# TRANSFER LEARNING FOR ONE-CLASS RECOMMENDATION BASED ON MATRIX FACTORIZATION

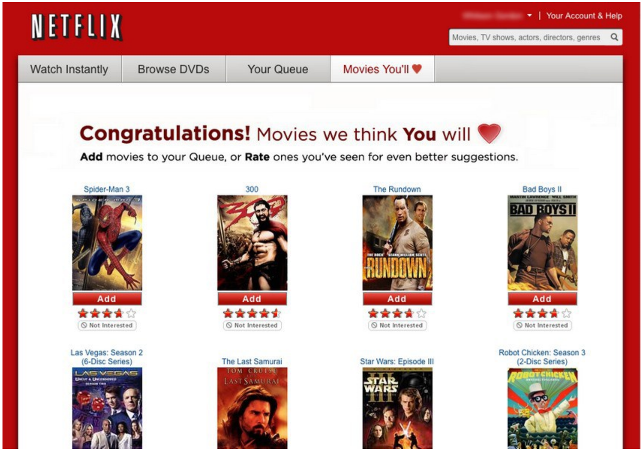
XIE Ruiming

2015.02.12

## Outline

## Outline

# Traditional recommender system

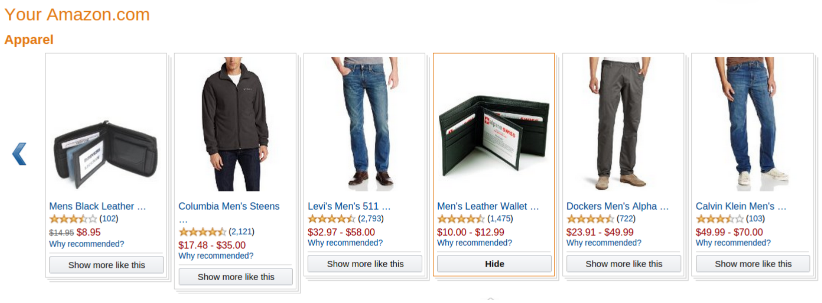


**Congratulations!** Movies we think **You** will ♥

**Add** movies to your Queue, or **Rate** ones you've seen for even better suggestions.

The data have multiple values(ratings from 1 to 5).

## One-class recommender system



Your Amazon.com

Apparel

| Mens Black Leather ... | Columbia Men's Steens ... | Levi's Men's 511 ... | Men's Leather Wallet ... | Dockers Men's Alpha ... | Calvin Klein Men's ... |
|---|---|---|---|---|---|
| (102) | (2,121) | (2,793) | (1,475) | (722) | (103) |
| ~~$14.95~~ $8.95 | $17.48 - $35.00 | $32.97 - $58.00 | $10.00 - $12.99 | $23.91 - $49.99 | $49.99 - $70.00 |
| Why recommended? | Why recommended? | Why recommended? | Why recommended? | Why recommended? | Why recommended? |
| Show more like this | Show more like this | Show more like this | Hide | Show more like this | Show more like this |

The data usually are binary. No ratings are available.

## Matrix factorization & Transfer Learning

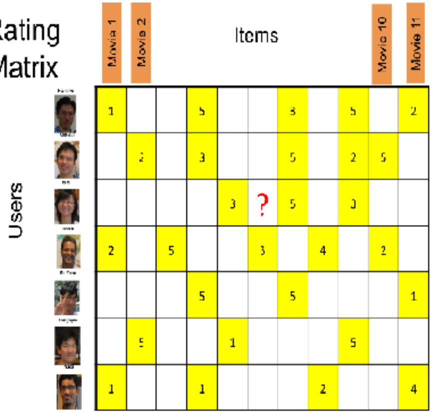
### Matrix factorization

Matrix factorization models map both users and items to a joint latent factor space of dimensionality f, such that user-item interactions are modeled as inner products in that space.
Matrix tri-factoriztion factorizes a matrix $M$ into three parts : $M = USV^T$. Where $U, V$ stand for user/item clusters and $S$ stands for cluster relationships.

### Transfer Learning

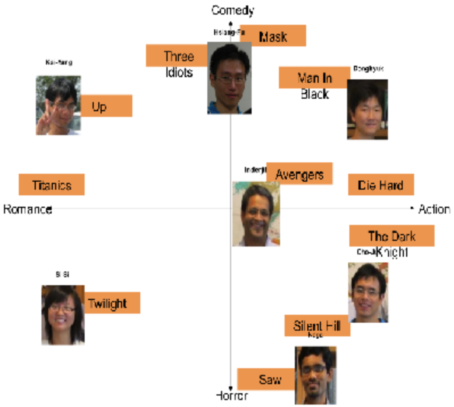
As a way to transfer knowledge from different domains, transfer learning can be applied in recommender systems to leverage data in different system with shared users.
There exists some research that use transfer learning to tackle CF problems.

## Rating Matrix



Items
Users

| Movie 1 | Movie 2 | | | | | | | | Movie 10 | Movie 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | 5 | | | 3 | | 5 | | 2 |
| | 2 | | 3 | | | 5 | | 2 | 5 | |
| | | | | 3 | ? | 5 | | 3 | | |
| 2 | | 5 | | | 3 | | 4 | | 2 | |
| | | | 5 | | | 5 | | | | 1 |
| | 5 | | | 1 | | | | 5 | | |
| 1 | | | 1 | | | | 2 | | | 4 |



## Outline

## Motivation

- Users often have different kinds of actions in one website. We can treat different actions as different domains and adopt transfer learning methods.
- The actions are mostly binary.
- Current methods can't solve transfer learning in one-class recommendation.

## Outline

## TRIMF

### Problem background

- In an online shopping site, there are two main actions - click and purchase. Both can form a matrix which consists of binary data. And the primary goal for the site is to increase conversion rate.
- The matrices are sparse, but add them together into one matrix is too simple to success. So we need to consider a transfer learning method to leverage data better.

## Problem Definition

- Input: click matrix $X_c^{m_c*n_c}$, deal matrix $X_d^{m_d*n_d}$
  $m_c, m_d$ are the number of users in each matrix while $n_c, n_d$ denote the number of items.
- Output: prediction matrices $P_c^{m_c*n_c}, P_d^{m_d*n_d}$.

## Weighting scheme

Former one-class CF methods aim at giving entries different weights. Thus different confidence levels can be set. TRIMF combines two weight schemes together.

- Positive entries: $W_{ij} = 1 + log(n)$
    - Higher frequency can mean that we are more confident about the entry.
- Negative(Unknown) entries: $W_{ij} = log(\sum_j X_{ij})$
    - If a user has more positive examples, it is more likely that the user does not like the missing items.

## Clustering effects in shopping site

| Top click items | Top purchase items |
|---|---|
| Iphone 5s | Tissue |
| Xiaomi 3 | Laundry soap powder |
| Thinkpad | Xiaomi 3 |
| CPU | Snacks |
| Hard disk | Battery |
| Router | Iphone 5s |
| Earphone | Mouse |

Table : Top 10 click items and purchase items in Yixun.

There are some categories which users tend to buy after click. Also users have different shopping habits, some like window-shopping and some like to buy right after clicking.

## Transfer Learning in TRIMF

Traditional transfer methods in CF only share a certain part of matrices through matrix tri-factorization.
TRIMF shares user-latent factors and rating patterns, while designing a cluster-level mapping function.

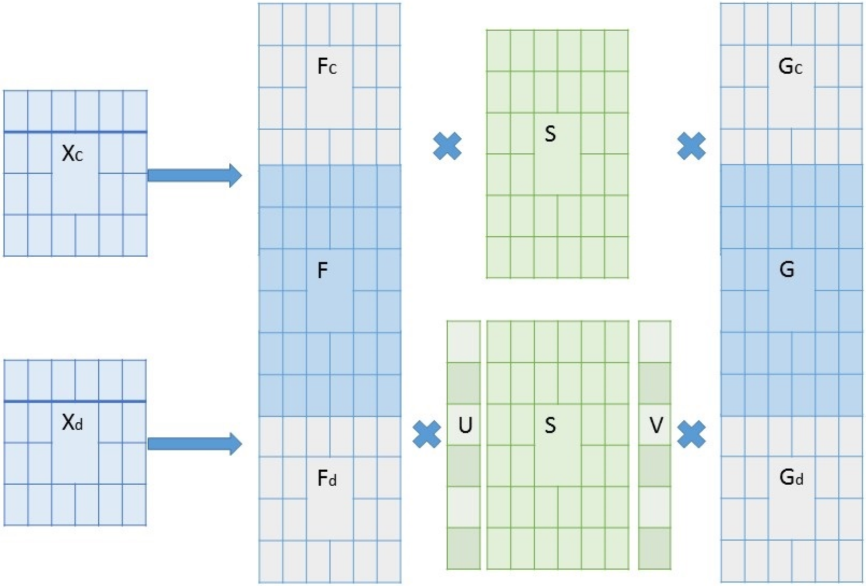
## TRIMF

### Objective function

$$min_{F,G,S,U,V} W_c \odot ||X_c - (F;F_c)S(G;G_c)'||_2$$
$$+W_d \odot ||X_d - (F;F_d)(USV)(G;G_d)'||_2$$

- $W_c, W_d$ are the weights for $X_C, X_d$.
- $F, G$ are the soft clustering result matrices for overlapped users(items), they are forced to be the same. $F_c, F_d, G_c, G_d$ are matrices for unique users(items).
- $U, V$ are two diagonal matrices, $U_{ii}$ scales every $S_{i*}$ to $U_{ii}S_{i*}$, and models the users' cluster-level transformation from click to deal. While $V_{jj}$ scales every $S_{*j}$ to $S_{*j}V_{jj}$, it models the items' cluster-level transformation from click to deal.

# TRIMF

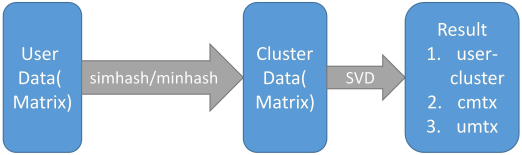


## Outline

## Limitation of TRIMF

When data are coming from multiple sources (e.g. click, pageview and add cart), TRIMF treats every source equally and puts each of them into one matrix which is very sparse. More actions will produce more matrices, increasing complexity.
What is more, in reality the data is much sparser than datasets in experiment. We cannot guarantee achieving equal performance.

## CBMF



we have developed a matrix factorization framework based on a clustering and scoring scheme(CBMF). And apply it in an online shopping site(Tencent).

## Clustering method in CBMF

In Tencent, we have 800,000,000 users in total, and their feature vectors dimensions can be as large as 1,000,000. Our method first convert sparse vectors into dense vectors(Simhash) then perform one-phase clustering(Minhash).

**Simhash**

Simhash is a kind of locality sensitive hashing(LSH) where similiar items are hashed to similiar hash values. We can use hamming distance between two hase values to compare their similarity.

## Clustering method in CBMF

### Jaccard similarity

The Jaccard similarity coefficient is a commonly used indicator of the similarity between two sets:

$$J(A, B) = \frac{A \cup B}{A \cap B}$$

### Minhash

Let $h$ be a hash function, and for any set $S$ define $h_{min}(S)$ to be the minimal member of $S$ with respect to $h$. The probability that $h_{min}(A) = h_{min}(B)$ is true is equal to the similarity $J(A, B)$.

We can produce multiple $h^i_{min}A$ for each set then concatenate them together as our hash-key. The probability that two sets $A, B$ will agree the concatenated hash-key is equal to $J(A, B)^p$.

## Simhash & Minhash using MapReduce

- Map(input:user vector $u_i$):
  - calculate Simhash of $u_i$, $s(u_i)$
  - calculate Minhash of $s(u_i)$ for $p$ times, concatenate them together.
  - output $(minhash_i, u_i)$, $minhash_i$ is the cluster id of $u_i$
- Reduce:
  - for one $minhash_i$, add all vectors that belong to $minhash_i$ together then normalize it. We can generate a cluster-vector $c_i$.
  - output $(minhash_i, c_i)$.

## Feature scoring in CBMF

- A user can have many actions, including click, purchase and pageview. CBMF integrate these actions into one matrix. For a specific action to an item, the score given should depend on how much impact the action has.
- For example, the conversion rate for item iPhone 6 is $CVR(iphone6_{all})$. The conversion rate for iPhone 6 in people who had clicked it is $CVR(iphone6_{click})$. Then their log ratio $log(\frac{CVR(iphone6_{click})}{CVR(iphone6_{all})})$ becomes our score for click.
- We add a weighted sum for all actions as our final score.

## Matrix factorization in CBMF

- Once the matrices are generated, we use Singular Value Decomposition(SVD) from mahout to produce our results.
- For expericeced users we provide a mixed result(direct and clustering), for new users we provide results from his cluster.

## Outline

## Offline experiments

### Datasets

- Yixun: A dataset from Yixun, a large online retailer for electronic products, that has been sampled from log data from two weeks.
- Tmall: A similar but smaller anonymized dataset from a shopping portal containing user interactions of different types.

| Datasets | Yixun | Tmall |
|---|---|---|
| users(click) | 16,240 | 884 |
| items(click) | 1,932 | 9,531 |
| sparsity(click) | 0.006 | 0.021 |
| users(purchase) | 2,520 | 884 |
| item(purchase) | 1,791 | 4,312 |
| sparsity(purchase) | 0.0003 | 0.001 |

Table : Dataset characteristics.

## Metric

We use *prec*@5 and *prec*@10 as our evaluation metrics. Precision is the fraction of clicked items that are shown to the user.

$$Precision = \frac{\|click\|}{\|Impression\|}$$

Precision takes all retrieved documents into account, but it can also be evaluated at a given cut-off rank, considering only the topmost results returned by the system. This measure is called **precision at n** or *prec*@*n*.

## Baseline methods

- non-transfer: For all non-transfer methods, we use three combinations of matrices as our training matrix:deal, click, deal+click, and report their best performance.
  - **Most Popular**: selects top-n items globally, and provides the same recommendation results for every user.
  - **Singular Value Decomposition(SVD)**: a typical method used in recommender system.
  - **Non-negative Matrix Factorization(NMF)**.
  - **Probabilistic Matrix Factorization(PMF)**: a recently proposed method for missing value prediction.
  - **BPRMF**: BPR is a generic optimization criterion for personalized ranking. Unlike traditional methods whose objective function is point-wise, BPR is a pair-wise object function. BPRMF implements BPR using matrix factorization.
  - **WRMF**: One-class collaborative filtering(WRMF) is a weighted low rank approximation method optimized for an implicit dataset.

## Baseline methods

- transfer methods:
  - **Collective Matrix Factorization(CMF)**: is proposed for jointly factorizing two matrices. CMF has been proven to be an effective cross-domain recommendation approach.
  - **TCF**: is a transfer learning method used to predict missing ratings via heterogeneous feedback.

## Offline experiment results

| Method | Prec@5 | Prec@10 |
|---|---|---|
| Most Popular | 0.0323 | 0.0289 |
| SVD | 0.0438 | 0.0367 |
| NMF | 0.0403 | 0.0324 |
| PMF | 0.0435 | 0.0372 |
| BPRMF | 0.0444 | 0.0364 |
| WRMF | 0.049 | 0.0403 |
| CMF | 0.0436 | 0.0350 |
| TCF | 0.0453 | 0.0369 |
| TRIMF | **0.0525** | **0.0410** |
| CBMF | **0.512** | **0.403** |

Table : Performance comparision on Yixun users who have deal data.

## Offline experiment results

| Method | Prec@5 | Prec@10 |
|---|---|---|
| Most Popular | 0.0090 | 0.0085 |
| SVD | 0.0123 | 0.00113 |
| NMF | 0.0091 | 0.0089 |
| PMF | 0.0121 | 0.0112 |
| BPRMF | 0.0142 | 0.0130 |
| WRMF | 0.0174 | 0.0144 |
| CMF | 0.0176 | 0.0139 |
| TCF | 0.0158 | 0.0127 |
| TRIMF | **0.0189** | **0.0153** |
| TRIMF(without remap) | 0.0175 | 0.0146 |
| CBMF | **0.0181** | **0.0144** |

Table : Performance comparision on Yixun users who have click data.

## Offline experiment results

| Method | Prec@5 | Prec@10 |
|---|---|---|
| Most Popular | 0.00508 | 0.00405 |
| SVD | 0.00453 | 0.00413 |
| NMF | 0.00401 | 0.00389 |
| PMF | 0.00421 | 0.00312 |
| BPRMF | 0.00542 | 0.00430 |
| WRMF | 0.00485 | 0.00345 |
| CMF | 0.00512 | 0.00432 |
| TCF | 0.00534 | 0.00502 |
| TRIMF | **0.00720** | **0.00606** |
| CBMF | **0.00612** | **0.00503** |

Table : Performance comparision on Tmall users.

## Offline experiment analysis

### The effects of cluster-level transformation

- In our assumption, $U, V$ are two mapping matrices that describe the difference in user clusters and item categories. To see whether $U, V$ really reflect the phenomenon, we manually check entries in $U, V$ with high and low values. We found that high values in $V$ reflect item clusters that people tends to buy after clicking, e.g. toothbrush, snacks. While low values of $V$ more reflects items that are popular but people may not buy immediately, e.g. cell phones and laptops.
- If we map $UV$ back to the click matrix. Thus the learned cluster-pattern $S$ is transformed from the click pattern to the purchase pattern.

## Offline experiment analysis

### The effects of latent vector sharing

- In TRIMF, for the same user the latent vectors are unique.
- To see that sharing $F, G$ really works, we randomly select another 6000 users and 1500 items from Yixun, make two new matrices $X'_c, X'_d$ to perform another test.
  - share$FG$ : TRIMF
  - not share : we update $F_c, G_c, F_d, G_d$ separately, pretending there is no overlapping users or items.
  - random share : we randomly choose 3000 users and 800 items, marking them as overlapping,

## Offline experiment analysis

| Method | Prec@5 | Prec@10 |
|---|---|---|
| share$FG$ | **0.0436** | **0.0350** |
| not share | 0.0335 | 0.0306 |
| random share | 0.0344 | 0.0299 |

Table : The effect of sharing.

## Scalability

We select the Yixun dataset. We implement TRIMF and CBMF using Python 2.7 with Numpy and Scipy, and run them using a laptop with Intel Core i5-4200 with 4 cores at 2.4GHz and 12GB memory.
For each method, the update algorithm stops while training loss is less than $n$.

- we set different threshold for $n$, and test the convergence speed for each method.
- we reduce the size of click matrix, and test the training speed regarding to dataset size.
- we set different number of parameters(latent dimension k) for each method, and test the training speed with regard to parameter size.

## Scalability

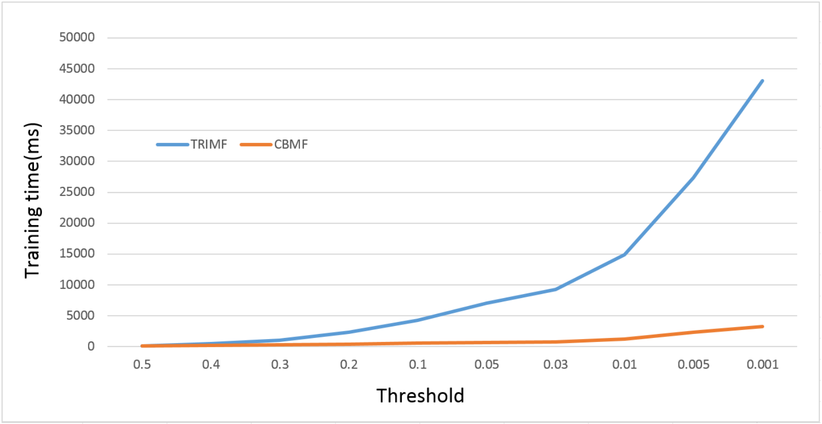


Figure : Convergence speed with regard to threshold.

## Scalability

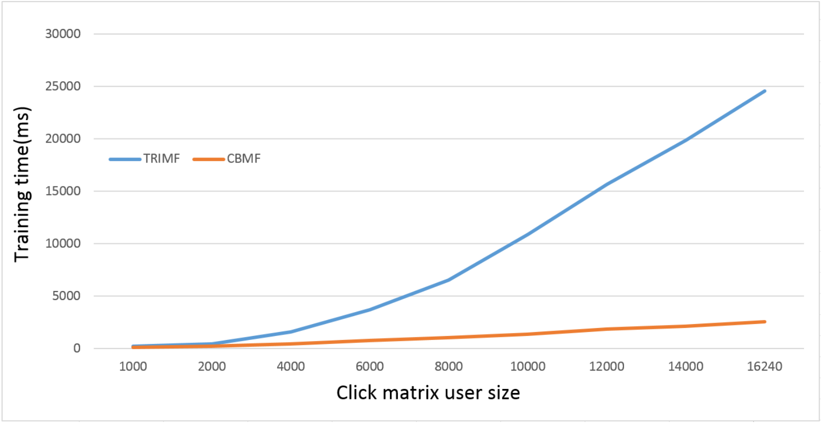


Figure : Convergence speed with regard to data size.

## Scalability

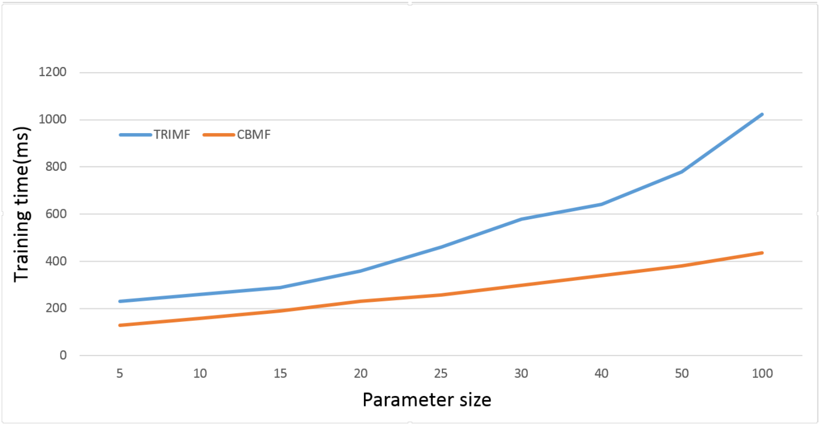


Figure : Convergence speed with regard to parameter size.

## Online experiments

## Online experiments metrics

An impression is a measure of the number of times an advertisement is seen.

- click-through-rate(CTR). The click-through rate is the number of times a click is made on the advertisement divided by the total impressions (the number of times an advertisement was served).

$$CTR = \frac{Clicks}{Impressions} * 100\%$$

- order amount per impression(OAPI).

$$OAPI = \frac{Order\ Prices}{Impressions}$$

- pay amount per impression(PAPI).

$$PAPI = \frac{Paid\ Money}{Impressions}$$

## Online experiment baselines

- **Co-clustering CF** (4313) is a scalable collaborative filtering framework based on co-clustering. Previous work showed that this method had a fast training time while providing decent accuracy.
- **Factorization Machine** (4314) is a generic approach that allows to mimic most factorization models by feature engineering. It provides high accuracy in several important prediction problems including recommender systems.
- **Item-based CF** (4315) is a typical recommending method proposed by Amazon. Cosine distance is used in calculating item similarities.
- **Efficient top-n recommendation** (cpsf) is a recommendation pipeline, which is the winner of the Million Songs Dataset (MSD) challenge.
- **CBMF**(4312): Our method.

## Online experiment results

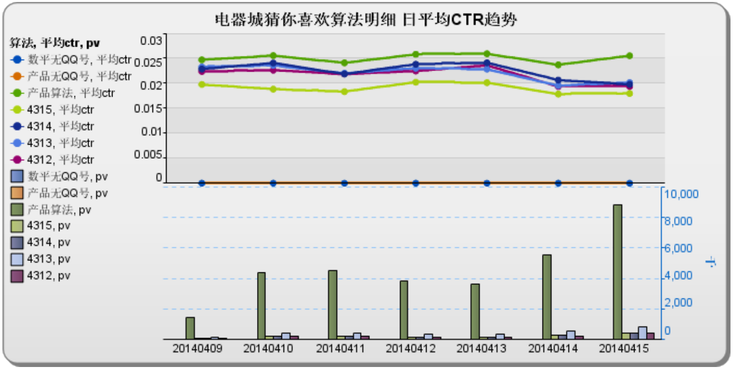


Figure : CTR of online algorithms from 0409 to 0415.

# Online experiment results

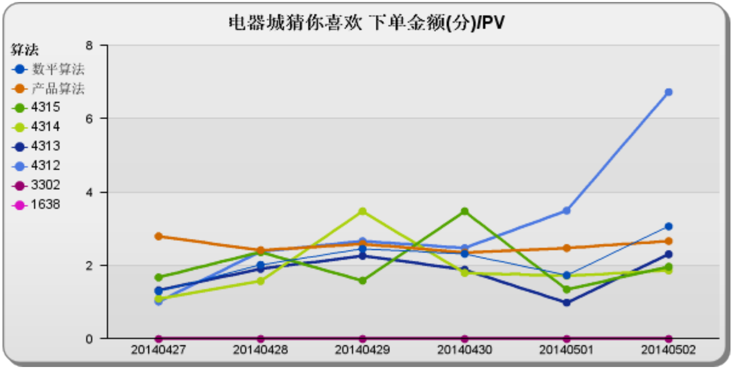


Figure : OAPI of online algorithms from 0427 to 0502.

# Online experiment results

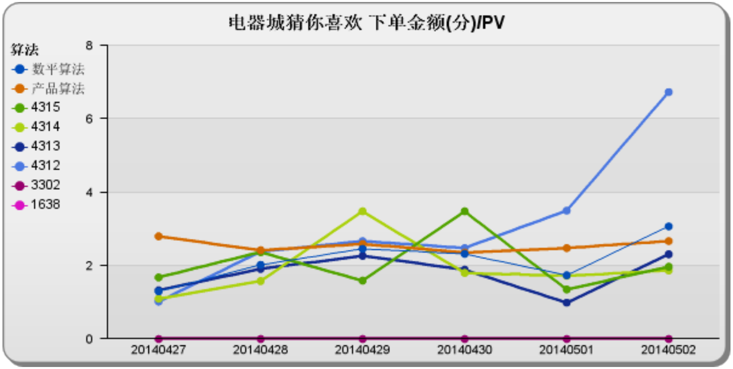


Figure : OAPI of online algorithms from 0427 to 0502.

## Online experiment results

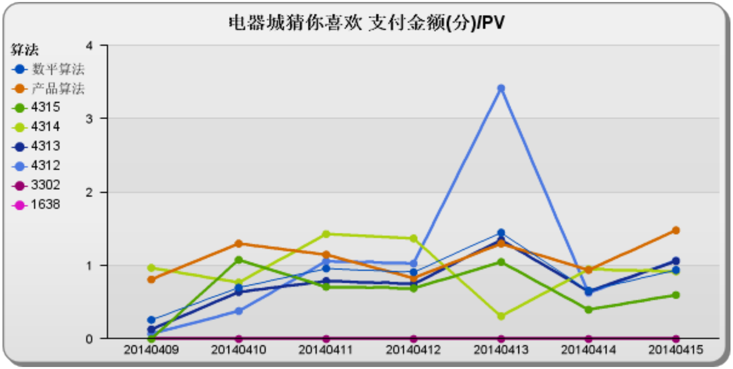


Figure : PAPI of online algorithms from 0409 to 0415.

## Outline

## Conclusion

- Transfer learning for one-class CF.
    - Matrix tri-factorization to share more data.
    - Clustering-pattern transfer function.
- Clustering based matrix factorization framework.

## Future work

- Pair-wise Transfer Learning in CF.
- Online Transfer Learning in CF.
- Transfer Learning in CF with multiple matrix.
- Time Complexity Optimization in CF.