Application of Dimensionality Reduction in Recommender System– A Case Study

B. Sarwar, G. Karypis, J. Konstan, J. Riedl

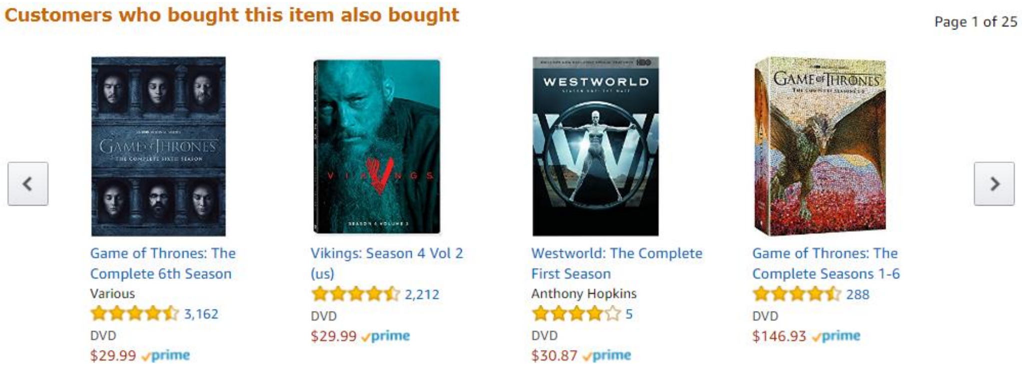
Presented By: Yuchen Luo

The Netflix Prize

From 2006 to 2009, Netflix sponsored a competition, offering a grand prize of $1,000,000 to the team that could take an offered dataset of over 100 million movie ratings and return recommendations that were 10% more accurate than those offered by the company's existing recommender system. On 21 September 2009, the grand prize of US$1,000,000 was given to the BellKor's Pragmatic Chaos team using tiebreaking rules

This competition energized the search for new and more accurate algorithms.

Introduction



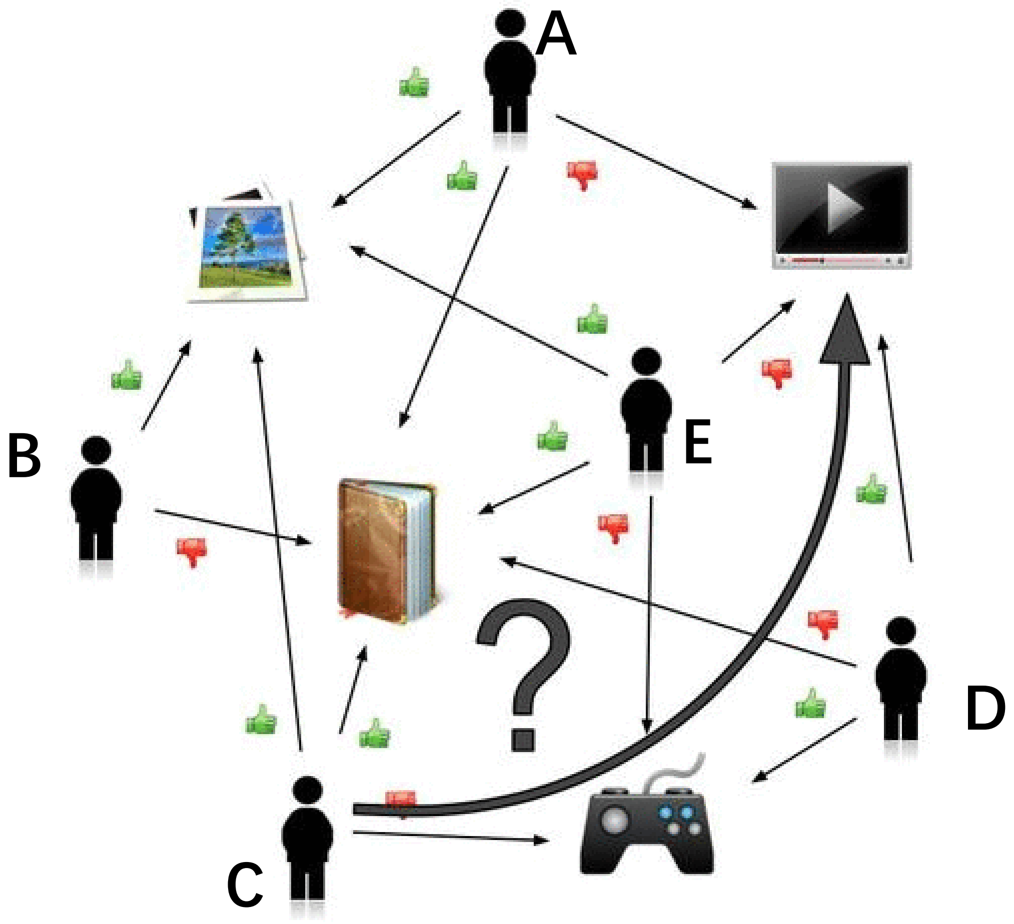
recommender system:

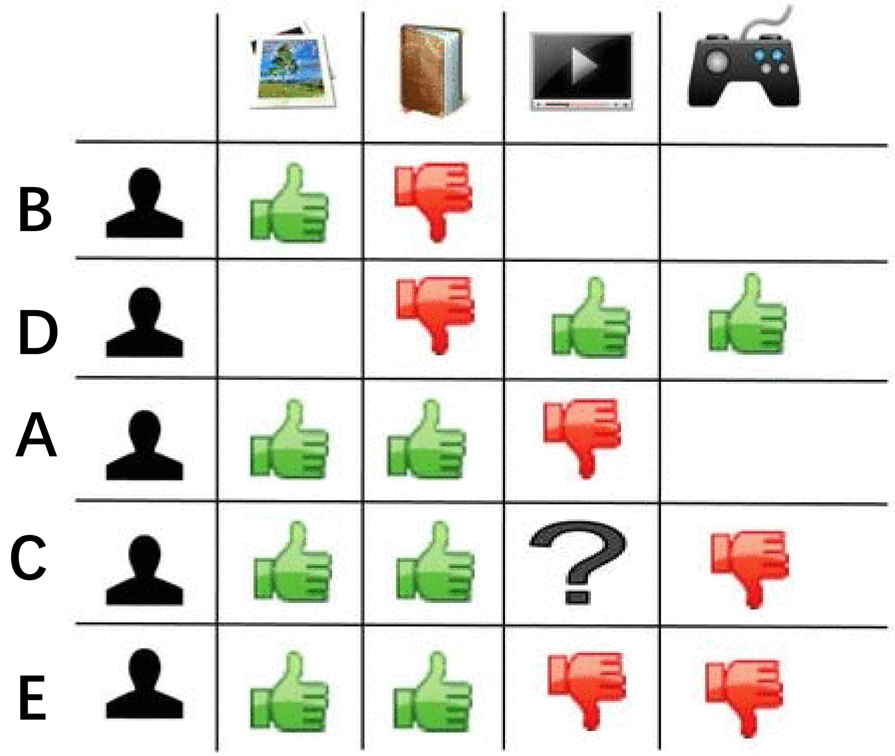
predict the "rating" or "preference" that a user would give to an item.
– From Wikipedia

Two Functions:

1) Increase efficiency (inventory management)
2) Sell more products (matching customers to products)

Problem Statement



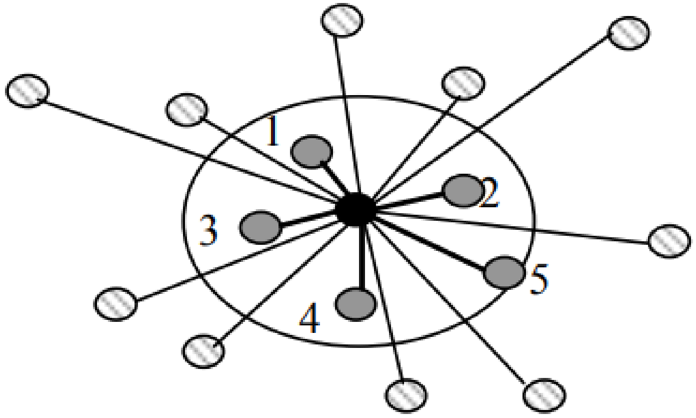


Existing Techniques– Collaborative Filtering(1)

Collaborative filtering (CF)

1) Tapestry, people from close-knit community, Group too small

2) Nearest-neighbor techniques: Neighbors (similar preference of products, likeminded customer)



Neighbor formation

I. Find out the purchasing order of different users
II. Use proximity measurement (cosine similarity, Pearson correlation)
III. Find top k (k= 5) neighbors for a certain user
IV. Based on neighbors choice, perform recommendation

Existing Techniques– Collaborative Filtering(2)

1. Preference prediction

$$C_{\mathrm{P}_{pred}} = \overline{C} + \frac{\sum_{J \in rates} (J_{\mathrm{P}} - \overline{J}) r_{CJ}}{\sum_{J} |r_{CJ}|}$$

It is a personalized prediction, and some simple way to handle is just take average rating of items by all the other customers

r_{CJ}: correlation between user C and neighbor J
J_P: neighbor J’s rating on product P
$\bar{J}$and $\bar{C}$ are average rating of the customer C and neighbor’s average rating

2. Top N products recommendation

once neighborhood formed, top N products rated by the neighbors can be recommended to the customer

Existing Techniques– Collaborative Filtering(3)

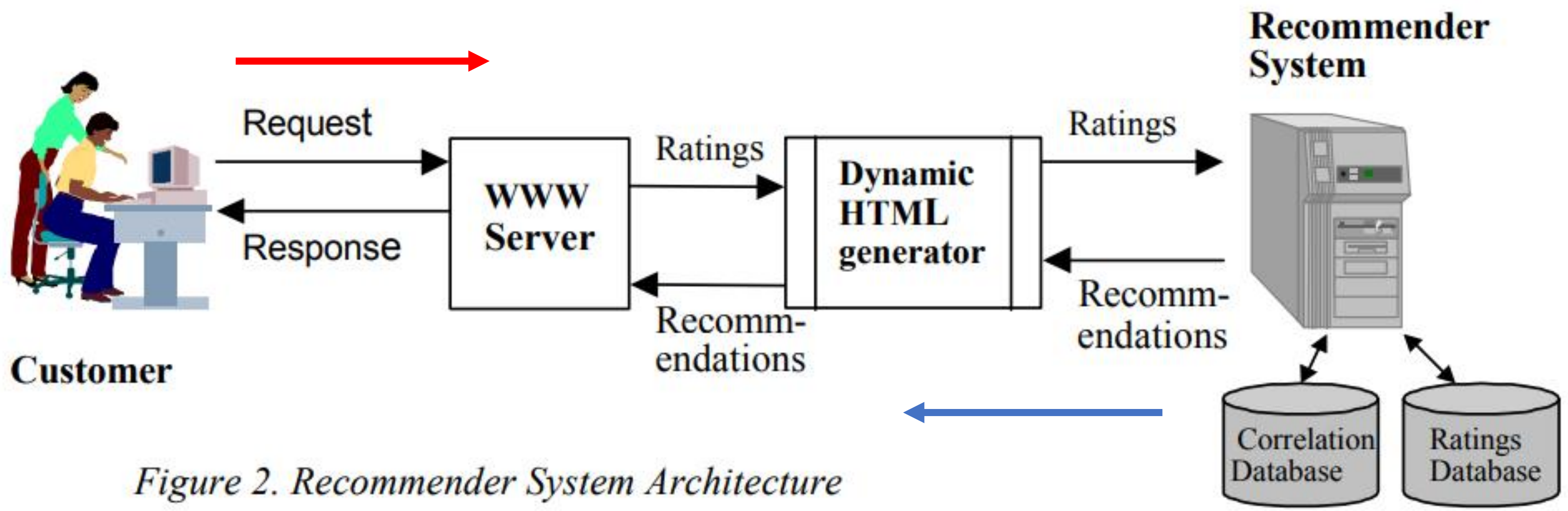


Figure 2. Recommender System Architecture

Recommendation Algorithms

Limitations of Collaborative Filtering

1)Sparsity

2)Scalability

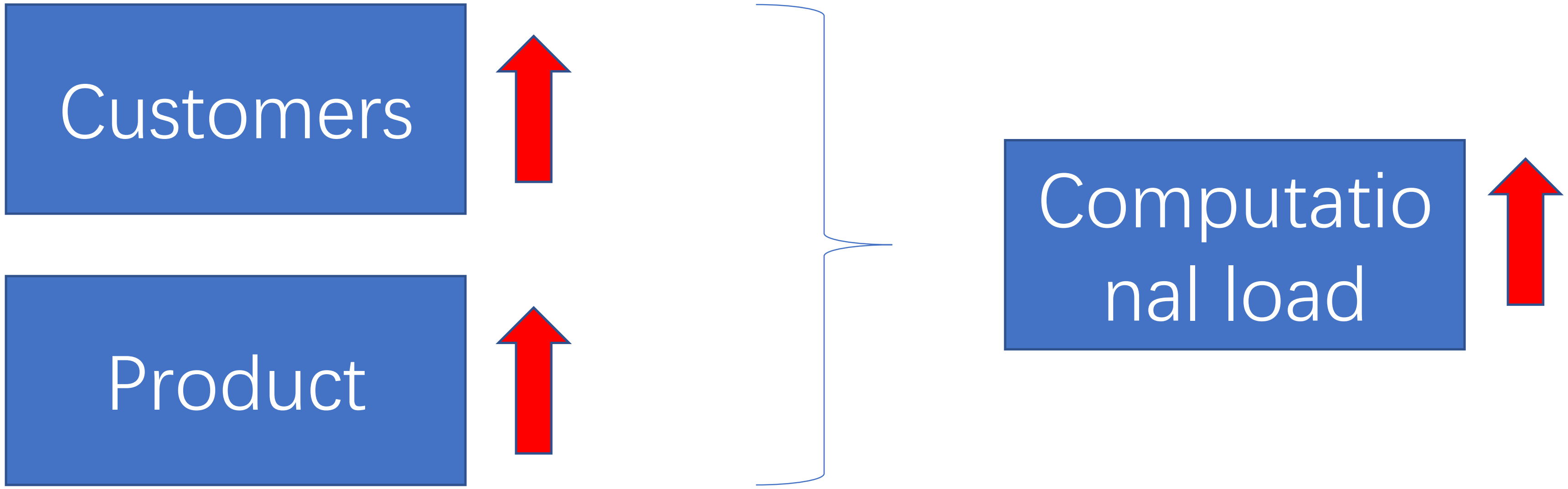
3)Synonymy.

Limitation of CF – Sparsity

1. Reduced coverage: Commercial recommender system has to work with sparse matrix, (1% products are rated), person nearest neighbor algorithm fails to provide recommendation to a customer if he has not rated any product at all.
2. Lack of Neighbor Transitivity: P=S, S=M→ P ≠ M.

   No enough common rating exists between P and M, maybe negative correlation established.

Limitation of CF – Scalability



Limitation of CF – Synonymy

Two words has same meanings, but not exactly the same

Recycled letter pad

Recycled memo pad

Recycled office product

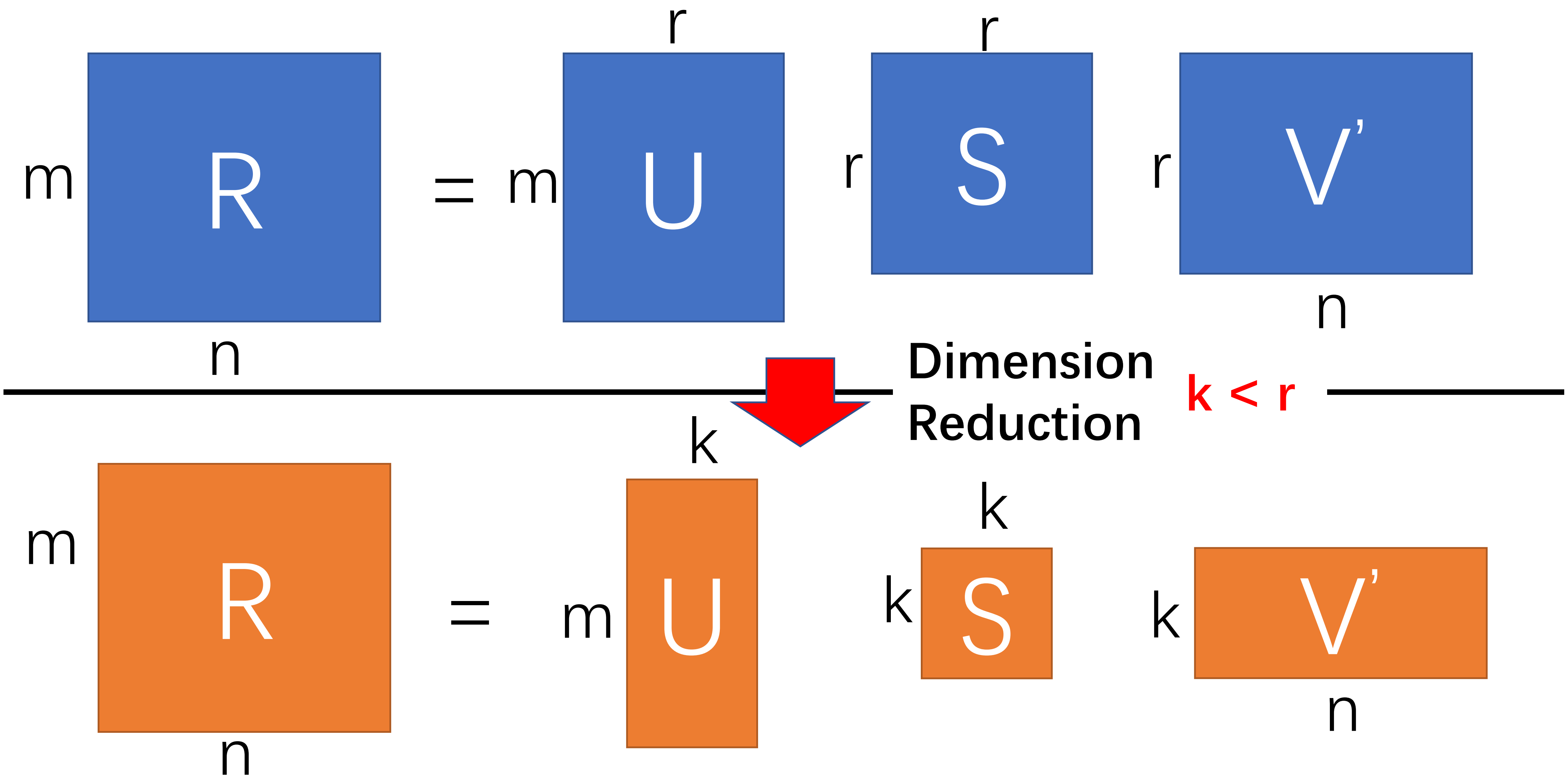
Motivation for the new– LSI/SVD

1) Semi-intelligent filtering agent to fight sparsity, fundamental problem unsolved

2) Latent Sematic Indexing: reduce dimension to make the matrix denser(sparsity), dimension reduction, less calculation (scalability)and good at synonymy problem (synonymy),

3) So LSI/SVD is chosen to incorporate with recommender system.

Method– SVD recommender algorithm(1)



Goal of the Study

1. **Preference Prediction** (Rating Prediction): Capture latent relationships between customers and products that allow us to compute the predicted likeliness of a certain product by a customer.
2. **Top-N Recommendation**: SVD to produce a low-dimensional representation of the original customer-product space and then compute neighborhood in the reduced space and then use that to generate a list of top-N product recommendations for customers

Preference Prediction (Experiment 1)

1) Procedure
2) Experiment Setup
3) Evaluation Metrics
4) Experiment Implementation
5) Result and Discussion

Prediction Generation Procedure(1)

1. filled in the sparse matrix
   a) average rating for a customer
   b) average rating for a product (better)
2. Normalized the matrix
   a) conversion of rating to z scores
   b) subtraction of customer average from each rating (better)

$$\mathrm{R}_{\mathbf{norm}} = \mathrm{R} + \underline{\mathrm{NPR}}$$

Fill-in, non-personalized recommendation

Prediction Generation Procedure(2)

3. Factor R_{norm} using SVD to obtain U,S and V
4. Reduce the matrix S to dimension k
5. Compute the square-root of the reduced matrix S_k to obtain $S_k^{1/2}$
6. Compute two resultant matrices: $U_k S_k^{1/2}$ and $S_k^{1/2} V_k'$
7. $C_{P_{pred}} = \bar{C} + < U_k S_k^{\frac{1}{2}}(c), S_k^{\frac{1}{2}} V_k'(P) >$, <> is used to denote dot product, c^{th} row of $U_k S_k^{1/2}$ and p^{th} column of $S_k^{1/2} V_k'$ are taken out.

Experiments Setup

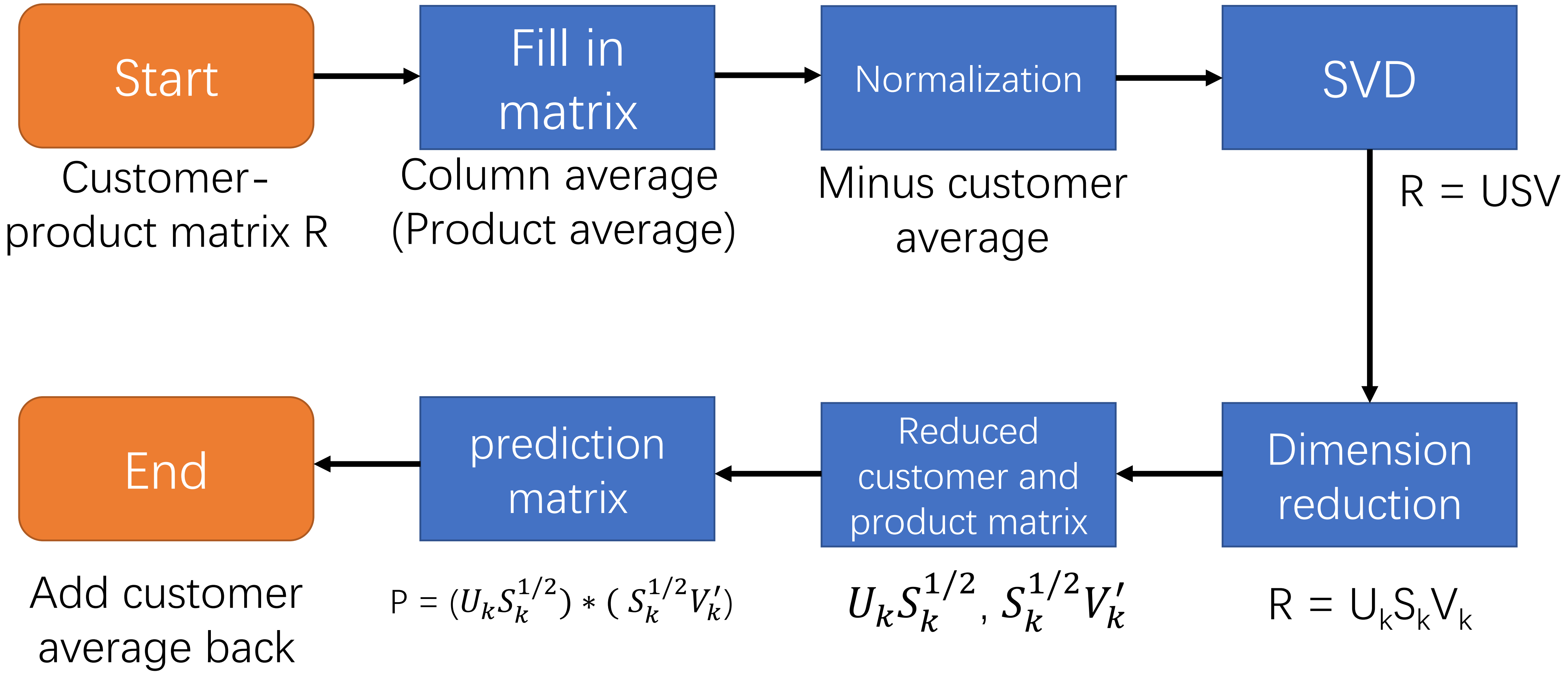
1. Data from MovieLens recommender system, with 100,000 rating-records. Rating-recorded formed in <customer, product, rating>.
2. Choose training ratio (# training/ total record) x= 0.3
3. Reformat the training set as a user-movie matrix with 943 rows and 1682 columns (1682 movies are rated by 943 customers)
4. Each entry represented the rating of i^{th} user to j^{th} movie.

Evaluation Metrics

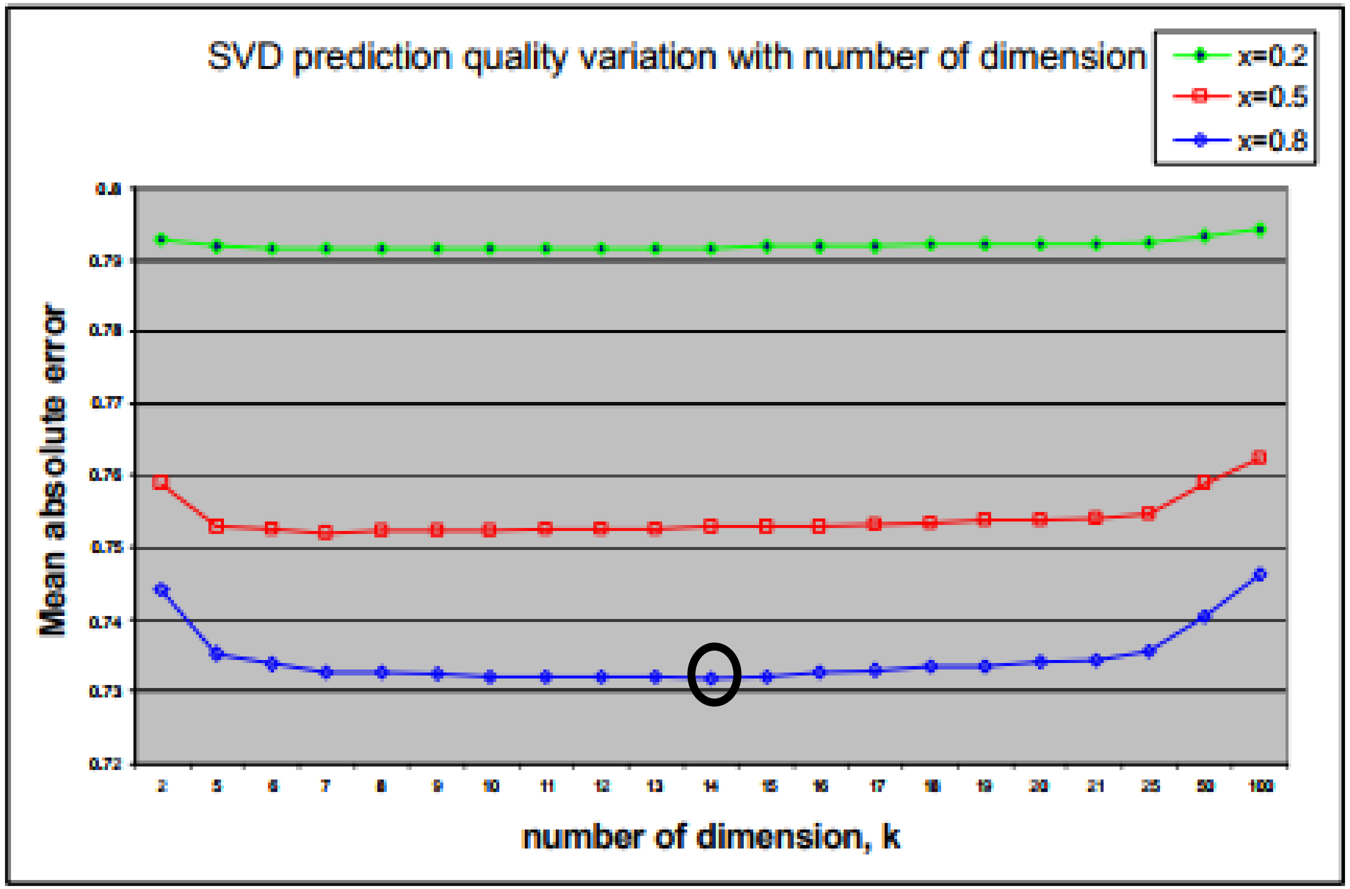
1. Coverage metrics: $\frac{\#customer-product\ (recommendable)}{\#customer-product(all\ possible\ )} \times 100\%$
2. Statistical accuracy: MAE, RMSE, Correlation between rating and prediction.
3. Decision support accuracy: reversal rate, weighted errors and ROC sensitivity.

MAE is used in the prediction evaluation experiment.

Experiment Implementation

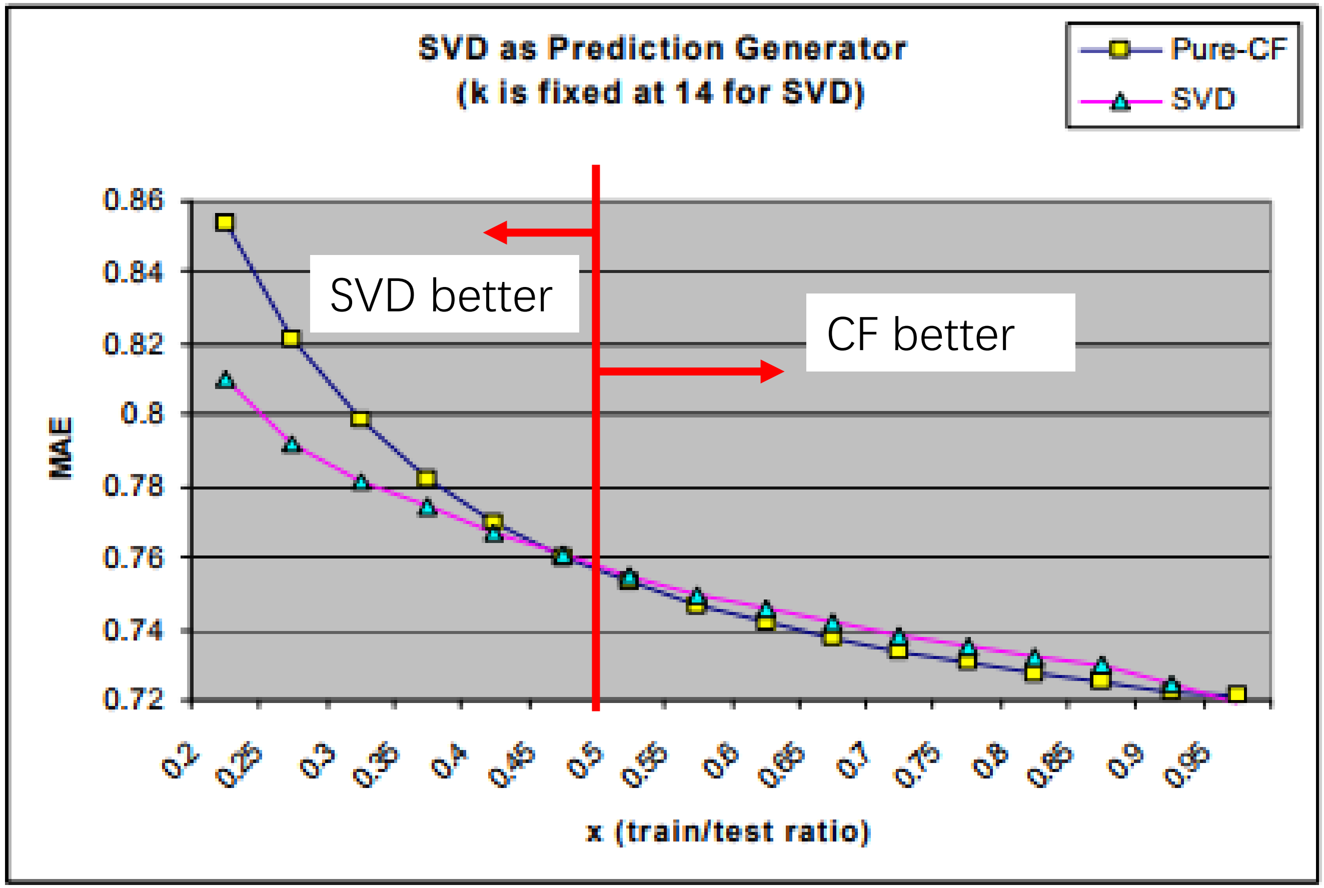


Result and Discussion(1)



Determine the optimal k value: it is found that when training ratio is 0.8, k=14 produces the minimum MAE.

Result and Discussion(2)



Fix k at 14, and vary the training ratio, compare with the result of Pure CF and SVD CF.

1) Low x, SVD is better
2) High x, Pure CF is better
3) Pure CF more sensitivity to x, namely the sparsity
4) SVD can resist sparsity problem by utilizing latent relationship.

Top-N Recommendation (Experiment 2)

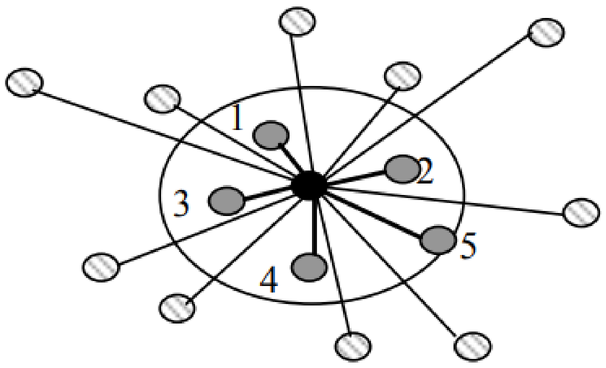
1) Procedure
2) Experiment Setup
3) Evaluation Metrics
4) Experiment Implementation
5) Result and Discussion

Procedure(1)

1. SVD of original customer-product matrix A= USV
2. Reduce S to rank k and do similar operation to U and V
3. Obtain $U_k S_k^{1/2}$, with dimension $m \times k$. It is the m customers in the k dimension domain
4. Perform vector similarity to form neighborhood.

Procedure(2)

5. Scan through the purchase record of each of k neighbors and perform a frequency count on the product they purchased
6. Sort the product list and take the top N item to the customer



	P1	P2	…	PN
neighbor1	1	1		1
neighbor2	0	1		2
…	0	0		3
neighbor k	1	0		6

	#purchased
P3	100
P4	58
..	…
Pl	6

Experiments Setup

1. Data from historical catalog purchase data from a large e-commerce company.
2. 6503 users on 23,554 catalog items. Total 97,045 purchasing records
3. Each record is formed as a triple <customer, product, purchase amount>
4. Convert purchase amount to binary value, if larger then zero, then put 1.
5. Choose training ratio x.

Evaluation Metrics

Products that appear in both sets are member of the hit set

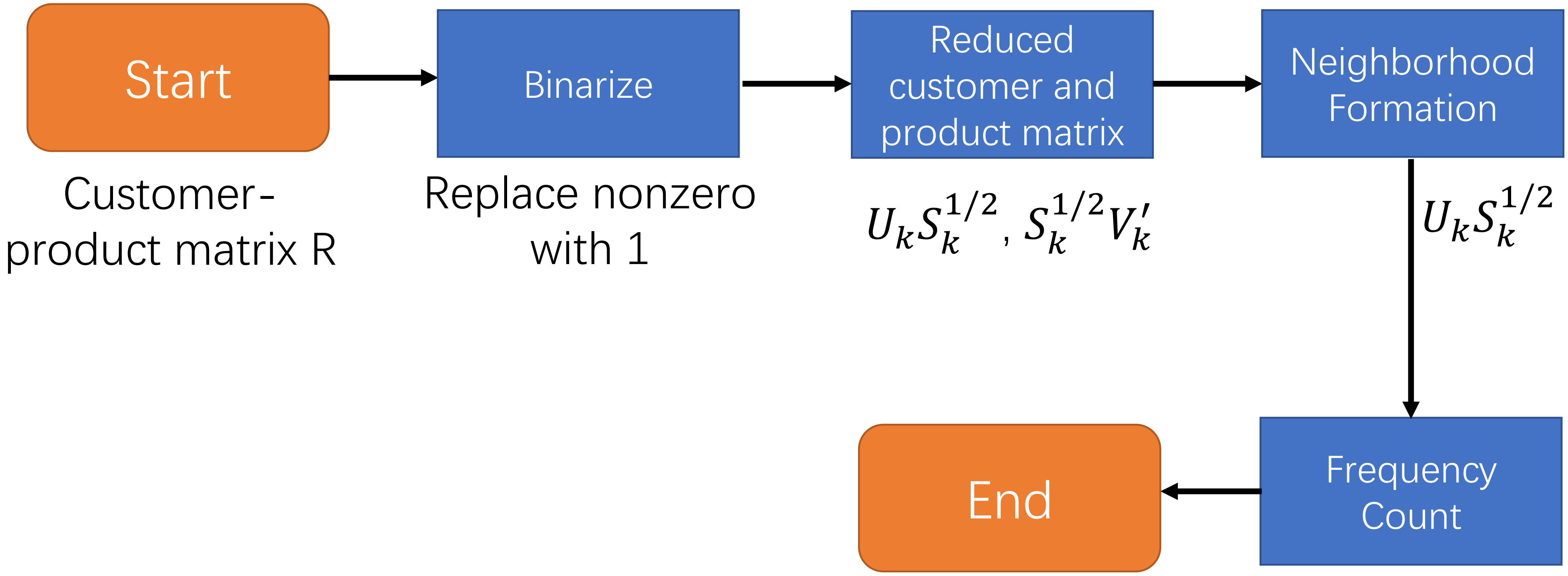
$$Recall = \frac{size\ of\ hit\ set}{size\ of\ test\ set} = \frac{|test \cap topN|}{|test|}$$

$$Precision = \frac{size\ of\ hit\ set}{size\ of\ topN\ set} = \frac{|test \cap topN|}{|topN|}$$

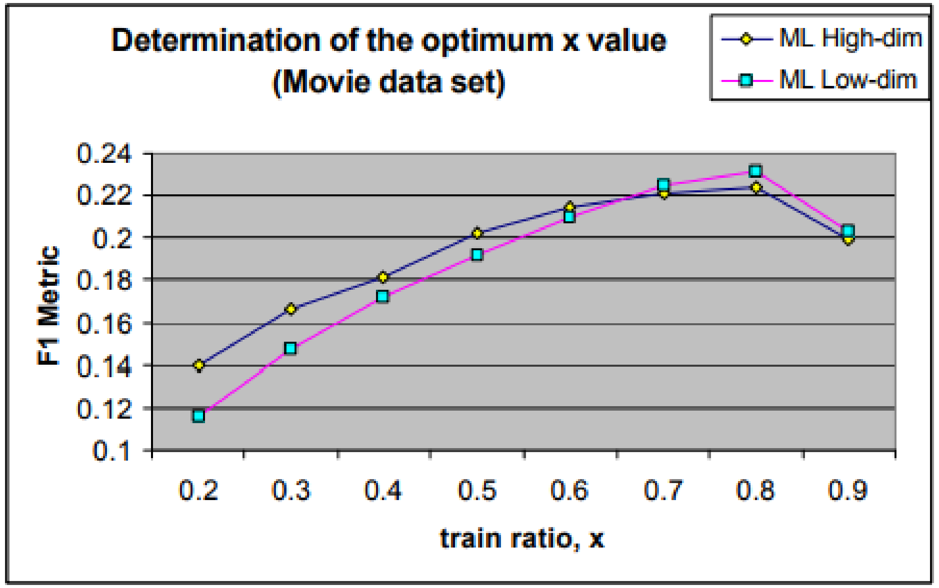
F1 is used in this study

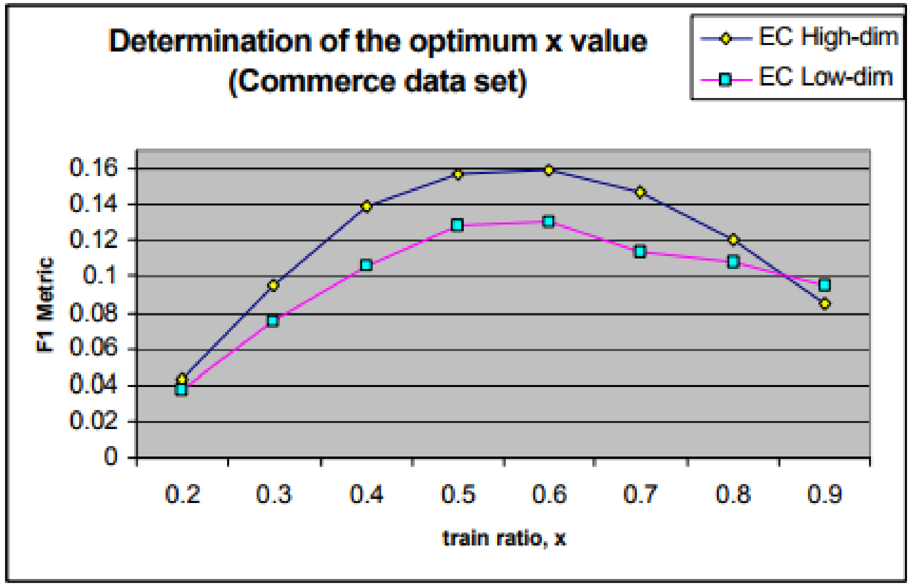
$$F1 = \frac{2 * Recall * Precision}{(Recall + Precision)}$$

Experiment Implementation



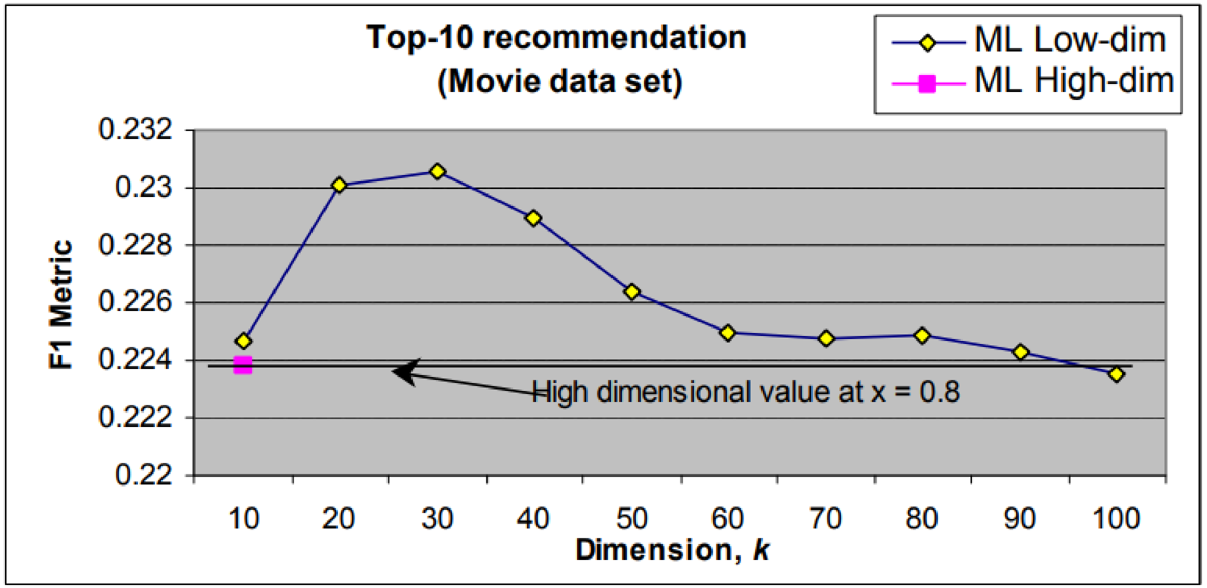
Results and Discussion(1)





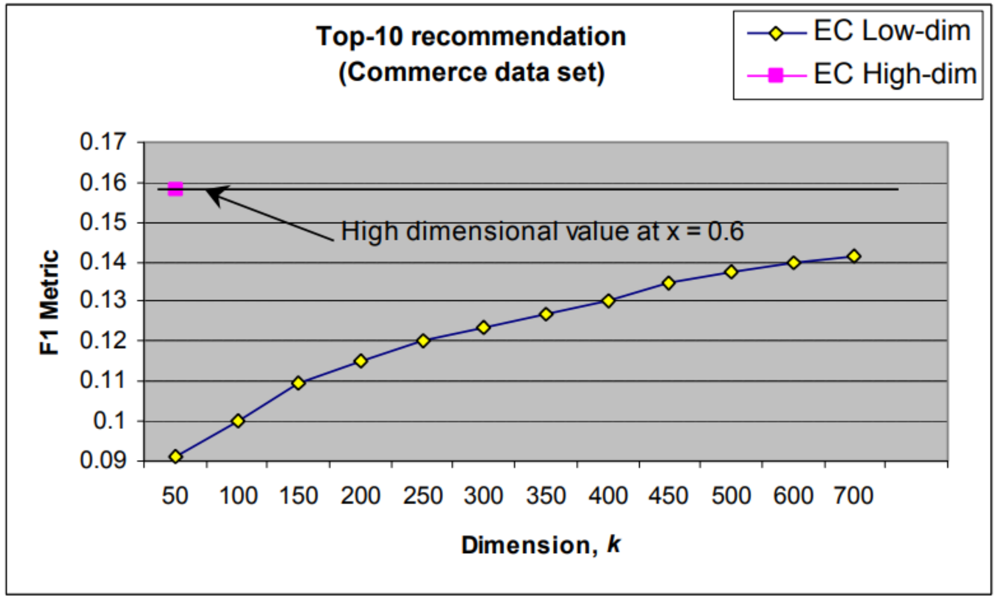
Fixing k, run low dimension and high dimension scheme for different training ratio for two different dataset, movie data set and E-commerce data set. It turns out that for movie data set the best training ratio is 0.8 and for E-commerce the best training ratio is 0.6.

Results and Discussion(2)



1. For movie data set apply x = 0.8, and vary the dimension of k in low dimension schemes.
2. As we can see that high dimension scheme (CF algorithm) does not have the option to change k value, so it is a horizontal line.
3. But for low dimension SVD case, the optimal k is at k = 20.

Results and Discussion(3)



1. For movie data set apply x = 0.6, and vary the dimension of k in low dimension schemes.
2. As we can see that high dimension scheme (CF algorithm) does not have the option to change k value, so it is a horizontal line.
3. High dimension (CF) continues shows better performance over low dimension SVD algorithm. As k increase, SVD algorithms is catching up.

Results and Discussion(4)

1. In the movie case, low dimension is better than high dimension case at all k.
2. In E-commerce data, till k=700, high dimension (CF) is still better than low dimension (SVD).
3. Reflection: hypothesis

   (a) as the E-commerce data is very high dimension, small value of k = 700, cannot provide a good approximation.

   (b) Sparsity: movie data base 95.4% sparse, E-commerce data is 99.996 % sparse
4. Validate the hypothesis: increase the sparsity in the move data case, F1 value reduces largely as well

Conclusion

1. SVD in CF recommender systems can provide good quality prediction. And SVD can provide better online performance than correlation-based systems.
2. In Top-10 recommendation, even a small fraction of dimensions, recommendation quality was better than corresponding high dimensional scheme.
3. Reduced dimension method has advantages in the neighborhood formation.
4. SVD does not do a good job in the very sparse matrix (sparsity larger or equal to 99.996%)

Limitation of LSI/SVD and future work

1. Does not perform well on the very sparse matrix
2. Understand why SVD does not perform well in some cases but well in the other.
3. How often SVD should be updated and how to update it more efficiently
4. Expand the application of SVD: use SVD in the neighborhood formation and visualization of the rating.

Thank you for your attention !

Sensitivity of Number of Dimension K

In the dimension reduction procedure, the chose of K become important.

1) We should keep k large enough to capture all the important structure in the matrix
2) We should also keep it small enough to avoid overfitting errors.

In the experiment, the k value has been studied by trying several different values.

Performance implications

The recommender algorithm can be divided into: online component and offline component

1) Offline component: large amount of computation, the SVD decomposition and reduced user and item matrix can be done offline
2) Online component: important to the performance of the recommender system, only dot product and frequency table formation and sorting.