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.

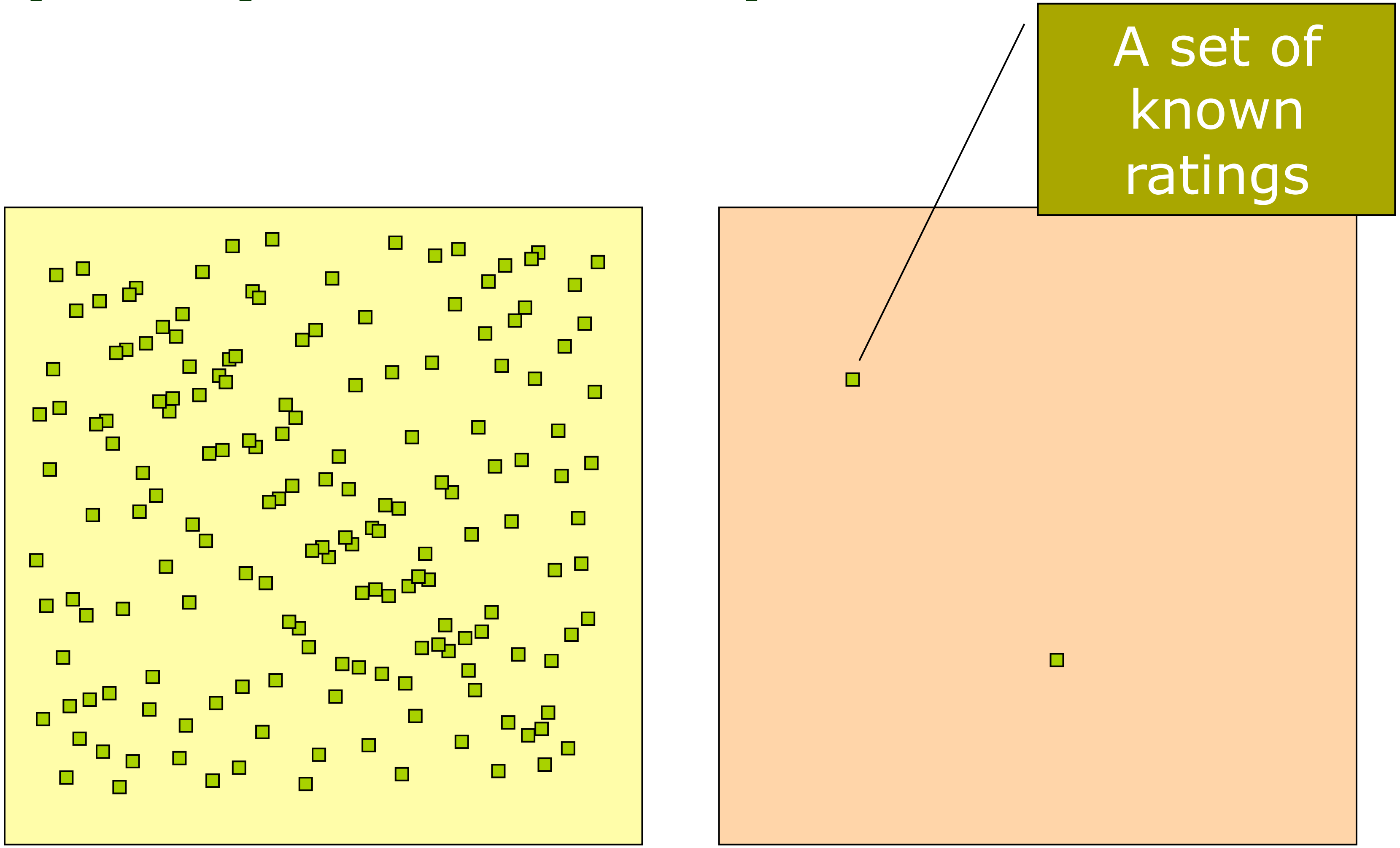
Example of Evaluation of a Collaborative Filtering Recommender System

- **Movie data:** 35,000 users, 3,000 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
  - #missing-ratings/#all-possible-ratings.

All the possible ratings

[Sarwar et al., 2000]

Sparsity: visual example



94% sparsity

99,9% sparsity

Evaluation Procedure

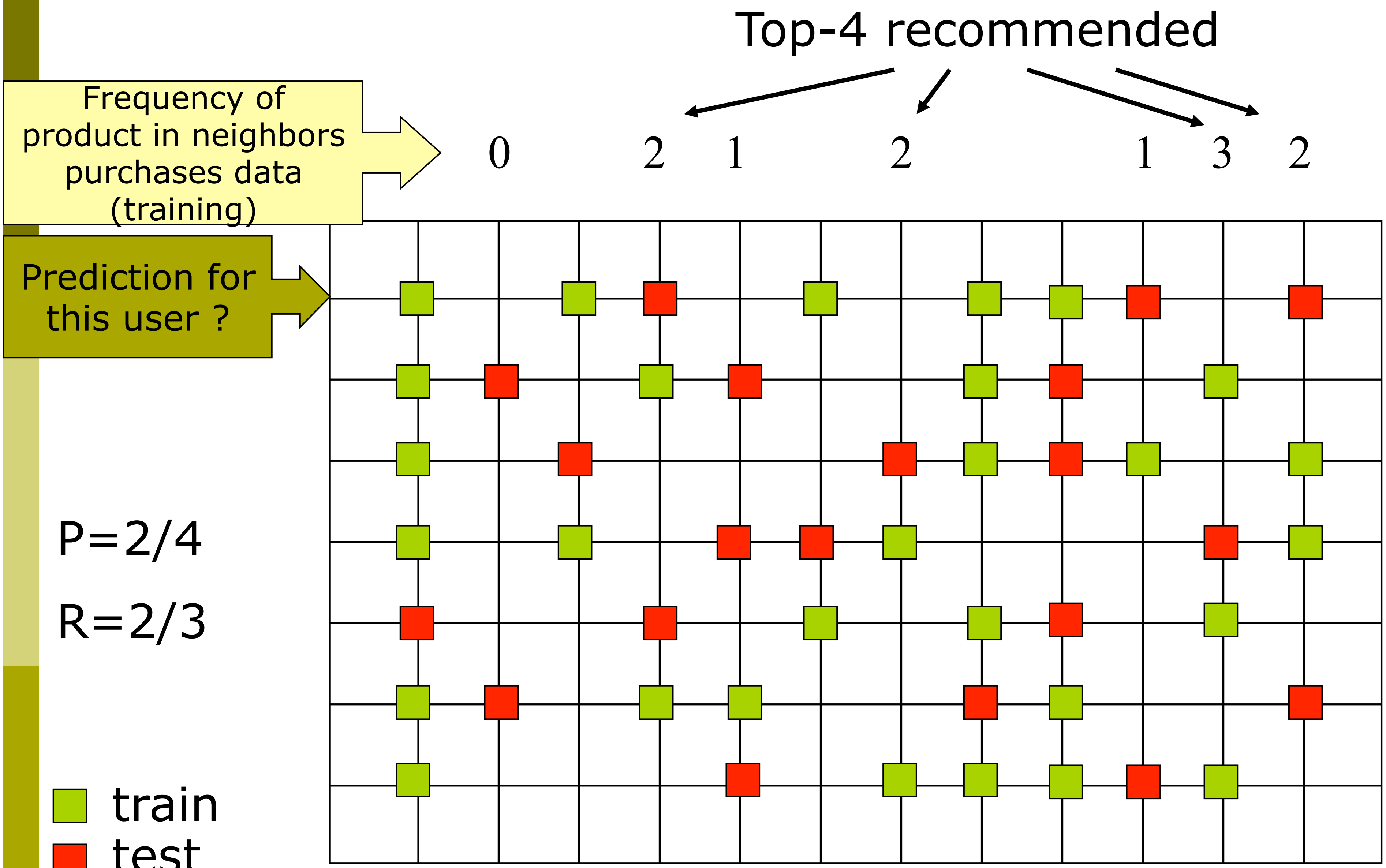
- They evaluate top-N recommendation (10 recommendations for each user)
- Separate ratings in training and test sets (80% Train - 20% Test)
- Use the training to compute the prediction (top-N)
- Compare (precision and recall) the items **in the test set of a user** with the top N recommendations **for 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 rated items are relevant*
- They used the cosine metric to find the neighbors.

Generation of recommendations

- Instead of using the weighted average of the ratings

$$r^*_{uj} = r_u + K \sum_{v \in N_j(u)} w_{uv} (r_{vj} - r_v)$$

- 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.
- **Most-frequent item recommendation** computes a recommendation list without computing any rating estimation.



P=2/4

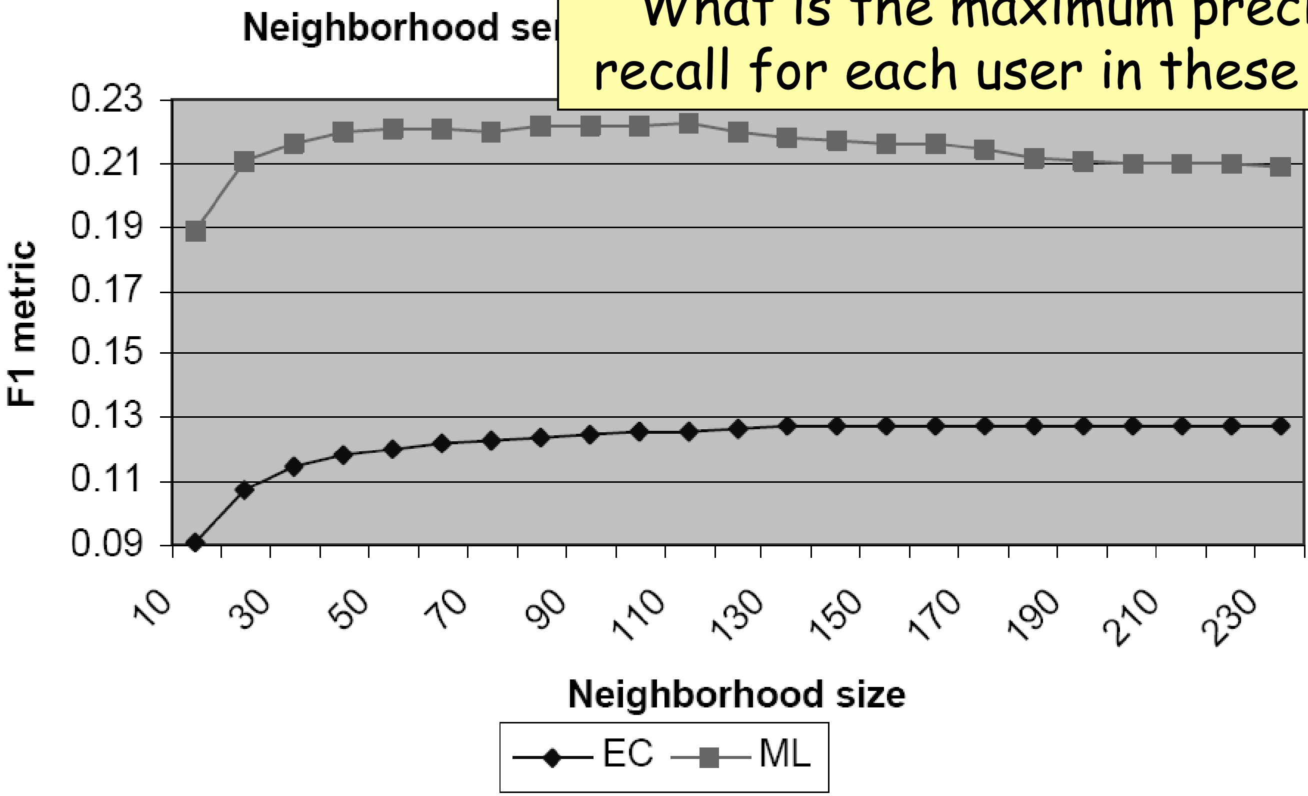
R=2/3

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 precision and recall for each user in these datasets?



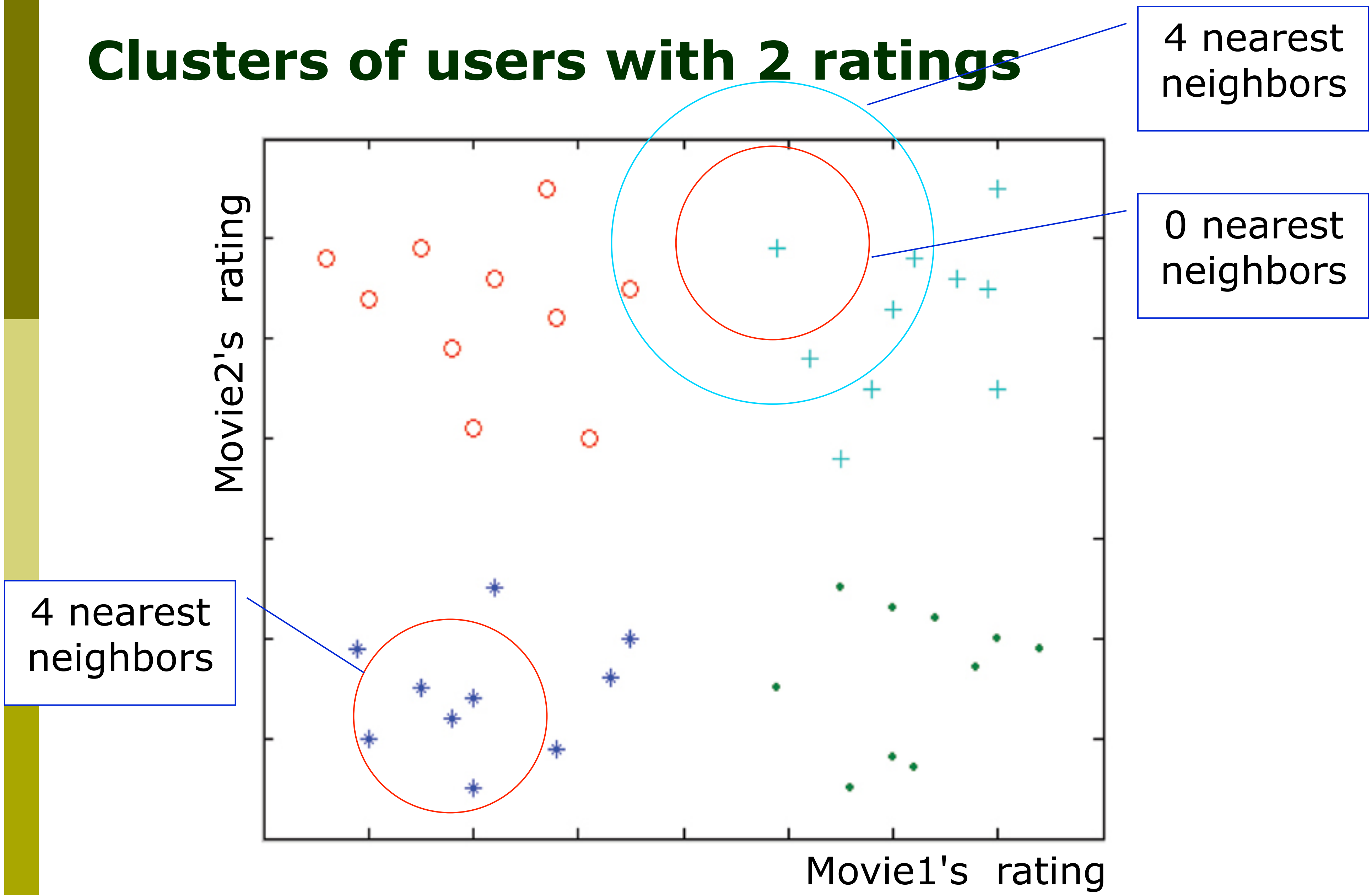
EC = eCommerce data; ML = MovieLens data

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

Top 10 recommendations

EC users rated 14.9 items (avg) – ML users rated 106 items (avg)

Clusters of users with 2 ratings



- Fixed similarity threshold in two different points (users) may mean completely different neighbors

Neighbor Size

- Reducing the neighbor size is important for performance considerations (why?)
- If the neighbor size k 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 **fixed k:**
  - Accuracy for users with more unique preferences will be lower – their k-nn are far away
- Selection can be made with a **threshold similarity**: the drawback is that as the number of ratings increases we may have too many neighbors

Remember the discussion on knn optimality

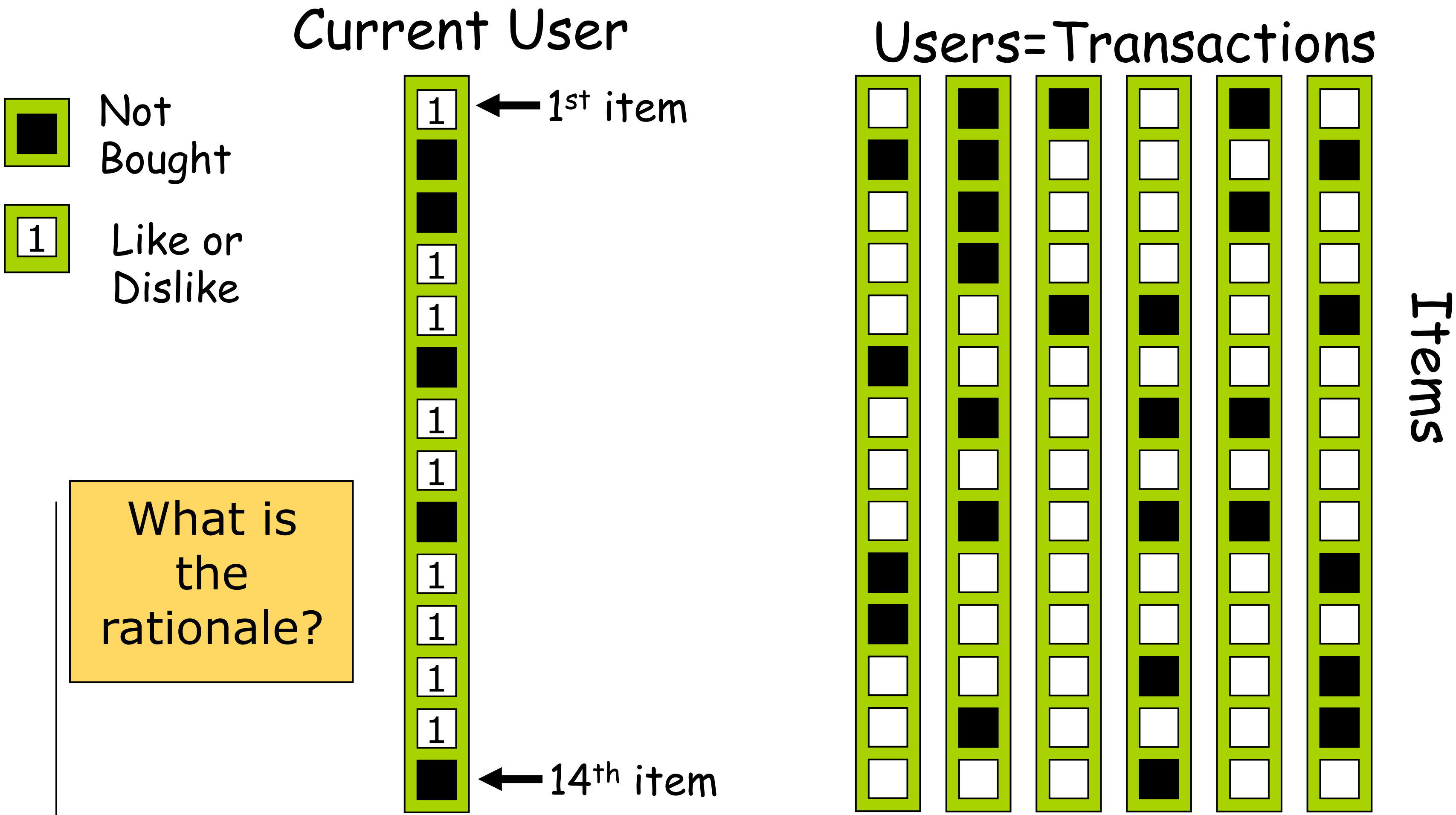
Neighbor Size (II)

- When using Pearson correlation it is common to discard neighbors with negative similarity
- *Advanced techniques use "adaptive" neighbor formation algorithm – the size depends on the global data characteristics and the user and item specific ratings.*

Association Rules

- Discovering association between sets of co-purchased products – the presence of a set of products “implies” the presence of others
- $\{p_1, \ldots, p_m\} = P$ 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 ($j=1, \ldots N$), 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/purchased by the user**.

Transactions and Ratings

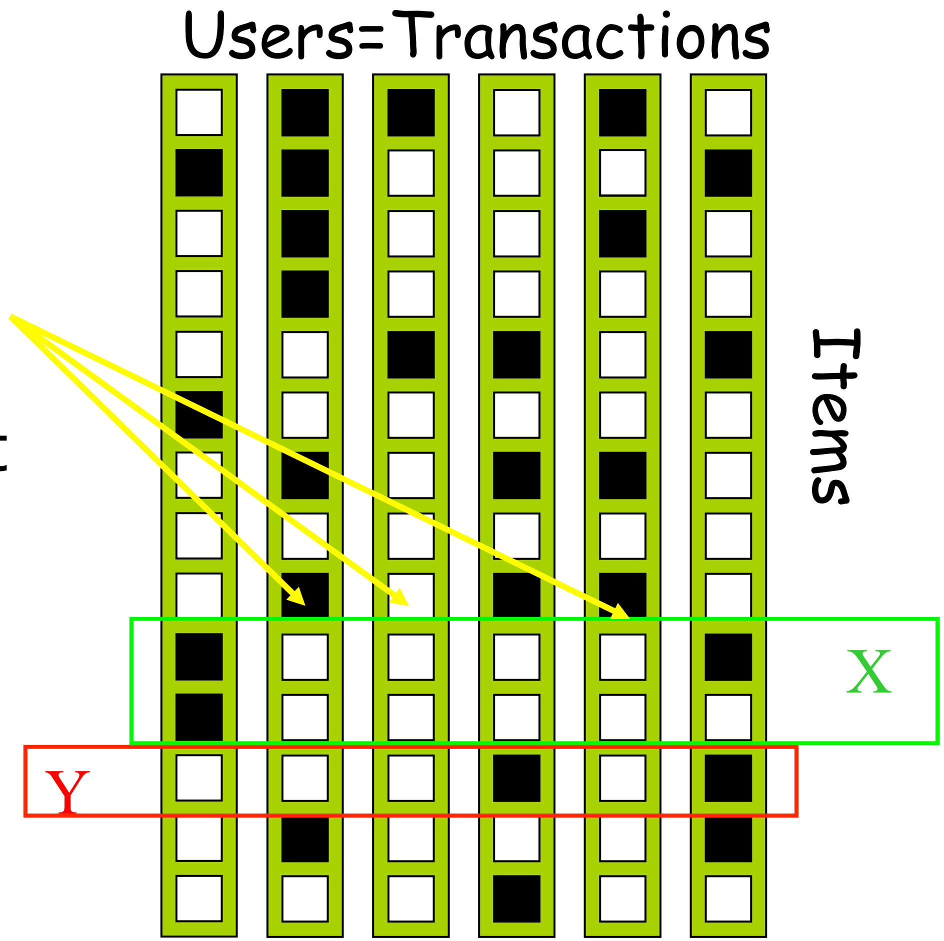


What is the rationale?

In [Sarwar et al., 2000] a transaction is made of all the products bought/rated by a user - not exploiting the rating values.

Example - X → Y

- **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



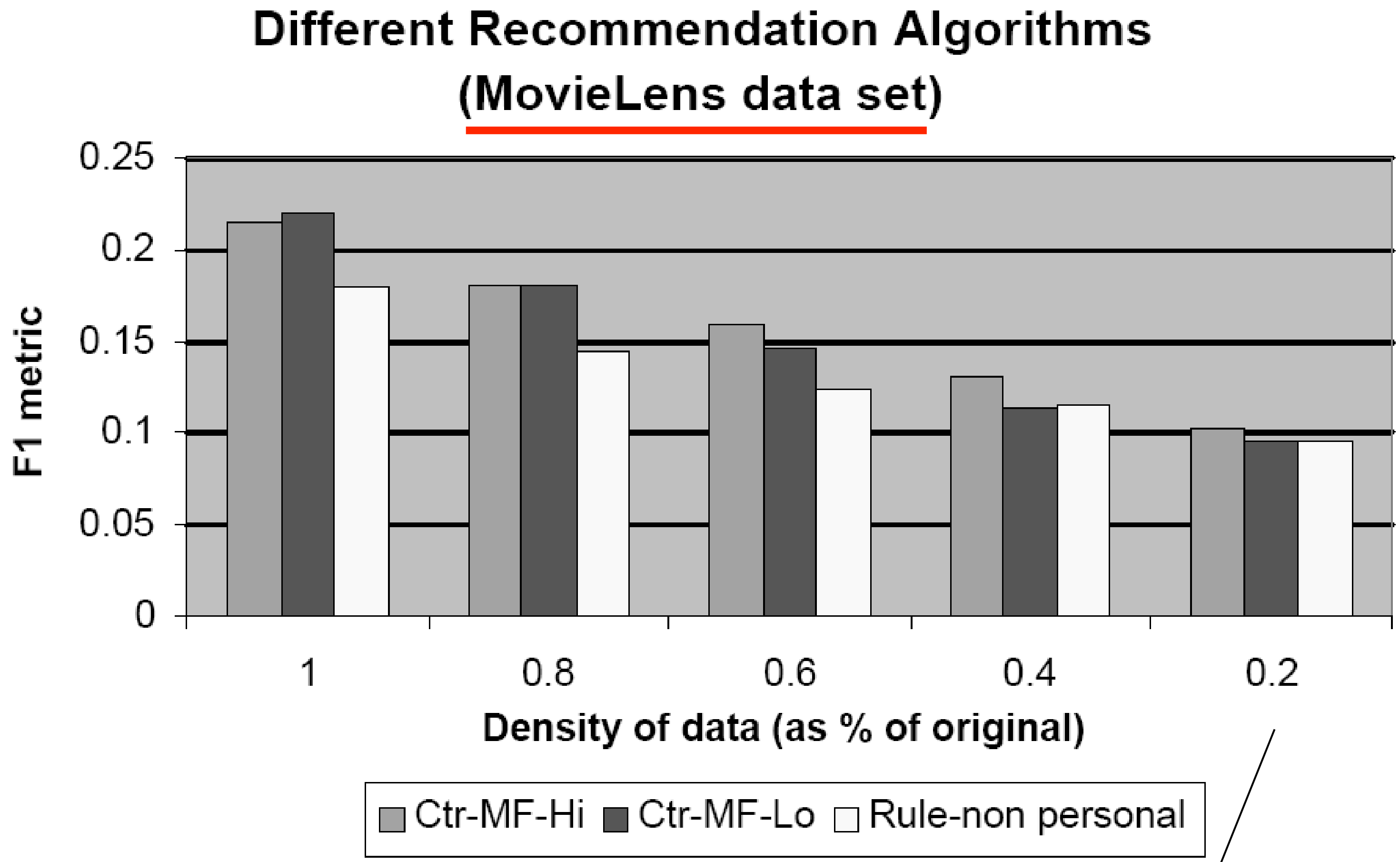
Association Rules and Recommender

- **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
   - [Sarwar et al., 2000] used all the users-transactions to build the association rules
   - You may *use only the users close to the target (kind of mix between AR and CF)*
2. Find the rules R supported by a user profile: X is in the **training** part of 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

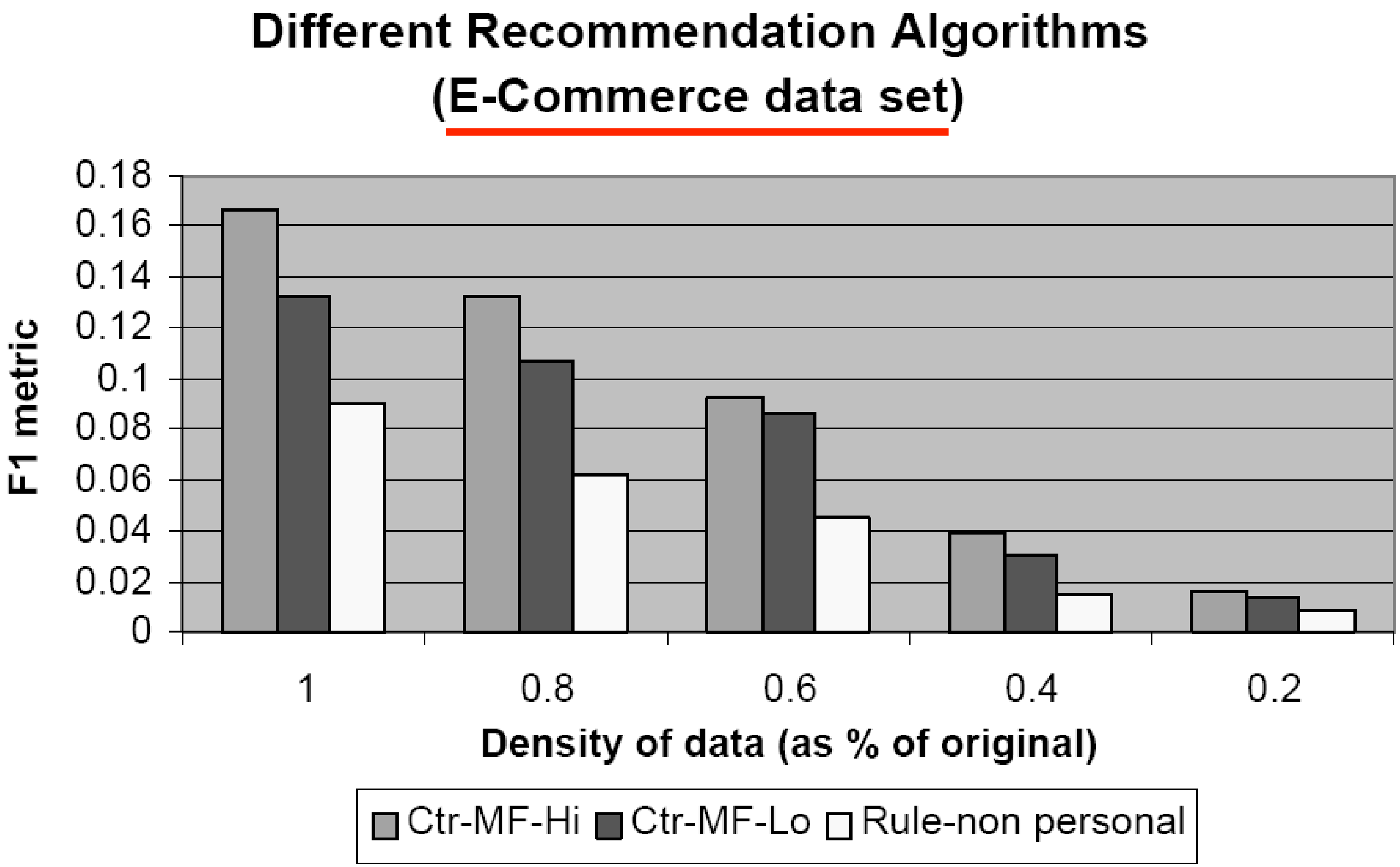
[Sarwar et al., 2000]

Comparison with Association Rules



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

Comparison with Association Rules



Classification Learning

outlook	temperature	humidity	windy	Play/CLASS
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

outlook	temperature	humidity	windy	Play duration
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

Items

Users

	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 described by their ratings
- Class = the **rating** given by the users/instances to a target **product**
- *Or items=instances and class = the rating to the items/ instances given by a target user*
- Differences with classical ML problems
  - Data are sparse
  - There is no preferred class/item-rating to predict
  - **In fact, the main problem is related to determining the classes (item ratings) that it is better to predict**
  - You cannot (?) predict all and then choose the item with higher predicted ratings (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 (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:

$$r_{uj}^{*} = E[r_{uj}] = \sum_{r=1}^{5} r * P(r_{uj} = r \mid \boxed{\{r_{uk}, k \in I_u\}}) \quad \text{User model}$$

- Where r=1, …5 are the possible values of the rating and I_u is the set of items rated by user u
- *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(H) *[P(E|H) / P(E)]
- Example:
  - P(flue | fever) = P (flue) * [P(fever | flue) / P(fever)]
  - P(flue | fever) = P(flue) * [0.99 / 0.03] = 0.01 * 33= 0.33
- A class variable is the rating for a particular item (e.g., the first item): X_i is a variable (feature) representing the rating for product i

$$P(X_1 = r \mid X_2 = r_{u2}, \ldots, X_n = r_{un}) = \frac{P(X_2 = r_{u2}, \ldots, X_n = r_{un} \mid X_1 = r) P(X_1 = r)}{P(X_2 = r_{u2}, \ldots, X_n = r_{un})}$$

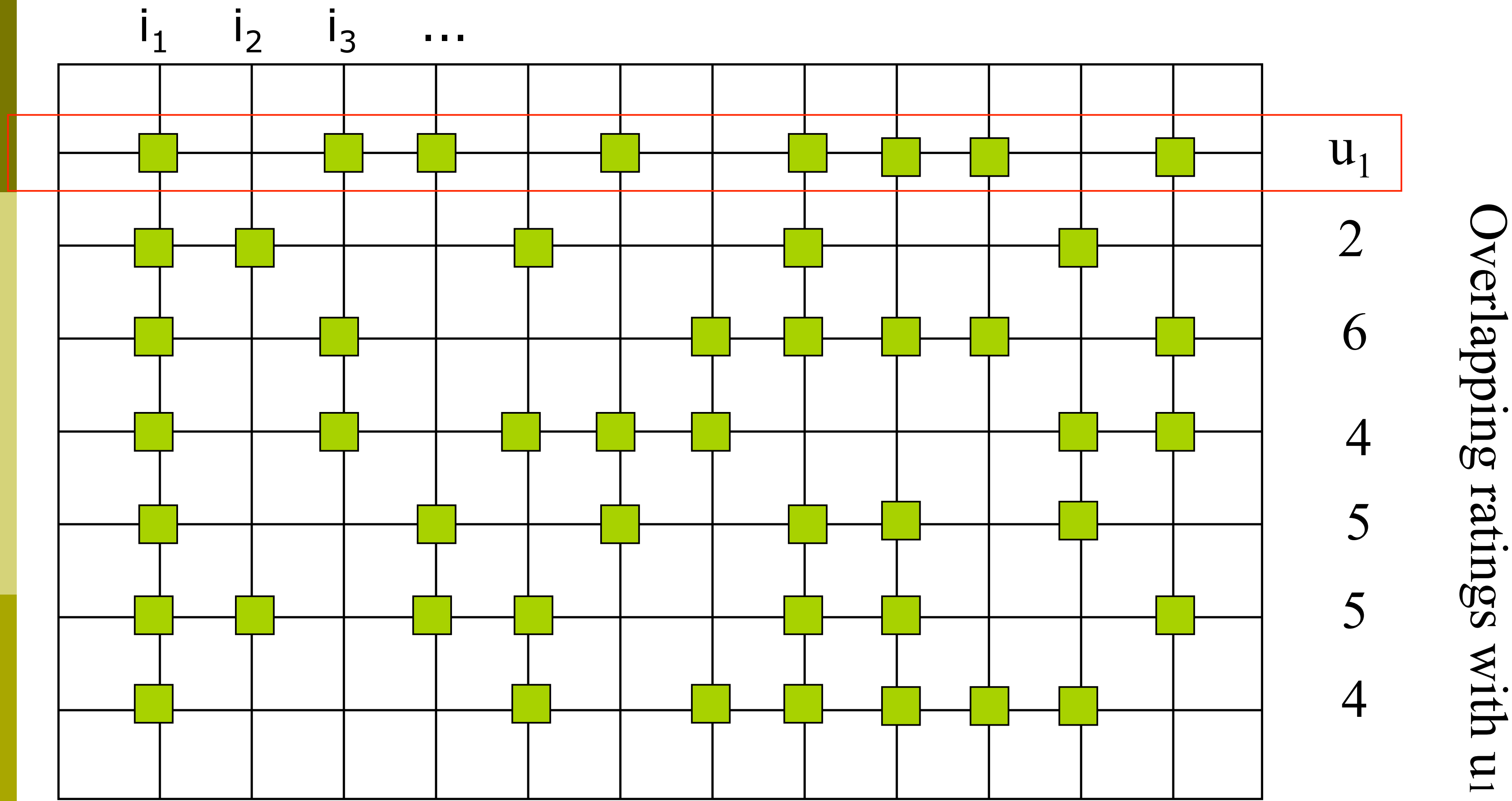
- Assuming the independence of the ratings on different products

$$P(X_1 = r \mid X_2 = r_{u2}, \ldots, X_n = r_{un}) = \frac{\prod_{j=2}^{n} P(X_j = r_{uj} \mid X_1 = r) P(X_1 = r)}{P(X_2 = r_{u2}, \ldots, X_n = r_{un})}$$

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 them:
  - The probability that two users, who have rated 100 books, have at least a common rated book (in a catalogue of 1 million books) is ~0.01 (with 50 ratings and 10 millions books is 0.00025) – if all books are equally likely to be bought!
  - Exercise: what is the probability that they have 10 books in common (stattrek.com/Tables/Binomial.aspx)
- Hence, if you have not a large set of users it may be difficult to find out a single neighbor for a target user
- But if there are 10,000,000 users then one can easily find $10^7 * 0.00025 = 2{,}500$ neighbors!

Reliability of the similarity measure



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

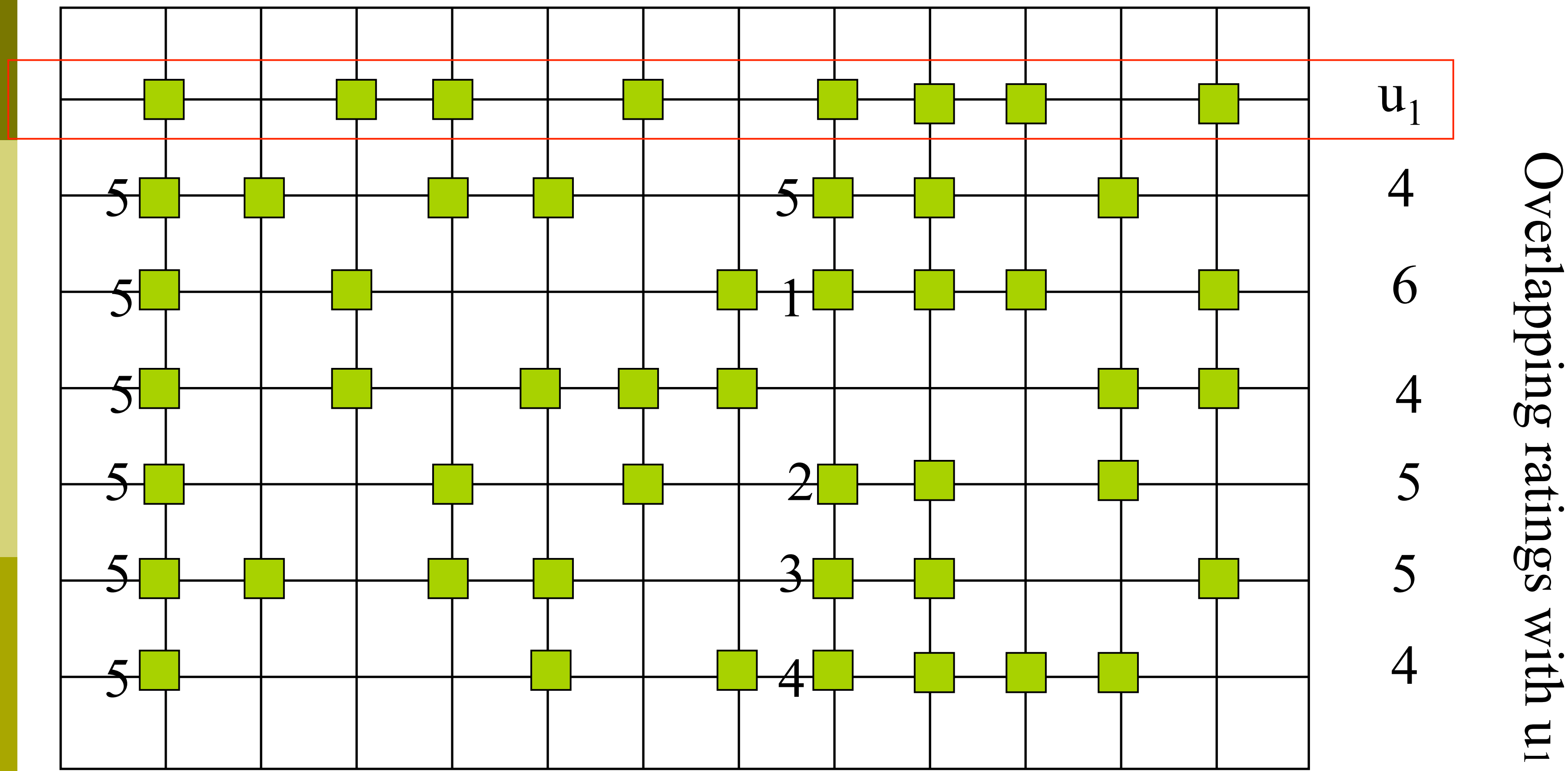
Significance of User-to-User Similarity

- Rating data are typically **sparse** – user-to-user similarity weights are often computed on few ratings given to common items: I_{uv}
- Take into account the **significance** of the user-to-user similarity metric: how dependable the measure of similarity is – and not only its value – when making a rating prediction [Herlocker et al., 1999]

$$w'_{uv} = \frac{\min\left\{\left|I_{uv}\right|, \gamma\right\}}{\gamma} \times w_{uv}$$

- γ is a parameter that must be cross validated – 50 gave optimal results (movielens)
  - This approach **improves** the prediction accuracy (MAE)

Low variance vs. High variance items



Improvements of CF

- 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
- [Baltrunas & Ricci, 2008] found **good results** using in the similarity computation only the most important items (feature selection: items with the highest Pearson correlation with the target item).

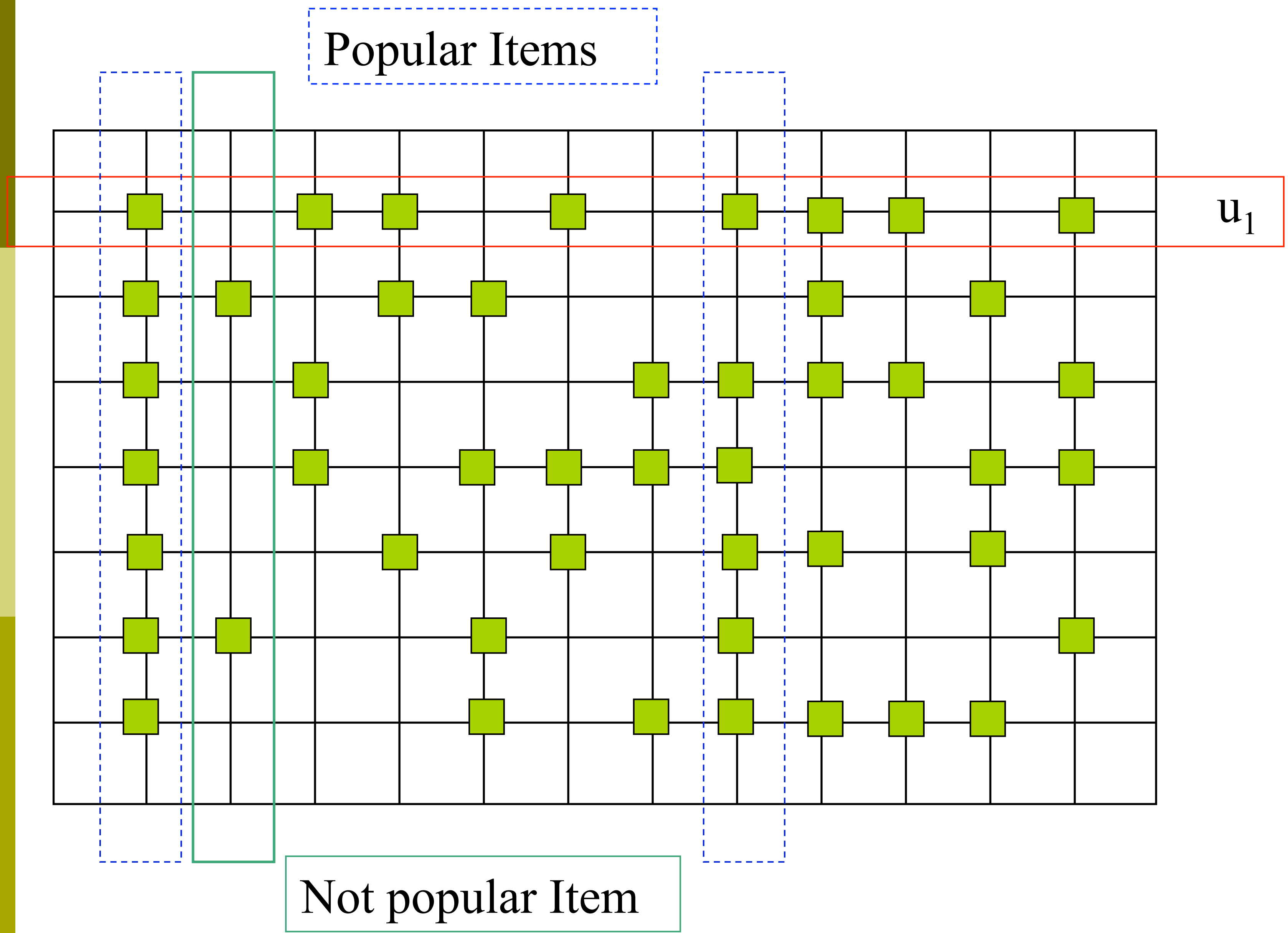
Inverse User Frequency

- Items rated by a small number of users may be more informative – similar idea to "terms present in a small number of documents are more informative"
- $\lambda_i = \log(|U|/|U_i|)$, U_i is the set of users that rated item *i* (assume that U_i is not empty)
- **Frequency-Weighted Pearson Correlation** similarity metric:

$$FWPC(u,w) = \frac{\sum_{i \in I_{uv}} \lambda_i (r_{ui} - r_u)(r_{vi} - r_v)}{\sqrt{\sum_{i \in I_{uv}} \lambda_i (r_{ui} - r_u)^2 \sum_{i \in I_{uv}} \lambda_i (r_{vi} - r_v)^2}}$$

- [Breese et al., 1998] found that this improves prediction accuracy.

Item Popularity



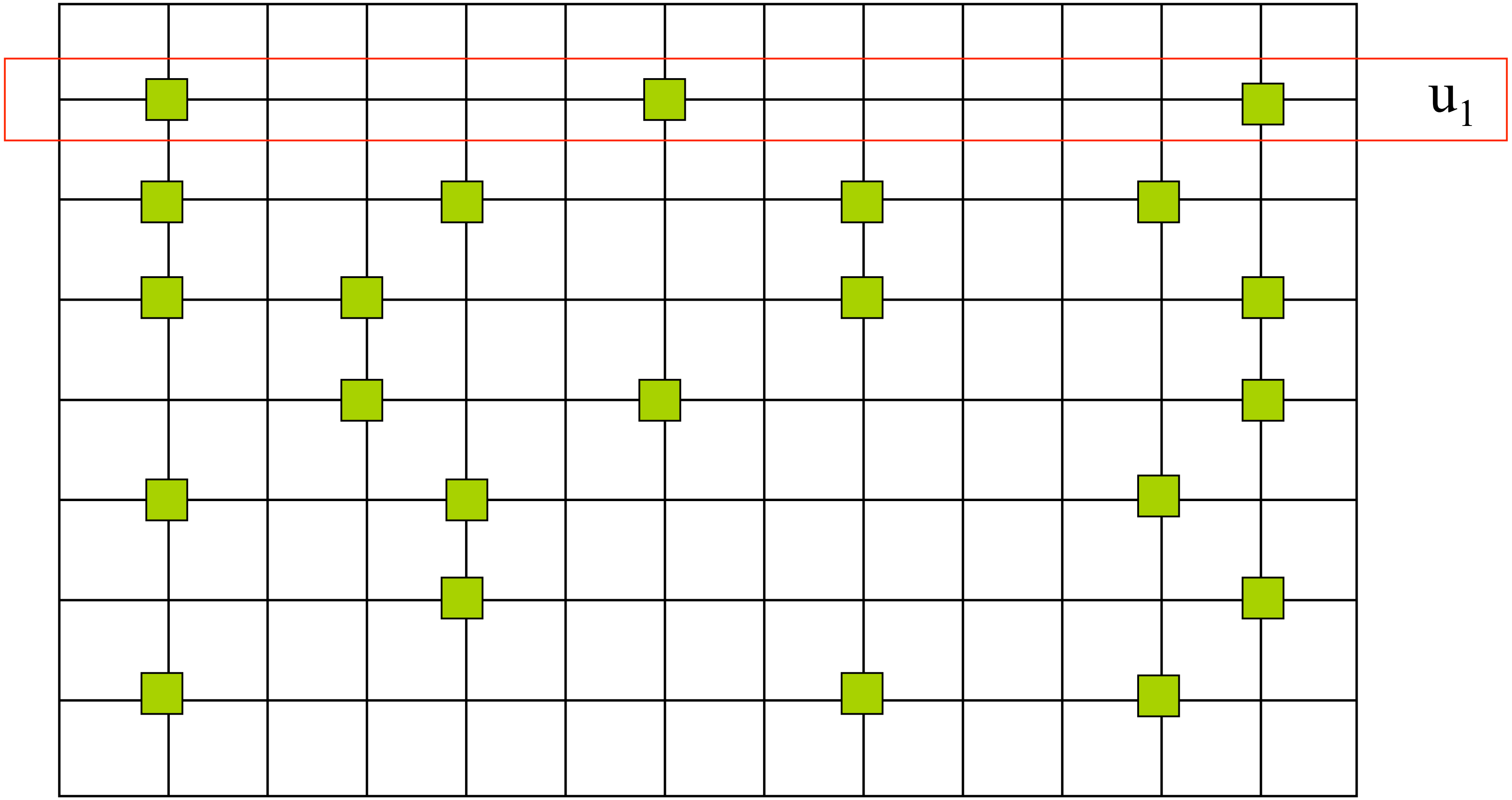
Popular vs Not Popular

- Predicting the rating of **popular** items is **easier** than for not popular ones:
  - The prediction can be based on the ratings of **many neighbor users**
- 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 in the user profile and are considered for computing the similarity
  - Then one loop on the *m* customers to compute similarity PLUS one on the *n* products to compute the prediction.

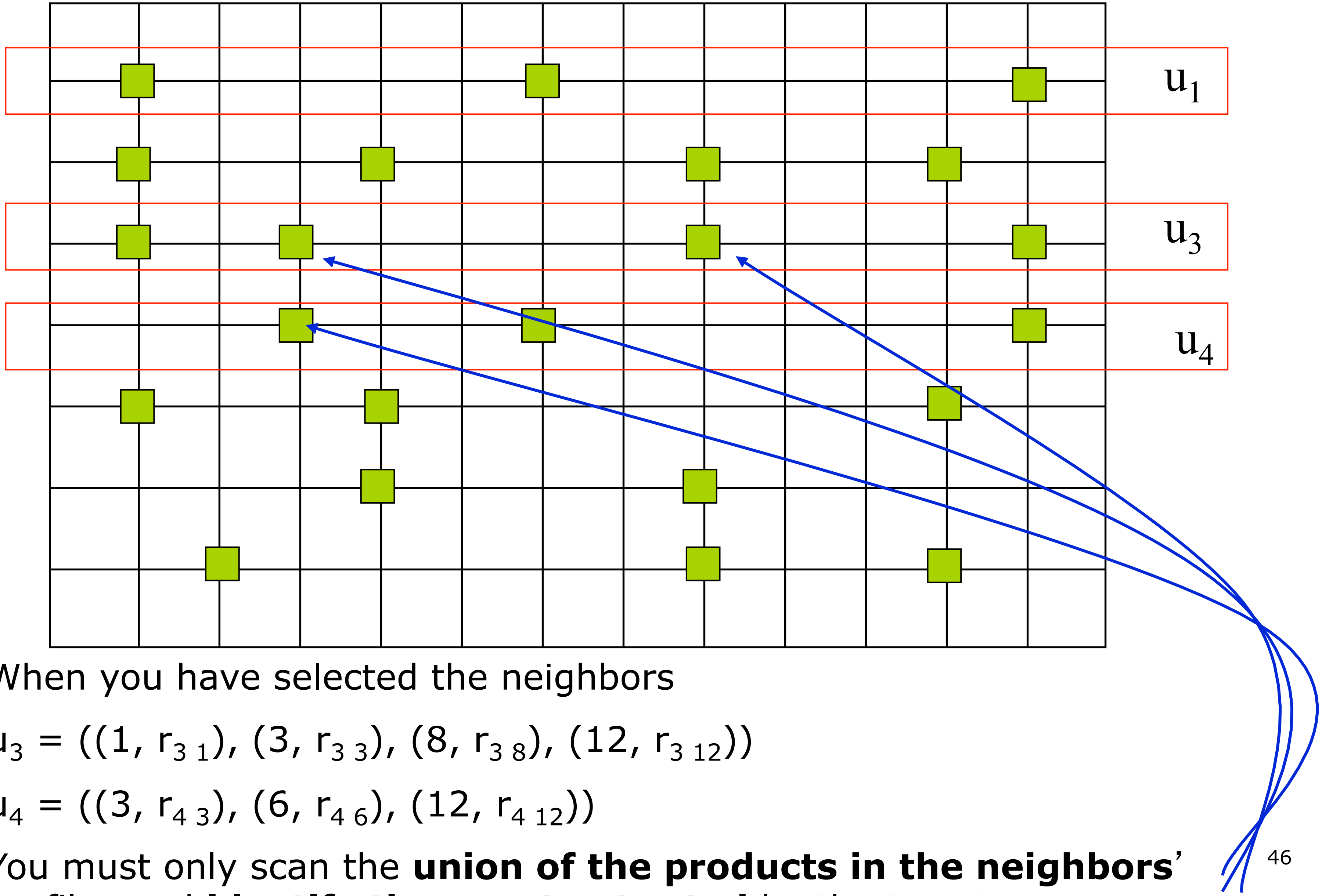
Computational Issues



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 $u_1 = ((1, r_{1\,1}), (6, r_{1\,6}), (12, r_{1\,12}))$

Computational Issues



When you have selected the neighbors

$u_3 = ((1, r_{3\,1}), (3, r_{3\,3}), (8, r_{3\,8}), (12, r_{3\,12}))$

$u_4 = ((3, r_{4\,3}), (6, r_{4\,6}), (12, r_{4\,12}))$

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]).

[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?