Recommendations for Streaming Data

K. Subbian[1] C. Aggarwal[2] K. Hegde[1]

[1]Department of Computer Science
University of Minnesota - Minneapolis, MN

[2]IBM Watson Research Center
Yorktown Heights, NY

CIKM 2016

Outline

1. Introduction
   - Motivation
   - Challenges
2. Methodology
   - The Setup
   - Our Algorithm
   - Our Algorithm contd.
3. Conclusion
   - Results
   - Summary

Recommendations in Streaming Setting

- Most current recommender systems are designed in the context of offline setting
- It is desirable to provide real-time recommendations in large-scale scenarios
- Some applications: social networks, movie/book suggestions, dating etc.

Challenges

- Social Network Example:
  - New items keep appearing
  - The underlying user patterns keep changing
  - This causes the recommendations to vary with time
- In-core memory for memory-resident operations is quite limited
- Classical methods like neighborhood-based and latent factor models have shortcomings
  - They require a computationally expensive offline phase
  - Factorizing large matrices is cumbersome when the said matrices are rapidly changing with time

The Setup

- The ratings are received in the format: `<userID, itemID, rating>`
- If rating is drawn from $\{-1, +1\}$, then let users who have given a rating of $+1$ to item i at time t be represented by $P(i, t)$ and -1 by $N(i, t)$
- Exact similarity computation is intensive. Compute it probabilistically by imposing a sort order on the users with the help of hash functions
  - Use d mutually independent hash functions
  - Each hash function takes in an identifier of a user and outputs a random number uniformly distributed in $(0, 1)$

Probabilistic Similarity

- Now, for a given sort order, what is the probability that the first user with a positive rating for item i is the same as the first user with a positive rating for item j?
- This is the probability that both i and j take on the value $+1$ when at least one of them takes on the value of $+1$
- This is

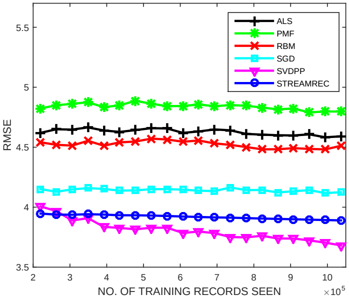
$$\frac{P(i,t) \cap P(j,t)}{P(i,t) \cup P(j,t)}$$

- The above expression represents the similarity between items i and j with respect to positive ratings

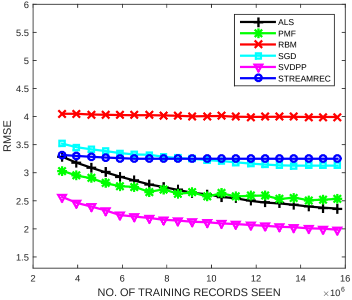
Probabilistic Similarity

- The d mutually independent hash functions are applied to the user indices that have rated item j positively
- For each hash function, the least hash value (*min-hash value*) among these positive users and the corresponding user index (*min-hash index*) are maintained
- d of such pairs are maintained for the n items seen which are easily updatable. This drastically reduces memory requirements
- Similarly, the process is repeated for negatively rated items. The system can be extended for scenarios with multiple ratings as well

Results



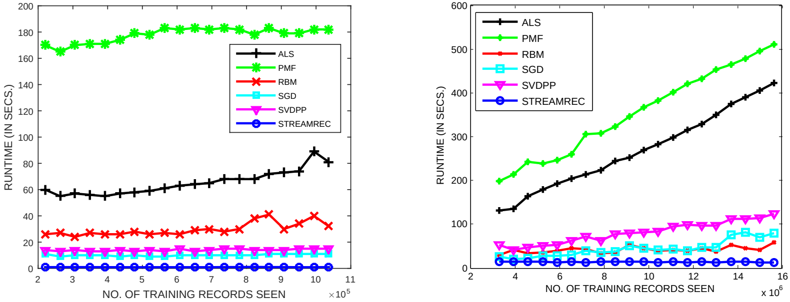
(a) Books RMSE



(b) Dating RMSE

Figure: Efficiency

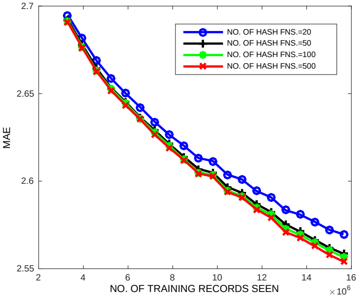
Results Contd.



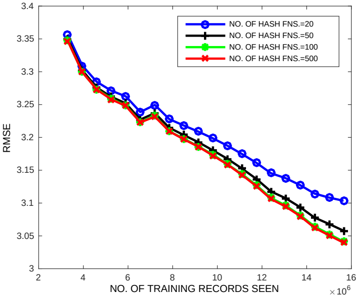
(a) Books Runtime (b) Dating Runtime

Figure: Runtime

Results Contd.



(a) Sensitivity Dating MAE



(b) Sensitivity Dating RMSE

Figure: Sensitivity

Summary

- Proposed an efficient algorithm to perform streaming recommendations using a probabilistic model
- Compact representation of ratings matrix allows for recommendation in online time