-based approaches
    - User-based CF
    - **Item-based CF**
  2. Model-based approaches

## ◆ 3) Hybrid approaches.

# **RECOMMENDATION SYSTEMS**

## **COLLABORATIVE FILTERING**

### **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.

# Vector Cosine Similarity for two items

- ◆ Vector  $A = \{x_1, y_1\}$
- ◆ Vector  $B = \{x_2, y_2\}$
- ◆ Vector cosine similarity between A and B is:

$$w_{A,B} = \cos(\vec{A}, \vec{B}) = \frac{\vec{A} \bullet \vec{B}}{\|\vec{A}\| * \|\vec{B}\|} = \frac{x_1 x_2 + y_1 y_2}{\sqrt{x_1^2 + y_1^2} \sqrt{x_2^2 + y_2^2}}$$

# Pearson Correlation between items $i, j$ (Cont'd)

	1	2	...	$i$		$j$	...	$m-1$	$m$
1				$R$		?			
2				$R$		$R$			
$\vdots$									
$l$				$R$		$R$			
$\vdots$									
$n-1$				?		$R$			
$n$				$R$		$R$			

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.

## Recall: Pearson Correlation Coefficient

- ◆ Pearson correlation measures extent to which two variables linearly relate
- ◆ For users  $u, v$ : Pearson correlation is

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

where the  $i \in I$  summations are over the items that both the users  $u$  and  $v$  have rated and  $\bar{r}_u$  is the average rating of the co-rated items of the  $u$ th user.

➤ And  $r_{u,i}$  is rating of item  $i$  by user  $u$

- ◆ Note: When calculating these similarities, look only at the co-rated items.

# 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

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

where  $r_{u,i}$  is the rating of user  $u$  on item  $i$ ,  $\bar{r}_i$  is the average rating of the  $i$ th 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  $i$
- ◆  $\bar{r}_i$  is average rating of  $i^{th}$  item by those users.



# Make Item-Based Predictions Using a Simple Weighted Average

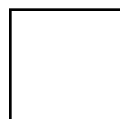
- ◆ Predict rating for user  $u$  on item  $i$
- ◆  $w_{i,n}$  is weight between items  $i$  and  $n$
- ◆  $r_{u,n}$  is rating for user  $u$  on item  $n$
- ◆ Summation over **neighborhood set  $N$  of items** rated by  $u$  **that are most similar to  $i$**

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

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 ( $|N|=2$ )

		users											
		1	2	3	4	5	6	7	8	9	10	11	12
movies	1	1		3			5			5		4	
	2			5	4			4			2	1	3
	3	2	4		1	2		3		4	3	5	
	4		2	4		5			4			2	
	5			4	3	4	2					2	5
	6	1		3		3			2			4	



- unknown rating



- rating between 1 to 5

# Item-Item CF ( $|N|=2$ )

		users											
		1	2	3	4	5	6	7	8	9	10	11	12
movies	1	1		3		?	5			5		4	
	2			5	4			4			2	1	3
	3	2	4		1	2		3		4	3	5	
	4		2	4		5			4			2	
	5			4	3	4	2					2	5
	6	1		3		3			2			4	



- estimate rating of movie 1 by user 5

# Item-Item CF ( $|N|=2$ )

First: what is similarity between items?

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

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

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

12

Neighborhood size is 2: pick movies 3 and 6

# Item-Item CF ( $|N|=2$ )

movies	users												Similarity (made up): $w_{1,j}$
	1	2	3	4	5	6	7	8	9	10	11	12	
	1		3		?	5			5		4		
	2		5	4			4			2	1	3	
	<u>3</u>	2	4		1	2	3		4	3	5		
	4		2	4		5		4			2		
	5			4	3	4	2				2	5	
	<u>6</u>	1		3		3		2			4		

Similarity weights:  
 $w_{1,3}=0.41$ ,  $w_{1,6}=0.59$

Here we use Pearson correlation as similarity:  
 1) Subtract mean rating  $m_i$  from each movie  $i$   
 $m_1 = (1+3+5+5+4)/5 = 3.6$   
 row 1: [-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]  
 2) Compute cosine similarities between rows

# Item-Item CF ( $|N|=2$ )

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

Predict by taking weighted average:

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

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

# 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 showed examples of co-rated items
  - We can use all ratings as well: based on all ratings:  $\text{avg}(r_1) = 10/3$ ,  $\text{avg}(r_2)^{15} = 8/3$

# 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:  $r_1 = 10/3$ ,  $r_2 = 8/3$
- ◆ Pearson correlation for  $w_{1,2}$ : *Similarity between items 1 and 2*
- ◆ Set U includes U1 and U4
- ◆  $w_{1,2} = ((r_{U1,I1} - 10/3)(r_{U1,I2} - 8/3) + (r_{U4,I1} - 10/3)(r_{U4,I2} - 8/3)) /$   
 $(\text{sqrt}((r_{U1,I1} - 10/3)^2 + (r_{U4,I1} - 10/3)^2) * \text{sqrt}((r_{U1,I2} - 8/3)^2 + (r_{U4,I2} - 8/3)^2))$
- ◆  $w_{1,2} = \frac{(2-10/3)(1-8/3) + (5-10/3)(3-8/3)}{\text{sqrt}((2-10/3)^2 + (5-10/3)^2) * \text{sqrt}((1-8/3)^2 + (3-8/3)^2)}$

$$= 2.778 / 3.628 = 0.765$$

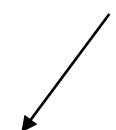


# Example: Item-to-Item Collaborative Filtering with Pearson Similarity

	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:  $r_1 = 7/2$ ,  $r_2 = 4/2$
- ◆ Pearson correlation for  $w_{1,2}$ : *Similarity between items 1 and 2*
- ◆ Set U includes U1 and U4
- ◆  $w_{1,2} = ((r_{U1,I1} - 7/2)(r_{U1,I2} - 4/2) + (r_{U4,I1} - 7/2)(r_{U4,I2} - 4/2)) /$   
 $(\text{sqrt}((r_{U1,I1} - 7/2)^2 + (r_{U4,I1} - 7/2)^2) * \text{sqrt}((r_{U1,I2} - 4/2)^2 + (r_{U4,I2} - 4/2)^2))$
- ◆  $w_{1,2} = \frac{(2-7/2)(1-4/2) + (5-7/2)(3-4/2)}{\text{sqrt}((2-7/2)^2 + (5-7/2)^2) * \text{sqrt}((1-4/2)^2 + (3-4/2)^2)}$

$= 3/3 = 1$

# Item-Based Prediction

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

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

- ◆ If we have the following item similarities:  $w_{2,1} = 0.5$ ,  $w_{2,3} = 0.2$ ,  $w_{2,4} = 0.3$
- ◆ 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,  $|N| = \text{items 1,4}$

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

# **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:**

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

- ◆ **Item-Based CF:**

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

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

- ◆ Reduces noise in the data

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

- ◆ Favors high weights
- ◆ 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. proposed imputation-boosted collaborative filtering (IBCF)
- ◆ First uses an imputation technique to fill in missing data
  - **imputation** is the process of replacing missing data with substituted values
- ◆ 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 naive 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

### Imputation Example

- 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.

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](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)

# **EVALUATING RECOMMENDER SYSTEMS**

# The Netflix Prize

## ◆ Training data

- 100 million ratings, 480,000 users, 17,770 movies
- 6 years of data: 2000-2005

## ◆ Test data

- Last few ratings of each user (2.8 million)
- **Evaluation criterion:** Root Mean Square Error (RMSE) =

$$\frac{1}{|R|} \sqrt{\sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2}$$

- **Netflix's system RMSE: 0.9514**

## ◆ Competition

- 2,700+ teams
- **\$1 million** prize for 10% improvement on Netflix

# The Netflix Utility Matrix $R$

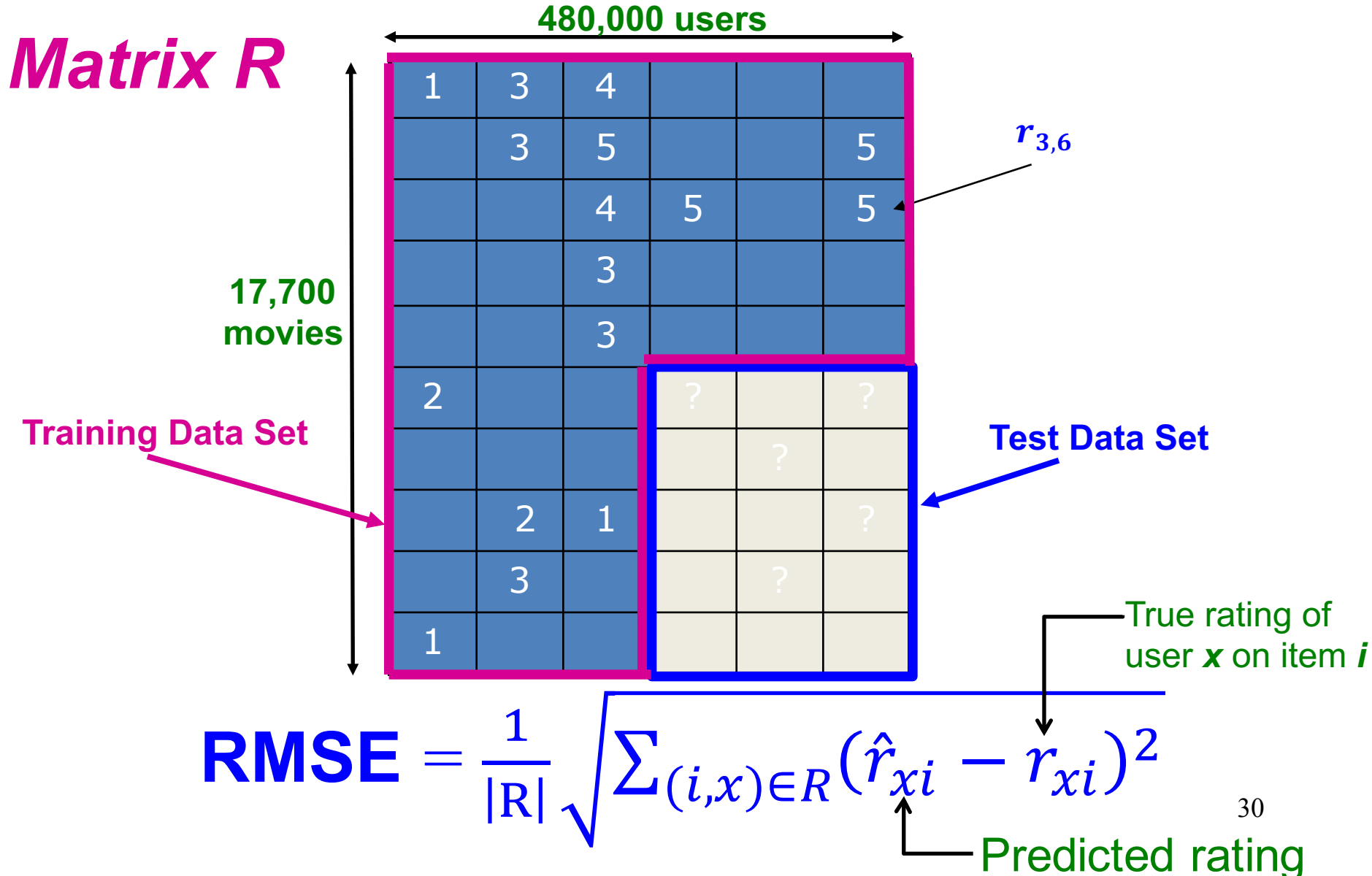
**Matrix  $R$**

480,000 users

17,700 movies

1	3	4			
	3	5			5
		4	5		5
		3			
		3			
2			2		2
				5	
	2	1			1
	3			3	
1					

# Utility Matrix $R$ : Evaluation



# Evaluation

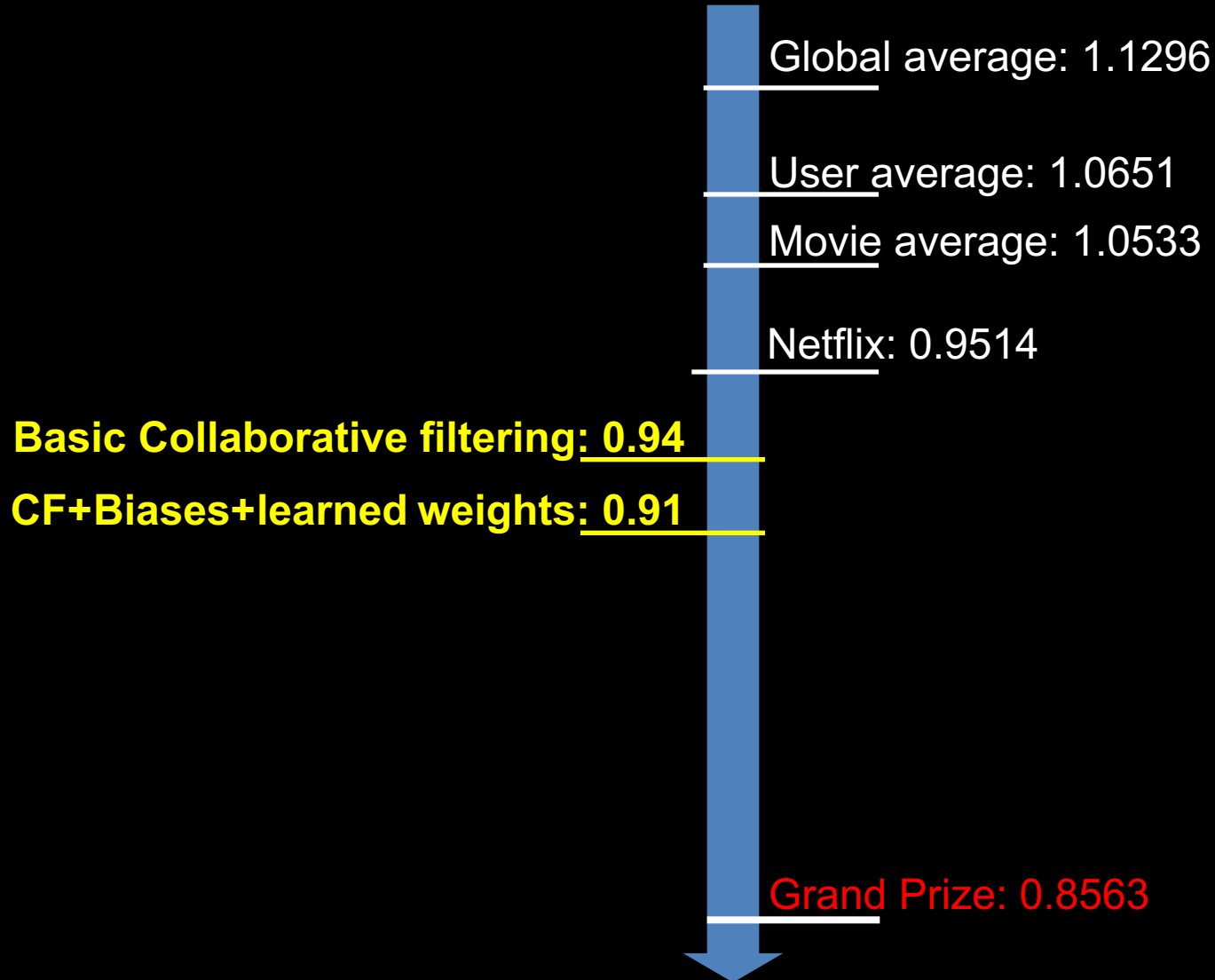
## ◆ Goal: Make good recommendations

- Quantify goodness using **RMSE**:  
**Lower RMSE  $\Rightarrow$  better recommendations**
- Want to make good recommendations on items that user has not yet seen. *Can't really do this!*

1	3	4			
	3	5			5
		4	5		5
		3			
		3			
2			2		2
				5	
	2	1			1
	3			3	
1					

# Performance of Various Methods

## Netflix prize





# Recall: 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 CF**

# Model-based Collaborative Filtering

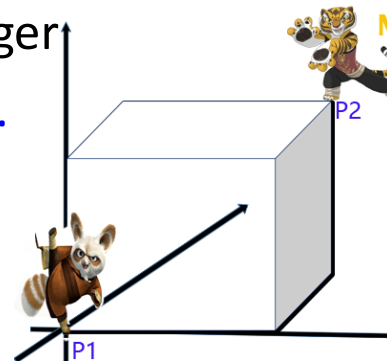
- Provide recommendations by estimating parameters of statistical models for user ratings
- Design and development of models can allow system to learn to recognize complex patterns
  - Based on training set – supervised learning
- Then make intelligent predictions for CF tasks based on the learned models
- 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**

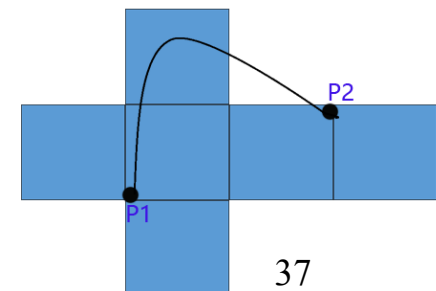
# Clustering CF Algorithms

- **Cluster = collection of data objects that are:**
  - Similar to one another within the same cluster
  - Dissimilar to objects in other clusters
- **Measurement of similarity between objects uses:**
  - 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  $X = (x_1, x_2, \dots, x_n)$ ,  $Y = (y_1, y_2, \dots, y_n)$
    - Where  $q$  is a positive integer
    - If  $q=2$ : Euclidean distance.

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



Minkowski Distance



# Clustering Algorithms

- Common clustering method
  - K-Means
  - Hierarchical Clustering
  - Mean-Shift
  - [More on Clustering in Chapter 7.](#)

# 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.

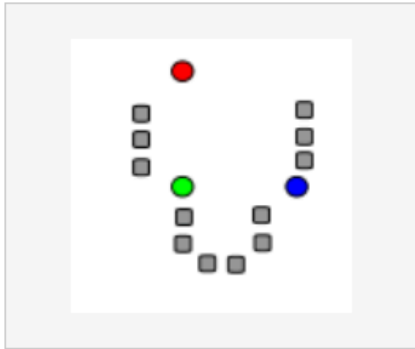
# ***K*-means Algorithm(s)**

- 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.

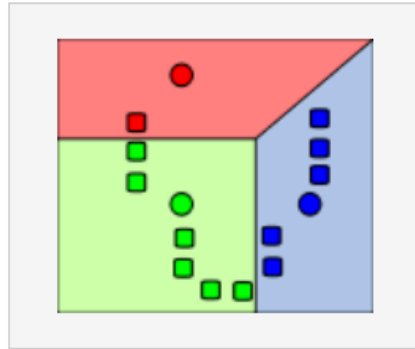


# Example: Standard K-Means in this case K-3

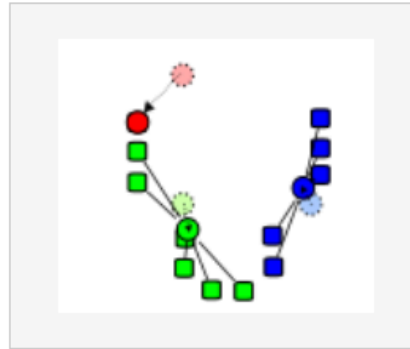
Demonstration of the standard algorithm



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.



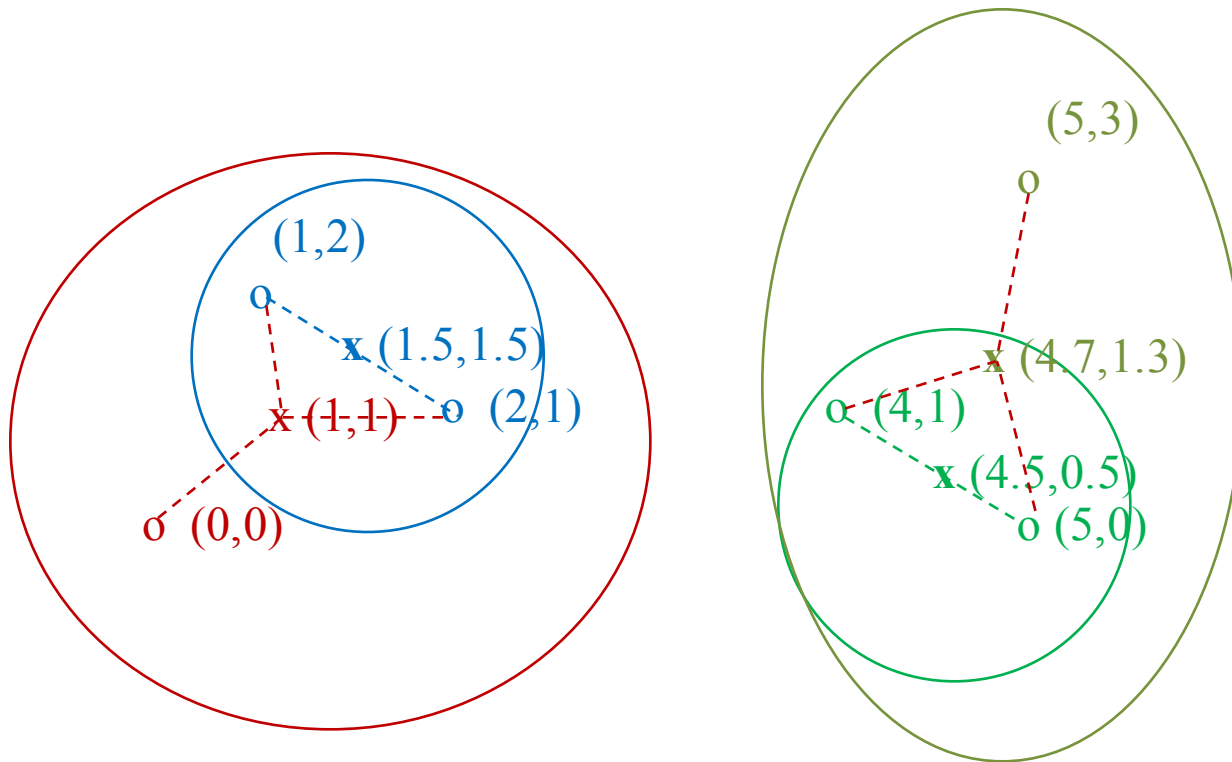
4. Steps 2 and 3 are repeated until convergence has been reached.

Source: [https://en.wikipedia.org/wiki/K-means\\_clustering](https://en.wikipedia.org/wiki/K-means_clustering)

# 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.

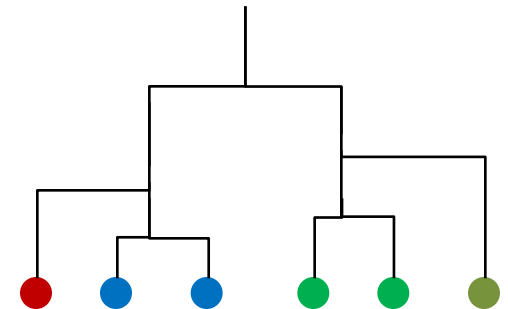
# Example: Hierarchical clustering



## Data:

o ... data point

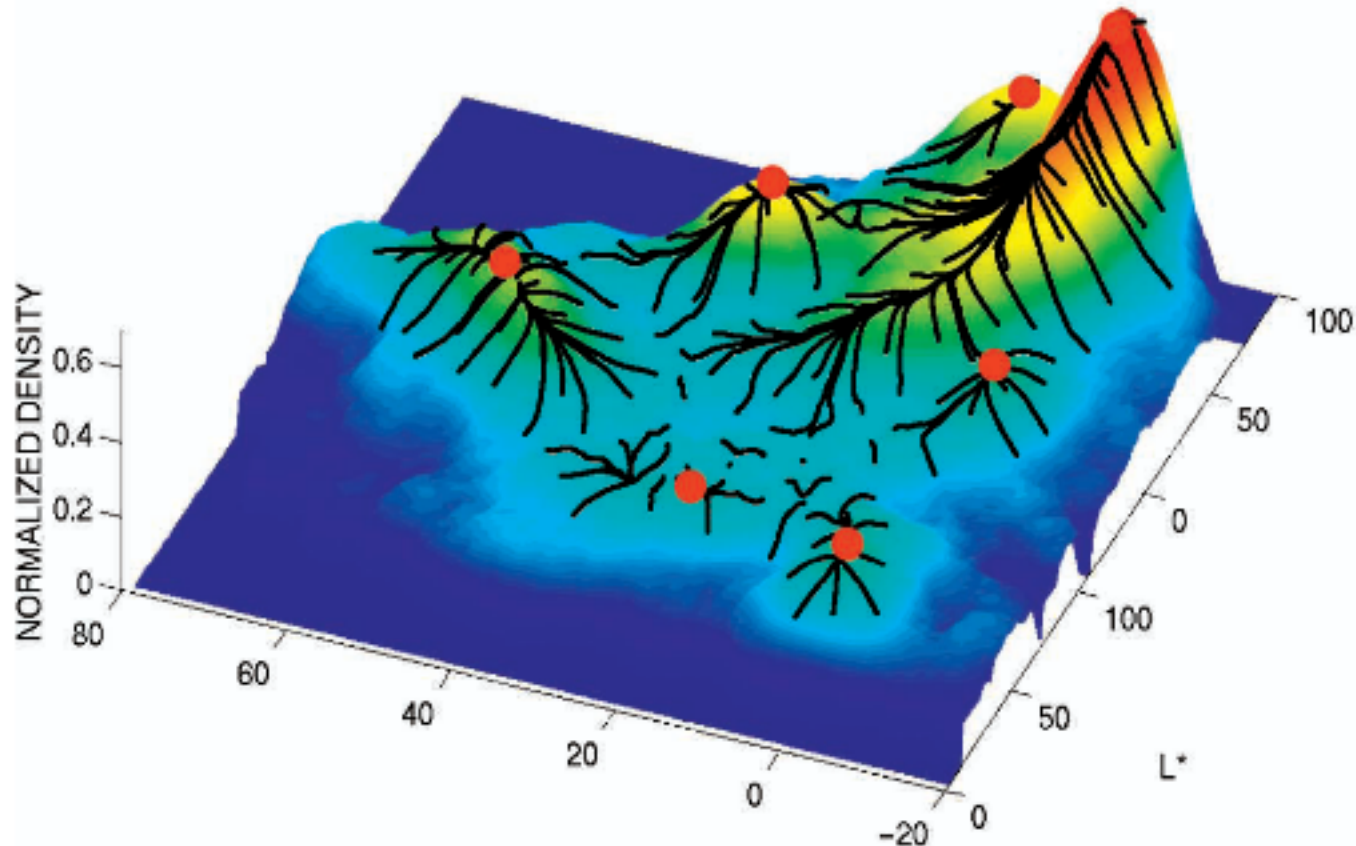
x ... centroid



Dendrogram<sup>43</sup>

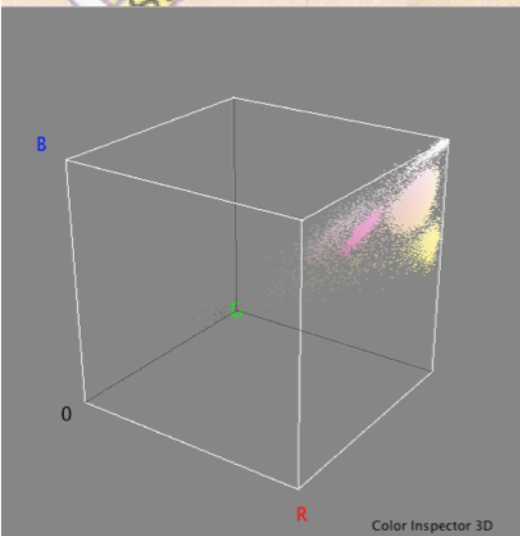
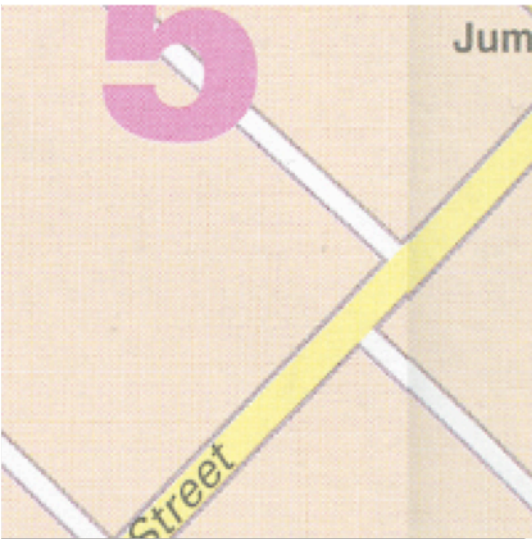
# Example: Mean-Shift clustering

Comaniciu and Meer 2002

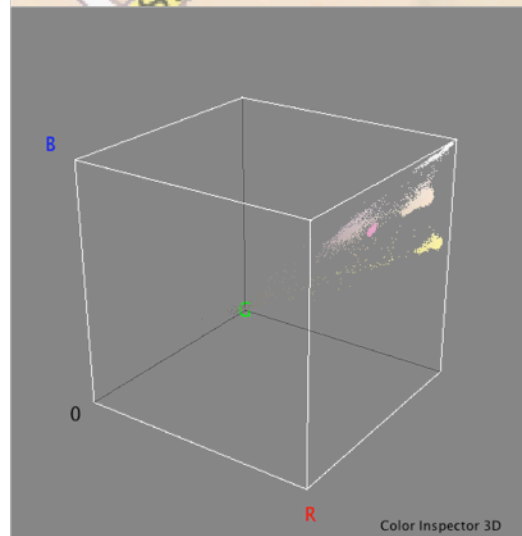
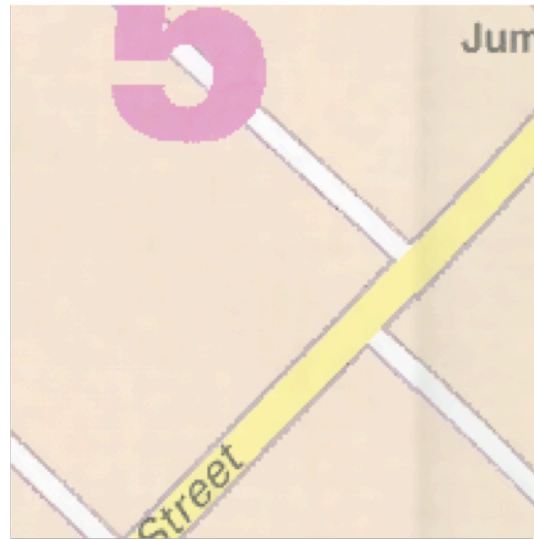


Mean shift is a powerful but computationally expensive method for nonparametric clustering and optimization. It iteratively moves each data point to its local mean until convergence.

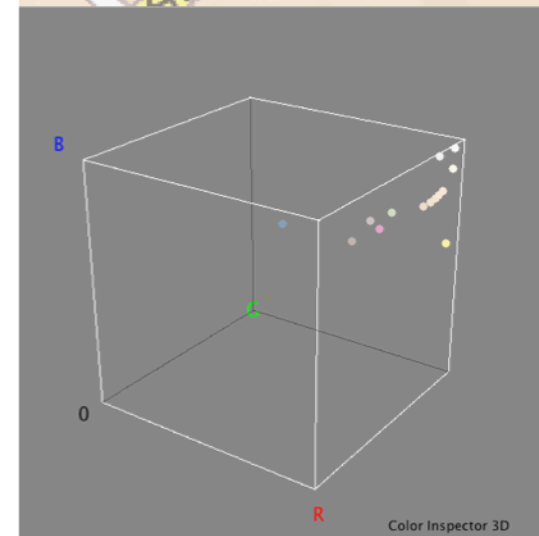
# Example: Hybrid Clustering



original



Mean-shift



Mean-shift->K-Means

# 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

- ◆ For memory-based CF algorithms: **in some cases, two rating vectors may have:**
  - large Euclidean distance
  - but have very high similarity using vector cosine or Pearson correlation measures - **noise**
- ◆ 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.**



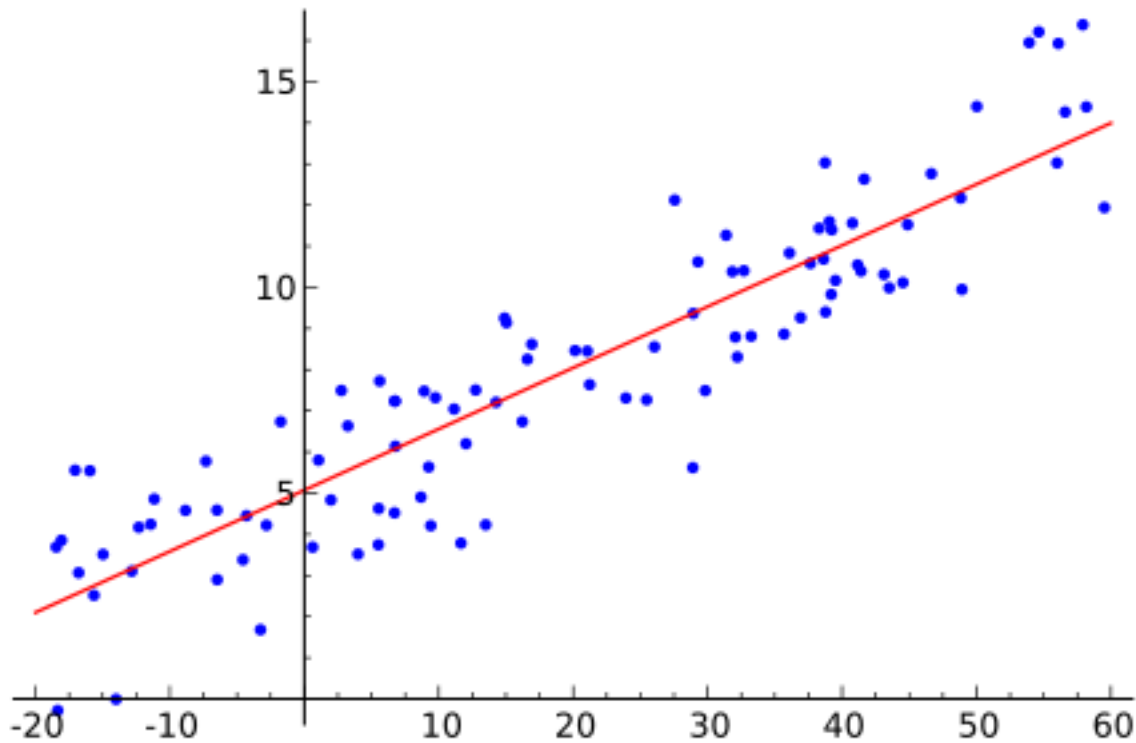
# Linear Regression

- ◆ 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  $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N)$  be a random variable representing user's preference on different items
- Linear regression method:
  - $\mathbf{Y} = \mathbf{\Lambda}\mathbf{X} + \mathbf{N}$
  - Where  $\mathbf{\Lambda}$  is an  $n \times k$  matrix
  - $\mathbf{N} = (\mathbf{N}_1, \dots, \mathbf{N}_n)$  is a random variable representing **Noise** in user choices
  - $\mathbf{Y}$  is an  $n \times m$  matrix where  $Y_{ij}$  is **rating of user i on item j** (typically very sparse)
  - $\mathbf{X}$  is a  $k \times m$  matrix with each column as estimate of the value of the random variable  $\mathbf{X}$  (user's rating in  $k$ -dimensional rating space for one user).

# Example: Linear Regression



Source: [https://en.wikipedia.org/wiki/Linear\\_regression](https://en.wikipedia.org/wiki/Linear_regression)

# **CHARACTERISTICS AND CHALLENGES OF COLLABORATIVE FILTERING**

# 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

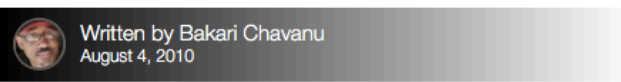
- **Data Sparsity (cont.)**
  - **Dimensionality reduction techniques**
  - **When users or items are discarded:**
    - **Useful information for recommendations may be lost**
    - **Recommendation quality may be degraded.**

# 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



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 "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.

amazon  
Try Prime

All ▾

Shop by  
Department ▾

Your Amazon.com

Today's Deals

Gift Cards

Sell

### What is Amazon Vine?

Amazon Vine invites the most trusted reviewers on Amazon to post opinions rank, which is a reflection of the quality and helpfulness of their reviews as independent opinions of the Vine Voices. The vendor cannot influence, modify Customer review from the Amazon Vine Program.

### Why do you have the Amazon Vine program?

The program was created to provide customers with more information includ

### How can I join the program?

Amazon Vine is an invitation-only program. Vine Voices are selected based on in the program. Customers who consistently write helpful reviews and develo

### How are Vine Voices selected?

We want the Voice program to reflect the best of our growing body of custom their reviews. factoring in the number of reviews they have written. More we



# 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.**

## Example SVD

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ u_{31} & u_{32} \\ u_{41} & u_{42} \\ u_{51} & u_{52} \end{bmatrix} \times \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \\ v_{21} & v_{22} & v_{23} & v_{24} & v_{25} \end{bmatrix}$$

Figure 9.9: UV-decomposition of matrix  $M$

# 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**
- Weights are determined on a per-user basis
- System determines optimal mix of content and CF-based recommendations

## ◆ 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.

# **Recommendation Systems**

## **Hybrid approaches**



# Three Approaches to Recommendation Systems

## ◆ 1) Content-based

- Use characteristics of an item
- Recommend items that have similar content to items user liked in the past
- Or items that match pre-defined attributes of the user

## ◆ 2) Collaborative filtering

- Build a model from a user's past behavior (items previously purchased or rated) and similar decisions made by other users
- Use the model to predict items that the user may like
- Collaborative: suggestions made to a user utilize information across the entire user base

## ◆ 3) Hybrid approaches

# **HYBRID RECOMMENDER SYSTEMS**

# Hybrid Recommender Systems

- **Implement two or more different recommenders and combine predictions**
  - Perhaps using a linear model
- **Example: Add content-based methods to collaborative filtering**
  - Item profiles for **new item problem**
  - Demographics to deal with **new user problem**
- **Additional reading: “Hybrid Web Recommender Systems” by Robin Burke**
  - Burke, R. (2007). Hybrid web recommender systems. In *The adaptive web* (pp. 377-408). Springer Berlin Heidelberg.
  - Some slides adapted from PPT by Jae-wook Ahn.

# Hybrid Recommender Systems

- **Mix of recommender systems**
- **Recommender system classification based on its knowledge source**
  - **Collaborative Filtering (CF)**
    - User ratings only
  - **Content-based (CN)**
    - Product features, user ratings
    - Classifications of users' likes/dislikes
  - **Demographic**
    - User ratings, user demographics
  - **Knowledge-based (KB)**
    - Domain knowledge, product features, user's need/query
    - Inferences about a user's needs and preferences.

# Netflix Example

- Before green-lighting *House of Cards*, Netflix knew:
  - A lot of users watched the David Fincher directed movie *The Social Network* from beginning to end
  - The British version of “*House of Cards*” has been well watched
  - Those who watched the British version “*House of Cards*” also watched Kevin Spacey films and/or films directed by David Fincher.
- Netflix Used Big Data To Identify The Movies That Are Too Scary To Finish
  - A typical Netflix customer will lose interest in 60 to 90 seconds when choosing something to watch
  - 80% of the content we watch on Netflix is influenced by the company’s recommendation system.

# Strategies for Hybrid Recommendation

- **Combination of multiple recommendation techniques together for producing output**
- Different techniques of *different types*
  - Most common implementations
  - Most promise to resolve cold-start problem
- Different techniques of the *same type*
  - Example: NewsDude – naïve Bayes + k-nearest neighbor.

# Seven Types of Hybrid Recommender Systems (Taxonomy by Burke, 2002)

1. **Weighted:** The score of different recommendation components are combined numerically
2. **Switching:** The system chooses among recommendation components and applies the selected one.
3. **Mixed:** Recommendations from different recommenders are presented together.
4. **Feature Combination:** Features derived from different knowledge sources are combined together and given to a single recommendation algorithm.
5. **Feature Augmentation:** One recommendation technique is used to compute a feature or set of features, which is then part of the input to the next technique.
6. **Cascade:** Recommenders are given strict priority, with the lower priority ones breaking ties in the scoring of the higher ones.
7. **Meta-level:** One recommendation technique is applied and produces some sort of model, which is then the input used by the next technique.

# **1. WEIGHTED HYBRID**



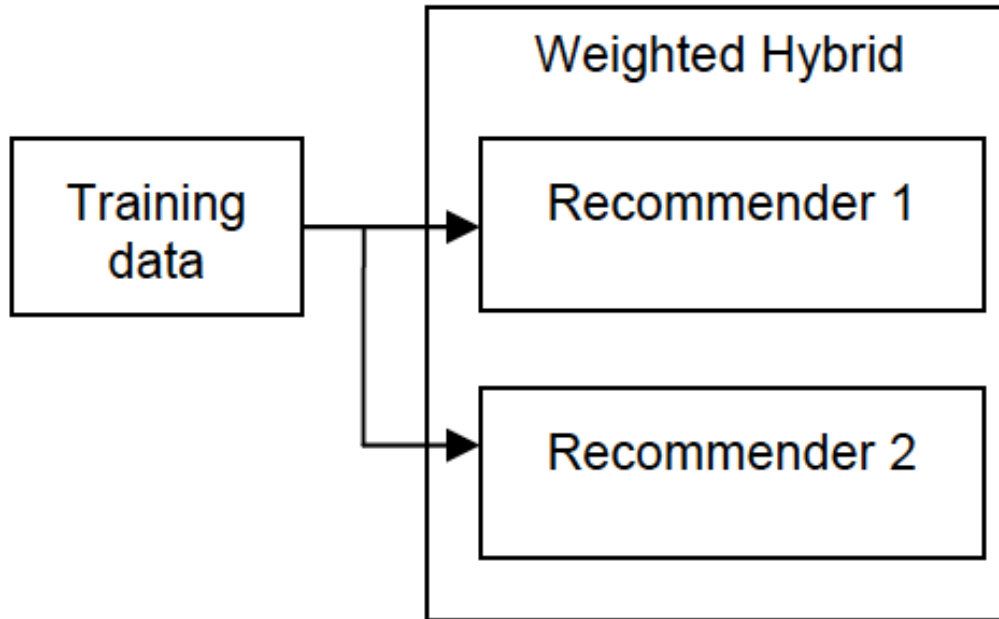
# 1. Weighted Hybrid

- Each component of the hybrid scores a given item and the *scores are combined* using a *linear formula*
- Combines evidence from both recommenders in a **static manner** (weighting doesn't change)
- Appropriate when component recommenders have consistent relative accuracy across the product space
- The movie recommender system in [32] has two components: one, using collaborative techniques, identifies **similarities between rating profiles and makes predictions** based on this information,
- The second component uses simple semantic knowledge about the **features of movies**, compressed dimensionally via **latent semantic analysis**, and recommends movies that are **semantically similar to those the user likes**,
- The output of the two components is combined using a linear weighting scheme.

# 1. Weighted Hybrid

- ◆ **Training phase: each individual recommender processes the training data**
- ◆ Note: all hybrid algorithms include training phase

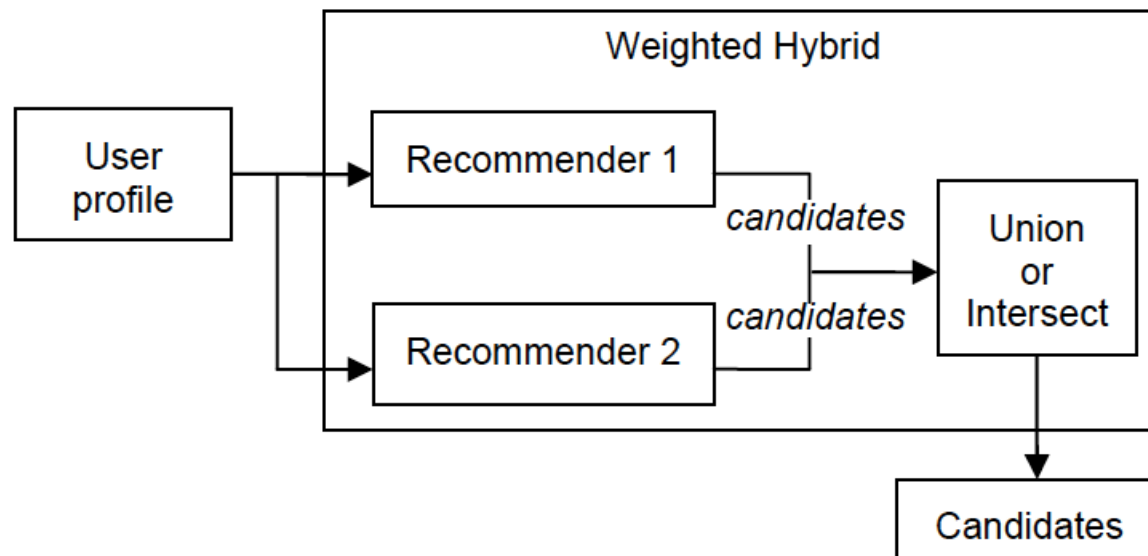
*Training phase*



# 1. Weighted Hybrid

- ◆ **Candidate generation: When a prediction is being generated for a test user, the recommenders jointly propose candidates**
  - **Some** recommendation techniques (e.g., content-based algorithms) make **predictions on any item**
  - **Others are limited** (e.g., collaborative filtering **can't make predictions** if there are **no peer users** who have rated the item)
  - **Candidate generation necessary to identify items that will be considered**

*Candidate generation*



# 1. Weighted Hybrid

- ◆ **The sets of candidates must then be rated jointly**

**Two approaches:**

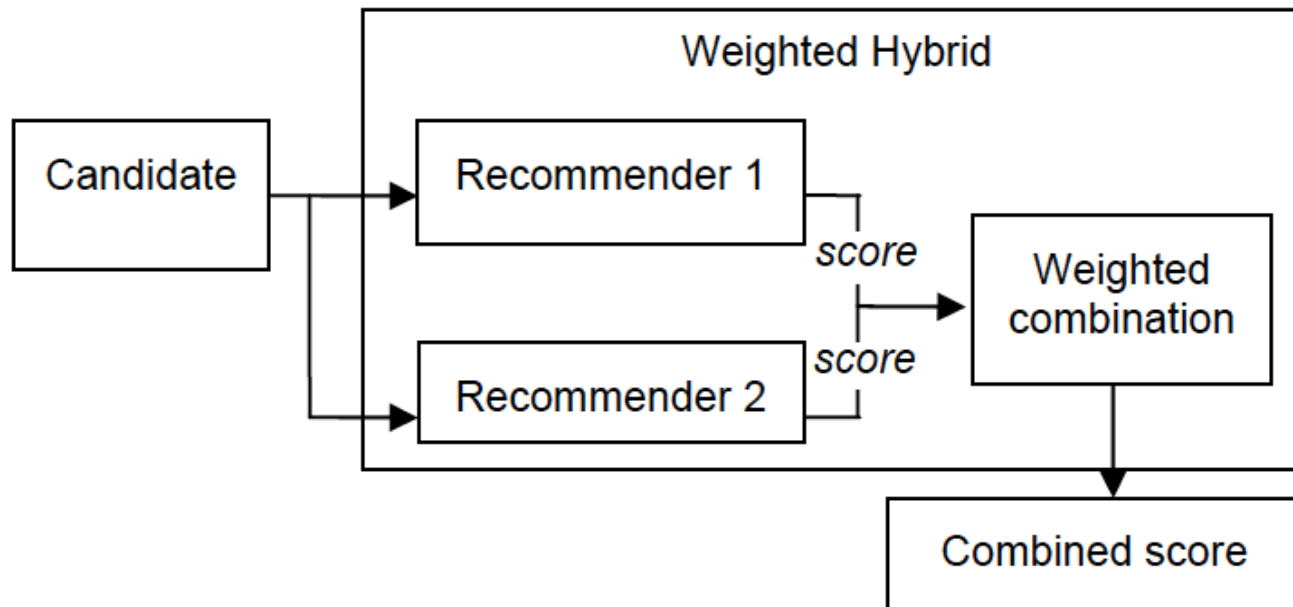
- ◆ **Intersection of candidate sets:** possible only a small number of candidates shared between candidate sets
- ◆ **Union of candidate sets:** must decide how to handle cases in which it is not possible for a recommender to rate a given candidate
  - May give such a candidate a neutral (neither liked nor disliked) score.

# 1. Weighted Hybrid

## Scoring:

- ◆ Each candidate is then rated by the two recommenders
- ◆ A linear combination of the two scores computed, which becomes the item's predicted rating

*Scoring*



# 1. Weighted Hybrid

## Weighted hybrid recommender:

- ◆ Burke classification discusses static, linear combination of weights
- ◆ More generally: Weights can be linear combination, use **weighted majority voting** or **weighted average voting**
- ◆ Example: P-Tango system
  - **Initially: CF and content-based recommenders have equal weight**
  - **Gradually adjusts weighting as predictions are confirmed or found incorrect.**

## **2. MIXED HYBRID**

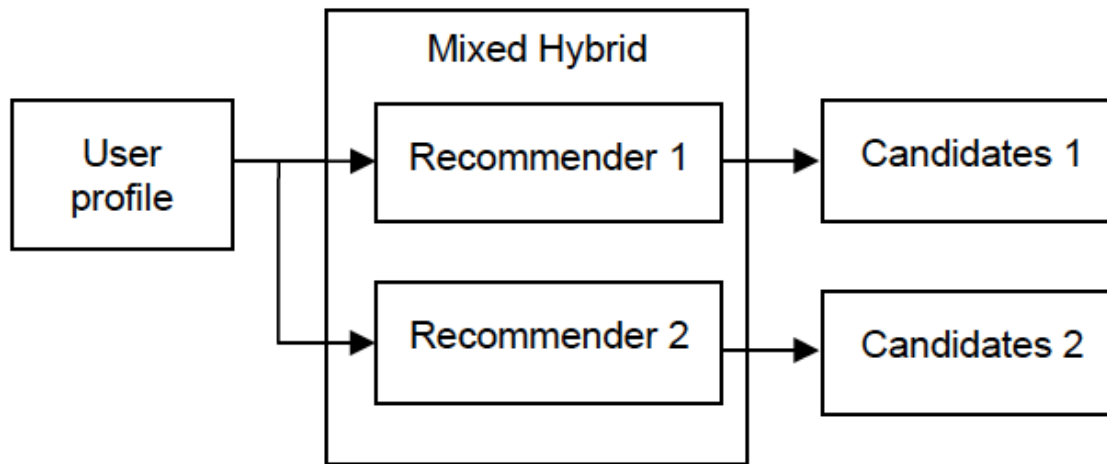
## 2. Mixed Hybrid

- ◆ A mixed hybrid **presents recommendations of its different components side-by-side** in a combined list
- ◆ There is **no attempt to combine evidence between recommenders**
- ◆ PTV recommends television shows [48]. It has both **content-based** and **collaborative components**, but because of the **sparsity** of the ratings and the content space, it is difficult to get both recommenders to produce a rating for any given show.
- ◆ Instead the components **each produce their own set of recommendations** that are combined before being shown to the user.

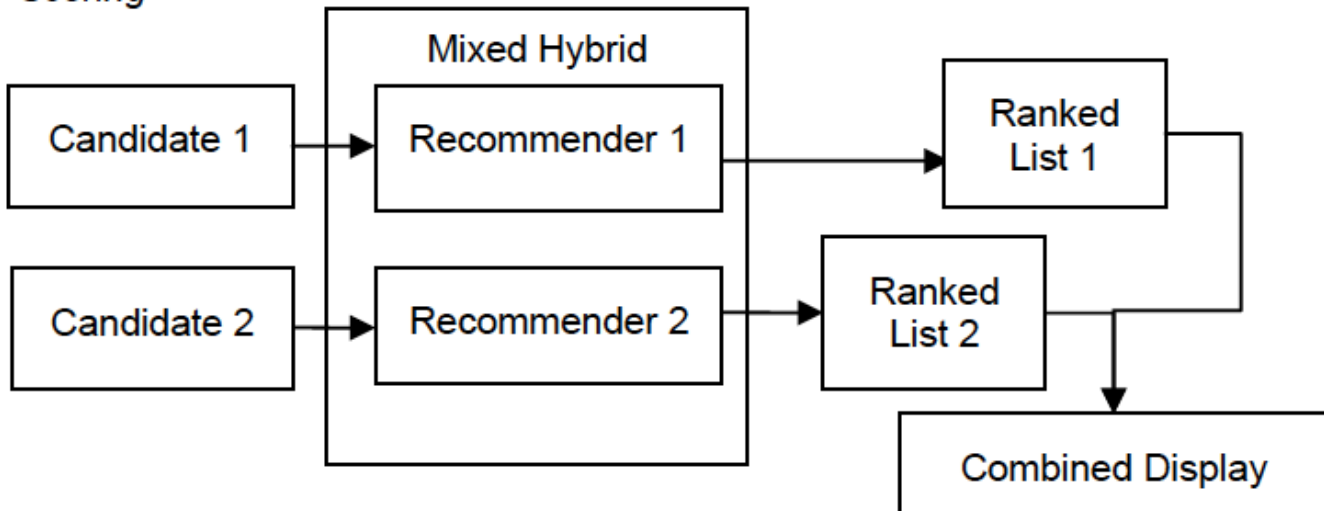


## 2. Mixed Hybrid

*Candidate generation*



*Scoring*



### **3. SWITCHING HYBRID**

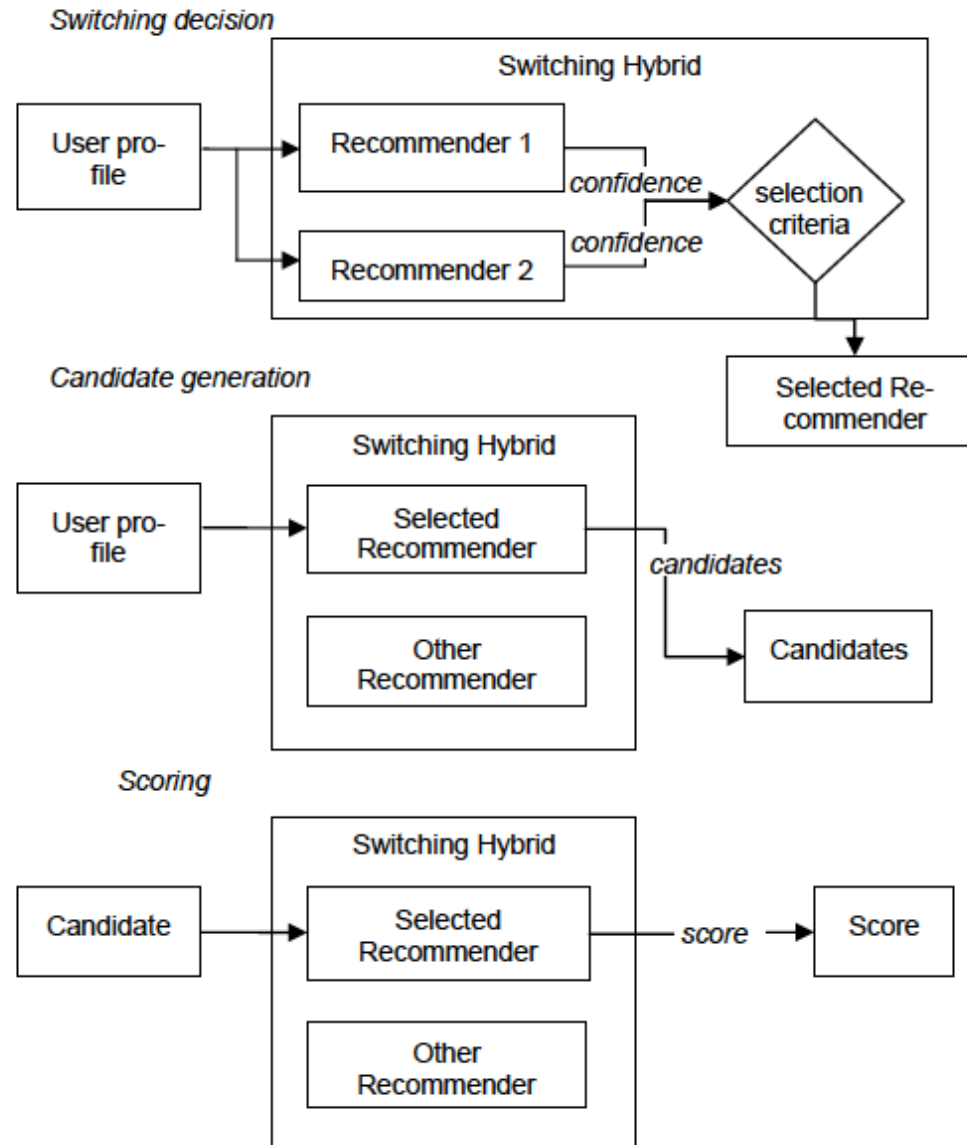
### 3. Switching Hybrid

- **Selects a single recommender from among its constituents based on the recommendation situation**
  - For a different profile, a different recommender might be chosen
- Takes into account that **components may not have consistent performance for all types of users**
- **Assumes that some reliable criterion is available on which to base the switching decision**
  - E.g., Confidence values inherent in the recommendation components.

# 3. Switching Hybrid

- ◆ NewsDude [4] recommends news stories.
- ◆ It has three recommendation components:
  - a content-based nearest-neighbor recommender,
  - a collaborative recommender and
  - a second content-based algorithm using a naive Bayes classifier.
  - The recommenders are ordered.
- The nearest neighbor technique is used first.
- If it cannot produce a recommendation **with high confidence**, then the collaborative recommender is tried, and so on, with the naive Bayes recommender at the end of line.

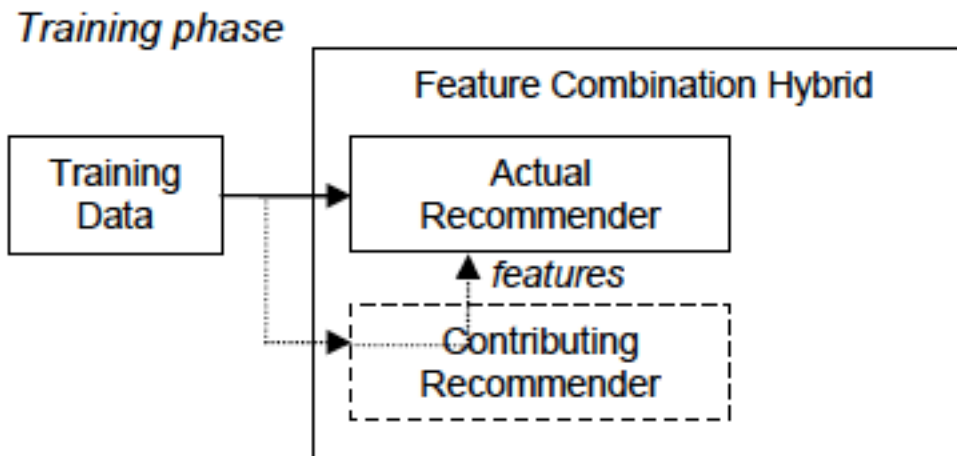
### 3. Switching Hybrid



## **4. FEATURE COMBINATION**

## 4. Feature Combination

- **Inject features of one source** (such as collaborative recommendation) **into an algorithm designed to process data with a different source** (such a content-based recommendation)
- **A virtual "contributing recommender"**
- Features would ordinarily be processed by one recommender are instead used as part of the input to the actual recommender
- **Hybrid** is the **knowledge sources** involved
- Borrows the recommendation logic from another technique rather than employing a separate component that implements it



## 4. Feature Combination

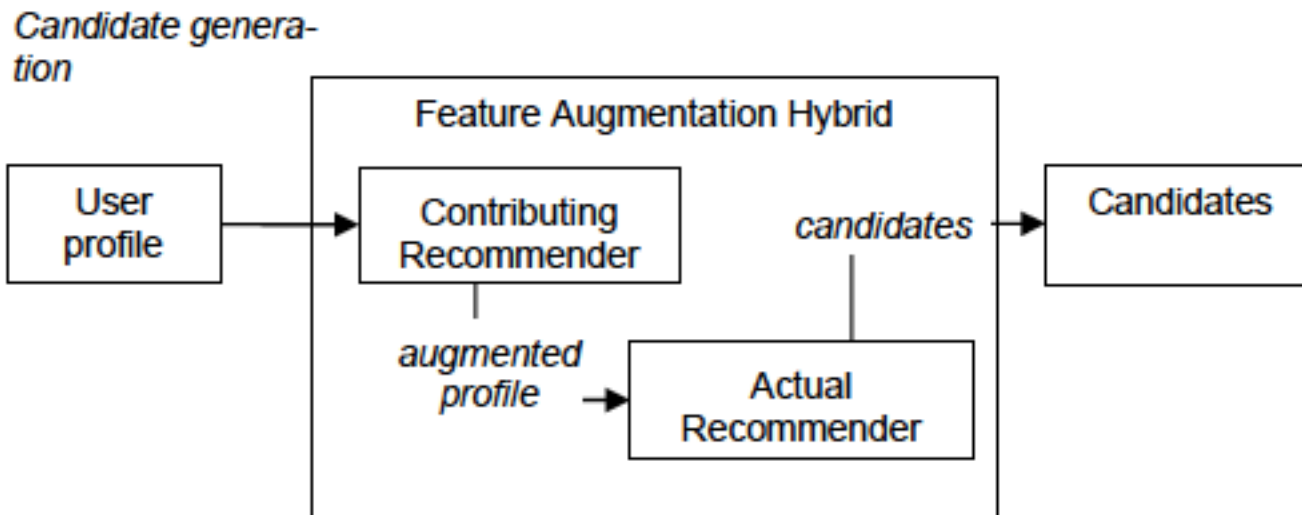
- ◆ Basu, Hirsh and Cohen [3] used the inductive rule learner Ripper [16] to learn content-based rules about user's likes and dislikes. They were able to improve the system's performance by adding collaborative features, thereby treating a fact like "User1 and User2 liked Movie X" in the same way that the algorithm treated features like "Actor1 and Actor2 starred in Movie X".
- ◆ **Content-based recommender** works in the typical way
- ◆ **Builds a learned model for each user (a user profile)**
- ◆ But **user rating data is combined with the product features**
- ◆ Content-based recommender **draws from a knowledge source associated with collaborative filtering recommendation.**



## **5. FEATURE AUGMENTATION**

## 5. Feature Augmentation

- ◆ Feature augmentation hybrid **generates a new feature for each item by using the recommendation logic of the contributing domain**
  - E.g. use **association rule mining** over the collaborative data to derive new content features for content-based recommendation
- ◆ At each step, the **contributing recommender intercepts the data headed for the actual recommender and augments it** with its own contribution
  - not raw features as in feature combination, but the result of some computation



## 5. Feature Augmentation

- ◆ Melville, Mooney and Nagarajan [30] coin the term "content-boosted collaborative filtering." This algorithm learns a content-based model over the training data and then uses this model to generate ratings for unrated items. This makes for a set of profiles that is denser and more useful to the collaborative stage of recommendation that does the actual recommending.
- ◆ Employed when there is:
  - a well-developed strong primary recommendation component
  - desire to add additional knowledge sources
- ◆ Augmentation can usually be done off-line.

# Content-Boosted CF Example

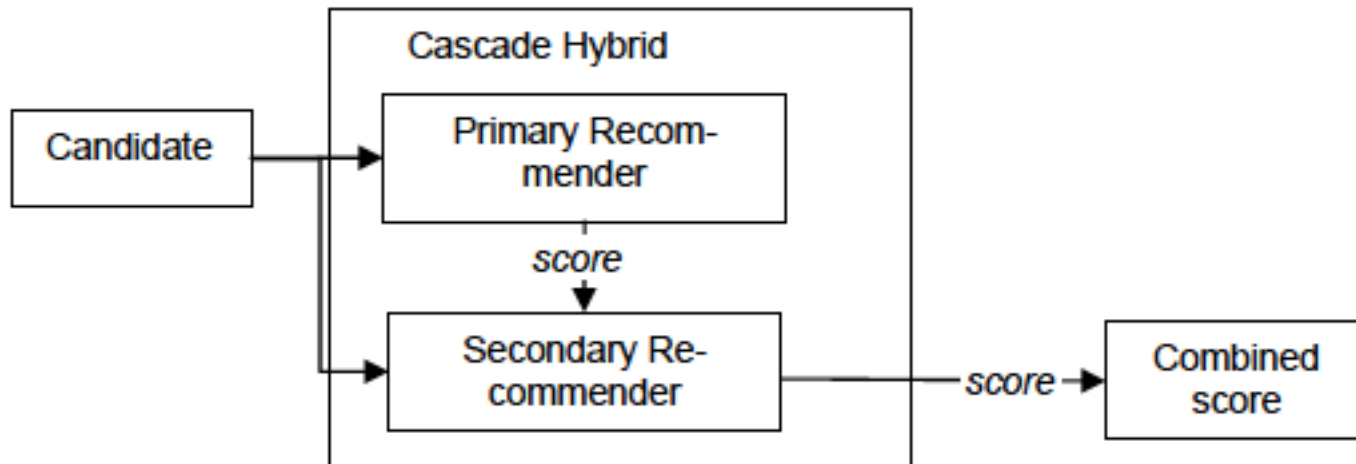
- ◆ Uses naïve Bayes as the content classifier
- ◆ Fills in missing values of rating matrix with predictions of the content predictor
  - Form a **pseudo rating matrix**: observed ratings unchanged, **missing ratings replaced by predictions of content predictor**
- ◆ Then **make predictions over the pseudo ratings matrix using weighted Pearson correlation based CF**
  - Give higher weight for item that more users rated
  - Gives higher weight for active user.

## **6. CASCADE HYBRID**

## 6. Cascade Hybrid

- ◆ Create a **strictly hierarchical hybrid**
- ◆ A **weak recommender cannot overturn decisions made by a stronger one**, but can merely **refine them**
- ◆ Order-dependence
- ◆ A cascade recommender uses a secondary recommender only to break ties in the scoring of the primary one

*Scoring*



## 6. Cascade Hybrid

- ◆ The knowledge-based Entree restaurant recommender [10] was found to **return too many equally-scored items**, which could not be ranked relative to each other. Rather than additional labor-intensive knowledge engineering (to produce finer discriminations), the hybrid EntreeC was created by **adding a collaborative re-ranking of only those items with equal scores**.
- ◆ This taxonomy is very strict
- ◆ **Real-world systems might have other refinements that are not exclusive (e.g., only breaking ties).**

## **7. META-LEVEL HYBRID**

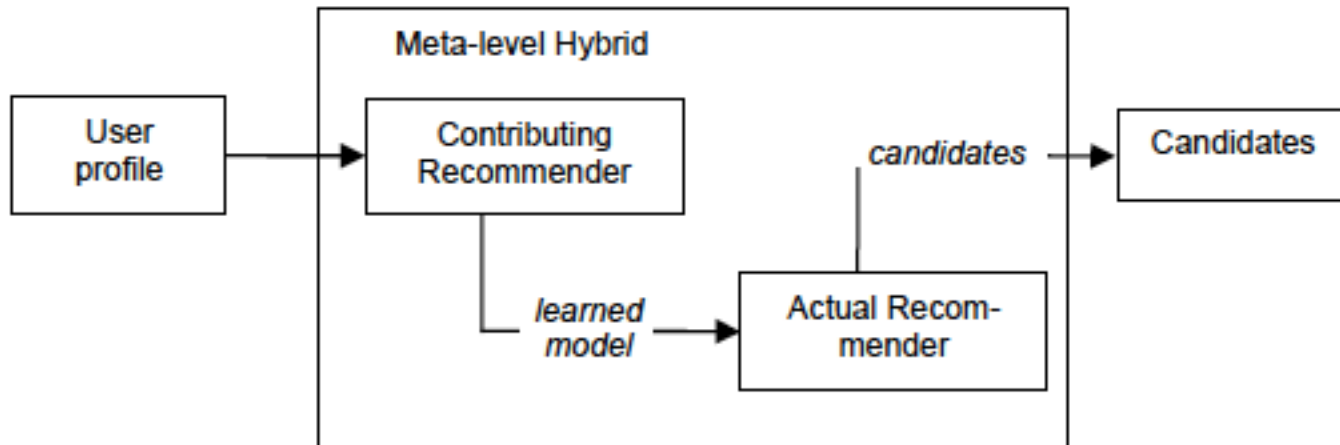


## 7. Meta-Level Hybrid

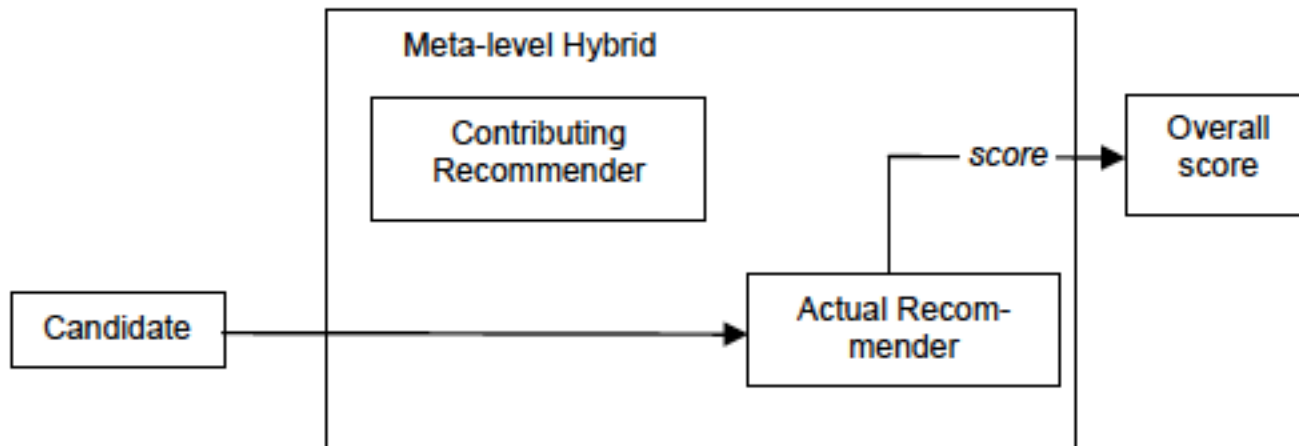
- ◆ **Uses a model learned by one recommender as input for another**
- ◆ Similar to the feature augmentation hybrid in that the **contributing recommender is providing input to the actual recommender**
- ◆ Difference: in meta-level hybrid, **contributing recommender completely replaces the original knowledge source with a learned model that the actual recommender uses**
  - Actual recommender does not work with any raw profile data
- ◆ Pazzani [36] used the term "collaboration through content" to refer to his restaurant recommender that used the naive Bayes technique to build models of user preferences in a content-based way. With each user so represented, a collaborative step was then be performed in which the vectors were compared and peer users identified.

## 7. Meta-level Hybrid

### *Candidate generation*



### *Scoring*



# Hybrid Recommendation Systems

## Personality diagnosis

- **Combines memory-based and model-based CF**
- Active user generated by choosing one of the other users uniformly at random, adding Gaussian noise to their ratings
  - \* *Gaussian noise equal to Normal distribution noise*
- **Given active user's known ratings, can calculate probability active user is same "personality type" as other users**
  - Predict probability they will like new items
- Can be regarded as a **clustering method** with one user per cluster
- Makes better predictions than:
  - Pearson correlation-based and vector similarity-based CF algorithms
  - Bayesian clustering and Bayesian networks.

# Summary: Seven Types of Hybrid Recommender Systems (Taxonomy by Burke, 2002)

1. **Weighted:** The score of different recommendation components are combined numerically
2. **Switching:** The system chooses among recommendation components and applies the selected one.
3. **Mixed:** Recommendations from different recommenders are presented together.
4. **Feature Combination:** Features derived from different knowledge sources are combined together and given to a single recommendation algorithm.
5. **Feature Augmentation:** One recommendation technique is used to compute a feature or set of features, which is then part of the input to the next technique.
6. **Cascade:** Recommenders are given strict priority, with the lower priority ones breaking ties in the scoring of the higher ones.
7. **Meta-level:** One recommendation technique is applied and produces some sort of model, which is then the input used by the next technique.

More on

# **LATENT FACTOR MODELS**

# Matrix Decomposition Techniques in Machine Learning and Information Retrieval



**Thomas Hofmann**

*Associate Professor  
Department of Computer Science  
Brown University*

th@cs.brown.edu  
www.cs.brown.edu/~th



# Latent Factor Models

## AIM3 – Scalable Data Analysis and Data Mining

11 – Latent factor models for Collaborative Filtering  
Sebastian Schelter, Christoph Boden, Volker Markl



Fachgebiet Datenbanksysteme und Informationsmanagement  
Technische Universität Berlin

<http://www.dima.tu-berlin.de/>

# Latent Structure



◆ Given a matrix that “encodes” data ...

◆ Potential problems

- too large
- too complicated
- missing entries
- noisy entries
- lack of structure
- ...

$$\mathbf{A} = \begin{pmatrix} a_{11} & \dots & a_{1j} & \dots & a_{1m} \\ \dots & \dots & \dots & \dots & \dots \\ a_{i1} & \dots & a_{ij} & \dots & a_{im} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & \dots & a_{nj} & \dots & a_{nm} \end{pmatrix}$$

◆ Is there a **simpler** way to **explain** entries?

◆ There might be a **latent structure** underlying the data.

◆ How can we “find” or “reveal” this structure?



# Latent Factor Models

## Idea

- ratings are deeply influenced by a set of **factors** that are very **specific to the domain** (e.g. amount of action in movies, complexity of characters)
- these factors are in general **not obvious**, we might be able to think of some of them but it's hard to estimate their impact on the ratings
- the goal is to infer those so called **latent factors** from the rating data by using mathematical techniques

# Matrix Decomposition

- ◆ Common approach: approximately **factorize** matrix

$$\mathbf{A} \approx \hat{\mathbf{A}} = \mathbf{L} \cdot \mathbf{R}$$

approximation      left factor      right factor

- ◆ Factors are typically constrained to be “**thin**”

$n$   $m$   $\mathbf{A}$   $\approx$   $n$   $q$   $\mathbf{L}$   $\cdot$   $q$   $m$   $\mathbf{R}$

reduction  
 $n \cdot m \gg n \cdot q + m \cdot q$   
factors = latent structure (?)

# Latent Factor Models

## ■ Approach

- users and items are characterized by **latent factors**, each user and item is mapped onto a **latent feature space**

$$u_i, m_j \in R^{n_f}$$

- each rating is approximated by the dot product of the **user feature vector** and the **item feature vector**

$$r_{ij} \approx m_j^T u_i$$

- **prediction of unknown ratings** also uses this dot product
- **squared error** as a measure of loss

$$(r_{ij} - m_j^T u_i)^2$$

# Latent Factor Models

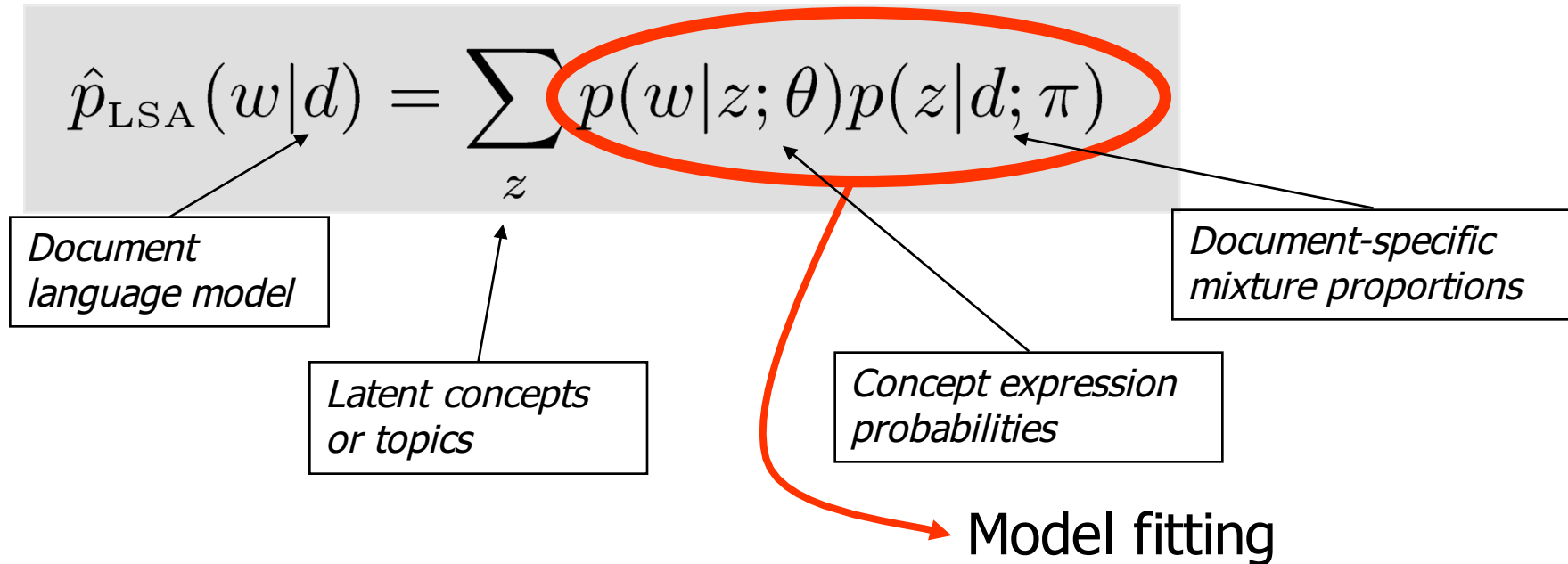
## ■ Approach

- **decomposition of the rating matrix** into the product of a user feature and an item feature matrix
- row in  $U$ : vector of a user's affinity to the features
- row in  $M$ : vector of an item's relation to the features
- closely related to **Singular Value Decomposition** which produces an optimal low-rank optimization of a matrix



# pLSA – Latent Variable Model

- ◆ Structural modeling assumption (**mixture** model)



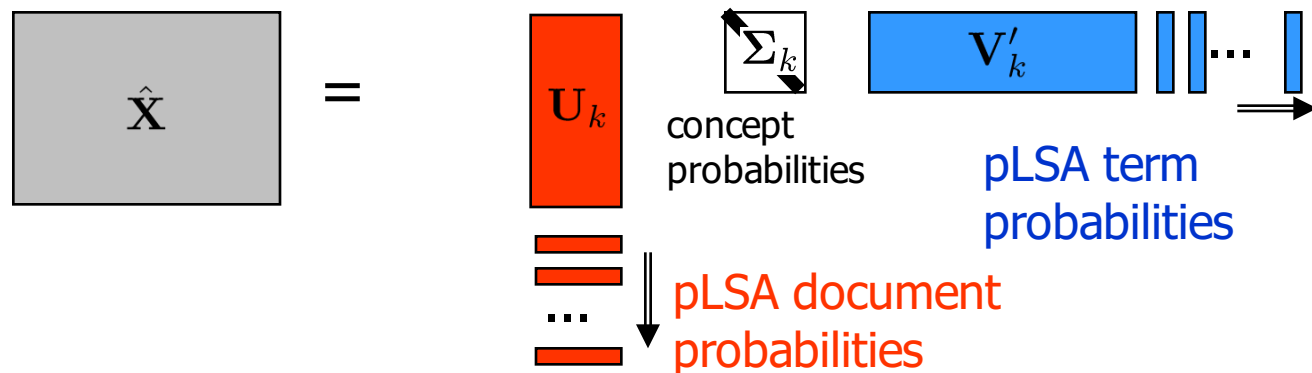
- ◆ \**Document language model*=A statistical **language model** is a probability distribution over sequences of words.
- ◆ [Hofmann, Proceedings ACM SIGIR, 1999]

# pLSA: Matrix Decomposition

- ◆ Mixture model can be written as a **matrix factorization**

- ◆ Equivalent symmetric (joint) model

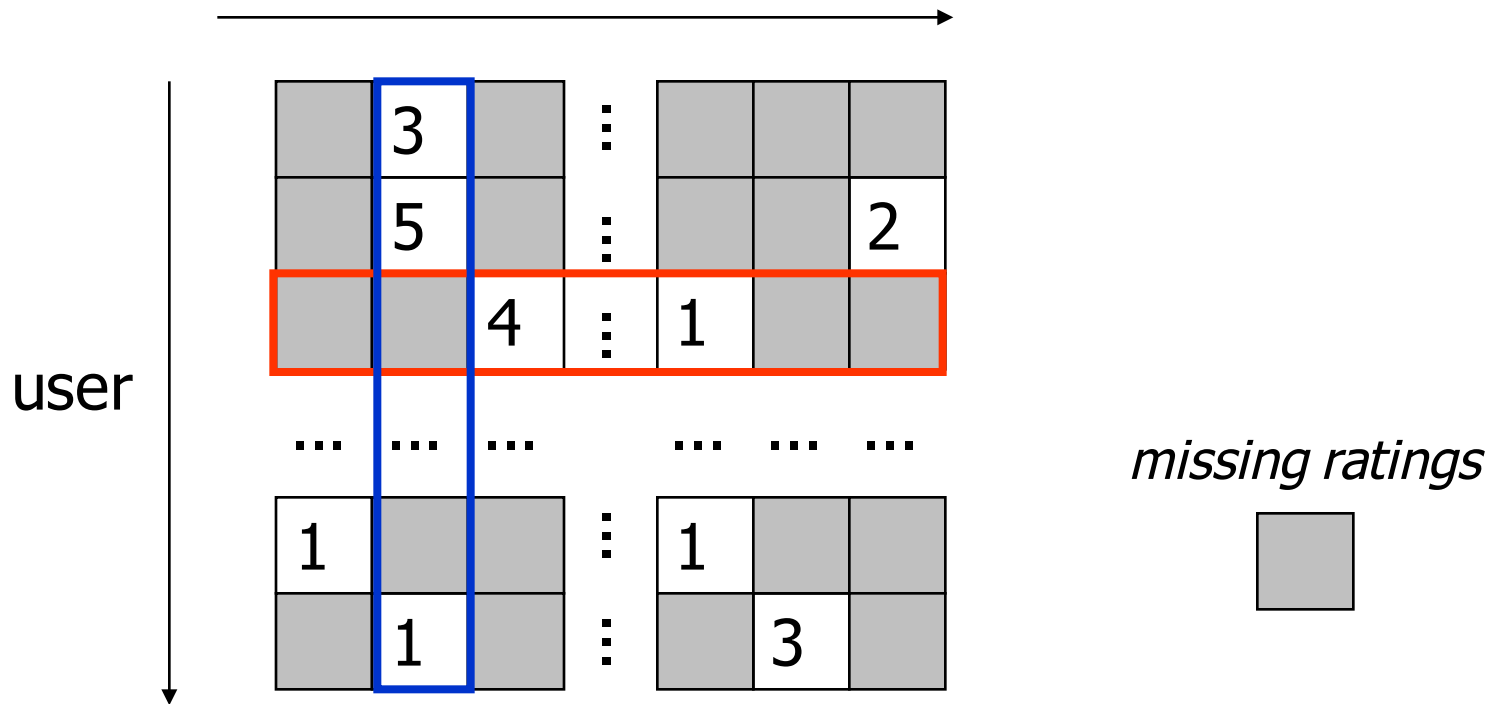
$$\hat{p}_{\text{LSA}}(d, w) = \sum_z p(d|z) p(z) p(w|z)$$



- ◆ Contrast to LSA/SVD: **non-negativity** and **normalization** (intimate relation to non-negative matrix factorization).

# Rating Matrix

- Rating matrix is typically a large matrix with many (mostly) **missing values**  
item



# pLSA-like Decomposition

- ◆ Generalization of pLSA (additional **rating variable**)

$$p_{\text{LSA}}(r, y|u) = \sum_z \underbrace{p(r|y, z; \rho)}_{\text{extension to predict ratings}} \underbrace{p(y|z; \theta)p(z|u; \pi)}_{\text{standard pLSA model to explain sparseness pattern}}$$

Explicit decomposition of user preferences (each user can have **multiple interests**)

- Probabilistic model can be used to **optimize** specific **objectives**
- Data **compression** and **privacy** preservation

- ◆ Details

- multinomial or Gaussian sampling model for rating variable
- EM algorithm for (approximate) model fitting