

# **Part 12: Advanced Topics in Collaborative Filtering**



**Francesco Ricci**

# Content

- ❑ Generating recommendations in CF using frequency of ratings
- ❑ Role of neighborhood size
- ❑ Comparison of CF with association rules (“traditional” data-mining)
- ❑ Classification and regression learning
- ❑ Memory-based CF vs. Model-Based CF
- ❑ Reliability of the user-to-user similarity
- ❑ Evaluating the importance of each rating in user-to-user similarity
- ❑ Computational complexity of collaborative filtering.

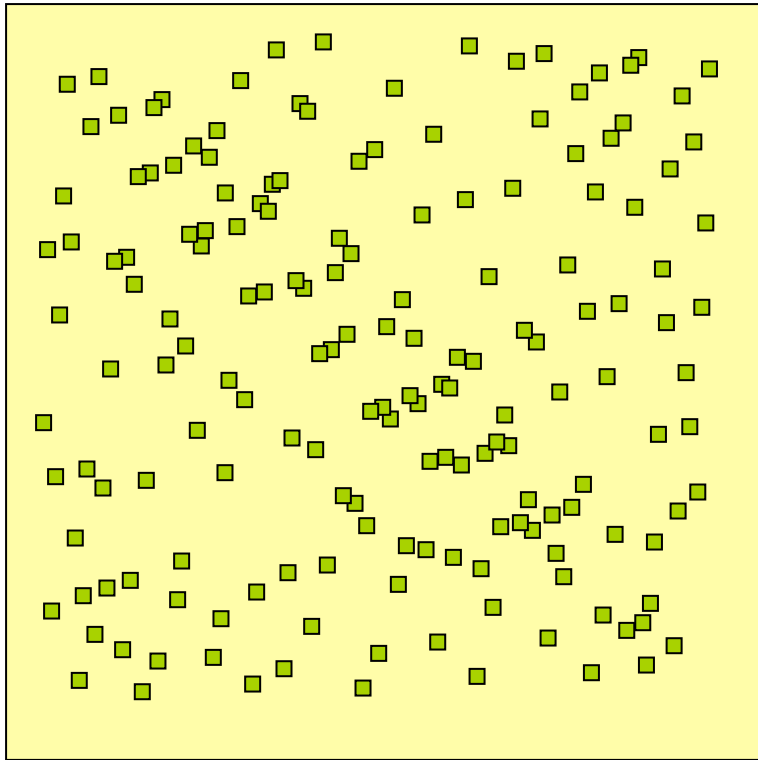
# Example of Evaluation of a Collaborative Filtering Recommender System

- ❑ **Movie data:** 3500 users, 3000 movies, random selection of 100,000 ratings – obtained a matrix of 943 users and 1682 movies
  - Sparsity =  $1 - 100,000 / (943 * 1682) = 0.9369$
  - On average there are  $100,000 / 943 = 106$  ratings per user
- ❑ **E-Commerce data:** 6,502 customers, 23,554 products and 97,045 purchase records
  - Sparsity = 0.9994
  - On average 14.9 ratings per user
- ❑ **Sparsity** is the proportion of missing ratings over all the possible ratings
  - $\# \text{missing-ratings} / \# \text{all-possible-ratings}.$

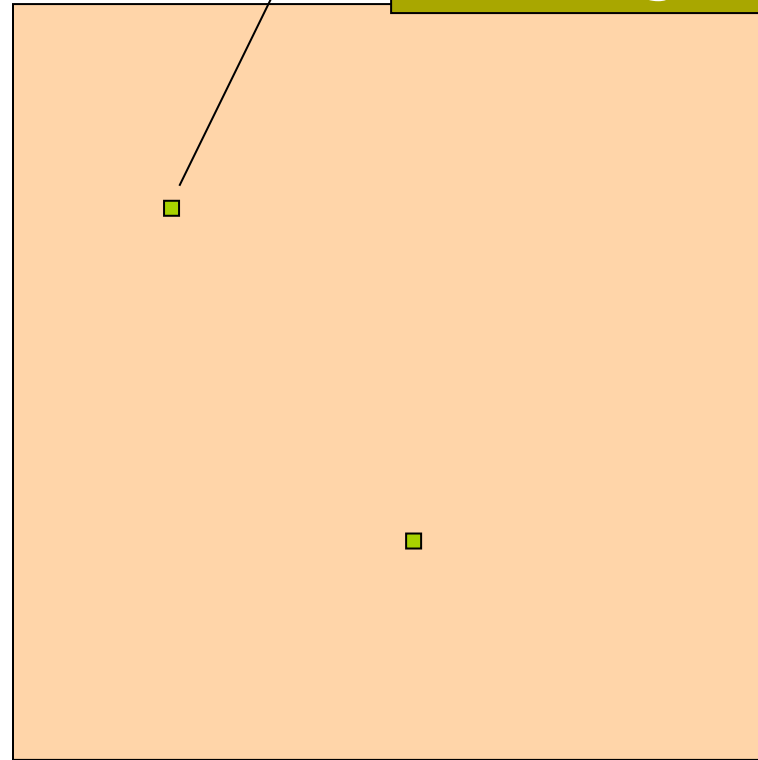
All the possible ratings

[Sarwar et al., 2000]

# Sparsity: visual example



94% sparsity



A set of known ratings

99,9% sparsity

# Evaluation Procedure

- They evaluate top-N recommendation (10 recommendations for each user)
- Separate ratings in training and test sets (80% Train - 20% Test)
- Use the training to compute the prediction (top-N)
- Compare (precision and recall) the items **in the test set of a user** with the top N recommendations **for that user**
- **Hit set** is the intersection of the top N with the test set (selected-relevant)
- Precision = size of the hit set / size of the top-N set
- Recall = size of the hit set / size of the test set
- *they assume that all the items not rated are not relevant*
- They used the cosine metric in the CF prediction method.

# Generation of recommendations

- Instead of using the weighted average of the ratings

$$v_{ij}^* = v_i + K \sum_{v_{kj} \neq ?} u_{ik} (v_{kj} - v_k)$$

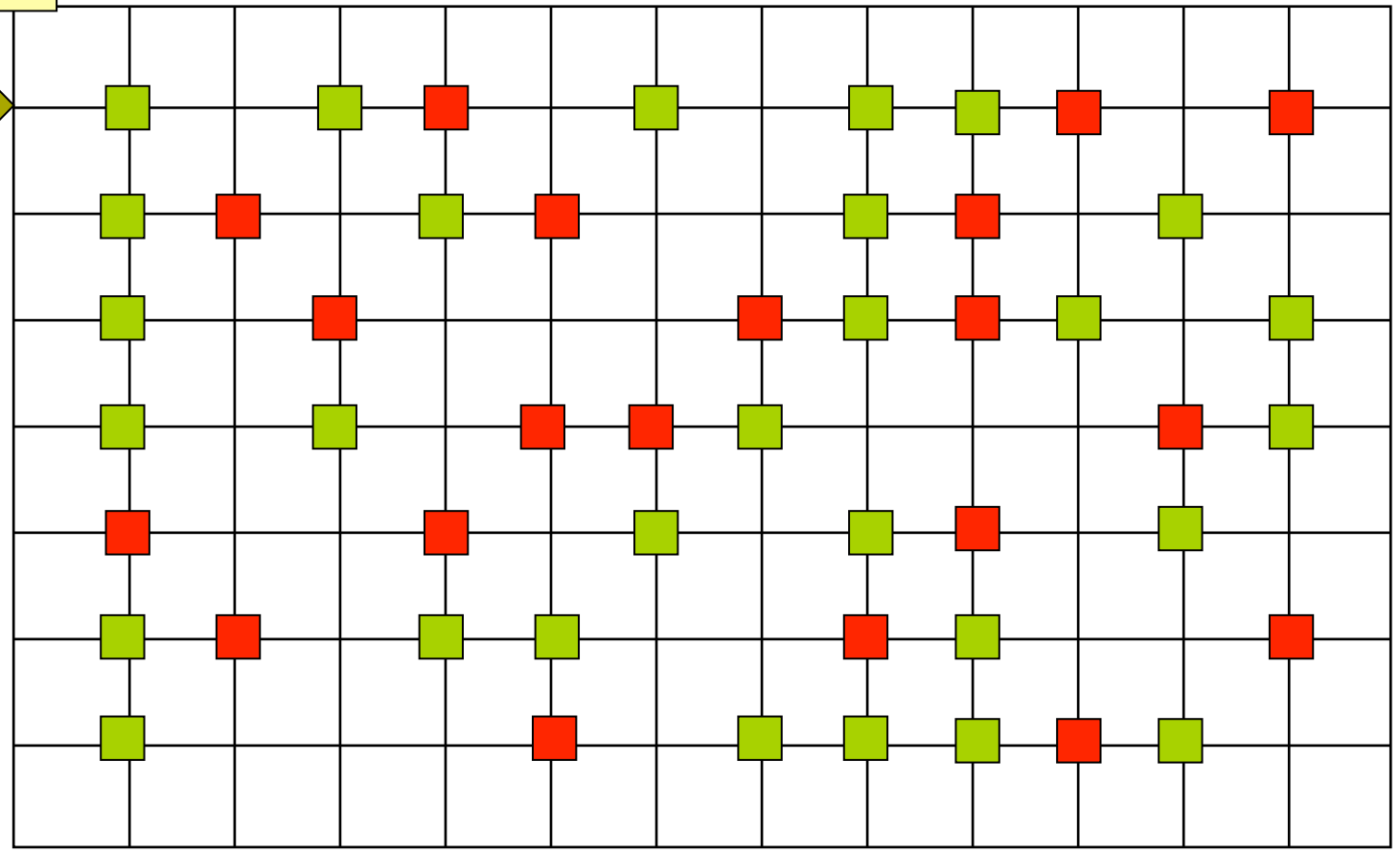
- They used the **most-frequent item recommendation** method
  - Looks in the neighbors (users similar to the target user) scanning the purchase data
  - Compute the **user frequency** of the products in the neighbors purchases - not already in the (training part of the) profile of the target user
  - Sort the products according to the frequency
  - Returns the N most frequent products.

# Top-4 recommended

Frequency of product in neighbors purchases data (training)

0 2 1 2 1 3 2

Prediction for this user ?



$P=2/4$

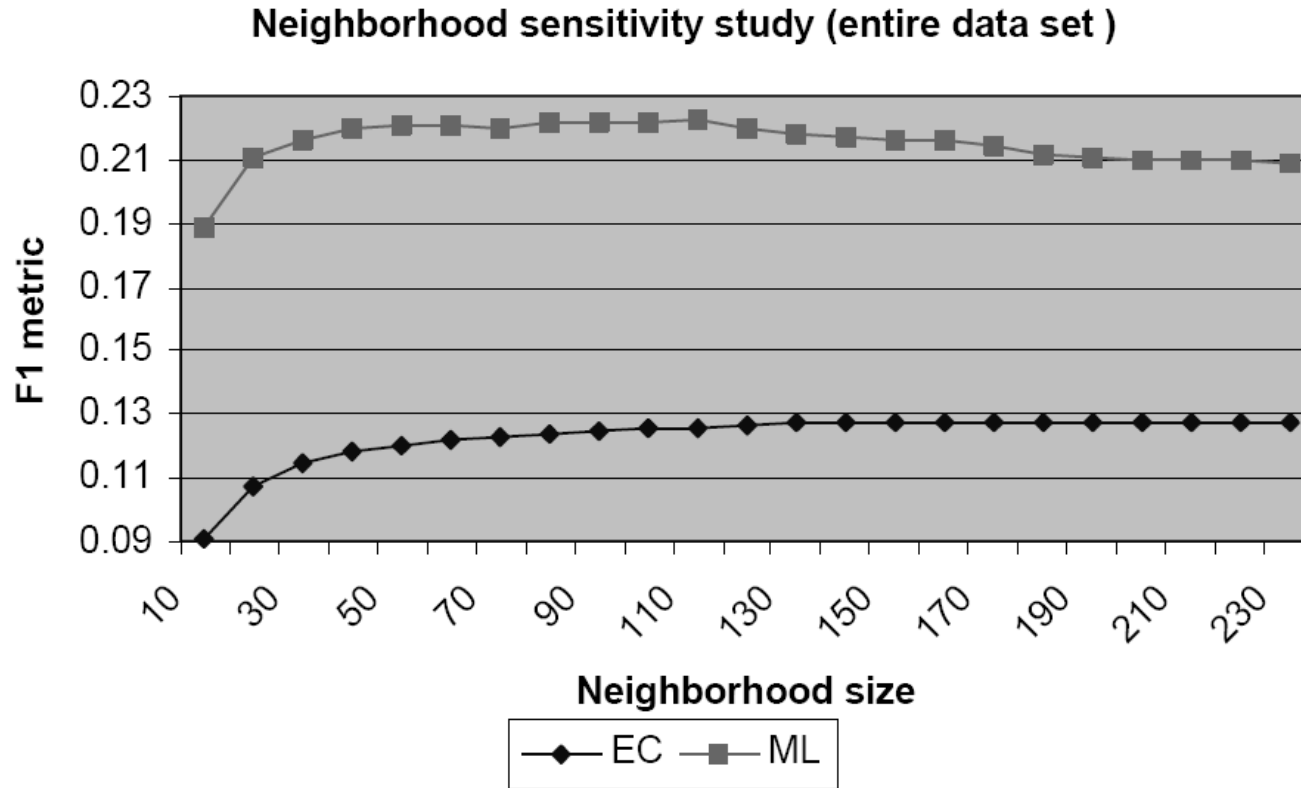
$R=2/3$

■ train  
■ test

Assume that all the depicted users are neighbors of the first one

# Neighbor Sensitivity

Why F1 is so small?  
What is the maximum average recall?



EC = eCommerce data; ML = MovieLens data

Splitting the entire data set into 80% train and 20% test

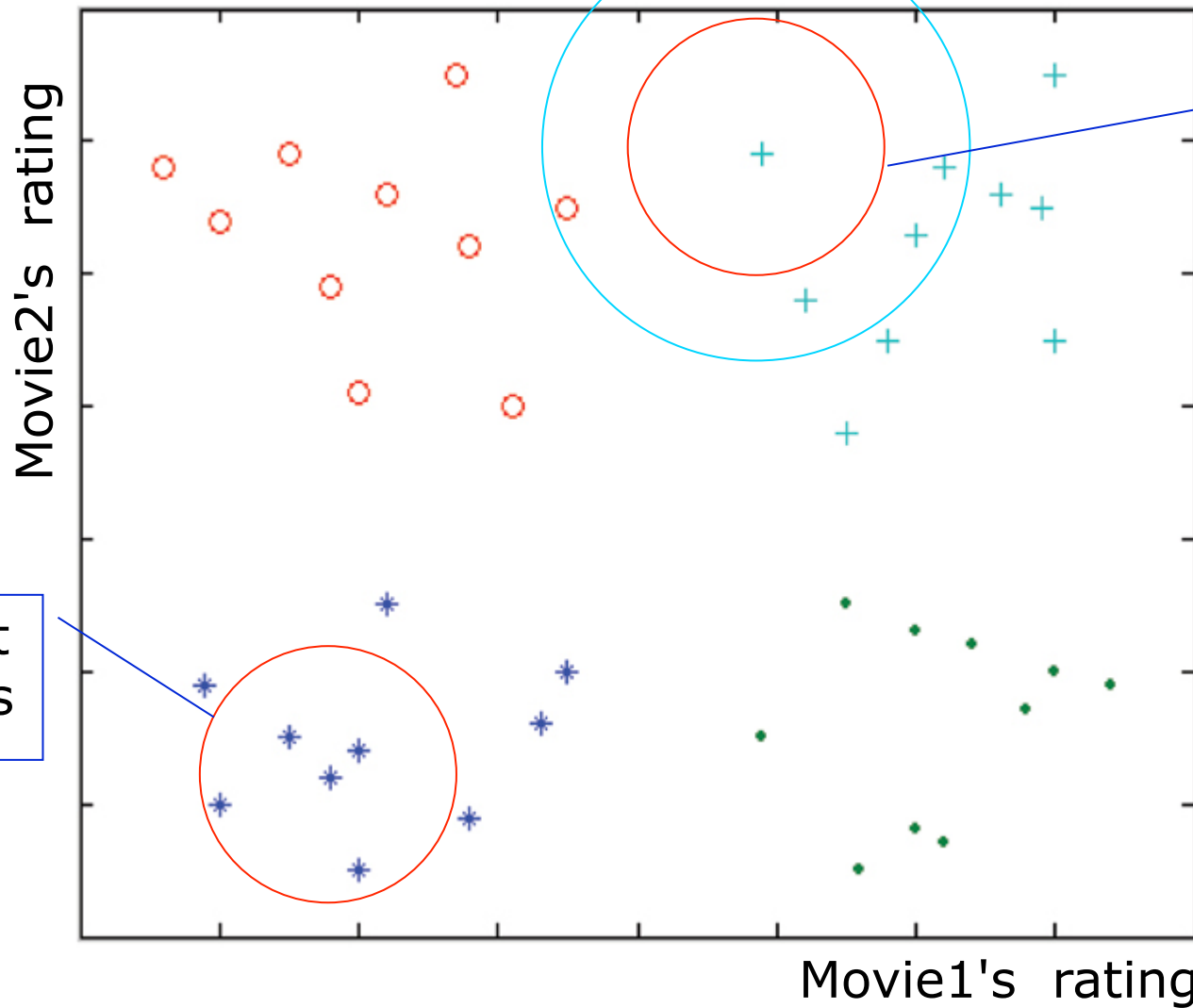
Top 10 recommendations.

# Neighbor Size

- ❑ Reducing the neighbor size is important for performance considerations (why?)
- ❑ If the neighbor size is too large then we are using in the prediction users that are **not very similar** to the target user – hence accuracy should decrease
- ❑ Selection can be made with a **threshold similarity**: the drawback is that as the number of ratings increases we may have too many neighbors
- ❑ Selection can be made with a **fixed k** – accuracy for users with more unique preferences will be lower
- ❑ *Advanced techniques use “adaptive” neighbor formation algorithm – the size depends on the global data characteristics and the user and item specific ratings.*

Remember the discussion on knn optimality

# Users with 2 ratings



4 nearest neighbors

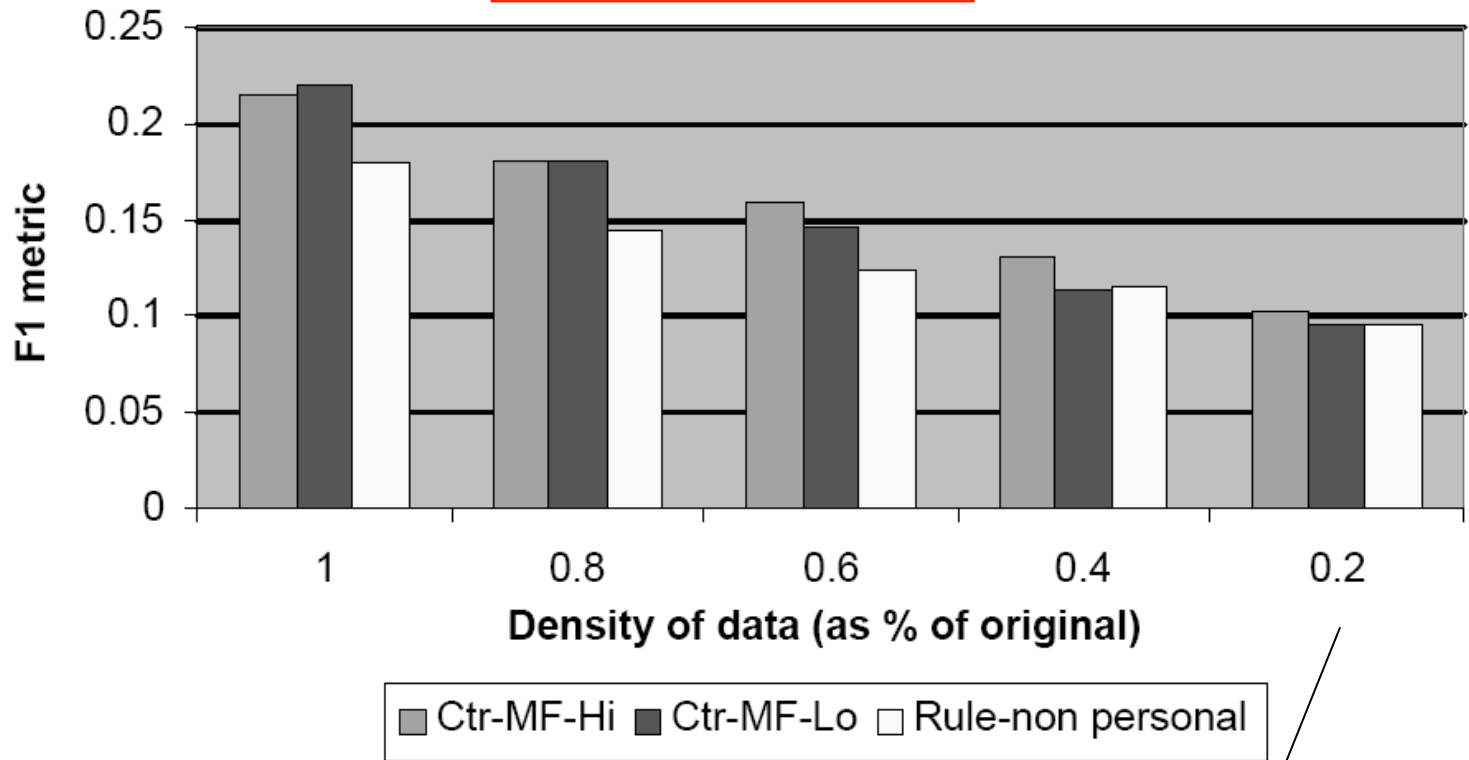
0 nearest neighbors

4 nearest neighbors

- ▣ Fixed similarity threshold in two different points (users) <sup>10</sup> may mean completely different neighbors

# Comparison with Association Rules

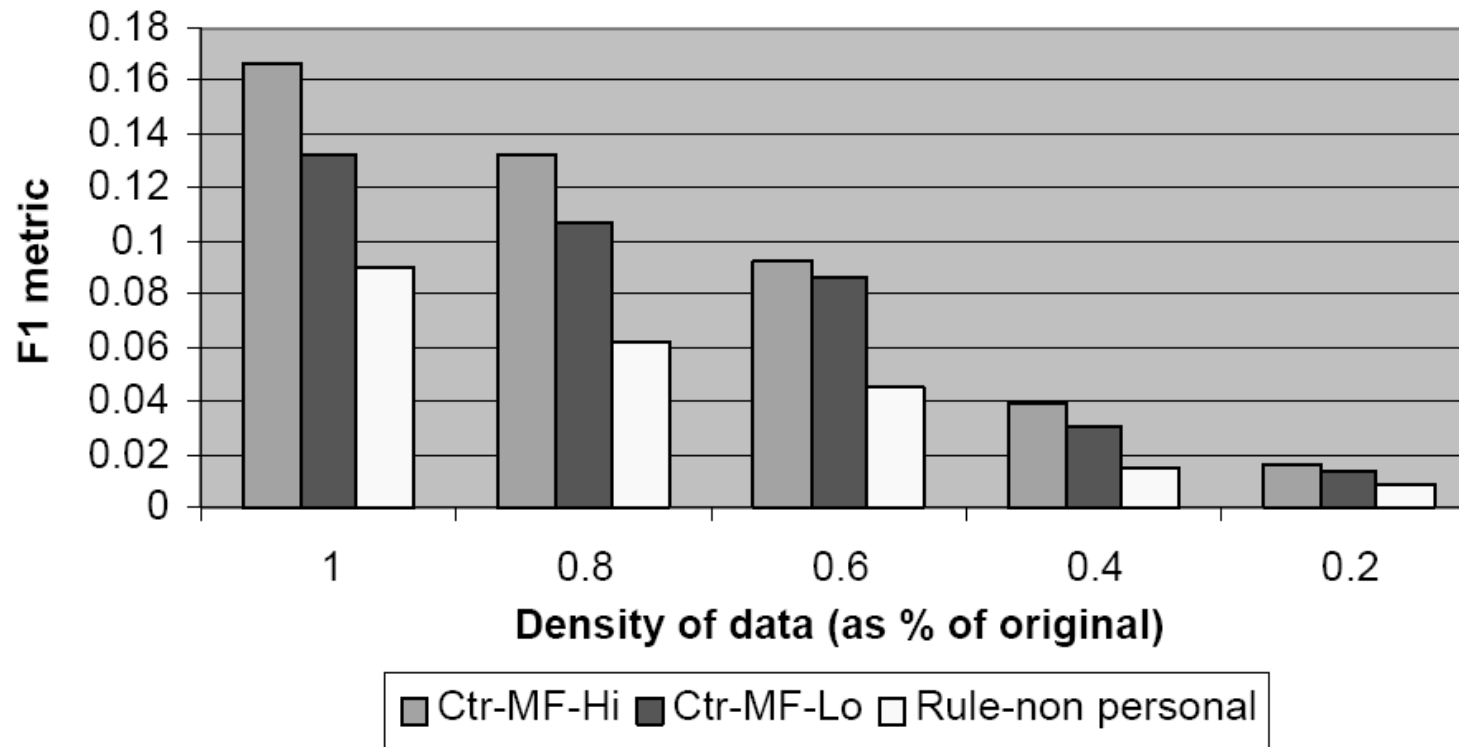
Different Recommendation Algorithms  
(MovieLens data set)



When we do not have enough data personalization is not useful – all these methods tend to perform similarly

# Comparison with Association Rules

Different Recommendation Algorithms  
(E-Commerce data set)



# Association Rules

- Discovering association between sets of co-purchased products – the presence of a set of products “implies” the presence of others
- $\{p_1, \dots, p_m\} = P$  are **products**, and a **transaction T** is a **subset of products P**
- **Association rule:**  $X \rightarrow Y$
- X,Y are **not overlapping subsets** of P
- The meaning of  $X \rightarrow Y$  is that in a collection of transactions  $T_j$  ( $j=1, \dots, N$ ), if X are present it is likely that Y are also present
- We may generate **a transaction for each** user in the CF system: contains **the products that have been rated by the user.**



# Very Basic Probability Concepts

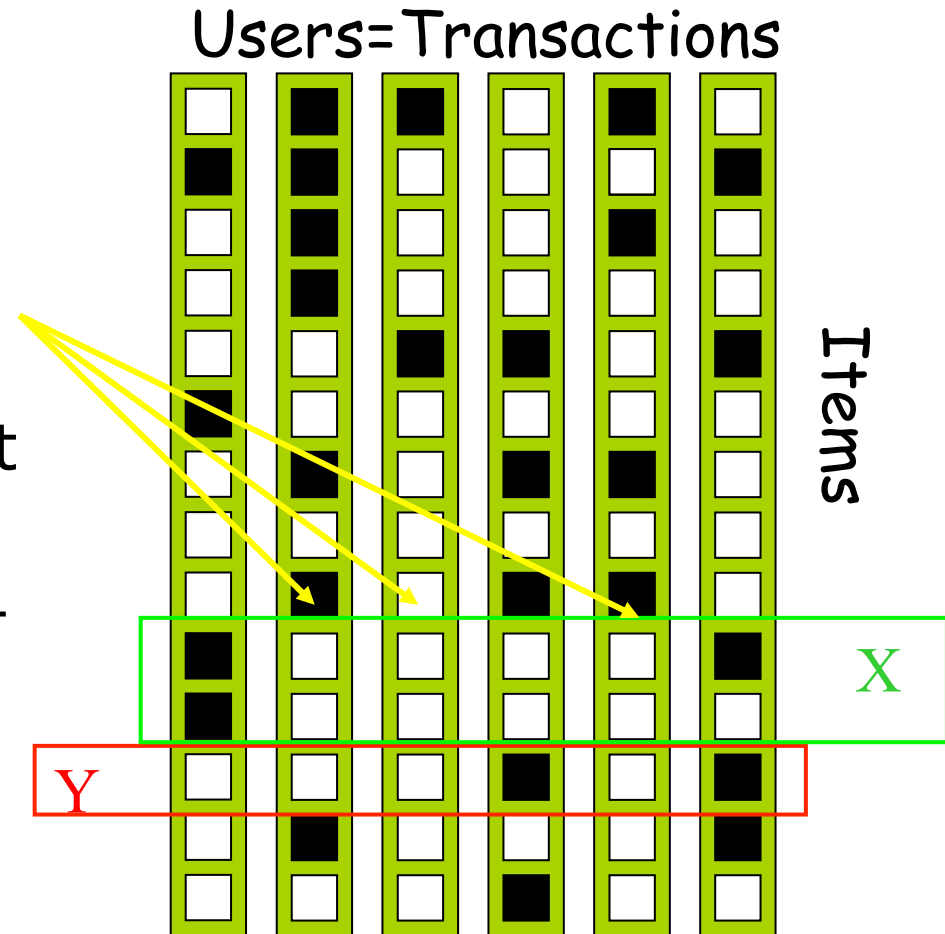
- Probability space  $(\Omega, F, P)$ :  $\Omega$  is the set of outcomes (states of nature),  $F$  is a set of subsets of  $\Omega$  (Events), and  $P$  is a probability measure
- Given a probability space  $(\Omega, F, P)$  and two events  $A$  and  $B$  with  $P(B) > 0$ , the conditional **probability of A given B** is defined by
  - $P(A|B) = P(A \cap B) / P(B)$
- If  $P(B) = 0$  then is undefined
- Two events  $A$  and  $B$  are statistically independent if  $P(A \cap B) = P(A) * P(B)$ 
  - Therefore in this case:  $P(A|B) = P(A)$ ,  $P(B|A) = P(B)$ .

# Support and Confidence

- **Support** is a measure of the **frequency** that a pattern (X and Y) is observed in the transactions
  - $S(X \rightarrow Y) = (\# \text{ transactions containing X and Y}) / (\# \text{ transactions})$
- **Confidence** is a measure of the **strength** of the rule, i.e., the probability that a *transaction that contains X will also contain Y*
  - $C(X \rightarrow Y) = (\# \text{ transactions containing X and Y}) / (\# \text{ transactions containing X})$
- Confidence is the **conditional probability** that Y can be observed in a transaction given the fact that we observed X
- $C(X \rightarrow Y) = S(X \rightarrow Y) / S(X)$ .

# Example

- Support = proportion of transactions that contains X and Y =  $3/6$
- Confidence = proportion of those that contains X **and** Y over those containing X =  $3/4$



# Example

<b>Beverage</b>	<b>Meat</b>	<b>Vegetables</b>	<b>Milk</b>	<b>Cafe</b>
Coke	Porky	Potato	No	No
Coke	Pork	Potato	Yes	No
Beer	Pork	Potato	No	Yes
Wine	Beef	Potato	No	Yes
Wine	Chicken	Tomato	No	Yes
Wine	Chicken	Tomato	Yes	No
Beer	Chicken	Tomato	Yes	Yes
Coke	Beef	Potato	No	No
Coke	Chicken	Tomato	No	Yes
Wine	Beef	Tomato	No	Yes
Coke	Beef	Tomato	Yes	Yes
Beer	Beef	Potato	Yes	Yes
Beer	Pork	Tomato	No	Yes
wine	Beef	Potato	Yes	No

# Item sets (some of them)

## One-item sets

beverage=coke (5)

beverage=beer (4)

beverage=wine(5)

meat=chicken (4)

meat=beef(6)

meat=pork (4)

## Two-item sets

Beverage=coke  
Meat=beef (2)

Beverage=coke  
Meat=pork (2)

Beverage=coke  
Vegetable=tomato  
(2)

Beverage=coke  
Vegetable=potato  
(3)

Beverage=coke  
milk=yes (2)

beverage=coke  
milk=no (3)

## Three-item sets

beverage=coke  
meat=pork  
vegetables=potato  
(2)

beverage=coke  
meat=pork  
cafe=no (2)

beverage=coke  
vegetables=tomato  
cafe=yes (2)

beverage=coke  
vegetables=potato  
milk=no (2)

vegetables=tomato  
milk=no  
cafe=yes (4)

beverage=coke  
vegetables=potato  
cafe=no (3)

## Four-item sets

beverage=coke  
meat=pork  
vegetables=potato  
cafe=no (2)

beverage=coke  
vegetables=potato  
milk=no  
cafe=no (2)

beverage=overcast  
meat=pork  
milk=no  
cafe=no (2)

beverage=rainy  
meat=beef  
milk=no  
cafe=yes (2)

beverage=rainy  
vegetables=tomato  
milk=no  
cafe=yes (2)

meat=cool  
vegetables=tomato  
milk=no  
cafe=yes (2)

# Rules

- **vegetables=tomato, milk=no, cafe=yes (4)** : leads to seven potential rules (why 7?)
  - If vegetables=tomato and milk=no → café=yes (4/4)
  - If vegetables=tomato and cafe=yes → milk=no (4/6)
  - If milk=no and café=yes → vegetables=tomato (4/6)
  - If vegetables=tomato → milk=no and café=yes (4/7)
  - If milk=no → vegetables=tomato and café=yes (4/8)
  - If café=yes → vegetables=tomato and milk=no (4/9)
  - If – → vegetables=tomato and milk=no and café=yes (4/14)
- The number in bracket is the confidence (the proportion of the transactions for which the rule is correct)
  - E.g.: (milk=no) is found 8 times but only in 4 cases the consequence (vegetables=tomato and café=yes) is true.
- The support of these rules is 4/14 (# transactions containing X and Y) / (# transactions)

# Association Rules and Recommender

- **Long term user profile:** a transaction for each user containing all the products both in the past
  - **Short term user profile:** a transaction for each bundle of products bought during a shopping experience (e.g. a travel)
1. Build a set of association rule (e.g. with the "Apriori" algorithm) with at least a **minimum** confidence and support (*using all the users or only the users close to the target*)
    - [Sarwar et al., 2000] used all the users-transactions to build the association rules
  2. Find the rules R supported by a user profile (i.e. X is in the user profile)
  3. Rank a product in the right-hand-side of some rules in R with the (maximal) confidence of the rules that predict it
  4. Select the top-N

[Sarwar et al., 2000]

# Classification Learning

<b>outlook</b>	<b>temperature</b>	<b>humidity</b>	<b>windy</b>	<b>Play/CLASS</b>
sunny	85	85	FALSE	no
sunny	80	90	TRUE	no
overcast	83	86	FALSE	yes
rainy	70	96	FALSE	yes
rainy	68	80	FALSE	yes
rainy	65	70	TRUE	no
overcast	64	65	TRUE	yes
sunny	72	95	FALSE	no
sunny	69	70	FALSE	yes
rainy	75	80	FALSE	yes
sunny	75	70	TRUE	yes
overcast	72	90	TRUE	yes
overcast	81	75	FALSE	yes
rainy	71	91	TRUE	?

Set of classified examples

Given a set of examples for which we know the class predict the class for an unclassified examples.

# Regression Learning

<b>outlook</b>	<b>temperature</b>	<b>humidity</b>	<b>windy</b>	<b>Play duration</b>
sunny	85	85	FALSE	5
sunny	80	90	TRUE	0
overcast	83	86	FALSE	55
rainy	70	96	FALSE	40
rainy	68	80	FALSE	65
rainy	65	70	TRUE	45
overcast	64	65	TRUE	60
sunny	72	95	FALSE	0
sunny	69	70	FALSE	70
rainy	75	80	FALSE	45
sunny	75	70	TRUE	50
overcast	72	90	TRUE	55
overcast	81	75	FALSE	75
rainy	71	91	TRUE	?

Set of examples - target feature is numeric

Given a set of examples for which we know the target (duration), predict it for examples where it is unknown.

# Learning

- In order to predict the class several Machine Learning techniques can be used, e.g.:
  - Naïve Bayes
  - K-nn
  - Perceptron
  - Neural Network
  - Decision Trees (ID3)
  - Classification Rules (AC4.5)
  - Support Vector Machine

# Matrix of ratings



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
a			1		4	5			4		3					2			4		2				
b			4								3						5	1		3					
c		5		4			4						3		5					4		5			
d								3				5				3			4		2			3	
e		3					5			4	5				5					1			5	4	
f			4				1		3	5		4	1		5	4	4		4				3		
g	2	4			4		2				5		1	4	5		4	2	4		5			4	
h			2		1		4		3	5		4	2		5	4	5						5		
i			1				3			5					5	4	4		5			4		3	
j			4			4				5		1		5		4		4					4		
k		5				4			2		5		1	5		4		2		4				2	
l					3			3				4	1		4		4	2	4					3	
m	5		3					5	3		5	4		5	5	3			4	4	5	4		4	
n			1		4	5				4	5		1	5		4		3		4		4	3		
o			4			4				5		4		5			4	2		5		5		3	
p				4				5							5	4		2	4	4	5	4		2	
q					3			3					1	5		4	4		4			4		3	
r		4			1	4		2					2		5		4				5	4		4	
s			2		4		4			5			1			4		2	4		4		5		
t		1		4			3					4		5	5		4			4				3	
u			2		1		4		3				1		5	4		2	4		5	4			
v					4	5				4	3		5			2					2			5	
w				2			2		3			5			4	5		4	2		3	4			
x	4			5				3		3				4	5						1				
y			1			3				2	3							3	3		5	4			



# Recommendation and Classification

- ❑ Users = instances (*or items are the instances?*)
- ❑ Class = any product for which you'd like to predict the actual rating
- ❑ Class-value = rating
- ❑ Differences
  - Data are sparse
  - There is no preferred class-item to predict
  - **In fact, the main problem is related to determining the classes (items) that it is better to predict**
  - You cannot (?) predict all the class values and then choose the classes-items with higher class values (too expensive)
  - Unless there are few items (? Generalize the notion of item ?).

# Lazy and Eager Learning

- **Lazy:** wait for query before generalizing
  - *k*-Nearest Neighbor, Case based reasoning
- **Eager:** generalize, i.e., build a model, before seeing query
  - Radial basis function networks, ID3, Neural Networks, Naïve Bayes, SVM
- Does it matter?
  - Eager learner must create global approximation
  - Lazy learner can create many local approximations

# Model-Based Collaborative Filtering

- Previously seen approach is called “lazy” or memory-based as the original examples (the vectors of user ratings) are used when a prediction is required (no computation when data is collected)
- Model based approaches build and store a probabilistic model and use it to make the prediction

$$v_{ij}^* = E(v_{ij}) = \sum_{r=1}^5 r * P(v_{ij} = r \mid \{v_{ik}, k \in I_i\}) \quad \text{User model}$$

- Where  $r=1, \dots, 5$  are the possible values of the rating and  $I_i$  is the set of (indexes of) products rated by user  $I$
- $E(X)$  is the Expectation (i.e., the average value of the random variable  $X$ )
- The probabilities above are estimated with a classifier producing the probability for an example to belong to a class (the class of products having a rating =  $r$ ), e.g., Naïve Bayes (but also  $k$ -nearest neighbor!).

# Naïve Bayes

- $P(H|E) = [P(E|H) * P(H)] / P(E)$
- Example:
  - $P(\text{flue} | \text{fever}) = P(\text{fever} | \text{flue}) * [P(\text{flue}) / P(\text{fever})]$
  - $P(\text{flue} | \text{fever}) = 0.99 * P(\text{flue}) / P(\text{fever}) = 0.99 * 0.01 / 0.03 = 0.33$

- In case of ratings, a class is a particular rating for a particular product (e.g., the first product).  $X_i$  is a variable representing the rating of a generic user for product  $i$

$$P(X_1 = r | X_2 = v_2, \dots, X_n = v_n) = \frac{P(X_2 = v_2, \dots, X_n = v_n | X_1 = r)P(X_1 = r)}{P(X_2 = v_2, \dots, X_n = v_n)}$$

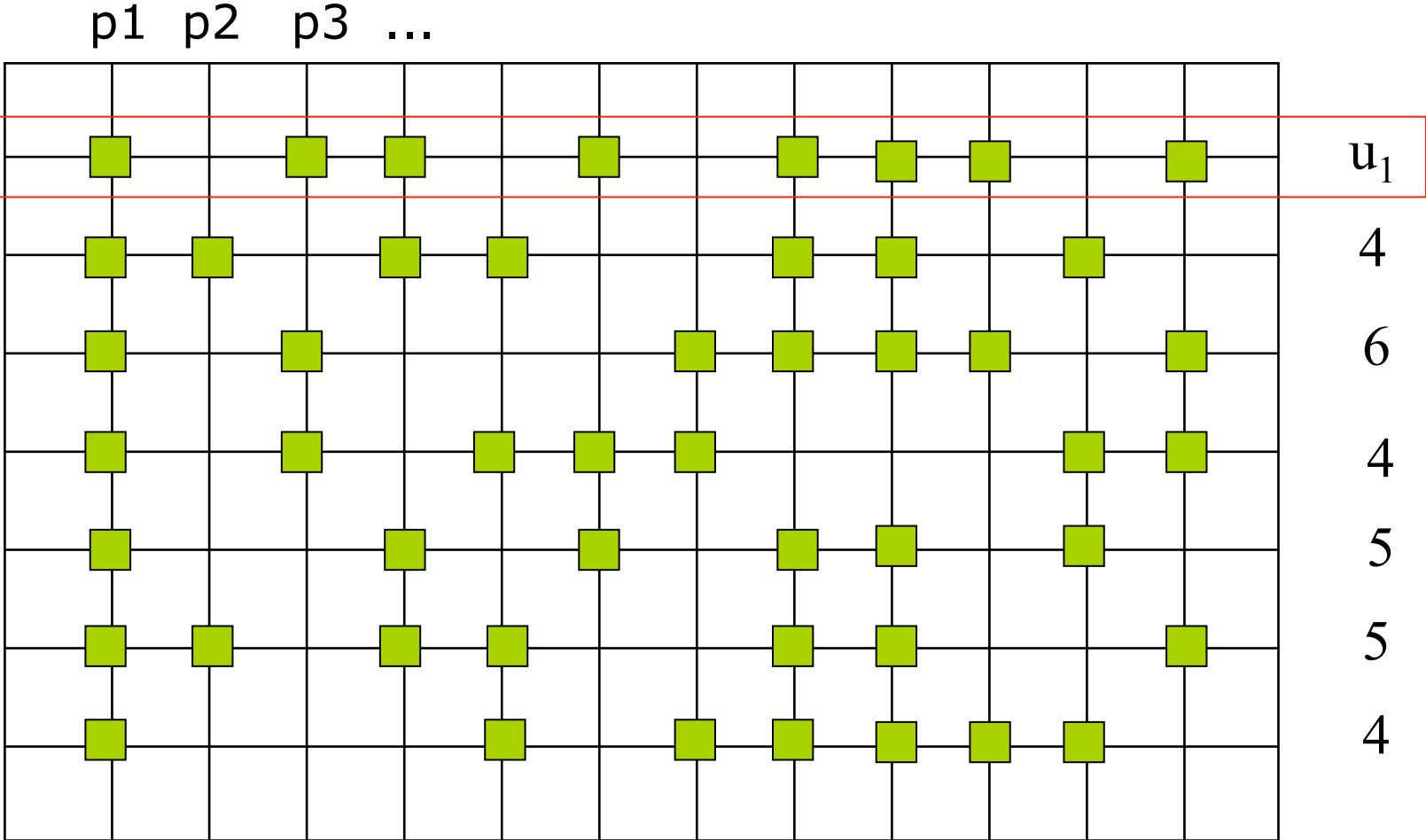
- Assuming the independence of the ratings on different products

$$P(X_1 = r | X_2 = v_2, \dots, X_n = v_n) = \frac{\prod_{j=2}^n P(X_j = v_j | X_1 = r)P(X_1 = r)}{P(X_2 = v_2, \dots, X_n = v_n)}$$

# Improvements of CF

- Using a **significance weighting** that tells how dependable the measure of similarity is – measured as a count of the number of items used in the similarity computation [Herlocker et al., 1999]
  - $\text{Min}(|P_i \cap P_j|, 50)/50$  : where  $P_i, P_j$  are the product rated by user  $i$  and  $j$  (50 is a parameter – 50 items are considered enough to have optimal similarity computation)
  - This approach **improves** the prediction accuracy (MAE)
- Not all items may be informative in the similarity computation – items that have low variance in their ratings are supposed to be less informative than items that have more diverse ratings [Herlocker et al., 1999]
  - He gave more importance, in the similarity computation, to the products having larger variance in ratings – this did **not improve** accuracy of the prediction.

# Dependability in the similarity measure



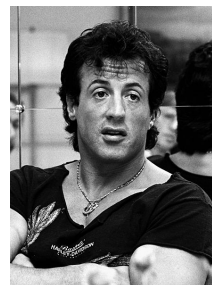
Overlapping ratings with  $u_1$

■ Means that there is a rating for that user-item pair

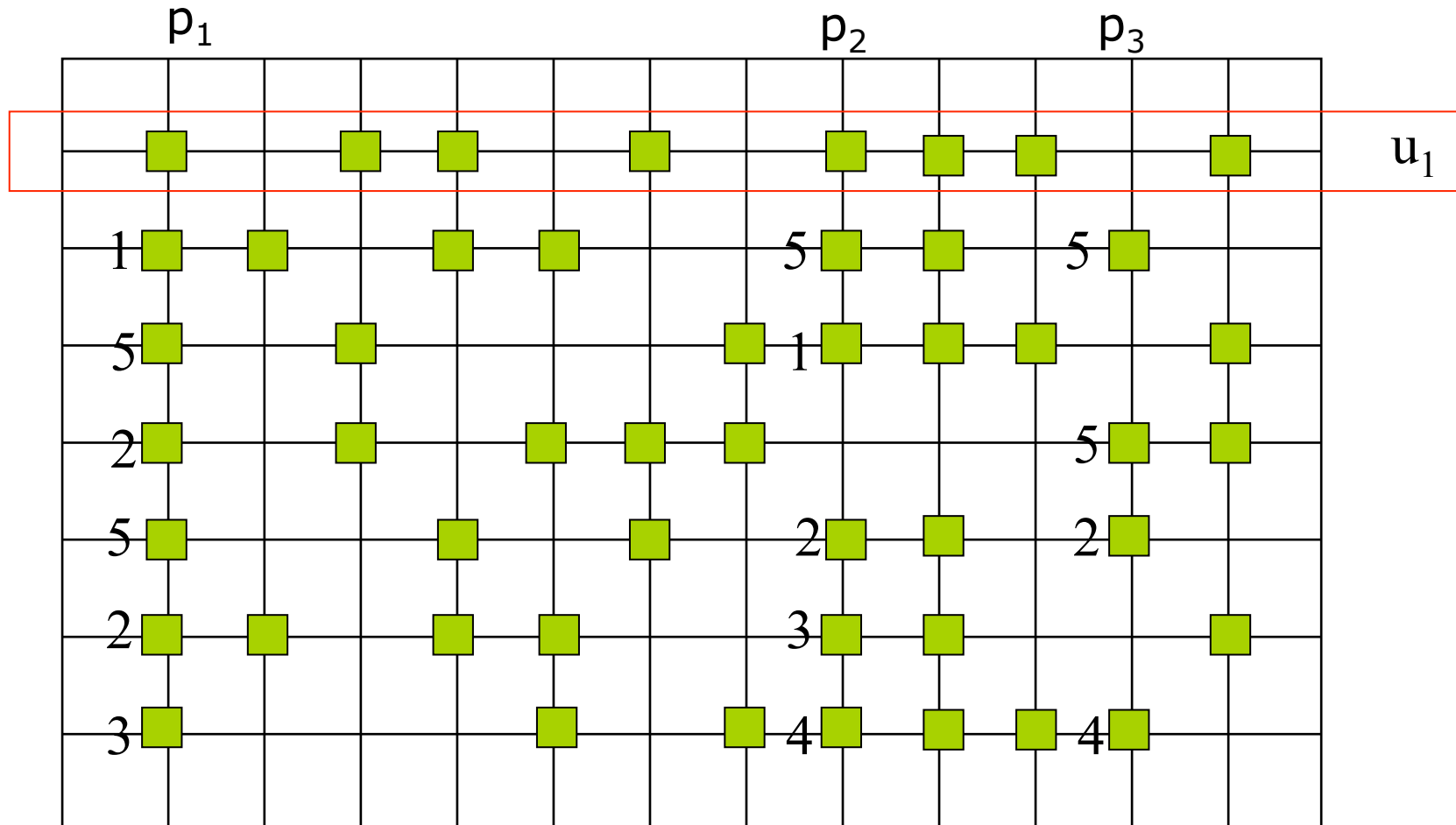


# Adaptive Similarity Metric

- Is it possible to identify - for each user-item pair the best set of ratings upon which to compute the user-to-user similarity?
- Feature selection
- **Example:** when predicting a rating for a movie by Schwarzenegger use in the user-to-user similarity computation only the ratings for other movies by Schwarzy and Stallone.



# Correlated vs. De-correlated items

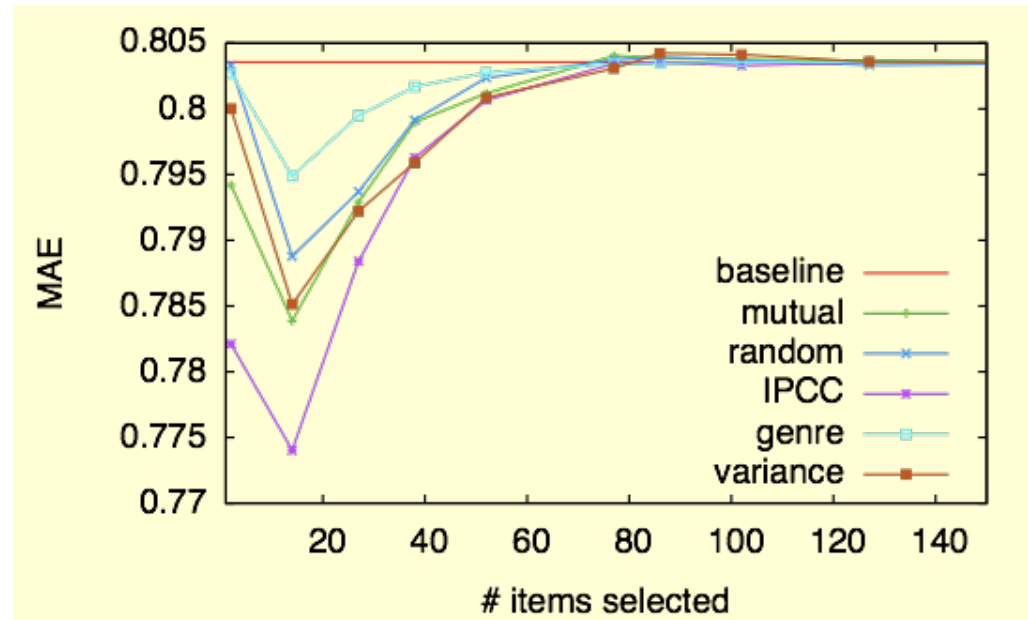


Assume that we want to predict the rating of user  $u_1$  for item  $p_3$ .  
 Would it be more useful to base the similarity on the ratings  
 for item  $p_1$  or  $p_2$ ?

# Item Selection

- In **adaptive** user-to-user similarity use item  $p$  if:
  - there are ratings for item  $p$  in both users' profiles
  - the item  $p$  has a **great correlation/importance** for the item whose rating has to be predicted (target)
- **Items Correlation Weights**
  - **Variance Weighting:** the larger the variance of the item, the bigger is the weight
  - **IPCC:** the larger is the Pearson correlation, the larger the weight
  - **Mutual Information:** weight is computed as the information that predictive item provides to the knowledge of target item
  - **Genre weighting:** the more genres the two items share the larger is the weight.

# Performance of Adaptive Item Selection



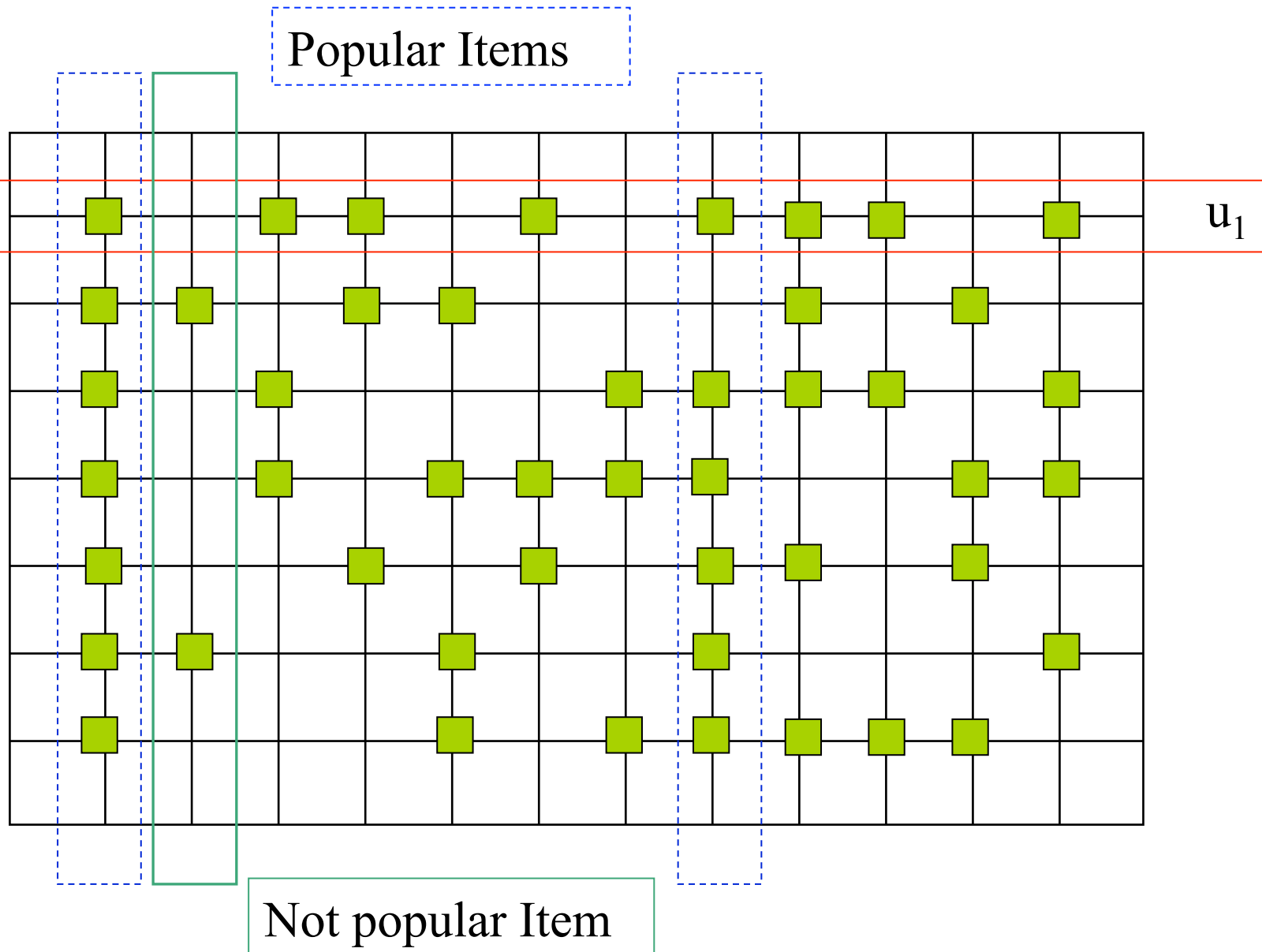
Movielens  
Dataset

- ❑ Less data can lead to a better prediction
- ❑ Selecting a small number of items overlapping in the profiles of the users whose similarity is computed improves the accuracy.

# Problems of CF : Sparsity

- Typically we have large product sets and user ratings for a small percentage of them
- Example Amazon: millions of books and a user may have bought hundreds of books –
  - the probability that two users that have bought 100 books have at least a common book (in a catalogue of 1 million books) is 0.01 (with 50 and 10 millions is 0.0005) – if all books are equally likely to be bought!
  - Exercise: what is the probability that they have 10 books in common.

# Sparsity



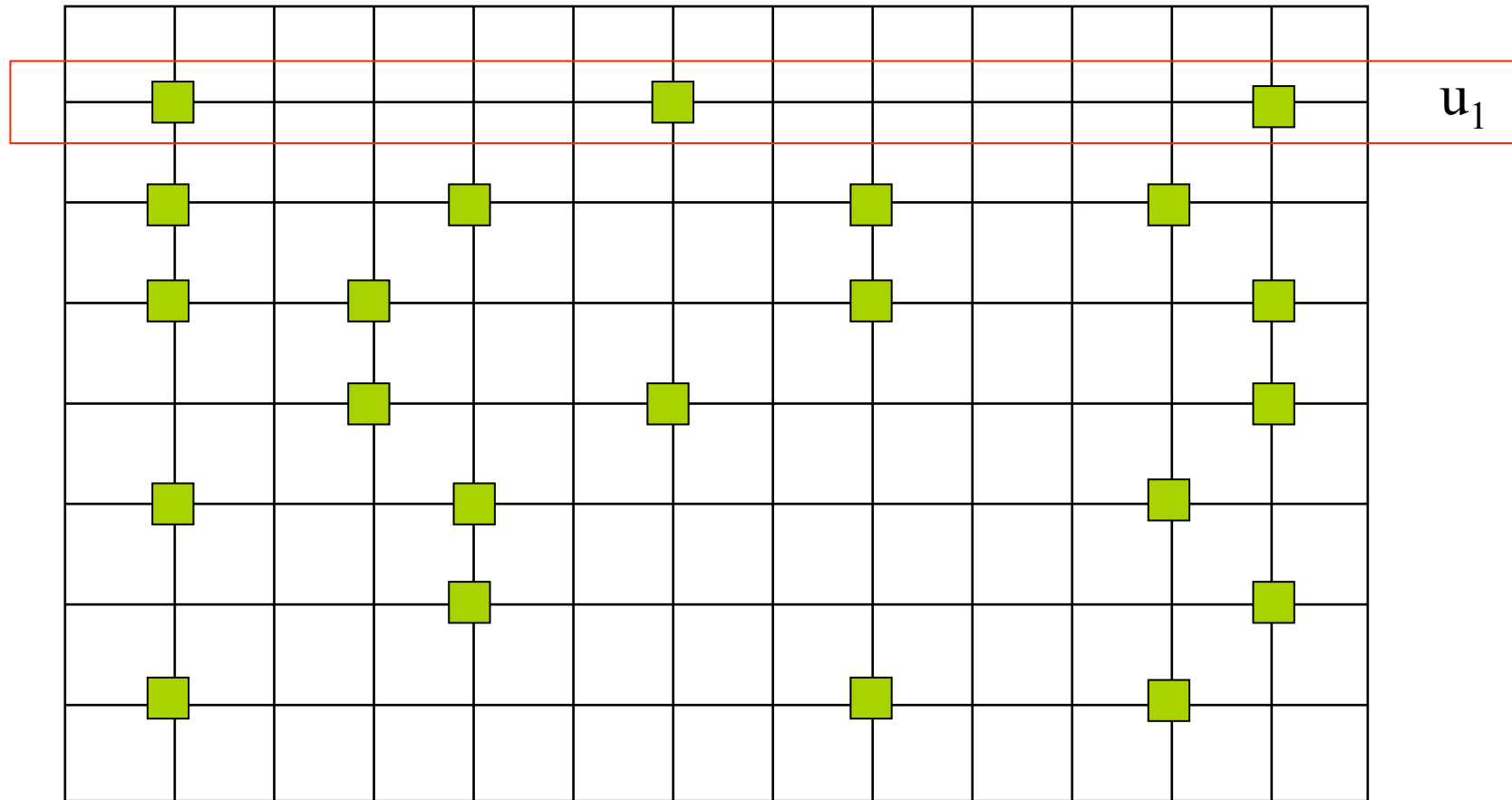
# Popular vs Not Popular

- Predicting the rating of popular items is easier than for not popular ones
  - The prediction can be based on many neighbors
- The usefulness of predicting the rating of popular items is questionable:
  - It could be guessed in a simpler way
  - The system will not appear as much smart to the user
- Predicting the rating of unpopular items is
  - Risky: not many neighbors on which to base the prediction
  - But could really bring a lot of value to the user!

# Problems of CF : Scalability

- Nearest neighbor algorithms require computations that grows with both the number of customers and products
- With millions of customers and products a web-based recommender will suffer serious scalability problems
- The **worst case complexity** is  $O(m*n)$  ( $m$  customers and  $n$  products)
- But in practice the complexity is  $O(m + n)$  since for each customer only a small number of products are considered (one loop on the  $m$  customers to compute similarity PLUS one on the  $n$  products to compute the prediction).

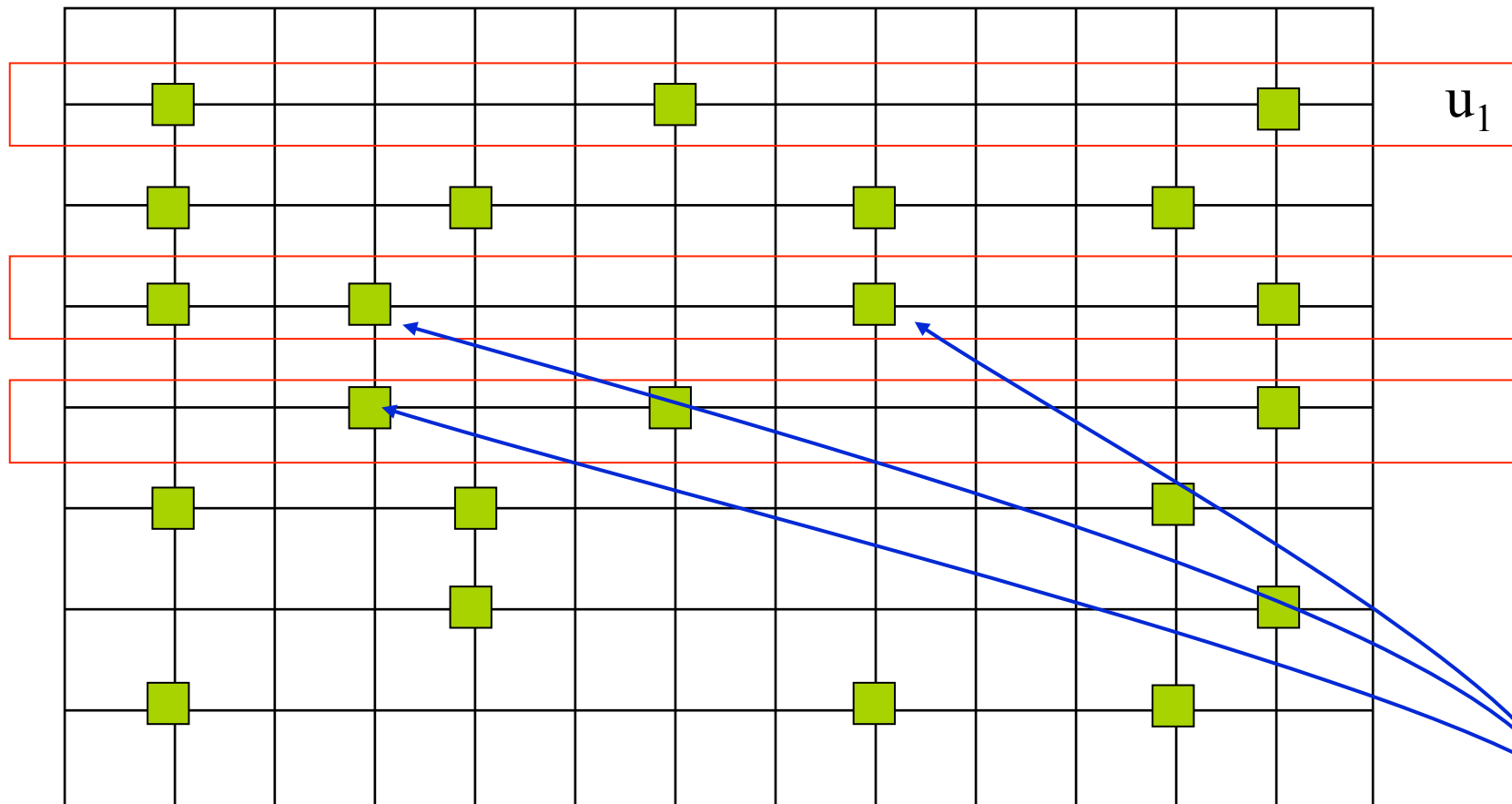
# Computational Issues



To compute the similarity of  $u_1$  with the other users we must scan the users database  $m$  (large) but only 3 products will be considered (in this example)

Represent a user as  $u_1 = ((1, v_{1_1}), (6, v_{1_6}), (12, v_{1_{12}}))$

# Computational Issues



When you have selected the neighbors

$$u_3 = ((1, v_{3_1}), (3, v_{3_3}), (8, v_{3_8}), (12, v_{3_{12}}))$$

$$u_4 = ((3, v_{4_3}), (6, v_{4_6}), (12, v_{4_{12}}))$$

You must only scan the **union of the products in the neighbors'** profiles and **identify those not yet rated** by the target user  $u_1$

## Some Solutions for Addressing the Computational Complexity

- ❑ Discard customers with few purchases
- ❑ Discard very popular items
- ❑ Partition the products into categories
- ❑ Dimensionality reduction (LSI or clustering data)
- ❑ *All of these methods also reduce recommendation quality (according to [Linden et al., 2003]).*

# Summary

- Example of usage of precision and recall in the evaluation of a CF system
- CF using only the presence or absence of a rating
- Association rules
- Comparison between CF and association rules
- Illustrated the relationships between CF and classification learning
- Briefly discussed the notion of model-based CF
- Naïve Bayes methods
- Discussed some problems of CF recommendation: sparsity and scalability
- Discussed the computational complexity of CF

# Questions

- ❑ What are the two alternative approaches for selecting the nearest neighbors, given a user-to-user similarity metric?
- ❑ How could be defined a measure of reliability of the similarity between two users?
- ❑ What is the computational complexity of a naïve implementation of the CF algorithm? What is its complexity in practice?
- ❑ How CF compares with association rules?
- ❑ What are the main differences between the recommendation learning and classification learning?
- ❑ How CF would work with very popular or very rare products?
- ❑ Will CF work better for popular or not popular items?