Fast Matrix Factorization for Online Recommendation with Implicit Feedback

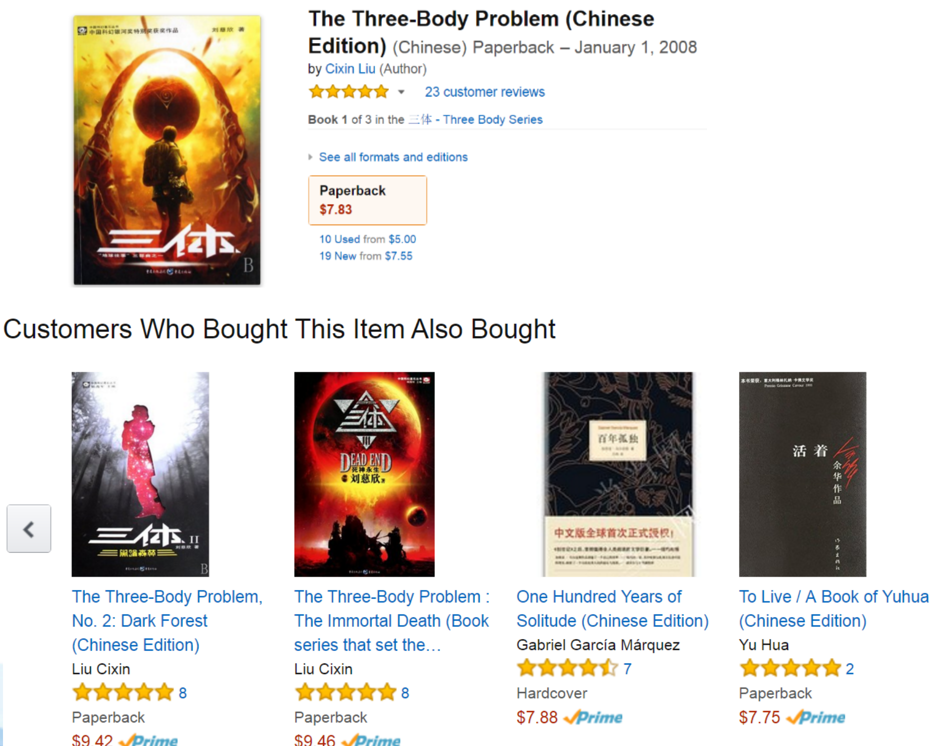
Xiangnan He, Hanwang Zhang, Min-Yen Kan, Tat-Seng Chua

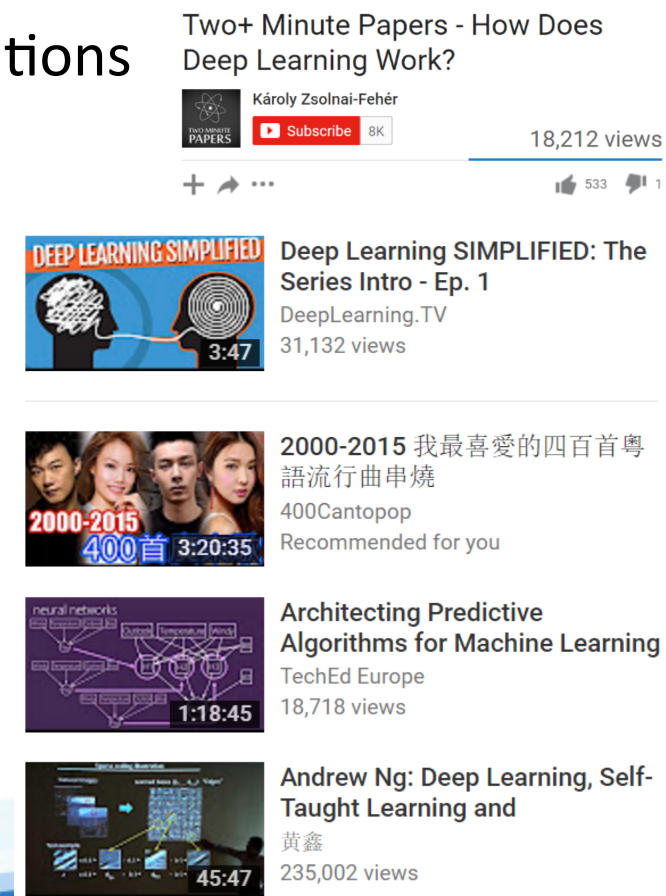
National University of Singapore (NUS)

SIGIR 2016, July 20, Pisa, Italy

Value of Recommender System (RS)

- Netflix: 60+% of the movies watched are recommended.
- Google News: RS generates 38% more click-through
- Amazon: 35% sales from recommendations





Statistics come from Xavier Amatriain

Collaborative Filtering (CF)

- Explicit Feedback
  - Rating prediction problem
  - Popularized by the Netflix Challenge
  - Only observed ratings are considered.
  - But, it is sub-optimal (missing-at-random assumption) for Top-K Recom. (*Cremonesi and K*

items

1	?	5	?
?	2	?	?
4	?	5	?
1	?	?	?
2	?	?	4

users

Real-valued Rating matrix

- Implicit Feedback
  - Ranking/Classification problem
  - Aims at recommending (unconsumed) items to users.
  - Unobserved missing data (0 entries) is important!

items

1	0	0	1
0	1	0	0
1	1	0	0
1	0	0	1

users

0/1 Interaction matrix

Outline

- Introduction
- Technical Background & Motivation
- Popularity-aware Implicit Method
- Experiments (offline setting)
- Experiments (online setting)
- Conclusion

Matrix Factorization (MF)

- MF is a linear latent factor model:

items

users				
	1	0	0	1
	0	1	0	0
	1	1	0	0
	1	0	0	1

0/1 Interaction matrix

User 'u' interacted with **item 'i'**

Learn latent vector for each user, item:

$$1 \times K$$

$$v_u^U$$

$$v_i^I$$

Affinity between user 'u' and item 'i':

$$\hat{y}_{ui} = < v_u, v_i >$$

Previous Implicit MF Solutions

Pair-wise Ranking Method (BPR, Rendle et al, UAI 2009)	Regression-based Method (WALS, *Hu et al, ICDM 2008*)

Sampling negative instances:

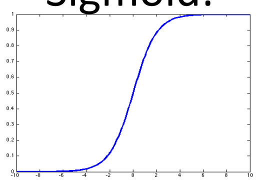
LIKELIHOOD:

$$p(.|\Theta) = \prod_{u} \prod_{i \in \mathcal{R}_u} \prod_{j \notin \mathcal{R}_u} \sigma(\hat{y}_{ui} - \hat{y}_{uj})$$

All Items bought by *u*

Items not bought by *u*

Sigmoid:



Pros:

- \+ **Efficient**
- \+ **Optimized for ranking (good precision)**

Cons:

- \- **Only model partial data (low recall)**

Treating all missing data as negative:

LOSS: Weight for Missing data

$$L(\Theta) = \sum_{(u,i) \in \mathcal{R}} (y_{ui} - \hat{y}_{ui})^2 + \boxed{w_0} \sum_{(u,i) \notin \mathcal{R}} (0 - \hat{y}_{ui})^2$$

Prediction on observed entries

Prediction for missing data

Address the effectiveness and efficiency issue of regression method.

Pros:

\+ Model the full data (good recall)

Cons:

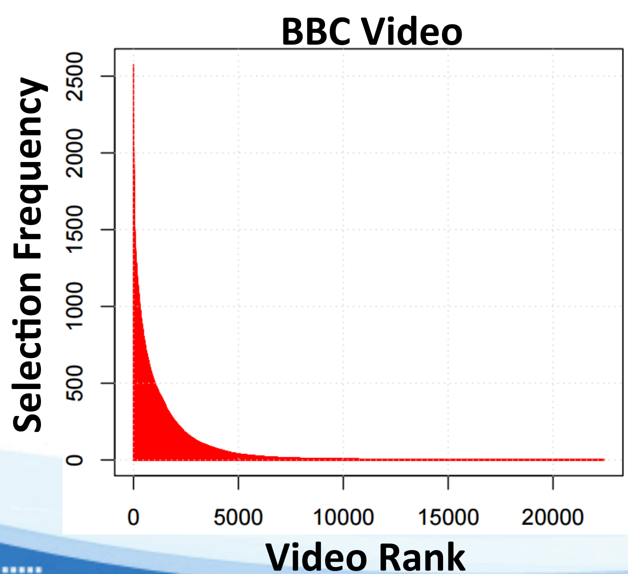
- \- **Less efficient**
- \- **Uniform weighting on missing data.**

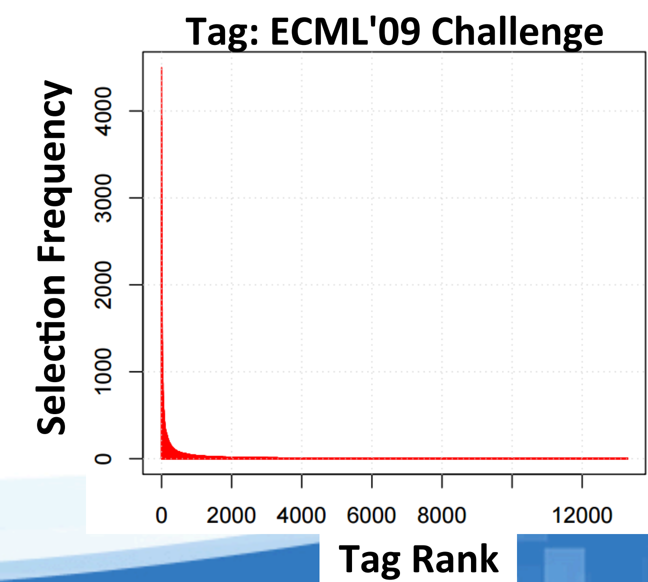
Drawbacks of Existing Methods

(whole-data based)

Uniform Weighting
- Limits model's fidelity and flexibility

- **Uniform weighting on missing data assumes that**
  "all missing entries are equally likely to be a negative assessment."
  - The design choice is for the optimization efficiency --- an efficient ALS algorithm (*Hu, ICDM 2008*) can be derived with uniform weighting.
- **However, such an assumption is unrealistic.**
  - Item popularity is typically non-uniformly distributed.
  - Popular items are more likely to be known by users.





Low Efficiency
- Difficult to support online learning

- An analytical solution known as *ridge regression*
  - Vector-wise ALS
  - Time complexity: $O((M+N)K^3 + MNK^2)$

    M: # of items, N: # of users, K: # of latent factors

Scary complexity and unrealistic for practical usage

- With the uniform weighting, Hu can reduce the complexity to $O((M+N)K^3 + |R|K^2)$

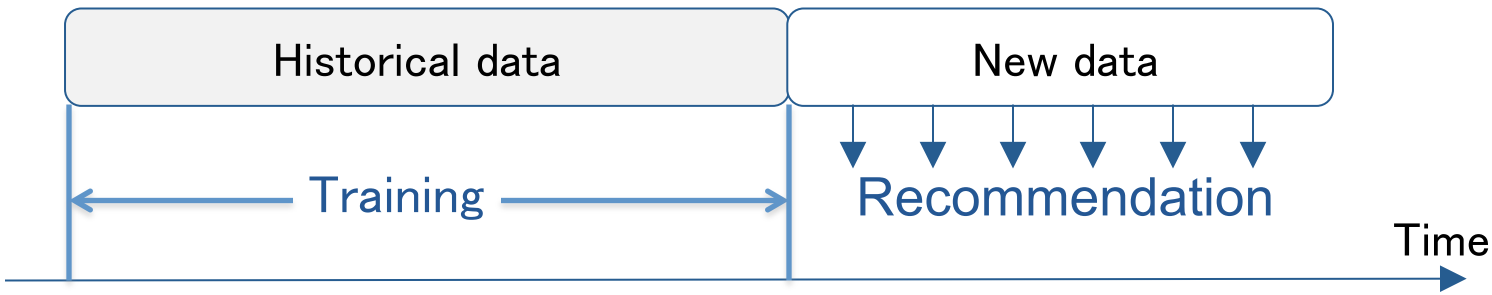
  |R| denotes the number of observed entries.

- However, the complexity is too high for large dataset:
  - K can be thousands for sufficient model expressiveness

    e.g. YouTube RS, which has over billions of users and videos.

Importance of Online Learning for RS

- Scenario of Recommender System:



- New data continuously streams in:
  - New users;
  - Old users have new interactions;
- It is extremely useful to provide *instant personalization* for new users, and *refresh recommendation* for old users, but retraining the full model is expensive

  => Online Incremental Learning

Key Features

Our proposal

- Non-uniform weighting on Missing data
- An efficient learning algorithm (K times faster than Hu's ALS, the same magnitude with BPR-SGD learner)
- Seamlessly support online learning.

#1. Item-Oriented Weighting on Missing Data

Old Design: $$L(\Theta) = \sum_{(u,i)\in\mathcal{R}} (y_{ui} - \hat{y}_{ui})^2 + w_0 \sum_{(u,i)\notin\mathcal{R}} (0 - \hat{y}_{ui})^2$$

Our Proposal: $$L(\Theta) = \sum_{(u,i)\in\mathcal{R}} (y_{ui} - \hat{y}_{ui})^2 + \sum_{u} \sum_{i\notin\mathcal{R}_u} c_i (0 - \hat{y}_{ui})^2$$

The confidence that item *i* missed by users is a true negative assessment

Popularity-aware Weighting Scheme:

- Intuition: a popular item is more likely to be known by users, thus a missing on it is more probably that the user is not interested with it.

Similar to frequency-aware negative sampling in word2vec.

$$c_i = c_0 \frac{f_i^\alpha}{\sum_{j=1}^{N} f_j^\alpha}$$

Smoothness: 0.5 works well

Overall weight of missing data

Frequency of item

#2. Optimization (Coordinate Descent)

- Existing algorithms do not work:
  - SGD: needs to scan all training instance O(MN).
  - ALS: requires a uniform weight on missing data.
- We develop a Coordinate Descent learner to optimize the whole-data based MF:
  - Element-wise Alternating Least Squares Learner (eALS)
  - Optimize one latent factor with others fixed (greedy exact optimization)

Property	eALS (ours)	ALS (traditional)
Optimization Unit	Latent factor	Latent vector
Matrix Inversion	No	Yes (ridge regression)
Time Complexity	O(MNK)	$O((M+N)K^3 + MNK^2)$

Lab for Media Search

#2.1 Efficient eALS Learner

- An efficient learner by using memoization.
- Key idea: memoizing the computation for missing data part:

$$L(\Theta) = \sum_{(u,i)\in\mathcal{R}} (y_{ui} - \hat{y}_{ui})^2 + \boxed{\sum_{u} \sum_{i \notin \mathcal{R}_u} c_i (0 - \hat{y}_{ui})^2}$$

Bottleneck: Missing data part

- Reformulating the loss function:

$$L(\Theta) = \sum_{(u,i)\in\mathcal{R}} [(y_{ui} - \hat{y}_{ui})^2 - c_i \hat{y}_{ui}^2] + \boxed{\sum_{u} \sum_{i} c_i \hat{y}_{ui}^2}$$

Sum over all user-item pairs, can be seen as a prior over all interactions!

This term can be computed efficiently *in O(|R| + MK^2),*
***rather than O(MNK)*. Algorithm details see our paper.**

#2.2 Time Complexity

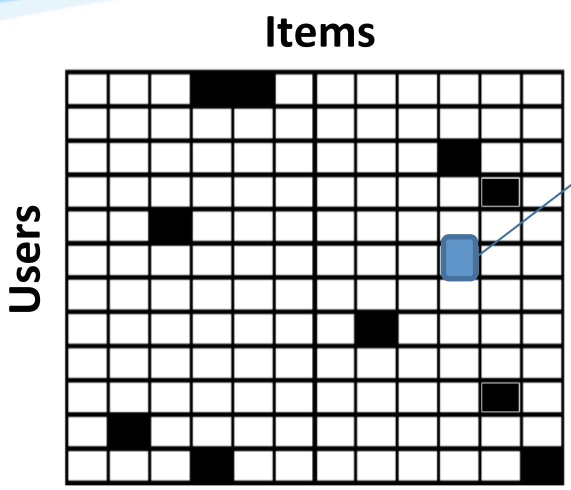
\# of latent factors

$$O((M+N)K^2 + |R|K)$$

\# of users   # of items   # of observed ratings

Linear to data size!

#3. Online Incremental Learning



Black: old training data
Blue: new incoming data

Given a new (*u, i*) interaction, how to refresh model parameters without retraining the full model?

Our solution: only perform updates for v_u and v_i

- We think the new interaction should change the **local features** for *u* and *i* significantly, while the global picture remains largely unchanged.

Pros:

+ Localized complexity: $O(K^2 + (|R_u| + |R_i|)K)$

Outline

- Introduction
- Technical Background & Motivation
- Popularity-aware Implicit Method
- Experiments (offline setting)
- Experiments (online setting)
- Conclusion

Dataset & Baselines

- Two public datasets (filtered at threshold 10):
  - Yelp Challenge (Dec 2015, ~1.6 Million reviews)
  - Amazon Movies (SNAP.Stanford)

Dataset	Interaction#	Item#	User#	Sparsity
Yelp	**731,671**	**25.8K**	**25.7K**	**99.89%**
Amazon	**5,020,705**	**75.3K**	**117.2K**	**99.94%**

- Baselines:
  - ALS (*Hu et al, ICDM'08*)
  - RCD (*Devooght et al, KDD'15*)

    Randomized Coordinate Descent, state-of-the-art implicit MF solution.
  - BPR (*Rendle et al, UAI'09*)

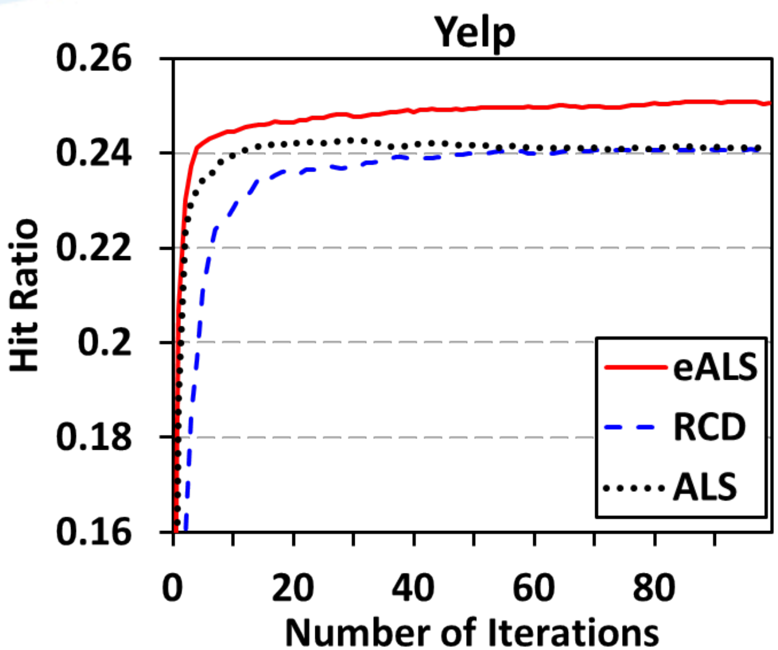
    SGD learner, Pair-wise ranking with sampled missing data.

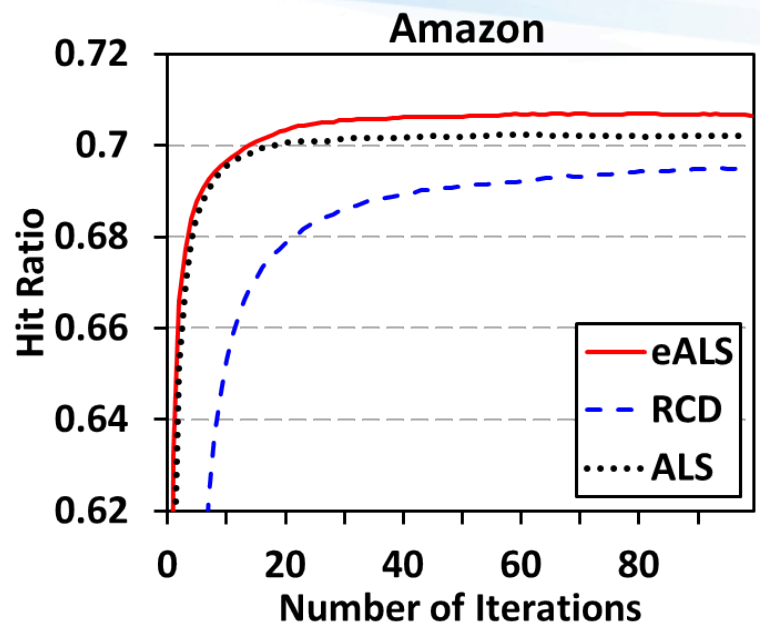
Offline Protocol (Static data)

- Leave-one-out evaluation *(Rendle et al, UAI'09)*
  - Hold out the latest interaction for each user as test (ground-truth).
- Although it is widely used in literatures, it is an artificial split that does not reflect the real scenario.
  - Leak of collaborative information!
  - New users problem is averted.

- Top-K Recommendation (K=100):
  - Rank **all** items for a user (very time consuming, longer than training!)
  - Measure: Hit Ratio and NDCG.
  - Parameters: #factors = 128 (others are also fairly tuned, see the paper)

Compare whole-data based MF



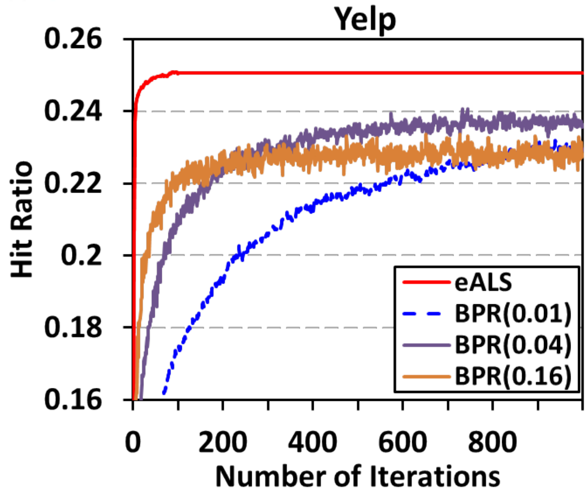


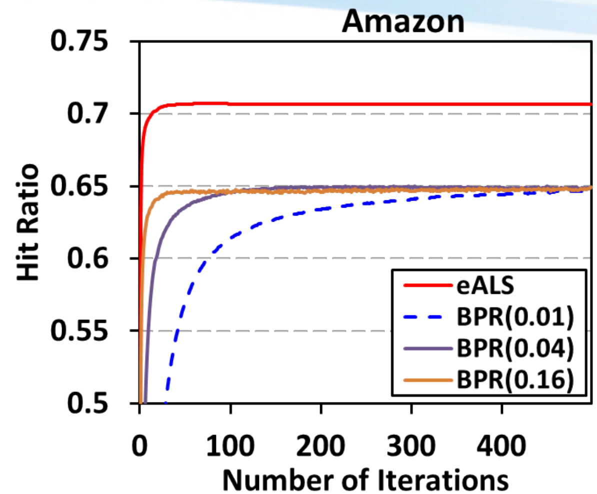
Analysis:

1. eALS > ALS: popularity-aware weighting on missing data is useful.
2. ALS > RCD: alternating optimization is more effective than gradient descent for linear MF model.

Compare with Sampled-based BPR

Hit Ratio

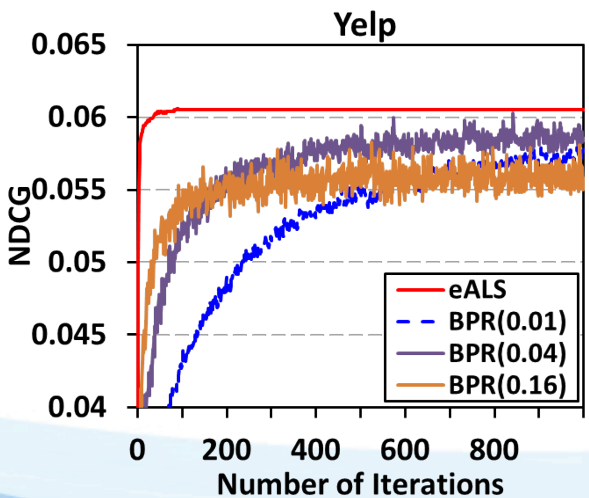


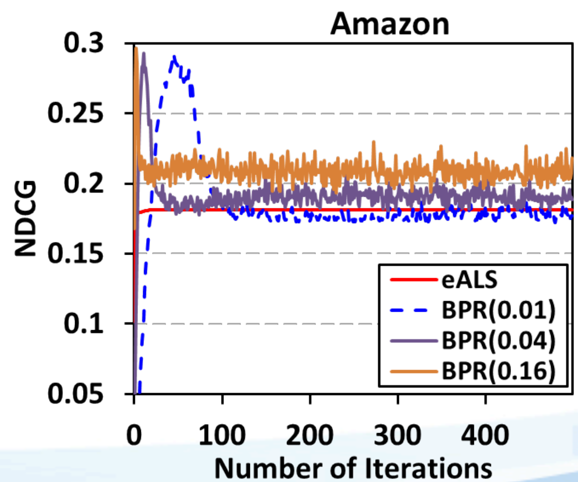


Observation:

1. BPR is a weak performer for Hit Ratio
(low recall, as it samples partial missing data only)

NDCG





2. BPR is a strong performer for NDCG
(high precision, as it optimizes a ranking-aware function)

Lab for Media Search

Efficiency Comparison

Training time per iteration (Java, single-thread)

	Yelp (0.73M)		Amazon (5M)	
Factor#	**eALS**	**ALS**	**eALS**	**ALS**
32	1 s	10 s	9 s	74 s
64	4 s	46 s	23 s	4.8 m
128	13 s	221 s	72 s	21 m
256	1 m	23 m	4 m	2 h
512	2 m	2.5 h	12 m	11.6 h

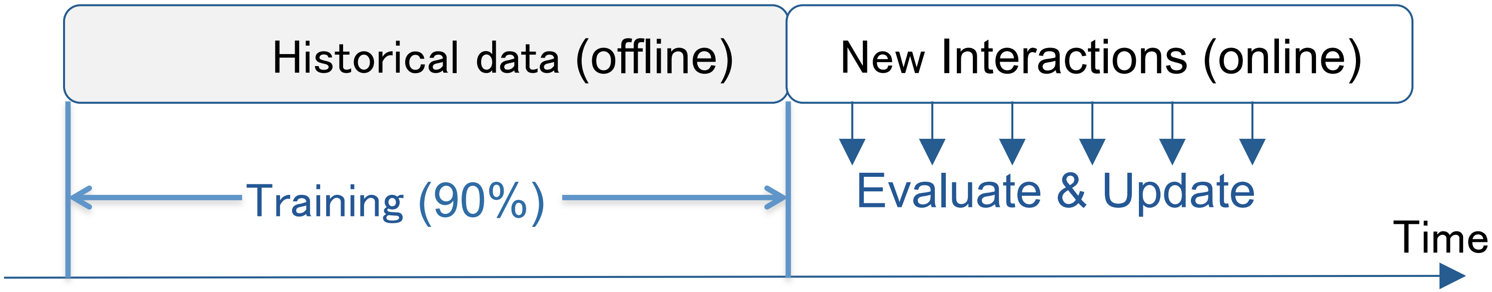
Analytically:
eALS: $O((M+N)K^2 + |R|K)$
ALS: $O((M+N)K^3 + |R|K^2)$

We used a fast matrix inversion algorithm: $O(K^{2.376})$

eALS has the similar running time with RCD (KDD'15), which only supports uniform weighting on missing data.

Online Protocol (dynamic data stream)

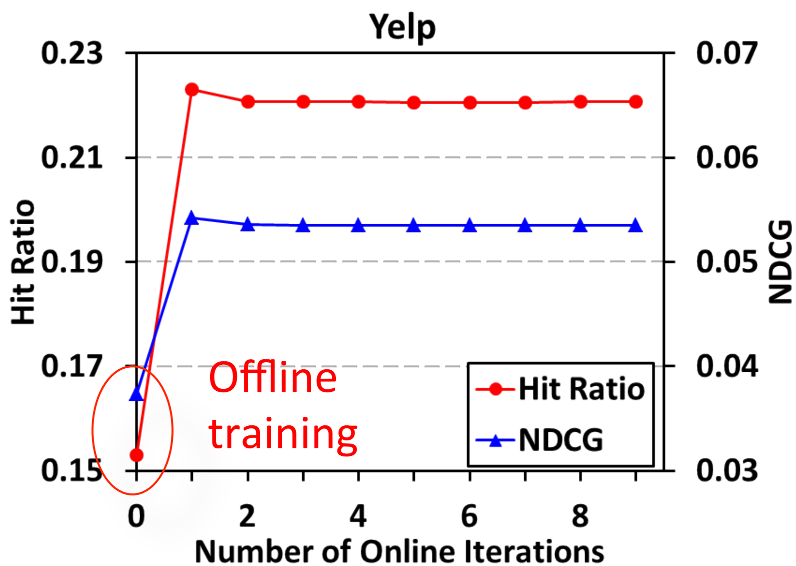
- Sort all interactions by time
  - Global split at 90%, testing on the latest 10%.

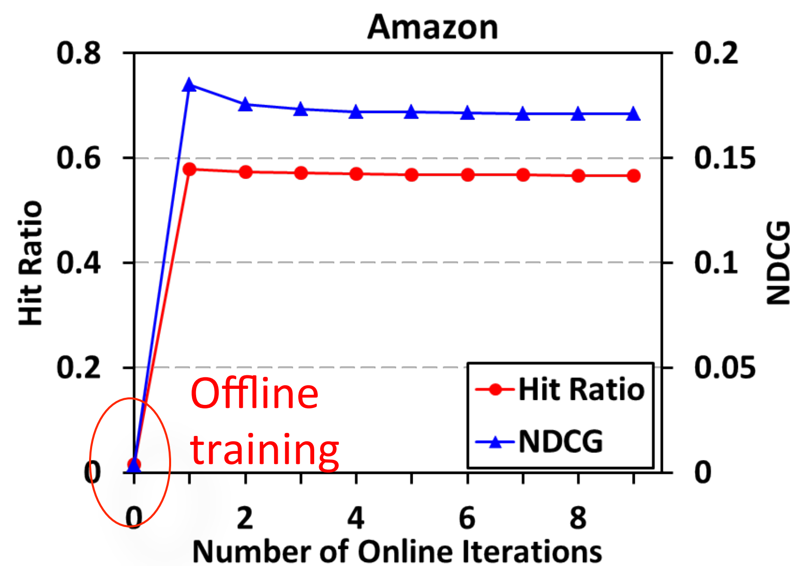


- In the testing phase:
  - Given a *test interaction* (i.e., *u-i* pair), the model recommends a Top-K list to evaluate the performance.
  - Then, the *test interaction* is fed into the model for an incremental update.
- New users problem is obvious:
  - 57% (Amazon) and 14% (Yelp) test interactions are from new users!

Number of Online Iterations

Impact of online iterations on eALS:



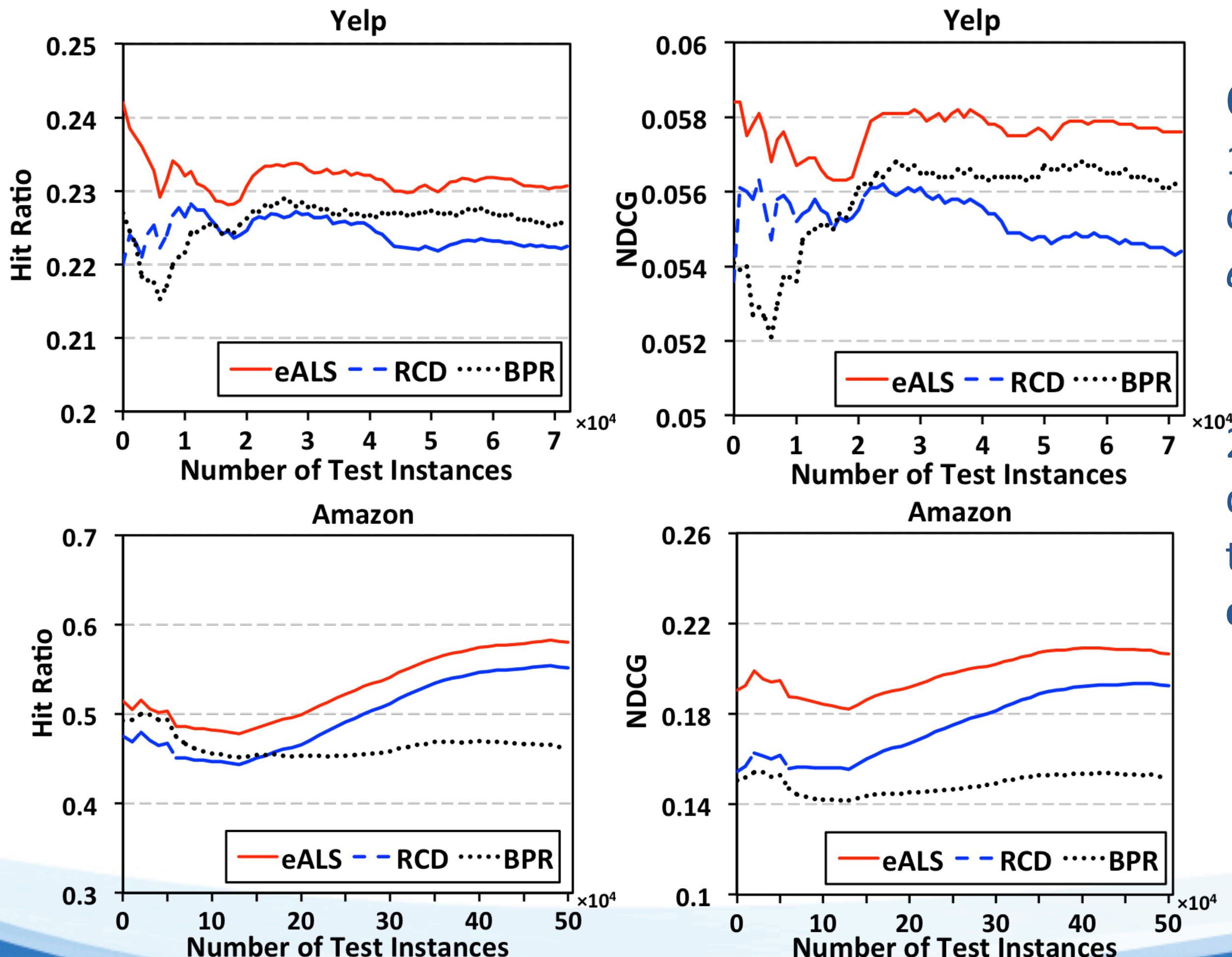


One iteration is enough for eALS to converge!

While BPR (SGD) needs 5-10 iterations.

Compare dynamic MF methods

Performance evolution *w.r.t.* number of test interactions:



Observations:

1. Our eALS consistently outperforms RCD (*Devooght et al, KDD'15*) and BPR

2. Performance trend – first decreases (**cold-start cases**), then increases (**usefulness of online learning**).

Conclusion

- Matrix Factorization for Implicit Feedback
  - Model the full missing data leads to better prediction recall.
  - Weight the missing data non-uniformly is more effective.
  - Develop an efficient algorithm that supports online incremental learning.

- Explore a new way to evaluate recommendation in a more realistic, better manner.
  - Simulate the dynamic data stream.

- Our efficient eALS technique is a generic solution, which can solve MF with any weighting scheme of missing data.
  - Item-oriented (this work) is just a special case.

Future Work

- Online Recommendation:
  - Balance Short-term (online data) and Long-term preference (offline data).

- Our technique is promising for other applications, e.g., in representation learning of words:
  - GloVe models observed entries only.
  - Word2vec samples negative entries.
  - Recently, Google develops Swivel that accounts for the full missing data, leading to better embedding but very high time complexity.

Thanks!

Codes available: https://github.com/hexiangnan/sigir16-eals