

# **Part 16: Context Dependent Recommendations**



**Francesco Ricci**

# Content

- What is "context"?
- Context modelling in Collaborative Filtering
- Data warehouse model of the ratings
- Multiple recommendation problems
- Segmentation of rating data
- Finding good contextual conditions
- Evaluation: contextual vs. standard

Much of this lecture is based on:

Adomavicius, G., Sankaranarayanan, R., Sen, S., and Tuzhilin, A. (2005).  
Incorporating contextual information in recommender systems using a  
multidimensional approach. *ACM Trans. Inf. Syst.*, 23(1):103-145.

# Motivating Examples

- Recommend a vacation
  - **Winter** vs. **summer**
- Recommend a purchase (e-retailer)
  - **Gift** vs. for **yourself**
- Recommend a movie
  - For a **Saturday night** with his **girlfriend** in a **movie theater** vs. at **home** with a **group of friends**
- Recommend music
  - When you have a **happy** vs. **sad mood**.

# What simple rec. techniques ignore?

- ❑ What is the user **doing** when asking for a recommendation?
- ❑ Where (and when) the user is **located**?
- ❑ What does the user **really want** (e.g., improve his knowledge or really buy a product)?
- ❑ Is the user **alone** or with other **fellows**?
- ❑ Are there **many** products to choose or only **few**?
- ❑ What is the **mood** of the user when searching for recommendations?
- ❑ Is the word economy growing or falling?
- ❑ ...

# Contextual Computing

- ❑ **Contextual computing** refers to the **enhancement of a user's interactions** by understanding the user, the context, and the applications and information being used, typically across a wide set of user goals
- ❑ Actively adapting the computational environment - for each and every user - at each point of computation
- ❑ Contextual computing approach focuses on understanding the **information consumption patterns** of each user
- ❑ Contextual computing focuses on the **process** not only on the output of the search process.

[Pitkow et al., 2002]

# Contextualization and Individualization

- **Contextualization:** the interrelated conditions that occur within an activity
  - It includes factors like the nature of information available, the information currently being examined, the applications in use, when, and so on
- **Individualization:** the totality of characteristics that distinguishes an individual
  - It encompasses elements like the user's goals, prior and tacit knowledge, past information-seeking behaviors, among others
- **Personalization must focus on the combination** of the user and the context within the application of search.

[Pitkow et al., 2002]

# Major obstacle for contextual computing

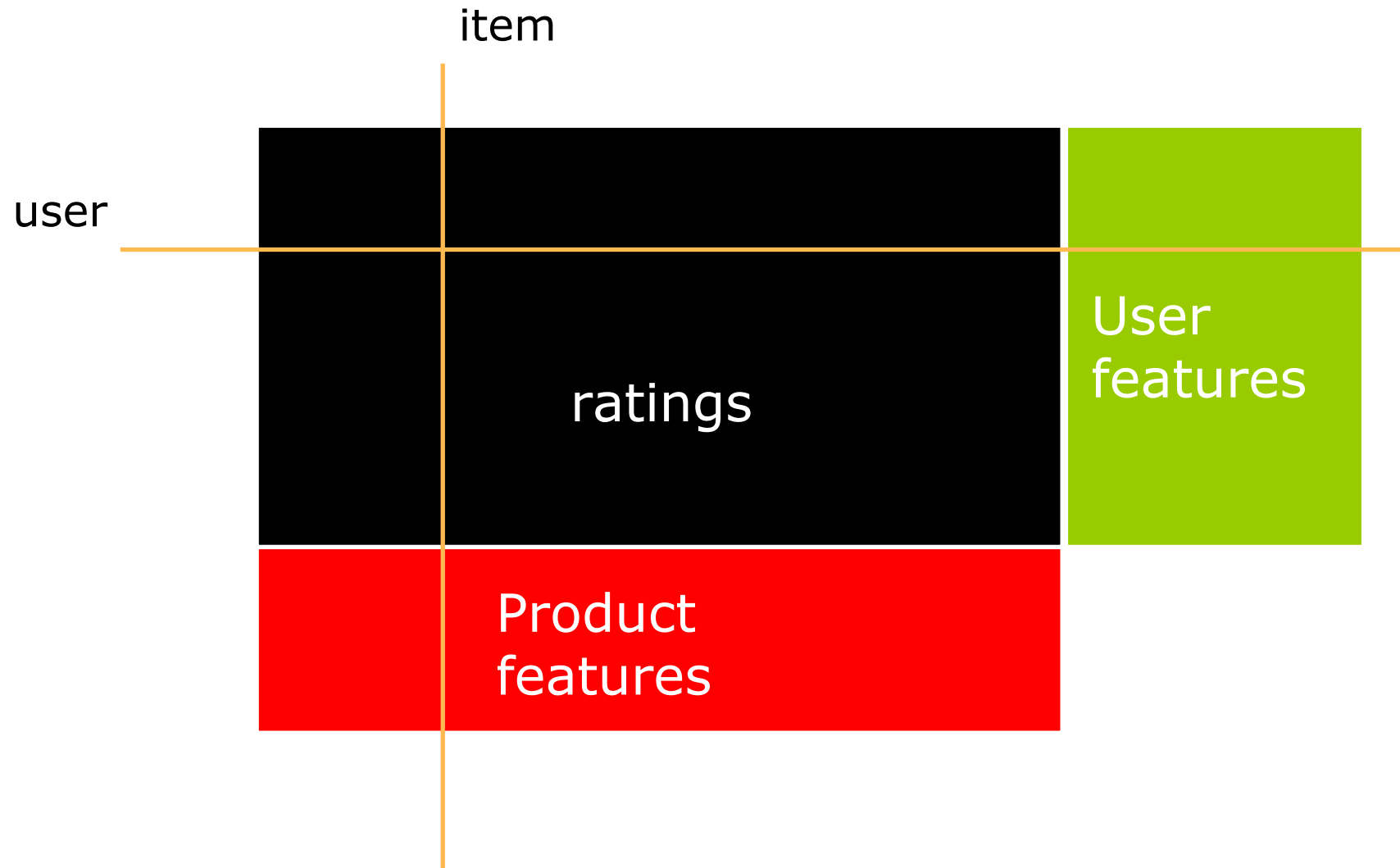
- **Obtain sufficient** and **reliable data** describing the user context
- **Selecting** the right information, i.e., relevant in a particular personalization task
- **Understand** the impact of contextual dimensions on the personalization process
- **Computational model** the contextual dimension in a more classical recommendation technology
  - For instance: how to extend Collaborative Filtering to include contextual dimensions?

# A Simplified Model of Recommendation

1. Two types of entities: **Users** and **Items**
2. A **background knowledge**:
  - A set of ratings: a map  $R: \text{Users} \times \text{Items} \rightarrow [0,1] \cup \{?\}$  – **R is a partial function!**
  - A set of “features” of the Users and/or Items
3. A **method** for **substituting** all or part of the ‘?’ values - for some (user, item) pairs – with good rating predictions
4. A method for **selecting the items to recommend**
  - Recommend to  $u$  the item  $i^* = \arg \max_{i \in \text{Items}} \{R(u,i)\}$

[Adomavicius et al., 2005]

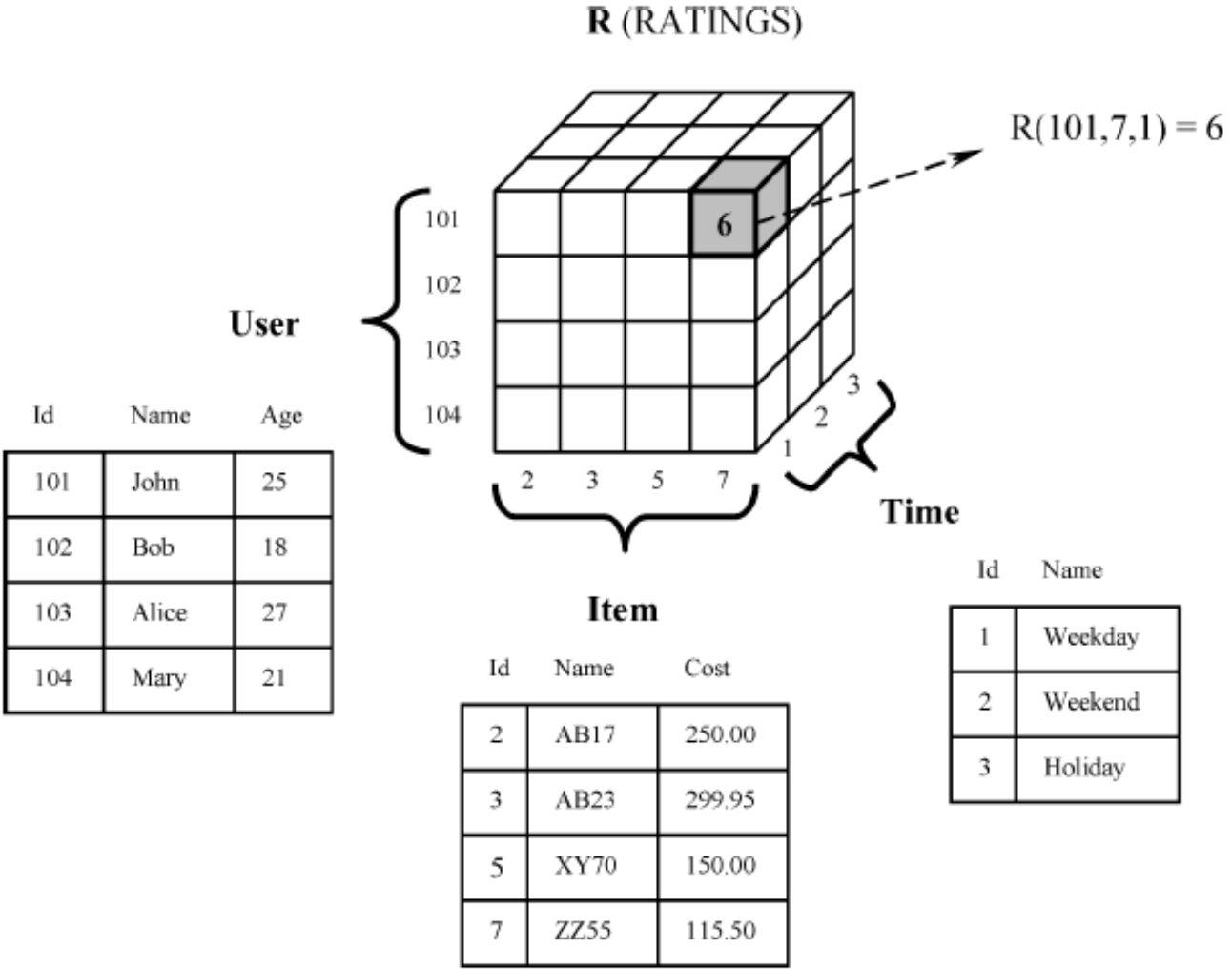
# A Bidimensional Model



# Bi-dimensional vs. multidimensional

- The previous model ( $R: \text{Users} \times \text{Items} \rightarrow [0,1] \cup \{?\}$ ) is **bi-dimensional**
- A general model may include some “contextual” dimensions, e.g.:
  - $R: \text{Users} \times \underline{\text{Time}} \times \underline{\text{Place}} \times \text{Items} \rightarrow [0,1] \cup \{?\}$
- **Assumption:** the rating function or, more in general, the recommendation evaluation is more complex than an assignment of each pair (user, product) to a rating
- There must be some "**hidden variables**" that contributes to determining the rating function
- This **multidimensional** data model approach was developed for data warehousing and OLAP.

# Multidimensional Model



# Formal Model

- $D_1, D_2, \dots, D_n$  are **dimensions**
- The **recommendation space** is n-dimensional:  
 $D_1 \times D_2 \times \dots \times D_n$
- Each dimension is a subset of the Cartesian product of some attributes  $D_i \subseteq A_{i(1)} \times \dots \times A_{i(k_i)}$  – **profile** of the dimension  $D_i$
- $i: [1, k_i] \rightarrow N$  (injective), and  $A_1, \dots, A_N$  is the set of all the attributes
- **General Rating function**
  - $R: D_1 \times D_2 \times \dots \times D_n \rightarrow [0,1] \cup \{?\}$

## Example

- User x Item x Time  $\rightarrow [0,1] \cup \{?\}$  – 3 dimensions
- User  $\subseteq$  UName x Address x Income x Age – 4 attributes
- Item  $\subseteq$  IName x Type x Price – 3 attributes
- Time  $\subseteq$  Year x Month x Day – 3 attributes
- Example:
  - **User** John Red (living in Bolzano, with Income 1000 and aged 34)
  - **rated** 0.6
  - a **vacation** at Miramonti Hotel (for 100E per night),
  - which he did **on** August 4-11, 2007.

# Recommendation Problem

- Assume that the rating function is **complete** (defined for each entry in  $D_1 \times D_2 \times \dots \times D_n$ )
- Recommendation problem:
  - **“what”** to recommend is a subset of the dimensions:  $D_{i_1}, \dots, D_{i_k}$  ( $k < n$ )
  - **“for whom”** is another subset of the dimensions:  $D_{j_1}, \dots, D_{j_l}$  ( $l < n$ )
  - The dimension in “what” and “for whom” have a void intersection, and

for whom	$\forall (d_{j_1}, \dots, d_{j_l}) \in D_{j_1} \times \dots \times D_{j_l},$	$(d_{i_1}, \dots, d_{i_k}) =$	what
	$\arg \max$	$R(d'_1, \dots, d'_n)$	
	$(d'_{i_1}, \dots, d'_{i_k}) \in D_{i_1} \times \dots \times D_{i_k}$	$(d'_{j_1}, \dots, d'_{j_l}) = (d_{j_1}, \dots, d_{j_l})$	
			This is given

## Example

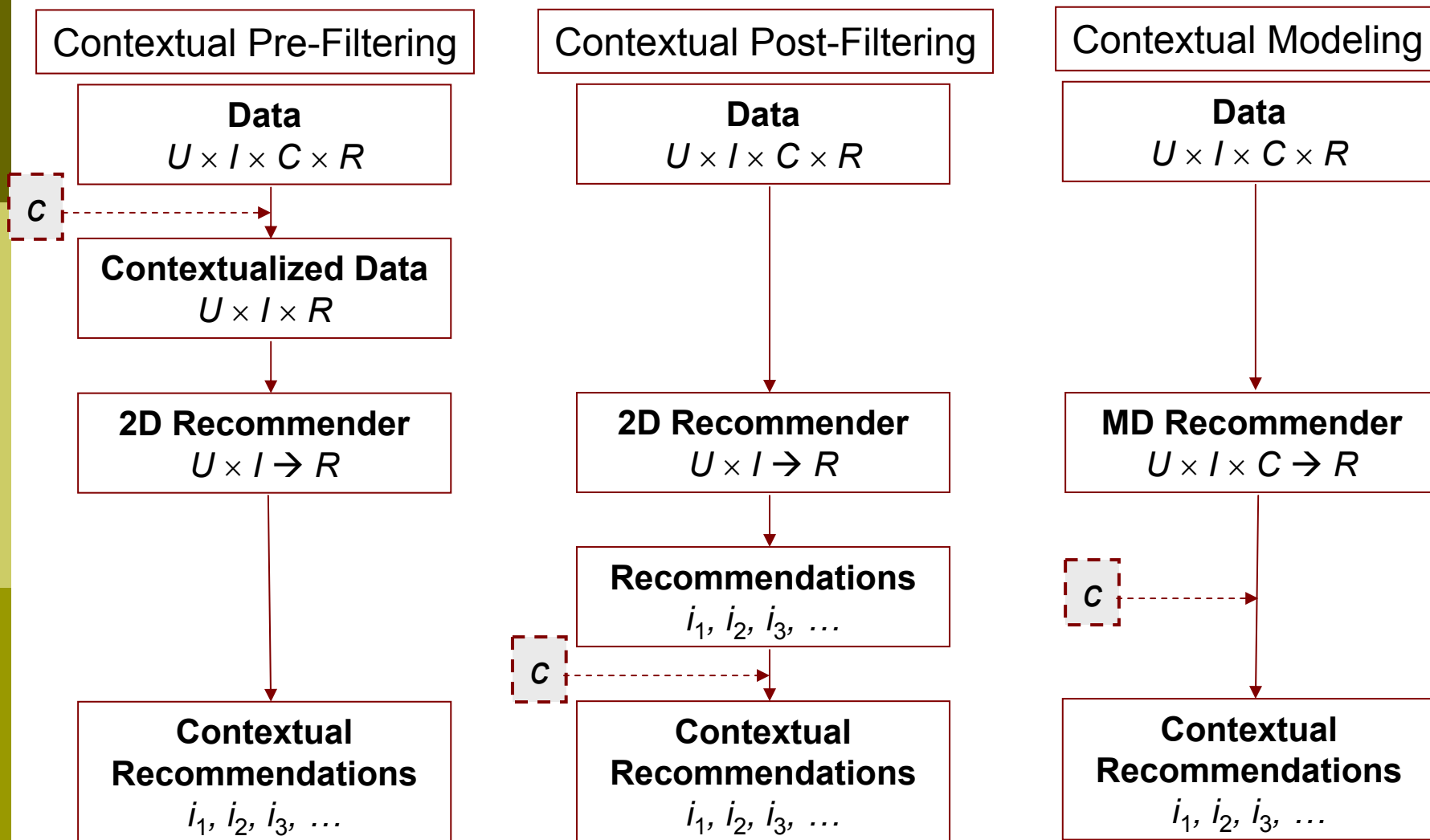
- **Movie:** defined by attributes *Movie(MovieID, Name, Studio, Director, Year, Genre, MainActors)*
- **Person:** defined by attributes *Person(UserID, Name, Address, Age, Occupation, etc.)*
- **Place:** a single attribute defining the listing of movie theaters and also the choices of the home TV, VCR, and DVD
- **Time:** the time when the movie can be or has been seen: *Time(TimeOfDay, DayOfWeek, Month, Year)*
- **Companion:** a person or a group with whom one can see the movie: a single attribute having values "alone," "friends," "girlfriend/boyfriend," "family," "co-workers," and "others."

## Example (cont)

R(movie, person, place, time, companion) context

- ❑ Recommend the best movies to users
- ❑ Recommend top 5 action movies to users older than 18
- ❑ Recommend top 5 movies to user to see on the weekend, but only if the personal ratings of the movies are higher than 0.7
- ❑ Recommend to Tom and his girlfriend top 3 movies and the best time to see them over the weekend
- ❑ Recommend movie genre to different professions using only the movies with personal ratings bigger than 6.

# Paradigms for Incorporating Context in Recommender Systems



# Hierarchies

- Some of the dimensions  $D_i$  may have **hierarchies** associated with these dimensions
- Examples:
  - **Products** may use a standard industrial product hierarchy (e.g.
  - **Time dimension** can use a temporal hierarchy such as minutes, hours, day, month, seasons, year
  - **Movies** can be classified according to genre and subgenre.
- Hierarchies can be used to define contexts and sub-contexts.

# Getting Ratings

- How to obtain (**enough**) ratings for the multidimensional cube?
- The problem is even more complex than for a “standard” bi-dimensional cube (User x Items)
- Some ratings are obtained in an **explicit** way (“we know the user rated 0.6 that item in that specific context”)
- Some ratings are obtained in an **implicit** way (e.g., the user browsed descriptions of that product, or the user inserted that product in a “whish list”)
- Ratings can be obtained in aggregated form: e.g., “John Doe assigned rating 0.6 to all action movies”.

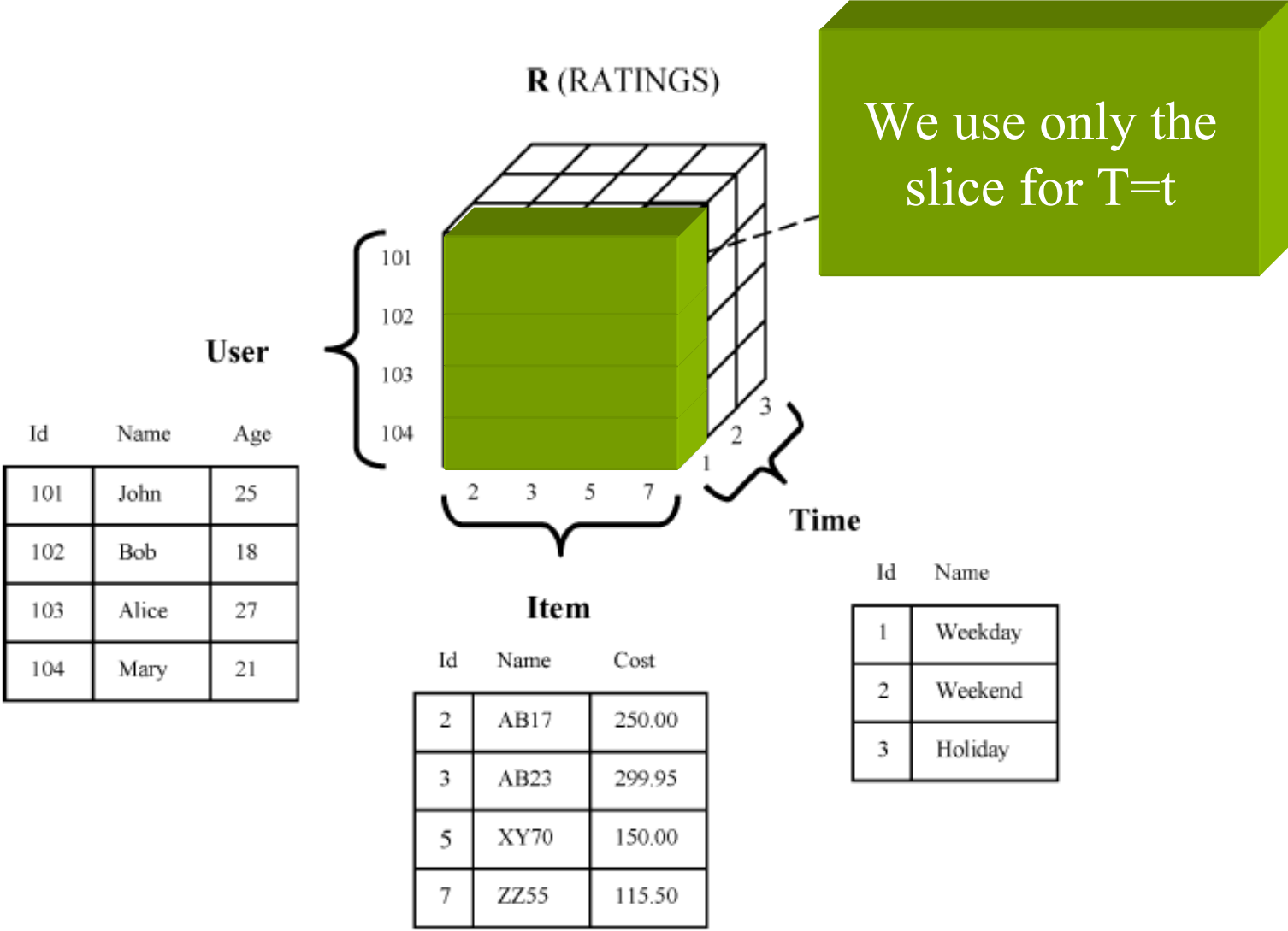
# Multi-level Multidimensional Rating Estimation Problem

- Given the initial (small) set of user-assigned ratings specified for *different* levels of the multidimensional cube of ratings, the task is to estimate *all* other ratings in the cube at *all* the levels of the OLAP hierarchies
- In other words:
  - Given a rating function  $R: D_1 \times D_2 \times \dots \times D_n \rightarrow [0,1] \cup \{?\}$
  - Compute an **estimation**  $R^D: D_1 \times D_2 \times \dots \times D_n \rightarrow [0,1]$
  - **D** is the subset of  $D_1 \times D_2 \times \dots \times D_n$  where R is **known**.

# Reduction-Based Approach

- Reduce the problem of **multidimensional** recommendation to the traditional **two-dimensional** User x Item
- *For each "value" of the contextual dimension(s) estimate the missing ratings with a traditional method*
- Example:
  - $R: U \times I \times T \rightarrow [0,1] \cup \{?\}$  ; User, Item, Time
    - $R^D(u, i, t) = R^{D[T=t]}(u, i)$
  - *The context-dependent estimation for  $(u, i, t)$  is computed using a traditional approach, in a two-dimensional setting, but using only the ratings that have  $T=t$ .*

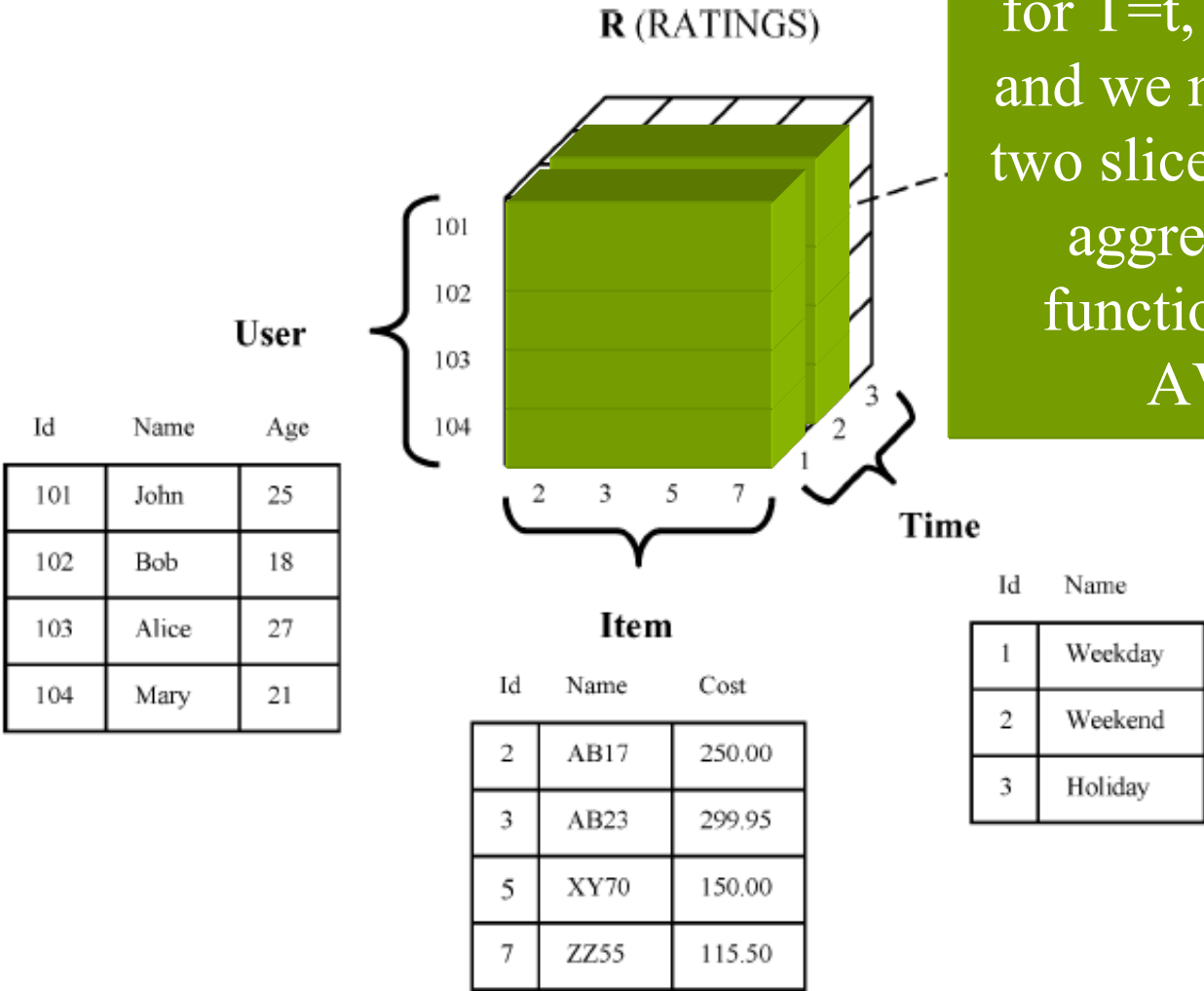
# Multidimensional Model



## Problems with the reduction

- The relation  $D[Time=t](User, Item, Rating)$  may not contain enough ratings for the two dimensional recommender algorithm to accurately predict  $R(u, i)$  for that specific value  $t$  of the Time variable
- **Approach:** use a “larger” contextual segment  $S_t$ , such that  $t \in S_t$
- Instead of  $R^D(u, c, t) = R^{D[T=t]}(u, c)$
- We have  $R^D(u, c, t) = R^{D[t \in S_t]}(u, c)$  **aggregated**
- **Example:** instead of considering only the ratings of a specific day, e.g., Monday, use the ratings of all the weekdays and aggregate them to produce a two-dimensional slice.

# Multidimensional Model



We use the slices for  $T=t$ , and  $T=t'$  and we merge the two slices with an aggregation function, e.g., **AVG**

# Generalization

- The same method could be used to reduce the original three-dimensional recommendation space to *Item x Time*, instead of *User x Item*
  - Example: recommend the movie to see on Monday
- The definition of what is the “**context**” of a recommendation problem is arbitrary (*from a pure mathematical point of view*)
- The method can be generalized to a generic reduction from an n-dimensional to an m-dimensional recommendation problem ( $m < n$ ).

# Research Problem

- ❑ **Local vs. Global model:** the local model exploits the local context "around" a (user, item) point to build the prediction, whereas the global model of CF would use all the (user, item) points ignoring the contextual information
- ❑ Will a local model **outperform** a global model?
- ❑ Is the local variability **worth exploiting**?
- ❑ When there is a "**dependency**" between context and rating?
- ❑ When the contextual dimensions will not reduce the available data to a too tiny subset?

# Example

	Item 1	Item n	
Context 1	0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6,	0.6, ?	Prediction is 0,6 using context and 0.45 without
Context 2	0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3,	0.3, ?	
Ratings for user x			Prediction is 0,3 using context and 0.45 without

Assume we are using item-to-item CF for prediction

# Example

	Item 1	Item n
Context 1	1.0, 1.0, 1.0, 1.0	0.1, 0.1, 0.1, 0.1, ?
Context 2	0.1, 0.1, 0.1, 0.1	1.0, 1.0, 1.0, 1.0, ?

Ratings for user x

Prediction is 1,0 if target is similar to orange items

Prediction is 0,1 if target is similar to pink items

Assume we are using item-to-item CF for prediction

Prediction is 0,55 if context is ignored – similarity to blue or red items is irrelevant

# Evaluation

- $\mu_{A,X}(Y)$  denotes the performance of an algorithm  $A$  trained on a set of ratings  $X$  and tested on a set  $Y$  ( $X \cap Y = \emptyset$ );  $X$  and  $Y$  are subsets of  $D$
- If  $d \in D$  then:
  - $d.R$  is the **actual rating** of  $d$
  - $d.R_{A,X}$  is the **predicted rating** of  $d$  (by algorithm  $A$  trained with data in  $X$ )
- **Example:** MAE (mean absolute error)

$$\mu_{A,X}(Y) = \left(1/|Y|\right) \sum_{d \in Y} |d.R_{A,X} - d.R|$$

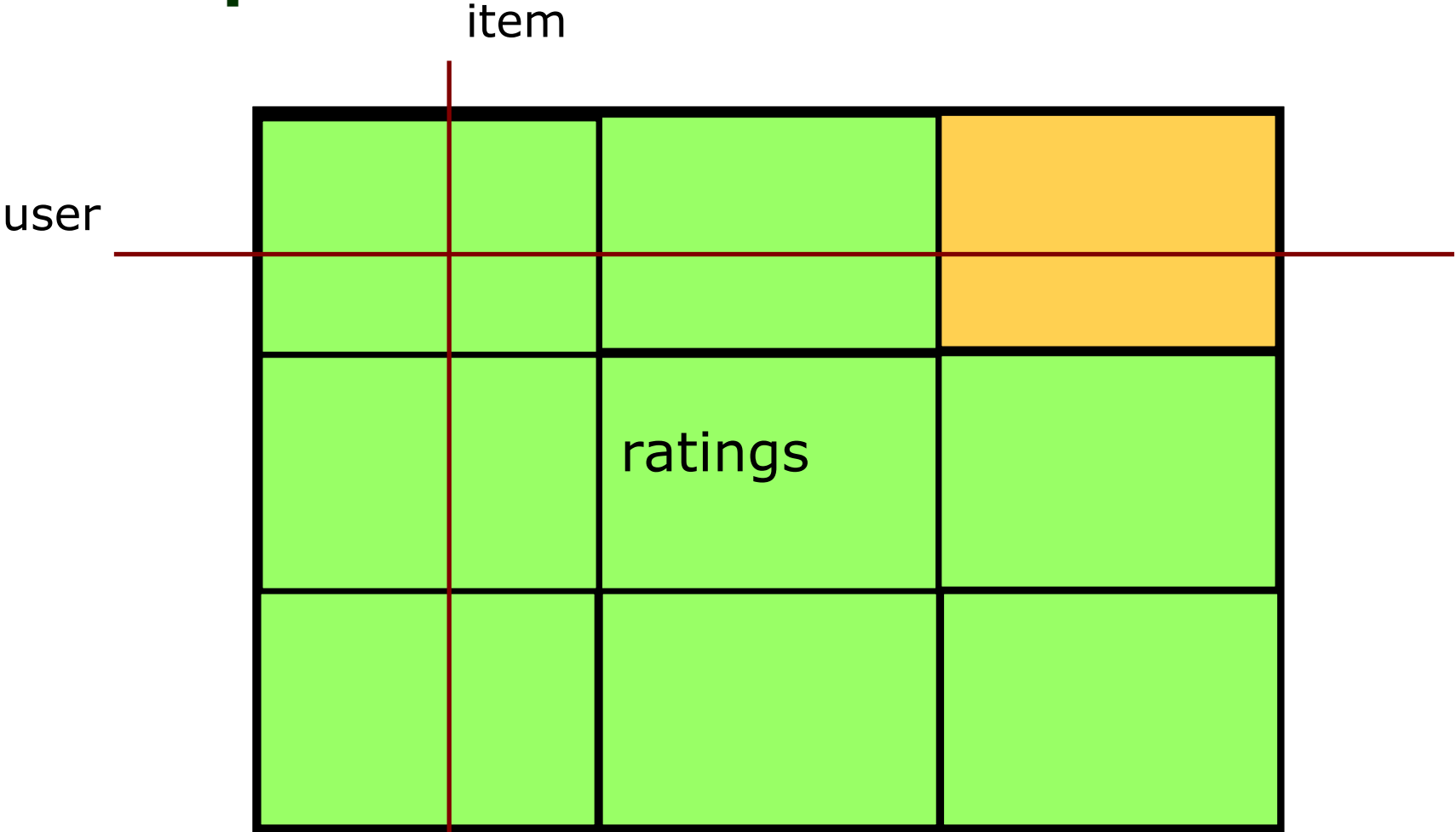
# Cross Validation

- Assume that  $T$  is the set of know ratings (i.e.,  $d$  such that  $d.R$  is not ?)
- Partition the set  $T$  into  $k$  folds, i.e.,  $k$  not-overlapping subsets  $X_i$  such that  $(\cup_{i=1,\dots,k} X_i) = T$
- Then

$$\mu_{A,T}(T) = 1/k \sum_{i=1}^k \mu_{A,(X_1 \cup X_2 \cup \dots \cup X_{i-1} \cup X_{i+1} \cup \dots \cup X_k)}(X_i)$$

Apparently you are using the same data for training and test, but that is not true! It is only a compact notation!

# Example



Used for training 

Then Predict 

# Finding high-performance segments

## Inputs:

- $T$  set of pre-specified ratings for a multidimensional recommendation space.
- $R_{A,T}$  rating estimation function based on algorithm  $A$  and training data  $T$ .
- $\mu$  performance metric function.
- $N$  threshold defining the minimal number of ratings for a “large” segment.

## Outputs:

$SEGM(T)$  – set of contextual segments on which the reduction-based approach based on algorithm  $A$  significantly outperforms the pure algorithm  $A$ .

## Algorithm:

1. Let  $SEGM(T)$  initially be the set of all large contextual segments for the set of ratings  $T$ .
2. For each segment  $S \in SEGM(T)$  compute  $\mu_{A,S}(S)$  and  $\mu_{A,T}(S)$ , and keep only those segments  $S \in SEGM(T)$  for which  $\mu_{A,S}(S)$  is *better*<sup>5</sup> than  $\mu_{A,T}(S)$ .
3. Among the segments remaining in  $SEGM(T)$  after Step 2, discard any segment  $S$  for which there exists a different segment  $Q$  such that  $S \subset Q$  and  $\mu_{A,Q}(Q)$  is better than  $\mu_{A,S}(S)$ . The remaining segments form  $SEGM(T)$ .

**The algorithm performs better if trained on the segment**

## Finding the “Large” segments

- ❑ A segment is a “logical” aggregation of ratings based on some contextual dimensions: e.g., the ratings collected in the “week end”, or the ratings in the “week end at home”
- ❑ Not easy to find all large segments with enough data
- ❑ Classical clustering/partitioning problem
- ❑ Rely on background information (such as those provided by a marketing expert) to determine the initial segments
- ❑ Use the “natural” hierarchies on the contextual dimensions to determine the segments.

# Combining the local and global predictions

- Basic idea of the **combined approach** here proposed for context exploitation:
  1. Use the prediction of the **best performing segments** to which a point belongs
  2. If there is **no segments that contain the point use the standard prediction**, that is, computed without the segments
- Hence the combined approach **will always work better or equal than the standard approach** (at the cost of the additional search on the set of segments)
- *BUT: how much better? Is it worth the extra effort?*

# Combining the local and global predictions

The larger the accuracy value the better the segment

## Inputs:

$SEGM(T) = \{S_1, \dots, S_k\}$ —where segments  $S_1$  through  $S_k$  are arranged in the decreasing order with respect to  $\mu$ , i.e.,  $\mu_{A,S_1}(S_1) \geq \dots \geq \mu_{A,S_k}(S_k)$ .  
 $d$ —data point for which we want to estimate the rating.

## Outputs:

$d.R$ —estimated rating for data point  $d$ .

## Algorithm:

```
done = false; i = 1;
while (i ≤ k) and (¬done) do {
    if d ∈ Si then { d.R = RA,Si(d);
                    done = true }
    i = i + 1 }
if (¬done) then d.R = RA,T(d) // i.e., d does not belong to any segment Si
```

Prediction based on algorithm A and data S<sub>j</sub>

# Experimental Evaluation

- Acquired movie ratings and contextual information related to
  - **Time:** weekday, weekend, don't remember
  - **Place:** movie theater, at home, don't remember
  - **Companion:** alone, with friends, with partner, with family, others
- Movies rated in a scale **from 1 to 13**
- Participants were **students**
- 1755 ratings by 117 students over a period of 12 months
- Dropped students that had rated less than 10 movies
- **Finally 62 students, 202 movies and 1457 ratings (the set D) – not very big!**

# Do these contextual dimension affect ratings?

- For each user the average rating for all the possible values of a dimension were computed:
  - For instance: the average rating of John in the **weekend** and in the **weekday**
  - *are there any differences for the average rating between different contextual conditions?*
- Then the averages were compared with a t-test
- For the “companion” dimension all the possible pairs were compared
- Conclusion: *for all the selected contextual dimensions there were significant differences in the average ratings!*

# Evaluation Procedure

- Split the original data set into 10 not-overlapping folds
- **For one fold**
  - Run the segment selection on the remaining 9 folds and identify the best segments (this set is called  $D_M$ )
- **For all the folds**
  - Train on the other 9 folds
  - Compute the prediction on the data in the fold
  - Compute the error on the fold
- **Compare** the error obtained on the ten folds with a traditional CF algorithm, i.e., not using the contextual dimensions
- Error computed using F-measure
  - $F = 2P * R / (P + R)$  P=precision, R=recall
  - A movie is Relevant if rated above 10 (on a 1-13 scale)

# Algorithms

- $\mu_{A,S}(S)$  is computed using only the ratings in the segment (contextual dependent)
- $\mu_{A,T}(S)$  is computed using all the data (in the train or in the test set depending when it is needed)
- To compute both  $\mu_{A,S}(S)$  and  $\mu_{A,T}(S)$  they use: **user to user collaborative filtering**

# Estimating the performance of standard CF

- $D_M$  is the data set where the (good) segments were found
- 500 splits of  $D_M$  in 29/30 for training and 1/30 for test
- $X$  is the subset of  $D_M$  for which CF (standard) can make a prediction
  - *You must have enough ratings for making a prediction, i.e., similar users must have rated the items whose rating you want to predict*
- The final prediction for a rating is computed by averaging all the prediction computed in some of these 500 splits

# Performance of standard CF

Performance Metric, $\mu$	Value, $\mu_{CF,DM}(X)$
MAE	2.0
Precision	0.617
Recall	0.356
F-measure	0.452

$$MAE_{CF,DM}(X) = \left(1/|X|\right) \sum_{d \in X} |d.R_{CF,DM} - d.R|$$

# Searching large segments

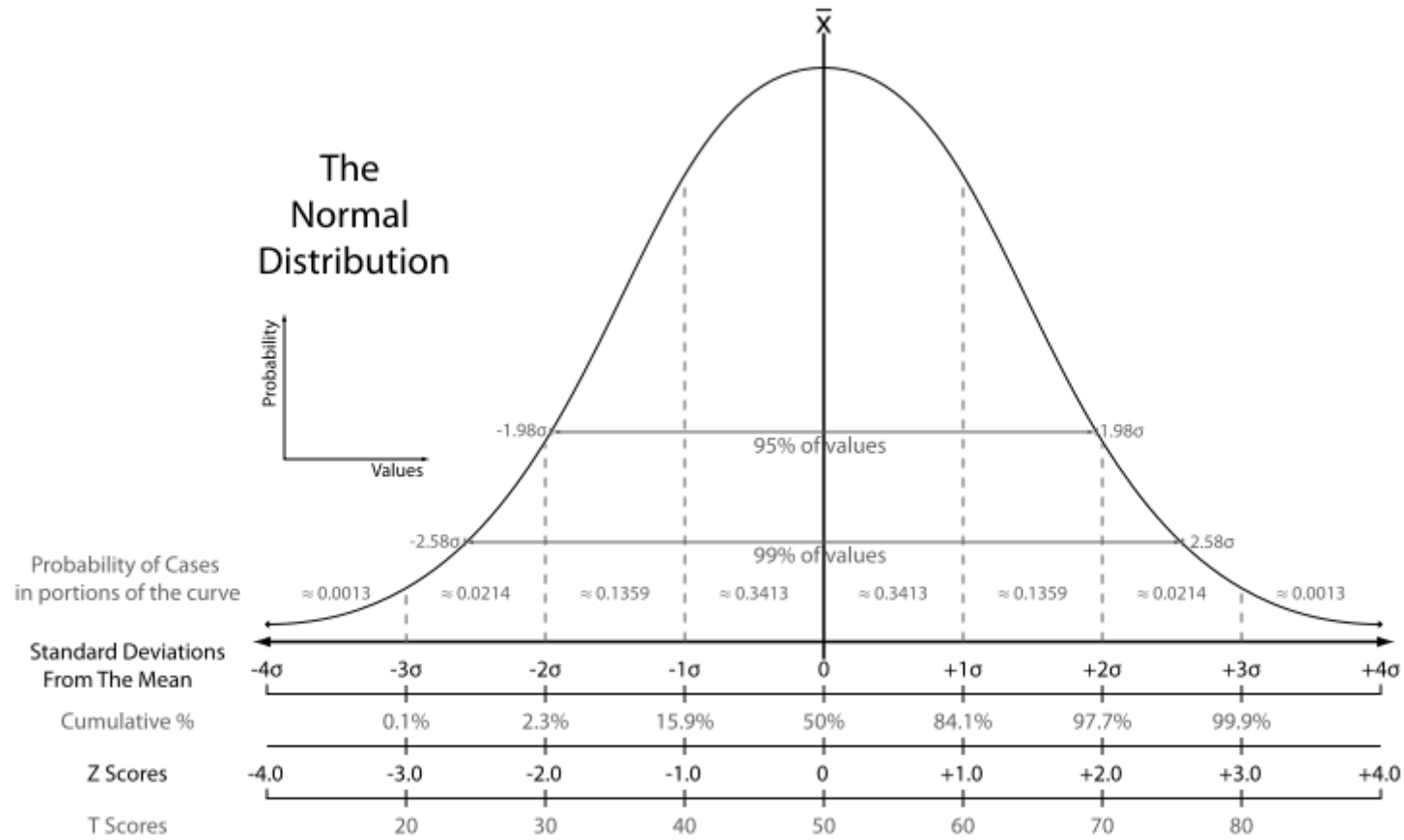
Name	Size	Description
Home	727	Movies watched at home
Friends	565	Movies watched with friends
NonRelease	551	Movies watched not during the 1st weekend of their release
Weekend	538	Movies watched on weekends
Theater	526	Movies watched in the movie theater
Weekday	340	Movies watched on weekdays
GBFriend	319	Movies watched with girlfriend/boyfriend
Theater-Weekend	301	Movies watched in the movie theater on weekends
Theater-Friends	274	Movies watched in the movie theater with friends

- These are obtained by performing an exhaustive search among the space of all possible segments (for the different dimensions try all different attribute values combinations)
- Each one of these segments has **more than 262 user-specified ratings** (more than 20% of the original dataset  $D_M$ )

# Comparison on each segment

Segment	Method (CF)	Precision	Recall	F-measure
<b>Home</b> Segment size: 727 Predicted: 658	Segment-based	0.527	0.319	0.397
	Whole-data-based	0.556	0.357	0.435
	<i>z-values</i>	<i>0.427</i>	<i>0.776</i>	
<b>Friends</b> Segment size: 565 Predicted: 467	Segment-based	0.526	<b>0.444</b>	0.482
	Whole-data-based	0.643	0.333	0.439
	<i>z-values</i>	<i>1.710</i>	<i>-2.051</i>	
<b>NonRelease</b> Segment size: 551 Predicted: 483	Segment-based	0.495	0.383	0.432
	Whole-data-based	0.500	0.333	0.400
	<i>z-values</i>	<i>0.065</i>	<i>-0.869</i>	
<b>Weekend</b> Segment size: 538 Predicted: 463	Segment-based	0.596	<b>0.497</b>	<b>0.542*</b>
	Whole-data-based	0.655	0.383	0.484
	<i>z-values</i>	<i>0.983</i>	<i>-2.256</i>	
<b>Theater</b> Segment size: 526 Predicted: 451	Segment-based	0.622	<b>0.595</b>	<b>0.608*</b>
	Whole-data-based	0.694	0.366	0.479
	<i>z-values</i>	<i>1.258</i>	<i>-4.646</i>	
<b>Weekday</b> Segment size: 340 Predicted: 247	Segment-based	0.415	0.349	0.379
	Whole-data-based	0.531	0.270	0.358
	<i>z-values</i>	<i>1.041</i>	<i>-0.964</i>	
<b>GBFriend</b> Segment size: 319 Predicted: 233	Segment-based	0.513	0.451	0.480
	Whole-data-based	0.627	0.352	0.451
	<i>z-values</i>	<i>1.292</i>	<i>-1.361</i>	
<b>Theater-Weekend</b> Segment size: 301 Predicted: 205	Segment-based	0.660	<b>0.623</b>	<b>0.641*</b>
	Whole-data-based	0.754	0.406	0.528
	<i>z-values</i>	<i>1.234</i>	<i>-3.161</i>	
<b>Theater-Friends</b> Segment size: 274 Predicted: 150	Segment-based	0.657	<b>0.564</b>	<b>0.607*</b>
	Whole-data-based	0.732	0.385	0.504
	<i>z-values</i>	<i>0.814</i>	<i>-2.245</i>	

# Z-score



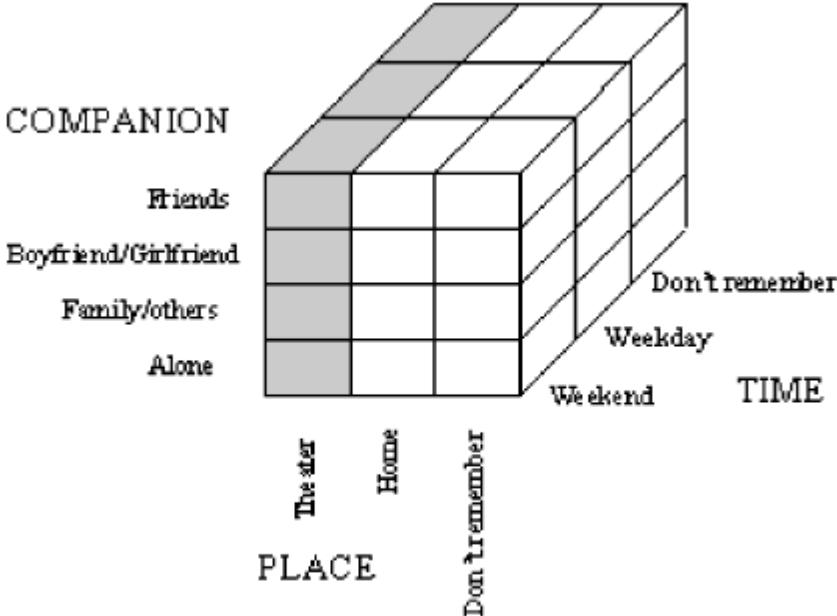
□  $Z = (X - \mu) / \sigma$

# Summary of the differences

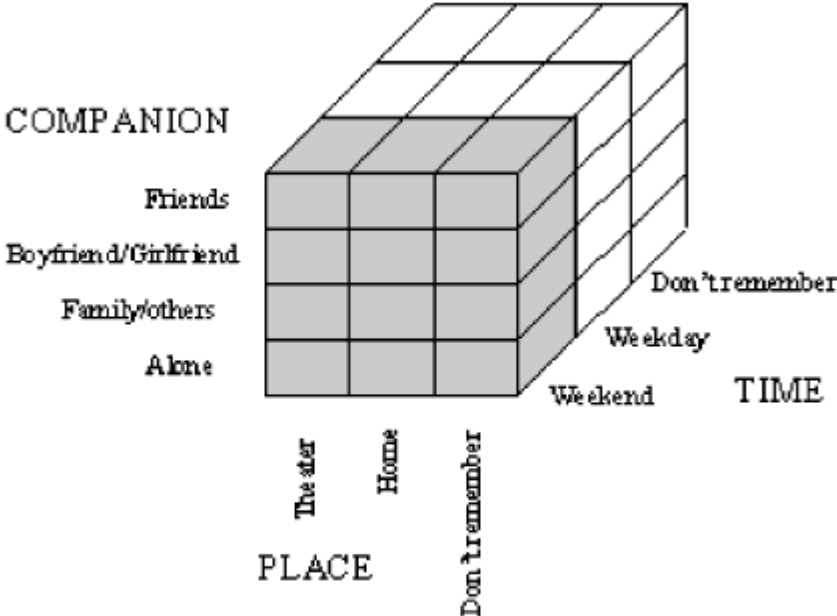
Segment	Segment-based F-measure	Whole-data-based F-measure
Theater-Weekend	0.641	0.528
Theater	0.608	0.479
Theater-Friends	0.607	0.504
Weekend	0.542	0.484

- ❑ Substantial improvement of F-measure on some segments
- ❑ Since Theater-Friends has lower F-measure than Theater then this is discarded (see the original algorithm)
- ❑ The final segments obtained are: Theater-Weekend, Theater and Weekend.

# Examples of segments



(a) Theater segment



(b) Weekend segment

# Comparison on all data

Comparison	Overall F-measure		Difference between F-measures
	Standard 2-dimensional CF	Combined reduction-based CF	
All predictions (1373 ratings)	0.463	0.526	<b>0.063*</b>
Predictions on ratings from <i>SEGM(T)</i> (743 ratings)	0.450	0.545	<b>0.095*</b>

- ❑ Cross validated F-Measure
- ❑ *SEGM(T)* are the three segments (theater-weekend, theater and weekend ) where we have discovered that the combined reduction-based approach performs better than standard CF
- ❑ The prediction on ratings that do not belongs to these three segments are equal for the two methods.

# Conclusions

- ❑ In this data set the **combined reduction-based** method performs **better** than standard CF (F-measure)
- ❑ The results may depend strongly on the contextual-dependency of the data
- ❑ In real situations the **COVERAGE** of the contextual-dependent approach may be much lower than an approach that do not discriminate according to the context
- ❑ We should identify in each situation the **right level of context** to be used
- ❑ Contextual prediction add a lot of overhead – important to measure if it pays!

# Questions

- ❑ What is the "context" of a recommendation?
- ❑ List some of the motivations why context matters
- ❑ What is the difference between "contextualization" and "individualization"?
- ❑ Describe the multidimensional model and compare it with the classical bi-dimensional one
- ❑ Using the multidimensional model, is it possible to generalize the concept of recommendation and recommend, for instance, "the time for a good action movie" (rather than simply the movie to watch)?
- ❑ How the segments where context-dependent recommendations are worthwhile have been found?
- ❑ What kind of evaluation measure do they were able to improve using contextual data?