Αξιοποιώντας Τεχνικές Συσταδοποίησης με βάση Υποχώρους σε Συστήματα Συστάσεων

Katharina Rausch[1], Ειρήνη Ντούτση[1],
Κώστας Στεφανίδης[2], Hans-Peter Kriegel[1]

Ελληνικό Συμπόσιο Διαχείρισης Δεδομένων
Ιούλιος 2014

1 Institute for Informatics, Ludwig-Maximilians-Universität (LMU) München, Germany

2 Foundation for Research and Technology - Hellas (FORTH), Institute of Computer Science, Heraklion, Greece.

Recommender Systems

Nowadays, due to the growing complexity of the Web, users find themselves overwhelmed by the mass of choices available

- E.g., shopping for DVDs, books or clothes online becomes more and more difficult, as the variety of offers increases rapidly and gets unmanageable.

Recommender systems facilitate users in their selection process, by providing suggestions on items, which could be interesting for the respective user.

How?

- Estimate preferences for items.
- Recommend items featuring the maximal predicted preference.

Use:

- Historical information on the users' interests, e.g., the users' purchase history.

Collaborative Filtering

In general, recommendation approaches are distinguished between:

- Content-based: recommendations are based on a description of each item and a profile of the user's preferences.
- Collaborative filtering: ratings are predicted using previous ratings of similar users

Recommendations for a user *u* are based on the ratings of his/her similar users.

- Compute the set of similar users, Friends F_u
- For all items i unrated by the user, estimate the user preference for the item

$$\hat{r}(u,i) = \frac{\sum_{u' \in F_u} sim(u,u') * r(u',i)}{\sum_{u' \in F_u} sim(u,u')}$$

- Present the top-k ranked items to the user

How do we compute the set of friends?

How do we compute the set of friends?

[Nearest neighbors approach]

Scan the whole database to find similar users

$$F_u = \{u' \in U: sim(u,u') \geq \delta\}$$

- *sim(u,u')*: a user similarity function (e.g., Pearson correlation)
- δ: a user similarity threshold

	1 (Titanic)	2 (Braveheart)	3 (Matrix)	4 (Inception)	5 (Hobbit)	6 (300)
Susan	5	?	5	5	4	?
Bill	3	3	?	1	?	1
Jenny	5	4	1	1	?	4
Tim	?	?	4	4	3	3
Thomas	?	1	1	4	?	4

Target user

Similar users w.r.t. all movies

$$\hat{r}(u,i) = \frac{\sum_{u' \in F_u} sim(u,u') * r(u',i)}{\sum_{u' \in F_u} sim(u,u')}$$

- Friends are defined in the full dimensional feature space
- Linear scan of the db to compute F_u
- Online computation of the set of friends for each query user

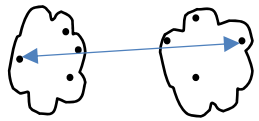
How do we compute the set of friends?

[Full Dimensional Clustering approach]

Users are grouped into clusters of similar users $\Theta = \{\theta_1, \theta_2, ..., \theta_k\}$, $\theta_i \cap \theta_j = \emptyset$.

- The friends of a user are the members of his corresponding cluster

$$F^{clu}_u = \{u' \in \theta_i : u \in \theta_i\}$$



- Agglomerative hierarchical clustering, complete link distance
  - Similarity between two clusters is the similarity of their *most dissimilar* members
  - Stop, if the similarity of the closest pair of clusters violates the user similarity threshold δ

	1 (Titanic)	2 (Braveheart)	3 (Matrix)	4 (Inception)	5 (Hobbit)	6 (300)
Susan	5	?	5	5	4	?
Bill	3	3	?	1	?	1
Jenny	5	4	1	1	?	4
Tim	?	?	4	4	3	3
Thomas	?	1	1	4	?	4

Target user

Members of her cluster

$$\hat{r}(u,i) = \frac{\sum_{u' \in F^{clu}_u} sim(u,u') * r(u',i)}{\sum_{u' \in F^{clu}_u} sim(u,u')}$$

- Faster than nearest neighbors approach
- For each u,u' $\in \theta_i$, *sim(u,u') ≥ δ* (correctness)
- $F^{clu}_u \subseteq F_u$ (incompleteness)
- For small clusters, F^{clu}_u too narrow.

Subspace Clustering Recommendations

For both cases, user similarity is evaluated *in the full high dimensional* feature space

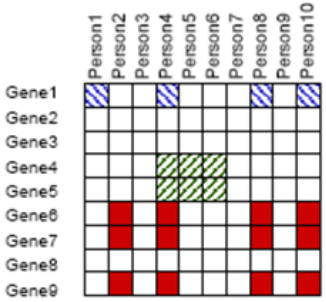
- Its difficult to find similar users when so many dimensions are considered

Its more probable users to exhibit similarity *in some subspace* of the feature space

- e.g., similar taste in comedies but not in dramas

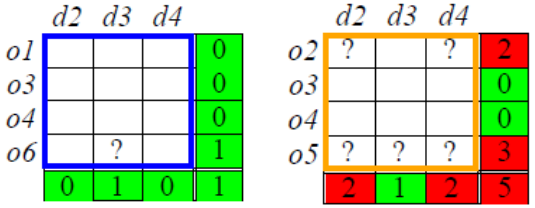
Subspace clustering

- Extract both clusters of users and dimensions, items in our case, based on which users are grouped together
- Clusters are defined in subspaces of the original feature space: $\theta = (U_\theta, I_\theta)$
- Subspace clustering: $\Theta=\{\theta_1, \theta_2, ..., \theta_k\}$, $\theta_i \cap \theta_j \neq \varnothing$ w.r.t. both users and items
- Traditional approaches do not consider missing values though



Fault tolerant subspace clustering

- Missing values are tolerated but *bounded* per cluster
  - User tolerance threshold ε_u
  - Item tolerance threshold ε_i
  - Rating tolerance threshold ε_g



How do we compute the set of friends?

[Subspace Clustering approach]

Subspace-based recommendations

- (Roughly) The friends of a user are the members of all his subspace clusters

$$F^{subclu}_{u} = \{u' \in \theta_i : u \in \theta_i\}$$

	1 (Titanic)	2 (Braveheart)	3 (Matrix)	4 (Inception)	5 (Hobbit)	6 (300)
Target user → Susan	5	?	5	5	4	?
Bill	3	3	?	1	?	1
Jenny	5	4	1	1	?	4
Tim			4	4	3	3
Thomas	?	1	1	4	?	4

Similar users w.r.t. movies 1,2

Similar users w.r.t. movies 3,4

Similar users w.r.t. movies 5,6

$$\hat{r}(u,i) = \frac{\sum_{u' \in F^{subclu}_{u}} sim(u,u') * r(u',i)}{\sum_{u' \in F^{subclu}_{u}} sim(u,u')}$$

Rough idea, we actually use a weighted ranking to select the top friends from the subspace cluster members

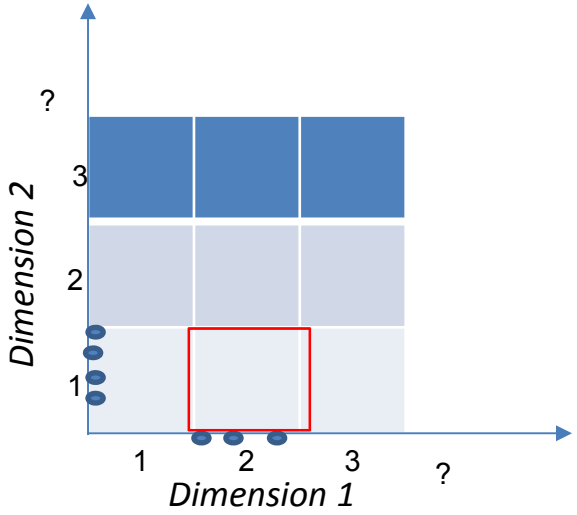
The benefits

- Improves clustering quality due to subspace partitioning
- Expands the set of friends as a user belongs to >1 clusters.
- Diversifies the set of friends as different friends might be chosen based on different items.

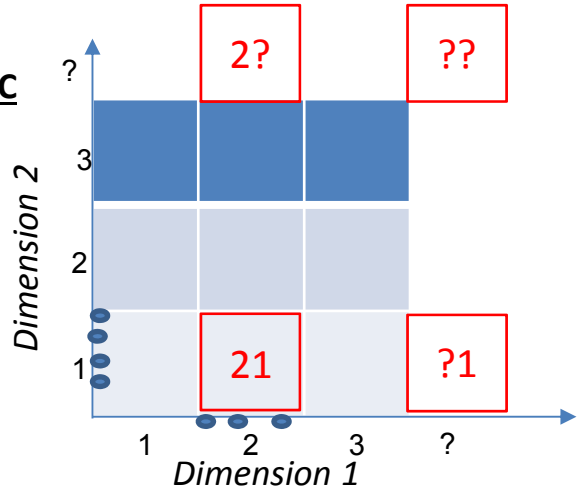
How to extract the fault tolerant subspace clusters?

- In the original paper [1], a grid-based approach was proposed (gridFTSC)
- The approach is based on **CLIQUE** algorithm
  - Each dimension is partitioned into *g* equal length intervals, called units.
  - A k-dimensional unit is the intersection of k-1 dimensional units from k different dimensions
  - A unit is dense if its points exceeds *minPts* points.
  - A cluster in a subspace is a maximal set of connected units in that subspace.
- Extension to fault tolerance
  - Also employ the missing values per dimension to extract cluster approximations

Clique



GridFTSC



[1] Flexible Fault Tolerant Subspace Clustering for Data with Missing Values, Gunneman et al, ICDM 2011.

How to extract the fault tolerant subspace clusters?

- hybridFTSC approach
  - 1-dimensional DBSCAN to detect density based clusters in each dimension
  - Objects with missing values in the dimension are filtered out before DBSCAN call
    - They form a “pseudo-cluster”
  - As in gridFTSC, we extend the clusters by combining them with the pseudo-cluster
  - DBSCAN is applied in single dimensions

- denFTSC approach
  - The approach is based on SUBCLU, a DBSCAN based approach where notions of reachability etc are defined per subspace.
  - 1-dimensional DBSCAN to detect density based clusters in each dimension
  - Objects with missing values in the dimension are filtered out before DBSCAN call
    - They form a “pseudo-cluster”
  - DBSCAN distance is based on the current subspace (ignoring missing values)

Reducing the search space: the significance threshold

To speed up the algorithms, we introduce the significance threshold for dimension pruning

Heuristic:

Consider only dimensions with significant information for subspace expansion.

Intuitively these are dimensions with *big clusters*.

Significant dimensions:

Those including c clusters featuring at least d % of the overall population

- Cluster threshold c: $c > 1$ & approximately half of rating values possible
- Data threshold d: $0.1 \leq d \leq 0.2$ depending on *c*

Weighted ranking to locate best friends

Through subspace clustering, we potentially receive several subspace clusters the query user u belongs to.

- Basic combining approach: Union of clusters (F)

Weighted ranking approach:

1) Combine all cluster members the query user u belong to (F)
2) Rank them according to their *weighted full dimensional distance* to u
3) Select as friends those below the weighte distance threshold β

Subspace-based

Full dimensional-based

Weighted full dimensional distance: Refine distance by weighting based on the number of common dimensions

$$dist^{weighted}(u,v) = \frac{1}{c_{uv}} dist(u,v)$$

Normalized #common dimensions

$$c_{uv} = \frac{|I_{uv}| - \min(|I_{uv'}|)}{max(|I_{uv'}|) - \min(|I_{uv'}|)}, v' \in F$$

Experiments

- ML-100K dataset: 983 users, 1682 movies, 100000 ratings.
- ML-1M dataset: 6040 users, 3952 movies, 1000000 ratings.

- Recommendation quality measures:
  - Mean absolute error (MAE)
  - Root mean squared error (RMSE)

- Compared approaches
  - Nearest neighbors approach [Naïve]
  - Full dimensional clustering approach [fullClu]
  - Fault tolerant grid-based approach [gridFTSC]
  - Fault tolerant hybrid approach [hybriFTSC]
  - Fault tolerant subclu-based approach [denFTSC]

Parameter settings, runtime and #clusters

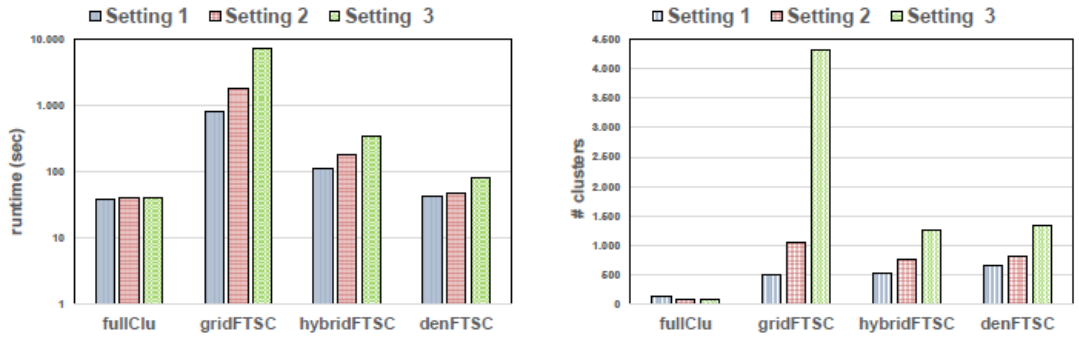
[ML-100K dataset]

Table 1: ML-100K dataset: Parameter settings and results

	Parameter settings	Runtime	# Clusters	Setting ID	
fullClu	$\delta = 0.2$	39s 585ms	141	1	Cluster cardinality: 3-15, 27, 38
	$\delta = 0.5$	41s 115ms	87	2	
	$\delta = 0.7$	41s 237ms	83	3	Cluster cardinality: 3-28, 138
gridFTSC*	minPts = 50, grid = 3	13min 33s 55ms	494	1	
	minPts = 40, grid = 3	29min 53s 364ms	1038	2	Dimensionality: 1-4, max 5
	minPts = 30, grid = 3	2h 3min 41s 655ms	4308	3	
hybridFTSC (*)(+)	minPts = 40, $\epsilon = 0.1$	1min 54s 555ms	516	1	
	minPts = 30, $\epsilon = 0.1$	3min 0s 345ms	754	2	Dimensionality: 1-3, mostly 1
	minPts = 20, $\epsilon = 0.1$	5min 51s 895ms	1265	3	
denFTSC (*)(+)	minPts = 35, $\epsilon = 0.1$	42s 399ms	667	1	
	minPts = 30, $\epsilon = 0.1$	48s 586ms	819	2	Dimensionality: 1-5
	minPts = 20, $\epsilon = 0.1$	1min 23s 15ms	1349	3	

(*) parameters for FTSC: $\epsilon_o = 0.4$, $\epsilon_s = 0.3$, $\epsilon_g = 0.4$

(+) parameters for significance threshold: d = 0.13, $c = 2$



(a) Runtime (b) # Clusters

Parameter settings, runtime and #clusters

[ML-1M dataset]

Table 2: ML-1M dataset: Parameter settings and results

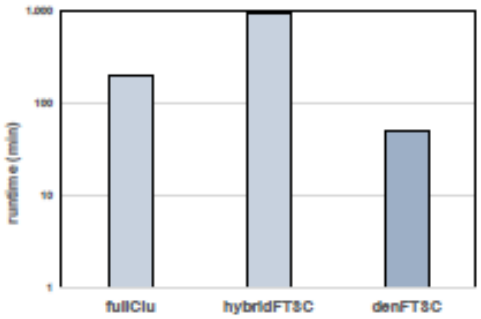
	Parameter Settings	Run-time	#Clusters
fullClu	$\delta = 0.5$	3h 22min 57s 869ms	384
hybridFTSC (*)	minPts = 100, $\epsilon = 0.1$, d = 0.17, c = 2	15h 43min 50s 50ms	2946
denFTSC (*)	minPts = 100, $\epsilon = 0.1$, d = 0.15, c = 2	51min 5s 957s	2894

(*) parameters for FTSC: $\epsilon_o = 0.4$, $\epsilon_s = 0.3$, $\epsilon_g = 0.4$)

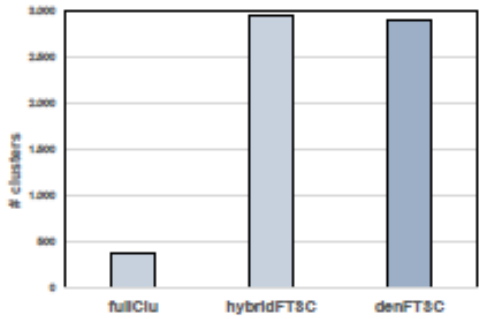
Many small clusters, a few big

Dimensionality: 1-4

Dimensionality: 1-3



(a) Runtime

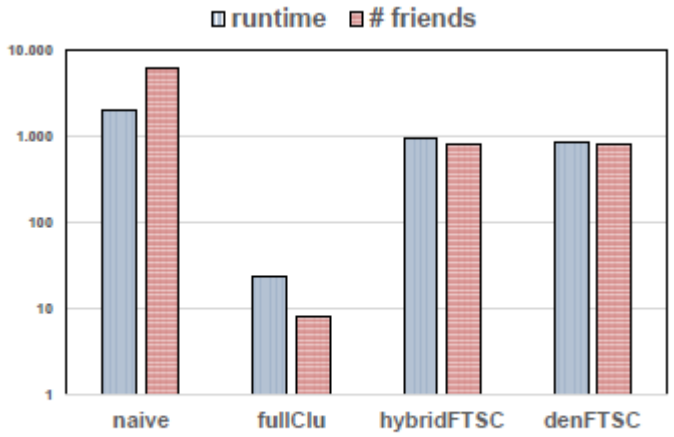


(b) # Clusters

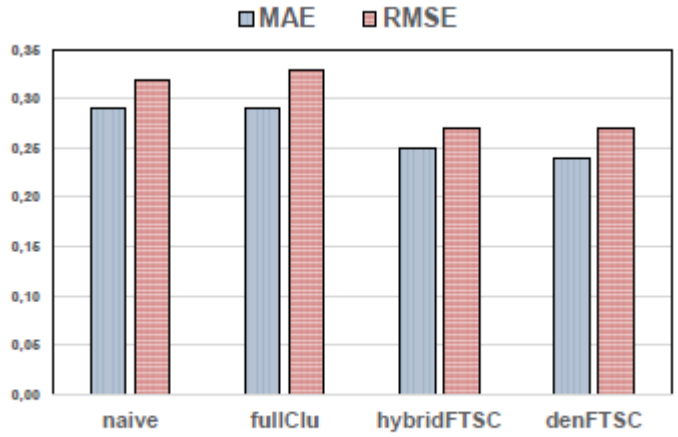
Quality of user recommendations

random user from [ML-1M dataset]

- To examine the qualitative differences of the approaches, we issued the 10 most promising recommendations to random query users.



(a) Runtime & #friends



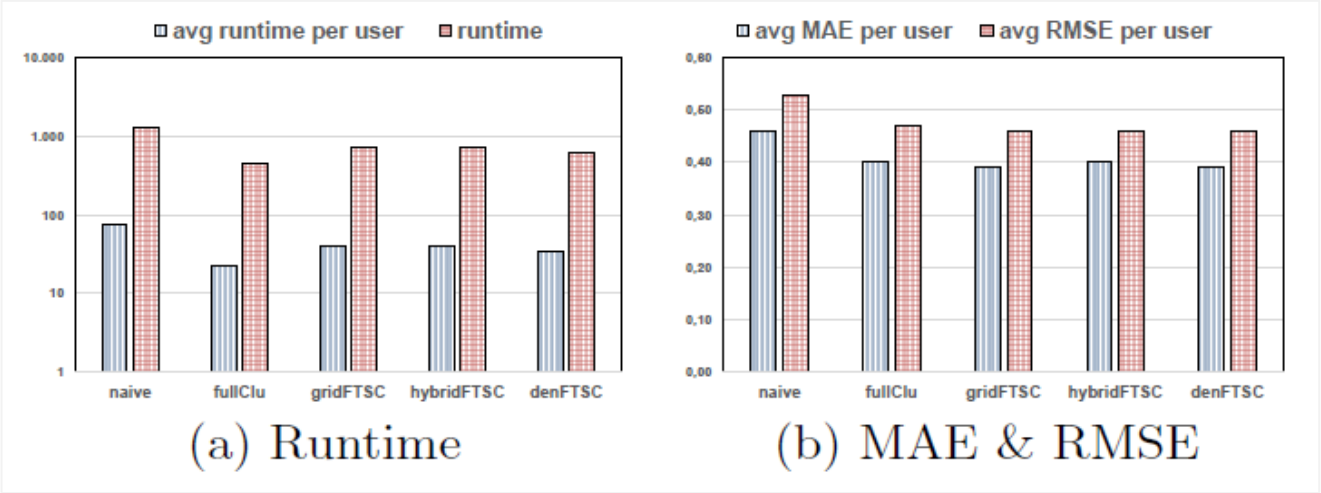
(b) MAE & RMSE

- For fullClu, the user was part of a cluster of 9 members → too narrow selection
- Naïve considers > half the users as similar → too broad selection
- Subspace also offers a broad selection of friends but this set is refined through weighted ranking and refinement → better quality of recommendations.

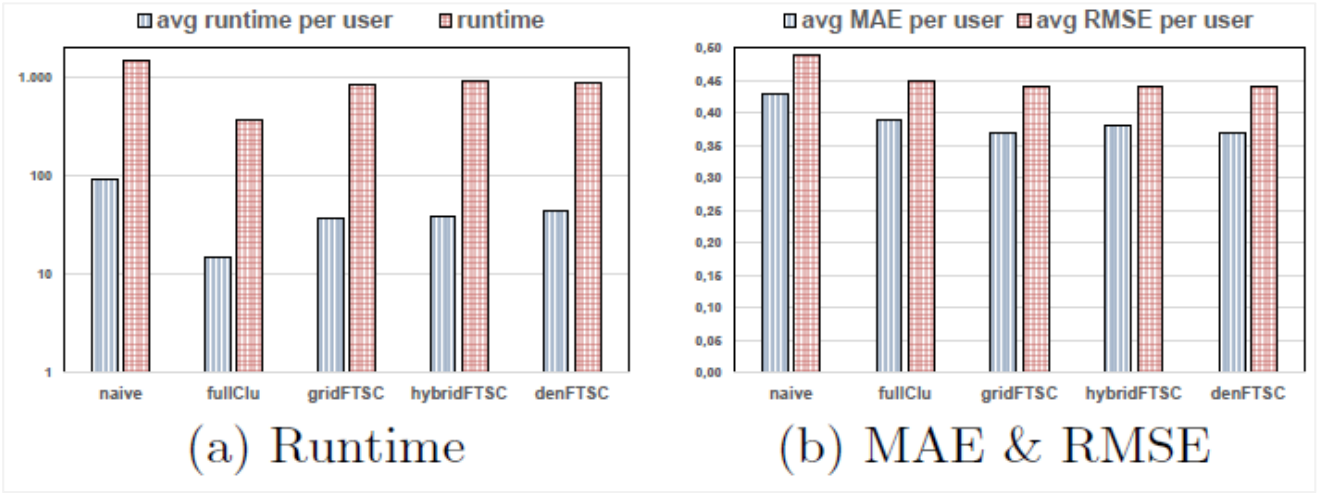
Quality of user recommendations

[ML-100K dataset]

Homogenous query group



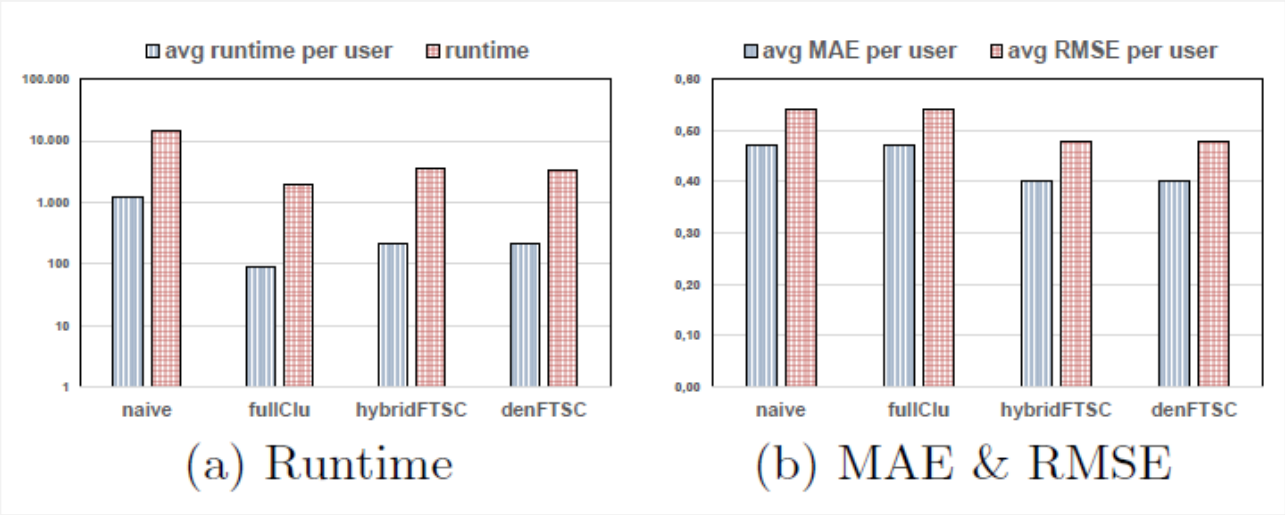
Heterogeneous query group



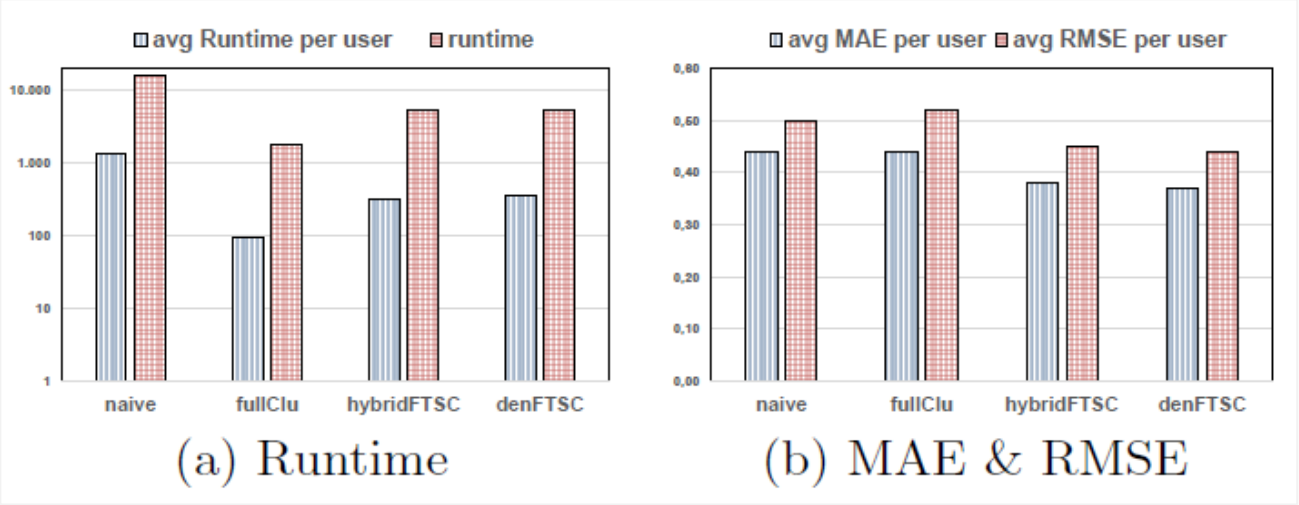
Quality of user recommendations

[ML-1M dataset]

Homogenous query group



Heterogeneous query group



Summary

- We introduce subspace clustering for recommendations
- We propose two new density-based approaches for fault tolerant subspace clustering, hybridFTSC and denFTSC
- We introduce the significance threshold to speed up computations.
- We propose a weighted ranking approach to combine multiple subspace clusters and select the most prominent users for a query user.

- Our results show that neither a narrow selection of frieds (fullClu), nor a broad selection of friends (naïve) perform well.
- Rather, a broad pool of diverse friends extracted through subspace clustering and a refinement of this set through weighted ranking offers the best quality of recommendations at a fair runtime.

- Open issues
  - Experimentation with more datasets
  - Matrix factorization methods
  - Combination of ratings with review texts

Σας ευχαριστώ πολύ!!!

Ερωτήσεις?

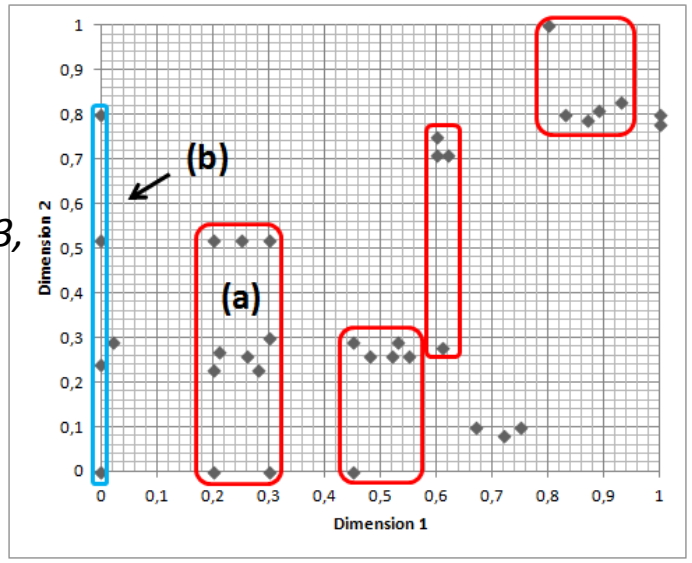
Παράληψη από χθες

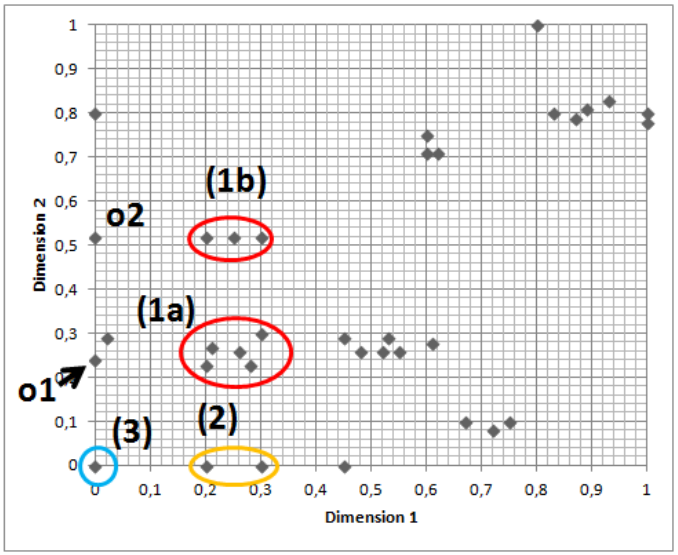
Η εργασία "*Discovering and Monitoring Product Features and the Opinions on them with OPINSTREAM*" είναι σε συνεργία με τους Max Zimmermann, Myra Spiliopoulou από University of Magdeburg, Germany.

New approach: HybridFTSC

- 1-dimensional DBSCAN to detect density based clusters in each dimension
- Objects with missing values in the dimension are filtered out before DBSCAN call
  - They form a “pseudo-cluster”

minPts = 3,
ε = 0.04





New approach: denFTSC

- DBSCAN employs a distance function, which is based on subspace & ignores missing values

minPts = 3,
ε = 0.04