BUSINESS INFORMATION SYSTEMS 2014
Larnaca, Cyprus
22-23 May, 2014

LCBM: Statistics-based Parallel Collaborative Filtering

Fabio Petroni [1], *Leonardo Querzoni* [1], *Roberto Beraldi* [1], *Mario Paolucci* [2]

SAPIENZA UNIVERSITÀ DI ROMA

CIS SAPIENZA
CYBER INTELLIGENCE AND INFORMATION SECURITY

CNR ISTC
INSTITUTE OF COGNITIVE SCIENCES AND TECHNOLOGIES

[1] Department of Computer Control and Management Engineering Antonio Ruberti, Sapienza University of Rome - petroni|querzoni|beraldi@dis.uniroma1.it

[2] Institute of Cognitive Sciences and Technologies, CNR - mario.paolucci@istc.cnr.it

Personalization

- Most of todays internet businesses deeply root their success in the ability to provide users with strongly personalized experiences.

- Recommender Systems: are a subclass of information filtering system that seek to predict the *rating* or *preference* that user would give to an item.

Collaborative Filtering

- Collaborative Filtering (**CF**) represents today's a widely adopted strategy to build recommendation engines.

- CF analyzes the known preferences of a group of users to make predictions of the unknown preferences for other users.

Filtering

filter the user data stream satisfying user personal interests

Discovery

recommends new content to the user that matches his interests

Challenges

- Key factors for recommendation effectiveness:

amount of data available

larger data-sets improve quality more than better algorithms

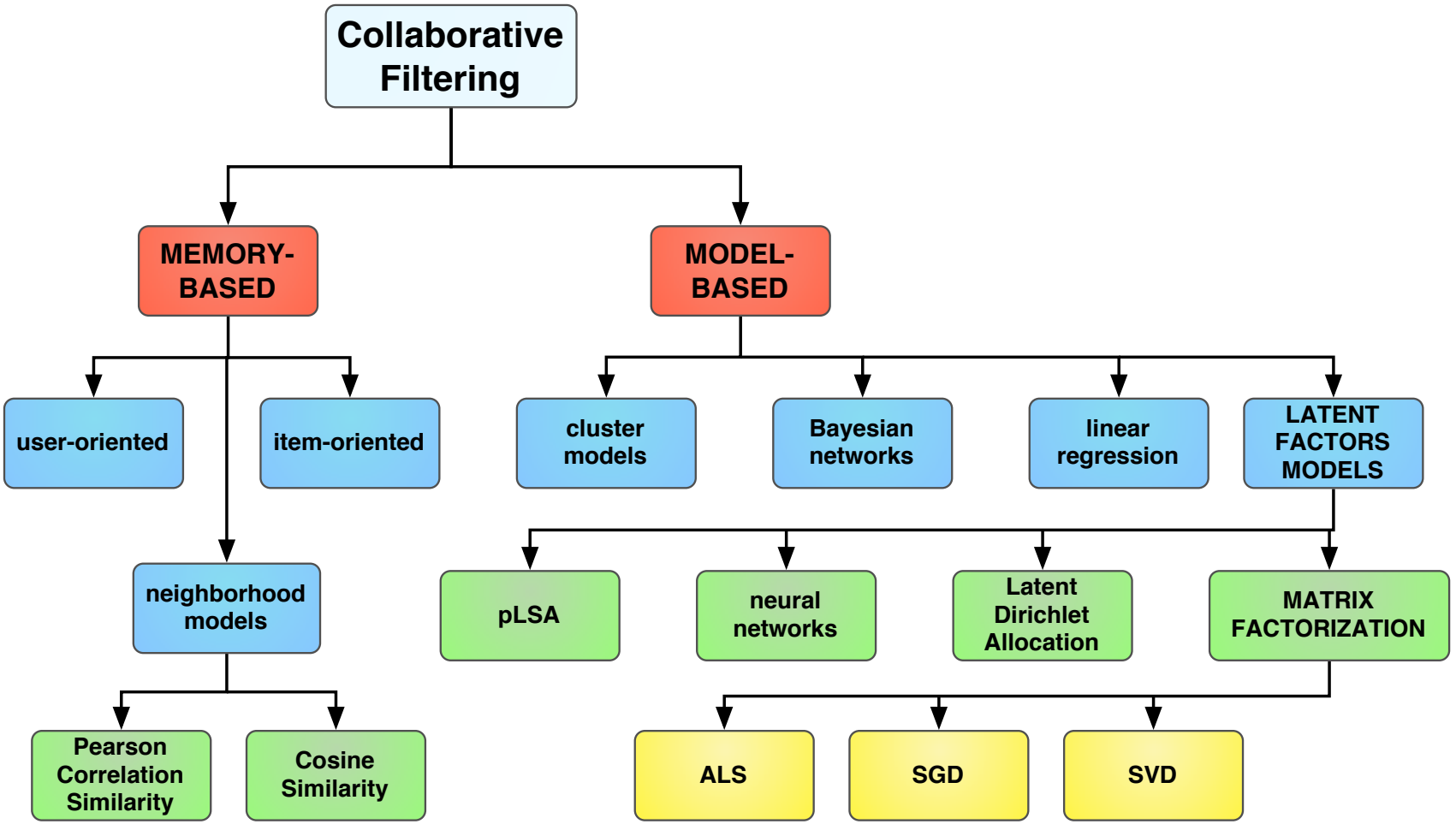
timeliness

timely deliver output to users constitutes a business advantage

✗ Most CF algorithms typically need to go through a very lengthy training stage, incurring in prohibitive computational costs and large time-to-prediction intervals when applied on large data sets.

✗ This lack of efficiency is going to quickly limit the applicability of these solutions at the current rates of data production growth.

Taxonomy Of CF Algorithms



Memory-Based CF Algorithms

- Operate on the entire set of ratings to generate predictions:
  - compute a similarity score, which reflects the distance between two users or two items.
    - *Pearson Correlation Similarity*
    - *Cosine Similarity*
  - Predictions are computed by looking at the ratings of the k most similar users or items (k-nearest neighbor).

✓ Simple design and implementation;

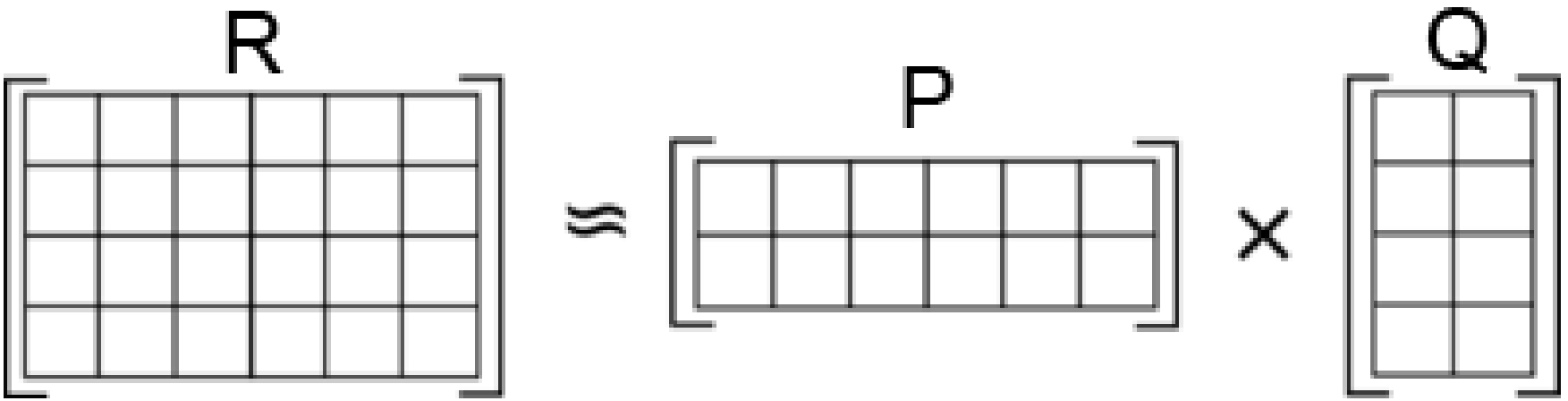
✓ used in a lot of real-world systems.

✗ Impose several scalability limitations;

✗ their use is impractical when dealing with large amounts of data.

Model-Based CF Algorithms

- Use the set of available ratings to estimate or learn a model and then apply this model to make rating predictions.
- The most successful are based on **matrix factorization** models.



Matrix Factorization 1/2

$$LOSS(P, Q) = \sum (R_{ij} - P_i Q_j)^2$$

- The most popular techniques to minimize *LOSS* are *Alternating Least Squares* (**ALS**) and *Stochastic Gradient Descent* (**SGD**).

ALS

ALS alternates between keeping P and Q fixed, solving from time to time a *least-squares* problem.

SGD

SGD works by taking steps proportional to the negative of the gradient of the LOSS function.

- Both algorithms need several passes through the set of ratings.

Matrix Factorization 2/2

- ✓ Provide high quality results;
- ✓ SGD was chosen by the top three solutions of KDD-Cup 2011;
- ✓ there exist parallel and distributed implementations.
- ✗ High computational costs;
- ✗ large time-to-prediction intervals;
- ✗ especially when applied on large data sets.

	SGD	**ALS**	**LCBM**
training time	$O(X \cdot K \cdot E)$	$\Omega(K^3(N+M)+K^2X)$	$O(X)$
prediction time	$O(K)$	$O(K)$	$O(1)$
memory usage	$O(K(N+M))$	$O(M^2+X)$	$O(N+M)$

In the table X is the number of collected ratings, K the number of hidden features, E the iterations, N and M the number of users and items respectively.

Contributions

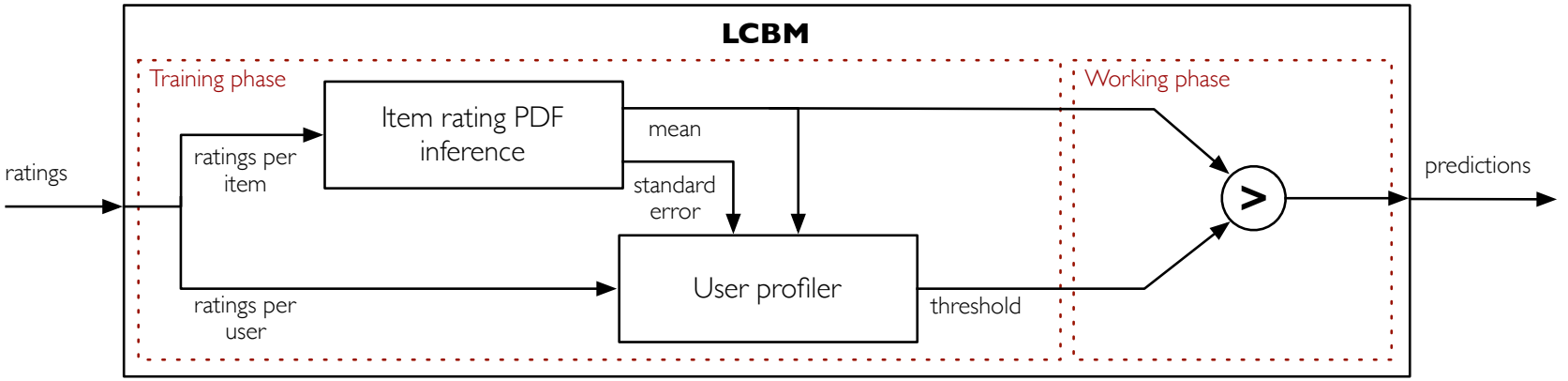
In this paper we present:

LCBM: Linear Classifier of Beta distributions Means

a novel collaborative filtering algorithm for binary ratings that:

- ✓ is inherently parallelizable;
- ✓ provides results whose quality is on-par with state-of-the-art solutions;
- ✓ in shorter time;
- ✓ using less computational resources.

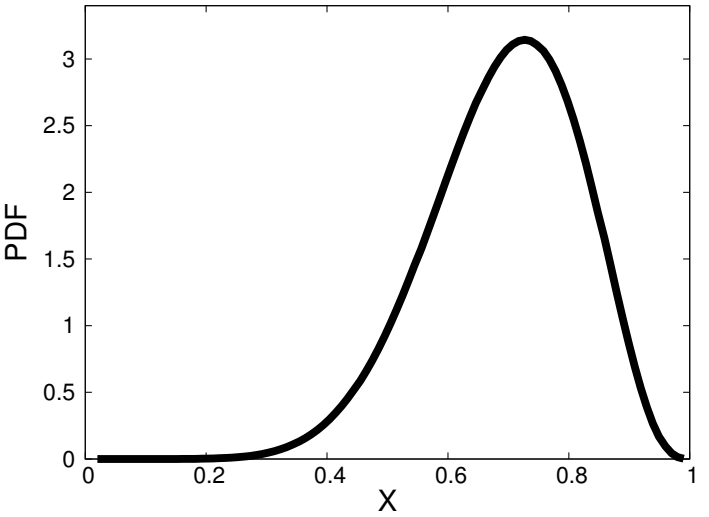
Overview



- We consider:
  - **items** as elements whose tendency to be rated positively/negatively can be statistically characterized using a *probability density function*;
  - **users** as entities with different tastes that rate the same items using different criteria and that must thus be profiled;
- The algorithm needs two passes over the set of available ratings to train the model (build users and items profiles).

Item Rating PDF Inference

- *Beta distribution*:
  - X = the relative frequency of positive votes that the item will receive.
  - $\alpha = OK + 1$, $\beta = KO + 1$



Item profile

$$MEAN = \frac{\alpha}{\alpha+\beta} = \frac{OK+1}{OK+KO+2}$$

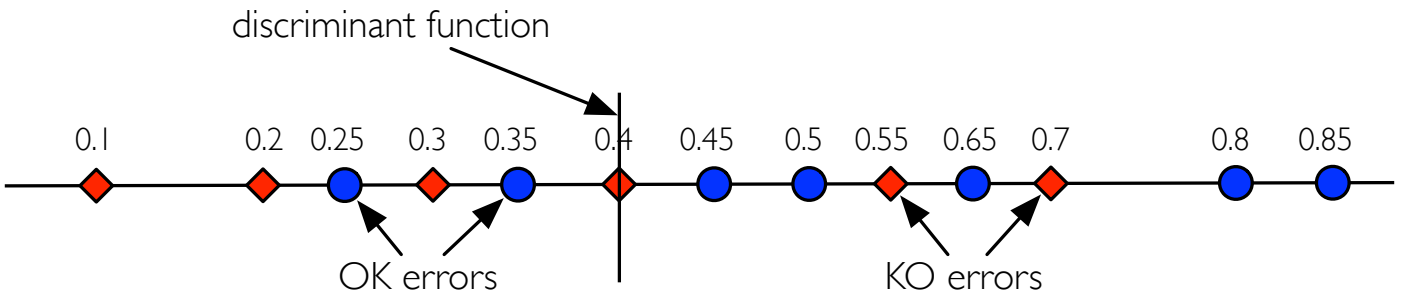
$$SE = \frac{1}{OK+KO+2}\sqrt{\frac{(OK+1)(KO+1)}{(OK+KO)(OK+KO+3)}}$$

- Chebyshev's inequality:

$$Pr(MEAN-\lambda SE \leq X \leq MEAN+\lambda SE) \geq 1-\frac{1}{\lambda^2}$$

- *Example*:
  - $OK = 8$, $KO = 3$
  - $MEAN \approx 0.7$
  - $SE \approx 0.04$
- $Pr(0.62 \leq X \leq 0.78) \geq 0.75$

User Profiler

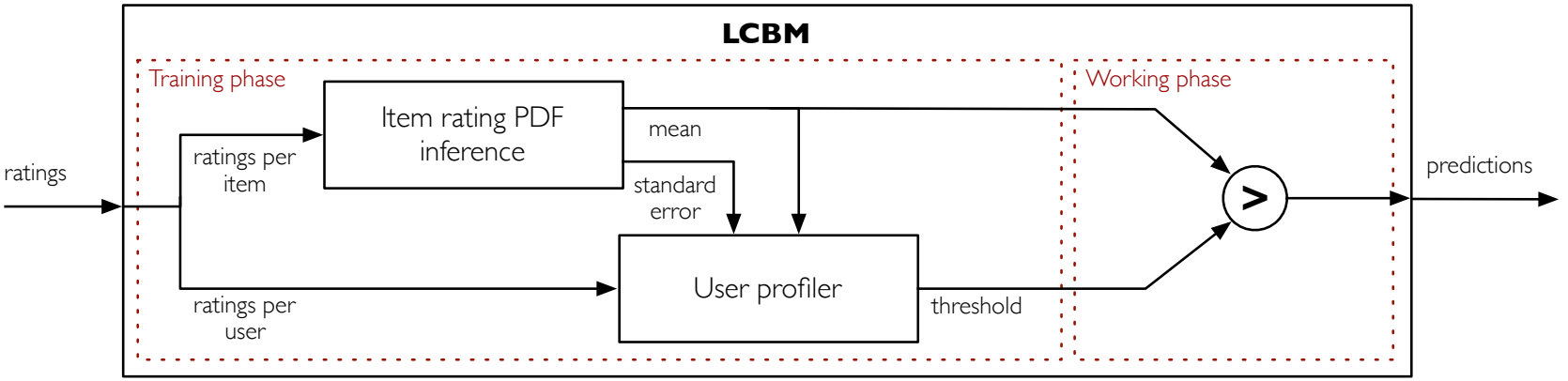


- Every rating is represented by a point with two attributes:
  - a boolean **value** containing the rating (OK or KO).
  - a **key** $\in [0, 1]$ that gives the point's rank in the order;
    - if value $= OK \rightarrow$ key $= MEAN + \lambda SE$
    - if value $= KO \rightarrow$ key $= MEAN - \lambda SE$

User profile

Quality threshold (QT) = the *discriminant function* of a *1D linear classifier* that separates the points with a minimal number of errors.

Working Phase



- In order to produce a prediction the algorithm simply compares QT values from the user profiler with MEAN values from the item's rating PDF inference block:
  - if $MEAN > QT \rightarrow \text{prediction} = OK$
  - if $MEAN \leq QT \rightarrow \text{prediction} = KO$

Experiments: Setting and The Data Sets

Dataset	MovieLens 100k	MovieLens 10M	Netflix
users	943	69878	480189
items	1682	10677	17770
ratings	100000	10000054	100480507

- 5-fold cross-validation approach:
  - This technique divides the dataset in 5 folds and then uses in turn one of the folds as test set and the remaining ones as training set.
- We compared LCBM against CF implementations in *Apache Mahout*:

  - ParallelSGDFactorizer: lock-free and parallel implementation of **SGD**
  - ALSWRFactorizer: **ALS** with Weighted-λ-Regularization

Experiments: Performance Metrics

- Confusion matrix:

true positives (TP)	false negatives (FN)
false positives (FP)	true negatives (TN)

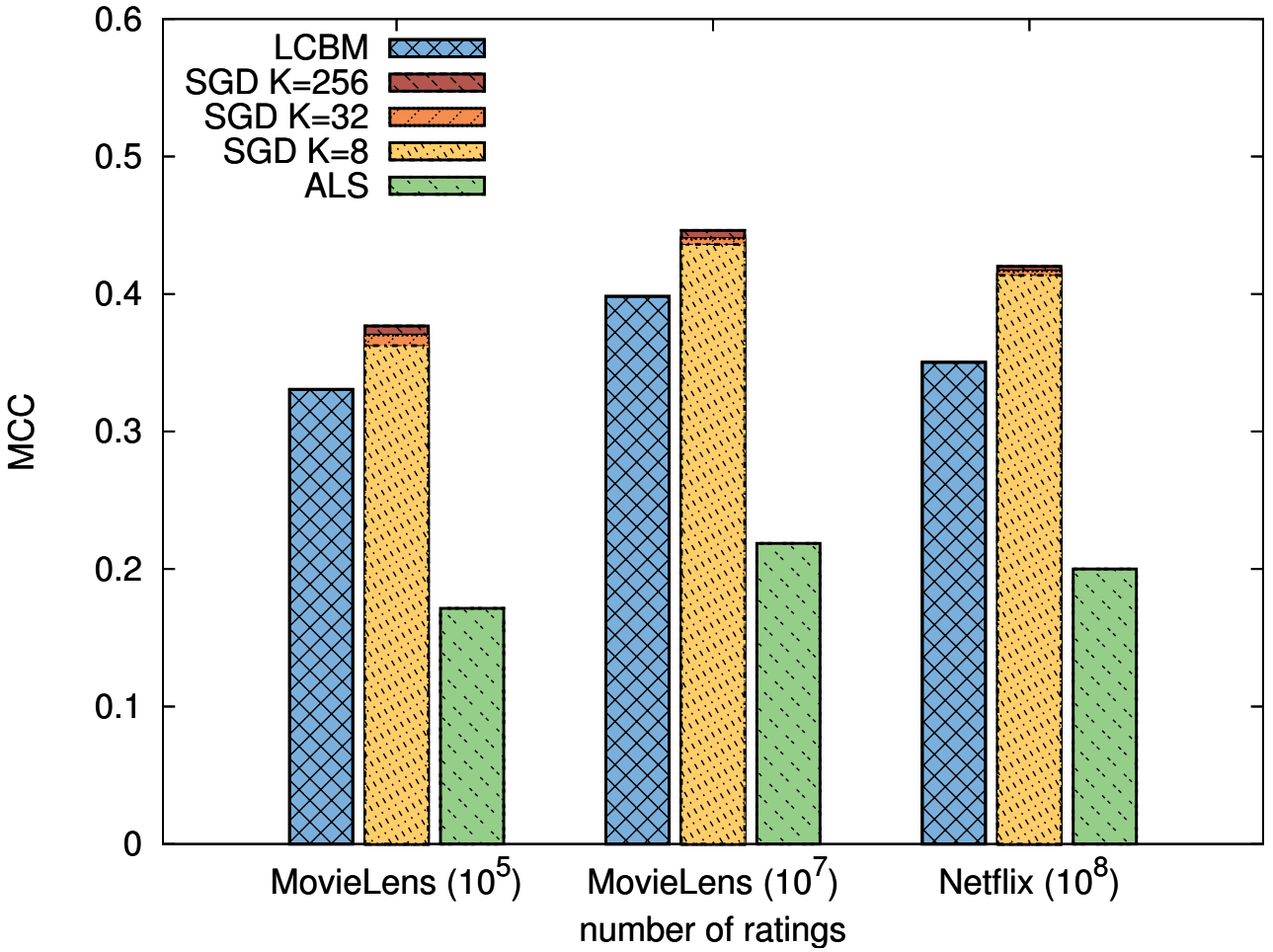
- Matthews correlation coefficient (**MCC**) $\in [-1, 1]$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}$$

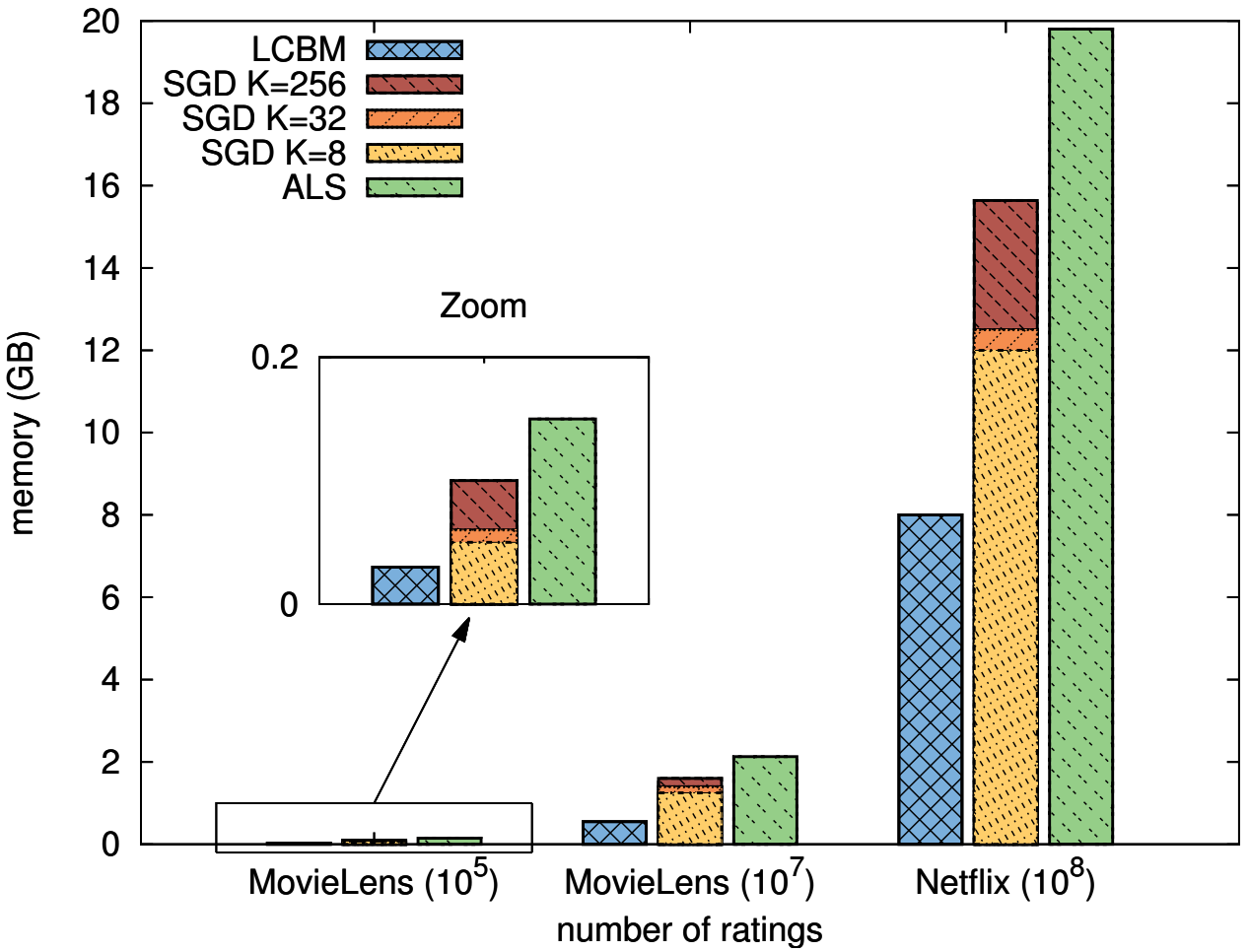
  - $1 \rightarrow$ perfect prediction;
  - $0 \rightarrow$ no better than random prediction;
  - $-1 \rightarrow$ total disagreement between prediction and observation.

- **time** needed to run the test;
- the peak **memory** load during the test.

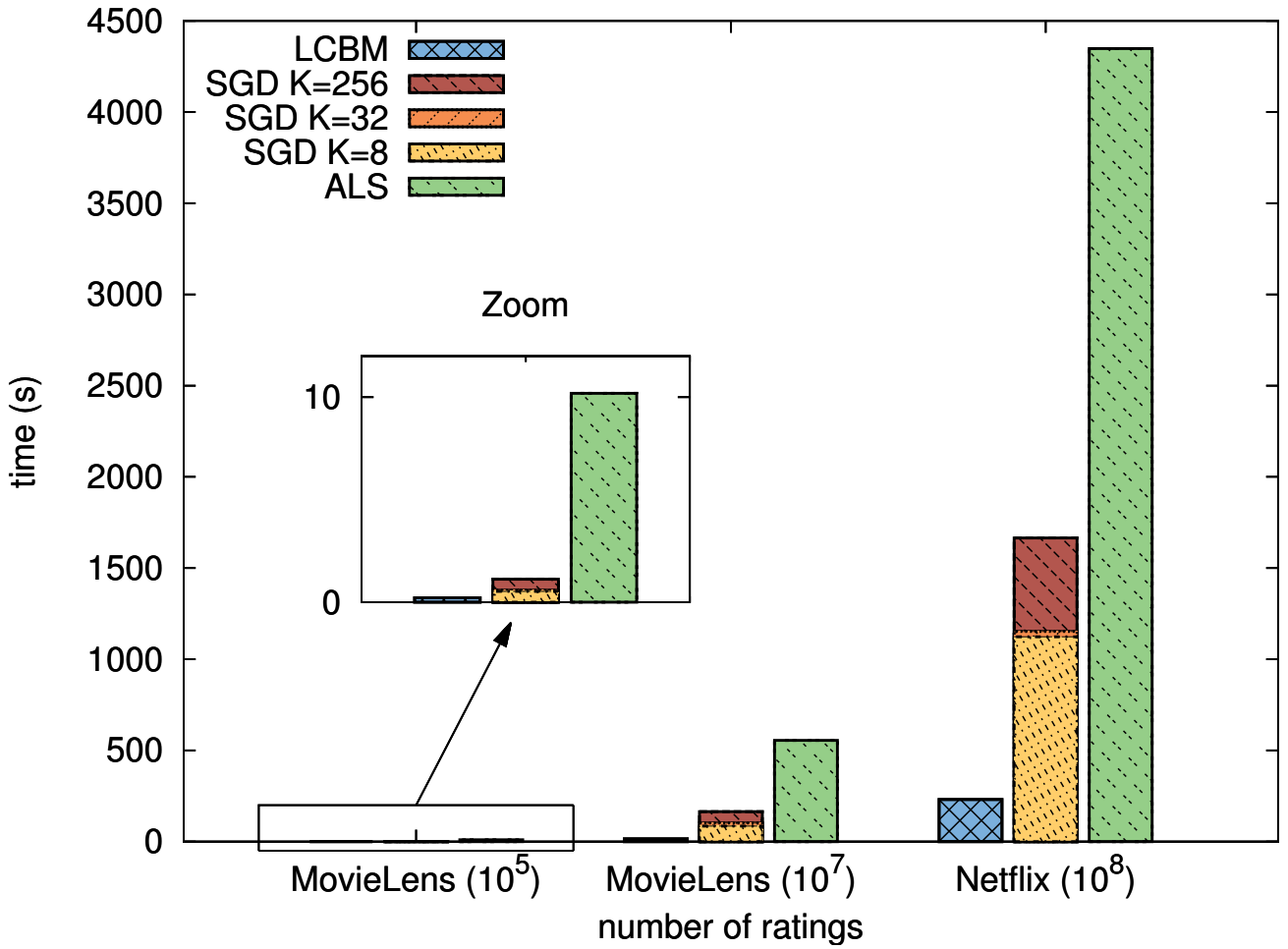
Results: Accuracy



Results: Computational Resources



Results: Timeliness



Conclusions

- LCBM is a novel CF algorithm with binary ratings.
- It works by analyzing collected ratings to:
  - infer a probability density function of the relative frequency of positive votes that the item will receive;
  - compute a personalized quality threshold for each user.
- These two information are used to predict missing ratings.
- LCBM is inherently parallelizable.

Future works:

- Handle multidimensional ratings.
- Provide recommendations: a list of items the user will like the most.

Thank you!

Questions?

Fabio Petroni

Rome, Italy

Current position:

PhD Student in Computer Engineering, Sapienza University of Rome

Research Areas:

Recommendation Systems, Collaborative Filtering, Distributed Systems

petroni@dis.uniroma1.it