Group 4
Implicit Feedback

- Papers:
  - Paper 12: Collaborating filtering for implicit Feedback Datasets
  - Paper 10: Fast ALS-based Matrix Factorization for Explicit and Implicit Feedback Datasets
- Presented by:
  - Mumtaz Hussain (271375)
  - Mofassir-ul-Islam Arif (271298)

Collaborating Filtering for Implicit Feedback Datasets

Yifan Hu, Yehuda Koren, Chris Volinsky

Mumtaz Hussain
271375

Instructor: Carlotta Schatten
Information Systems and Machine Learning Lab (ISMLL)
University of Hildesheim, Germany

November 29, 2016

outline

Introduction

- To improve customer experience through personalized recommendations by tracking user behavior in e-commerce
- No direct information from the user regarding their preferences
- Provide users personalized recommendations for products and services
- Profiling users and products to relate them
- Recommender system is based on different strategies

Basic Approaches of Recommender System

- Content Based Approach
  - Profile for each user or product to characterize its nature
  - Profiles might include demographic information or answers to a suitable questionnaire
  - Resulting profile allow programs to associate users with matching products

    $*$ It require gathering external information that might not available or not easy to collect
- Collaborative Filtering (CF)
  - Relies only on past user behavior without explicit profiles
  - Analyses relationship between users and interdependence among the products to find new user-item association
  - It is Domain Free yet can address the expects of data

    $*$ CF suffers the cold start problem

Unique Characteristics of Feedback

- Explicit Feedback
- Implicit Feedback
  - Purchase History, browsing history, search patterns or mouse movements
  - Analyzing watching habits of anonymized users
- No negative Feedback
- Implicit feedback is inherently noisy
- Preferences and Confidence
- Evaluation of implicit-feedback

Preliminaries

- In explicit feedback datasets, values will be the ratings that indicates the preferences
- In implicit feedback datasets, values would indicate observations
- Explicit ratings are typically unknown so algorithms works is needed

Neighborhood Models

- User-oriented method to estimate unknown ratings based on recorded ratings of like minded people
- Analogous item-oriented approach to estimate ratings using known ratings of same users on similar items
- Predicted value of r_{ui} is taken as weighted average of the ratings for neighboring items:

$$\widehat{r}_{ui} = \frac{\sum_{j \epsilon S^k(i;u)} s_{ij} r_{uj}}{\sum_{j \epsilon S^k(i;u)} s_{ij}}$$

Latent Factor Model

- To uncover latent features that explain observed ratings
- SVD Model have gained popularity due to their attractive accuracy and scalability

$$\min_{x_*, y_*} \sum_{r_{u,i} isknown} r_{ui} \left(r_{ui} - x_u^T y_i \right)^2 + \lambda \left(\| x_u \|^2 + \| y_i \|^2 \right)$$

λ is used for regularizing the model

Our Model

- Formalize the notion of confidence which r_{ui} variable measures
- Introduce set of binary variables p_{ui} which indicate the preferences of user u to item i

$$p_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \end{cases}$$

- Beliefs are associated with varying confidence levels (high or low)
- In case $p_{ui} = 0$ there can be many reasons beyond not liking it i.e. unaware of the existence or limited availability

Our Model

$$c_{ui} = 1 + \alpha r_{ui}$$

rate of increase is controlled by constant α

$$\min_{x_*, y_*} \sum_{u,i} c_{ui} \left(p_{ui} - x_u^T y_i \right)^2 + \lambda \left(\sum_u \parallel x_u \parallel^2 + \sum_i \parallel y_i \parallel^2 \right)$$

$$x_u = \left(Y^T C^u Y + \lambda I \right)^{-1} Y^T C^u p(u)$$

$$y_i = \left(X^T C^i X + \lambda I \right)^{-1} X^T C^i p(i)$$

$$c_{ui} = 1 + \alpha \log(1 + r_{ui/\epsilon})$$

Explaining Recommendations

- Explanation/Description of reason to recommend a specific product to a user to improve trust

$$x_u = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u p(u)$$

$$y_i^T (Y^T C^u Y + \lambda I)^{-1} Y^T C^u p(u)$$

- Least square model enables a novel way to compute explanations

$$\widehat{p_{ui}} = \sum_{j:r_{uj}>0} s_{ij}^u c_{uj}$$

- It reduces latent factor model into a liner model to predict preferences as a linear function of past actions

Data Description

- Data collected from Digital Television service on about 300,000 set top boxes
- Data collected with appropriate use end agreements and privacy policies
- Approximately 17,000 unique programs aired during in four week period-short period deteriorate results and long not add much value
- Training data contains r_{ui} real values for each user u and program i
- Toggle to zero all entries$r_{ui}^{t} > 0.5$ and log scaling scheme $\epsilon = 10^{-8}$

Evaluation Methodology

- Ordered list of shows sorted from the one predicted to be most preferred till least preferred
- recall oriented measures

$$\overline{rank} = \frac{\sum_{u,i} r^t_{ui} \mathrm{rank}_u i}{\sum_{u,i} r^t_{ui}}$$

- $rank_u i = 0$ percent means most desireable for user and $rank_u i = 100$ percent means least desireable

Evaluation Results

- Different number of factors (f) ranging from 10 to 200
- First Model: sorting all shows based on their popularity
- Second Model: Neighborhood based (item-item)
  - All as neighbors - not only a set of most popular ones
  - Cosine similarity for measuring item-item similarity

Comparison of Factor Model with PR and NM

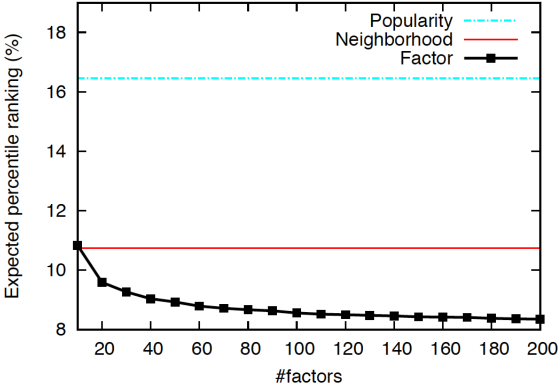


Figure: 1 Comparing factor model with popularity ranking and neighborhood model

cdf Probability

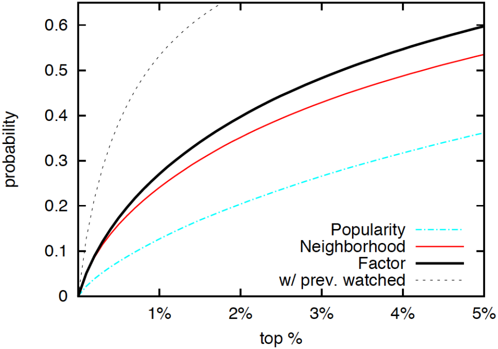


Figure: 2 Cumulative distribution function of the probability that a show watched in the test set falls within top x percentage of recommended shows

Data Description

- raw obsevations to distinct preference-confidence pairs
- First: Consider model work directly to given observations

$$\min_{x_*, y_*} \sum_{u,i} \left(p_{ui} - x_u^T y_i\right)^2 + \lambda_1 \left(\sum_u \| x_u \|^2 + \sum_i \| y_i \|^2 \right)$$

- Second: factorizing Deprived binary preferences values

$$\min_{x_*, y_*} \sum_{u,i} \left(p_{ui} - x_u^T y_i\right)^2 + \lambda_2 \left(\sum_u \| x_u \|^2 + \sum_i \| y_i \|^2 \right)$$

Analyzing Preference of Factor Model

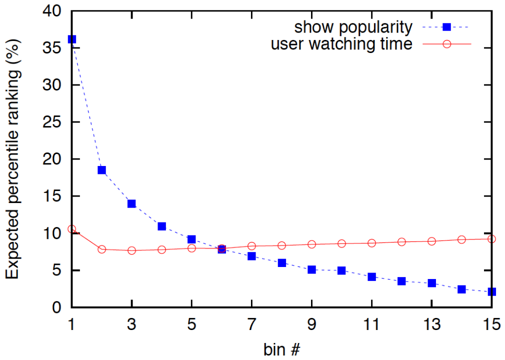


Figure: 3 Analyzing the performance of the factor model by segregating users shows based on different criteria

Conclusion

- Collaborative filtering on datasets with implicit feedback
- Main findings is that implicit user observations should be transformed into two pair magnitudes
  - Preferences (like/dislike)
  - Confidence Levels
- Latent factor algorithm that directly addresses the preference-confidence paradigm

Conclusion

So You Think You Can Dance	**Spider-Man**	**Life In The E.R.**
Hell's Kitchen Access Hollywood Judge Judy Moment of Truth Don't Forget the Lyrics	Batman: The Series Superman: The Series Pinky and The Brain Power Rangers The Legend of Tarzan	Adoption Stories Deliver Me Baby Diaries I Lost It! Bringing Home Baby
Total Rec = 36%	Total Rec = 40%	Total Rec = 35%

Table: 1 Three recommendations with explanations for a single user in our study. Each recommended show is recommended due to a unique set of already-watched shows by this user

Future Work/Recommendations

- Balance between unique properties of implicit feedback datasets and computational scalability
- Exploring modifications with a potential to improve accuracy at the expense of increasing computational complexity
- More careful analysis would split those zero values into different confidence level based on the availability of the item.
- Adding a dynamic time variable will lead to another possible extension of the model

References

Hu, Yifan and Koren, Yehuda and Volinsky, Chris, *Collaborative filtering for implicit feedback datasets*, 2008 Eighth IEEE International Conference on Data Mining, 2008, pp. 263–272.

Fast ALS-based Matrix Factorization for Explicit and Implicit Feedback Datasets

Presented By
Mofassir ul Islam Arif 271298
Instructor: Carlotta Schatten
Information Systems and Machine Learning Lab (ISMLL)
University of Hildesheim, Germany
November 29, 2016

Outline

- -Motivation
- -Hypothesis
- -Background: Explicit/Implicit issues
- -Matrix Factorization: Ridge Regression using ALS
- -Alternating Least Squares, Problems and Solutions
- -Toy Examples
- -Results
- -Comparison.

Motivation

- Explicit and Implicit feedback systems are a cornerstone of Recommender Systems enabling content to be delivered to user to boost profits. Changing nature of items, user preferences, style etc. make it necessary to keep the recommendations up-to-date.
- This means our models have to be retrained regularly. The accuracy of the models are dependent on the number of features we train the model with and that is where we see room for improvement.

Hypothesis

- Alternating least squares is a powerful tool for Matrix Factorization but the training time is proportional to K^3 . Where K is the number of latent factors. The paper presents as fast ALS variant with comparable accuracy to the original ALS method.

Explicit/Implicit Issues

- Explicit Feedback
  - A small subset of data is rated but over a finer scale
- Implicit Feedback
  - each user rates each item either positively (viewed) or negatively(did not view).
- Sparsity of the explicit rating matrix is usually less than 1%, the difference in the size of the data is usually several orders of magnitude.

- Time vs Accuracy Trade-Off.
  - Higher the time, Higher the accuracy.
    - Models are useful only if fast enough to keep up

Matrix Factorization 1/2

R	Item 1	Item 2	Item 3	.....	Item M
User 1					
User 2					
User 3		$\hat{r}_{ui}$			
...					
User N					

Q^T					
		q_i			

P	
p_u^T	

R_{NxM}: rating matrix

P_{NxK}: user feature matrix

Q_{MxK}: item feature matrix

N: #users

M: #items

K: #features

$K \ll M \, , K \ll N$

$$R = PQ^T \qquad \widehat{r_{ui}} = p_u^T . q_i$$

Matrix Factorization 2/2

$$(P^*, Q^*) = \underset{P,Q}{argmin} \sum_{u,i \in R} ({e_{ui}}^2 + \lambda. p_u^T . p_u + \lambda. q_i^T . q_i)$$

$$\lambda\, trades\ off\ between\ training\ error\ and\ small\ model\ wieght$$

- Minimize the error

$$e_{ui} = r_{ui} - \widehat{r_{ui}}$$

- Gradient descent or Alternating Least Squares

Ridge Regression

- Bell and Koren suggested Ridge Regression for CF
- Ridge Regression minimizes the cost function

$$\lambda \mathbf{w}^{\mathrm{T}}\mathbf{w} + \sum_{i=1}^{n} (\mathbf{w}^{\mathrm{T}}\mathbf{x}_i - y_i)^2,$$

- Cost function can be minimized with

$$\mathbf{w} = (\lambda \mathbf{I} + \mathbf{A})^{-1}\,\mathbf{d},$$

- Here

$$\mathbf{A} = \mathbf{X}^{\mathrm{T}}\mathbf{X}, \qquad \mathbf{d} = \mathbf{X}^{\mathrm{T}}\mathbf{y}.$$

- What could be the problem here?

Problem

- The cost

$$\mathbf{w} = (\lambda \mathbf{I} + \mathbf{A})^{-1}\,\mathbf{d},$$

- is dependent on computing

$$\mathbf{A} = \mathbf{X}^{\mathrm{T}}\mathbf{X}, \qquad \mathbf{d} = \mathbf{X}^{\mathrm{T}}\mathbf{y}.$$

- The calculation of A has a complexity of $O(K^2)$ and the matrix inversion is $O(K^3)$

Vanilla ALS (explicit feedback)

- We have already seen ALS a few times now. ALS alternates between two steps:
  - the P-step fixes Q and recomputes P.
  - the Q-step fixes P and recomputes Q.
- The recomputation of P is performed by solving a separate RR problem for each user.

$$\mathbf{A}_u = \mathbf{Q}[u]^{\mathrm{T}}\mathbf{Q}[u] = \sum_{i:(u,i)\in\mathcal{R}} \mathbf{q}_i\mathbf{q}_i^{\mathrm{T}},$$

A_u *is the covarience of inputs*

$$\mathbf{d}_u = \mathbf{Q}[u]^{\mathrm{T}}\mathbf{r}_u = \sum_{i:(u,i)\in\mathcal{R}} r_{ui}\cdot\mathbf{q}_i$$

d_u *is the covarience of inputs and outputs*

- p_u is recomputed as

$$\mathbf{p}_u = \left(\lambda n_u\mathbf{I} + \mathbf{A}_u\right)^{-1}\mathbf{d}_u.$$

Problem?

- In the P-step a ridge regression is solved for each user, which is

$$O(\sum_{u=1}^{N}(K^2 n_u + K^3)) \longrightarrow O(K^2|\mathcal{R}| + NK^3)$$

- Similarly, the Q-step requires

$$O(K^2|\mathcal{R}| + MK^3)$$

Vanilla ALS (Implicit feedback)

- Assignment of a confidence level to each pair of R $(R = N.M)$

$$(P^*, Q^*) = \underset{P,Q}{argmin} \sum_{u,i \in R} (c_{ui}.\, e_{ui}{}^2 + \lambda.\, p_u^T.\, p_u + \lambda.\, q_i^T.\, q_i)$$

- The authors use one restriction, namely if u has not watched i, then $r_{ui} = r_0 = 0$ and $c_{ui} = c_0 = 1$, where r_0 and c_0 are predefined constants, typically set to $r_0 = 0$ and $c_0 = 1$.
- A virtual user is imagined who hasn't viewed anything

Cont'd

- A_0 and d_0 are computed for the virtual user.

$$\mathbf{A}_0 = \sum_i c_0 \cdot \mathbf{q}_i \mathbf{q}_i^{\mathrm{T}}, \quad \mathbf{d}_0 = \sum_i c_0 \cdot r_0 \cdot \mathbf{q}_i.$$

- So our equations get updated as follows

$$\mathbf{A}_u = \mathbf{A}_0 + \sum_{i:\ (u,i) \in \mathcal{R}^{\pm}} (-c_0 + c_{ui}) \cdot \mathbf{q}_i^{\mathrm{T}} \mathbf{q}_i,$$

$$\mathbf{d}_u = \mathbf{d}_0 + \sum_{i:\ (u,i) \in \mathcal{R}^{\pm}} (-c_0 \cdot r_0 + c_{ui} \cdot r_{ui}) \cdot \mathbf{q}_i.$$

- Problem here is that Computationally it costs the same.

Solution: Explicit Feedback

- Reduce Complexity for

$$\mathbf{p}_u = (\lambda n_u \mathbf{I} + \mathbf{A}_u)^{-1} \mathbf{d}_u.$$

- Originally we were using Ridge Regression to recompute p_u every time while completely ignoring the results of the previous recomputation
- The proposition of the paper is to:
  - Update one parameter at a time, keeping the rest constant using one entry from the matrix X

$$\lambda \mathbf{w}^{\mathrm{T}} \mathbf{w} + \sum_{i=1}^{n} (\mathbf{w}^{\mathrm{T}} \mathbf{x}_i - y_i)^2, \qquad \forall_{i=1}^{n} \qquad w_k x_{ik} \approx y_i - \sum_{l \neq k} w_l x_{il},$$

  - Before updating the next value, we set it to zero and proceed to optimize it.

Pseudo Code

```
Data: P,Q
Result: P and Q step fixes
initialize P and Q
P ←User feature matrix
Q ←Item feature matrix
while until termination condition do
    /* P step optimization
    for users do
        run RR1 on pu for one cycle
    end
    /* Q step optimization
    for item do
        run RR1 on qi for one cycle
    end
end
```

Algorithm 1: ALS1

Pseudo Code

Input: n: number of examples,
K: number of features, λ: regularization factor,
$\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^{K \times 1}$: training examples,
$y_1, \ldots, y_n$: target variables, $c_1, \ldots, c_n$: weight of examples,
L: number of cycles, $\mathbf{w}$: initial weight vector, or $\mathbf{0}$.
Output: $\mathbf{w}$: the optimized weight vector

$\forall_{i=1}^{n} : e_i \leftarrow y_i - \mathbf{w}^{\mathrm{T}} \mathbf{x}_i$
for L *times* **do**
    one cycle:
    for $k \leftarrow 1$ **to** K **do**
        $\forall_{i=1}^{n} : e_i \leftarrow e_i + w_k x_k$
        $w_k \leftarrow 0$
        $a \leftarrow \sum_{i=1}^{n} c_i x_{ik} x_{ik}$
        $d \leftarrow \sum_{i=1}^{n} c_i x_{ik} e_i$
        $w_k \leftarrow d/(\lambda + a)$.
        $\forall_{i=1}^{n} : e_i \leftarrow e_i - w_k x_k$
    end
end

Pseudo Code

Input: n: number of examples,
K: number of features, λ: regularization factor,
$\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^{K \times 1}$: training examples,
$y_1, \ldots, y_n$: target variables, $c_1, \ldots, c_n$: weight of examples,
L: number of cycles, $\mathbf{w}$: initial weight vector, or $\mathbf{0}$.
Output: $\mathbf{w}$: the optimized weight vector

$\forall_{i=1}^{n} : e_i \leftarrow y_i - \mathbf{w}^{\mathrm{T}} \mathbf{x}_i$
for L *times* **do**
  one cycle:
  for $k \leftarrow 1$ **to** K **do**
    $\forall_{i=1}^{n} : e_i \leftarrow e_i + w_k x_k$
    $w_k \leftarrow 0$
    $a \leftarrow \sum_{i=1}^{n} c_i x_{ik} x_{ik}$
    $d \leftarrow \sum_{i=1}^{n} c_i x_{ik} e_i$
    $w_k \leftarrow d / (\lambda + a)$.
    $\forall_{i=1}^{n} : e_i \leftarrow e_i - w_k x_k$
  end
end

- How is this better?
- Univariate ridge regression as only one vector from the original user feature matrix gets used.
- Computationally $O(Kn)$

RR1 in action

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 12.2 & 18.3 & 13.3 & 8.6 & 13.6 \\ 18.3 & 37.3 & 25.6 & 18.9 & 23.9 \\ 13.3 & 25.6 & 29.4 & 22.6 & 23.8 \\ 8.6 & 18.9 & 22.6 & 21.0 & 17.3 \\ 13.6 & 23.9 & 23.8 & 17.3 & 25.3 \end{bmatrix}$$

$$\mathbf{X}^T\mathbf{y} = \begin{bmatrix} 14.4 \\ 28.7 \\ 24.4 \\ 19.1 \\ 22.6 \end{bmatrix}$$

Cost Matrix Inversion

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{y})$$

RR1 in action

- Optimize one feature at a time:
- Sum of squared errors: 24.6

$$\begin{bmatrix} 0.8 \\ 1.9 \\ 1.6 \\ 2.3 \\ 2.1 \\ 1.2 \\ 1.9 \\ 1.9 \end{bmatrix} \approx \begin{bmatrix} 1.3 & 1.2 & 0.1 & 0.6 & 0.1 \\ 1.7 & 2.9 & 0.4 & 0.3 & 1.8 \\ 1.7 & 2.7 & 2.0 & 0.2 & 1.1 \\ 1.3 & 1.7 & 2.5 & 2.0 & 2.9 \\ 0.1 & 2.9 & 1.3 & 1.3 & 1.3 \\ 0.0 & 0.2 & 2.2 & 2.1 & 1.8 \\ 0.7 & 2.2 & 2.5 & 2.9 & 0.5 \\ 1.6 & 2.0 & 2.5 & 1.4 & 2.7 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{0.0} \\ \mathbf{0.0} \\ \mathbf{0.0} \\ \mathbf{0.0} \\ \mathbf{0.0} \end{bmatrix}$$

RR1 in action

- Optimize one feature at a time:
- Sum of squared errors: 7.5

$$\begin{bmatrix} 0.8 \\ 1.9 \\ 1.6 \\ 2.3 \\ 2.1 \\ 1.2 \\ 1.9 \\ 1.9 \end{bmatrix} \approx \begin{bmatrix} 1.3 & 1.2 & 0.1 & 0.6 & 0.1 \\ 1.7 & 2.9 & 0.4 & 0.3 & 1.8 \\ 1.7 & 2.7 & 2.0 & 0.2 & 1.1 \\ 1.3 & 1.7 & 2.5 & 2.0 & 2.9 \\ 0.1 & 2.9 & 1.3 & 1.3 & 1.3 \\ 0.0 & 0.2 & 2.2 & 2.1 & 1.8 \\ 0.7 & 2.2 & 2.5 & 2.9 & 0.5 \\ 1.6 & 2.0 & 2.5 & 1.4 & 2.7 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{1.2} \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{bmatrix}$$

RR1 in action

- Optimize one feature at a time:
- Sum of squared errors: 6.2

$$\begin{bmatrix} 0.8 \\ 1.9 \\ 1.6 \\ 2.3 \\ 2.1 \\ 1.2 \\ 1.9 \\ 1.9 \end{bmatrix} \approx \begin{bmatrix} 1.3 & 1.2 & 0.1 & 0.6 & 0.1 \\ 1.7 & 2.9 & 0.4 & 0.3 & 1.8 \\ 1.7 & 2.7 & 2.0 & 0.2 & 1.1 \\ 1.3 & 1.7 & 2.5 & 2.0 & 2.9 \\ 0.1 & 2.9 & 1.3 & 1.3 & 1.3 \\ 0.0 & 0.2 & 2.2 & 2.1 & 1.8 \\ 0.7 & 2.2 & 2.5 & 2.9 & 0.5 \\ 1.6 & 2.0 & 2.5 & 1.4 & 2.7 \end{bmatrix} \cdot \begin{bmatrix} 1.2 \\ \mathbf{0.2} \\ 0.0 \\ 0.0 \\ 0.0 \end{bmatrix}$$

RR1 in action

- Optimize one feature at a time:
- Sum of squared errors: 5.7

$$\begin{bmatrix} 0.8 \\ 1.9 \\ 1.6 \\ 2.3 \\ 2.1 \\ 1.2 \\ 1.9 \\ 1.9 \end{bmatrix} \approx \begin{bmatrix} 1.3 & 1.2 & 0.1 & 0.6 & 0.1 \\ 1.7 & 2.9 & 0.4 & 0.3 & 1.8 \\ 1.7 & 2.7 & 2.0 & 0.2 & 1.1 \\ 1.3 & 1.7 & 2.5 & 2.0 & 2.9 \\ 0.1 & 2.9 & 1.3 & 1.3 & 1.3 \\ 0.0 & 0.2 & 2.2 & 2.1 & 1.8 \\ 0.7 & 2.2 & 2.5 & 2.9 & 0.5 \\ 1.6 & 2.0 & 2.5 & 1.4 & 2.7 \end{bmatrix} \cdot \begin{bmatrix} 1.2 \\ 0.2 \\ \mathbf{0.1} \\ 0.0 \\ 0.0 \end{bmatrix}$$

RR1 in action

- Optimize one feature at a time:
- Sum of squared errors: 5.4

$$\begin{bmatrix} 0.8 \\ 1.9 \\ 1.6 \\ 2.3 \\ 2.1 \\ 1.2 \\ 1.9 \\ 1.9 \end{bmatrix} \approx \begin{bmatrix} 1.3 & 1.2 & 0.1 & 0.6 & 0.1 \\ 1.7 & 2.9 & 0.4 & 0.3 & 1.8 \\ 1.7 & 2.7 & 2.0 & 0.2 & 1.1 \\ 1.3 & 1.7 & 2.5 & 2.0 & 2.9 \\ 0.1 & 2.9 & 1.3 & 1.3 & 1.3 \\ 0.0 & 0.2 & 2.2 & 2.1 & 1.8 \\ 0.7 & 2.2 & 2.5 & 2.9 & 0.5 \\ 1.6 & 2.0 & 2.5 & 1.4 & 2.7 \end{bmatrix} \cdot \begin{bmatrix} 1.2 \\ 0.2 \\ 0.1 \\ \mathbf{0.1} \\ 0.0 \end{bmatrix}$$

RR1 in action

- Optimize one feature at a time:
- Sum of squared errors: 5.0

$$\begin{bmatrix} 0.8 \\ 1.9 \\ 1.6 \\ 2.3 \\ 2.1 \\ 1.2 \\ 1.9 \\ 1.9 \end{bmatrix} \approx \begin{bmatrix} 1.3 & 1.2 & 0.1 & 0.6 & 0.1 \\ 1.7 & 2.9 & 0.4 & 0.3 & 1.8 \\ 1.7 & 2.7 & 2.0 & 0.2 & 1.1 \\ 1.3 & 1.7 & 2.5 & 2.0 & 2.9 \\ 0.1 & 2.9 & 1.3 & 1.3 & 1.3 \\ 0.0 & 0.2 & 2.2 & 2.1 & 1.8 \\ 0.7 & 2.2 & 2.5 & 2.9 & 0.5 \\ 1.6 & 2.0 & 2.5 & 1.4 & 2.7 \end{bmatrix} \cdot \begin{bmatrix} 1.2 \\ 0.2 \\ 0.1 \\ 0.1 \\ -\mathbf{0.1} \end{bmatrix}$$

RR1 in action

- Optimize one feature at a time:
- Sum of squared errors: 3.4

$$
\begin{bmatrix} 0.8 \\ 1.9 \\ 1.6 \\ 2.3 \\ 2.1 \\ 1.2 \\ 1.9 \\ 1.9 \end{bmatrix}
\approx
\begin{bmatrix}
1.3 & 1.2 & 0.1 & 0.6 & 0.1 \\
1.7 & 2.9 & 0.4 & 0.3 & 1.8 \\
1.7 & 2.7 & 2.0 & 0.2 & 1.1 \\
1.3 & 1.7 & 2.5 & 2.0 & 2.9 \\
0.1 & 2.9 & 1.3 & 1.3 & 1.3 \\
0.0 & 0.2 & 2.2 & 2.1 & 1.8 \\
0.7 & 2.2 & 2.5 & 2.9 & 0.5 \\
1.6 & 2.0 & 2.5 & 1.4 & 2.7
\end{bmatrix}
\cdot
\begin{bmatrix} \mathbf{0.8} \\ 0.2 \\ 0.1 \\ 0.1 \\ -0.1 \end{bmatrix}
$$

RR1 in action

- Optimize one feature at a time:
- Sum of squared errors: 0.055

$$
\begin{bmatrix} 0.8 \\ 1.9 \\ 1.6 \\ 2.3 \\ 2.1 \\ 1.2 \\ 1.9 \\ 1.9 \end{bmatrix} \approx \begin{bmatrix} 1.3 & 1.2 & 0.1 & 0.6 & 0.1 \\ 1.7 & 2.9 & 0.4 & 0.3 & 1.8 \\ 1.7 & 2.7 & 2.0 & 0.2 & 1.1 \\ 1.3 & 1.7 & 2.5 & 2.0 & 2.9 \\ 0.1 & 2.9 & 1.3 & 1.3 & 1.3 \\ 0.0 & 0.2 & 2.2 & 2.1 & 1.8 \\ 0.7 & 2.2 & 2.5 & 2.9 & 0.5 \\ 1.6 & 2.0 & 2.5 & 1.4 & 2.7 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{0.0} \\ \mathbf{0.4} \\ -\mathbf{0.01} \\ \mathbf{0.3} \\ \mathbf{0.3} \end{bmatrix}
$$

Solution: Implicit Feedback

- Use synthetic examples using the Eigenvalue decomposition of A_0

$\mathbf{A}_0 = \mathbf{S}\boldsymbol{\Lambda}\mathbf{S}^{\mathrm{T}}$	$\mathbf{S} \in \mathbb{R}^{K\times K}$	Orthogonal Matrix of eigenvectors $S^T.S = S.S^T = I$
	$\boldsymbol{\Lambda} \in \mathbb{R}^{K\times K}$	Diagonal Matrix, non negative eigenvectors
	$\mathbf{G}^{\mathrm{T}} = \mathbf{S}\sqrt{\boldsymbol{\Lambda}}.$	Feature matrix of synthetic examples
	$\mathbf{g}_j \in \mathbb{R}^{K\times 1}$	Feature vector of synthetic examples

$$\mathbf{A}_0' := \sum_{j=1}^{K} c_j \mathbf{g}_j \mathbf{g}_j^{\mathrm{T}} = \mathbf{G}^{\mathrm{T}}\mathbf{G} \qquad \mathbf{d}_0' := \sum_{j=1}^{K} c_j r_j \mathbf{g}_j = \mathbf{G}^{\mathrm{T}}\mathbf{r}.$$

- if a user rates the g_j examples with confidence level $c_j = 1$, the resulting covariance matrix A_0' will be equal to A_0

Pseudo Code

Data: P
Result: P
initialize
P $\leftarrow$ $User\ feature\ matrix$
rated g_j $as\ r_j\ with\ c_j = 1$
i:(u,i) $\in$ $R\ user\ rated\ q_i\ as\ r_0\ with\ -c_0$
i:(u,i) $\in$ R $user\ rated\ q_i\ as\ r_{ui}\ with\ c_{ui}$
while *until termination condition* **do**

```
/* P step optimization
```

  apply RR1
end

Algorithm 2: IALS1

IALS1

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 12.2 & 18.3 & 13.3 & 8.6 & 13.6 \\ 18.3 & 37.3 & 25.6 & 18.9 & 23.9 \\ 13.3 & 25.6 & 29.4 & 22.6 & 23.8 \\ 8.6 & 18.9 & 22.6 & 21.0 & 17.3 \\ 13.6 & 23.9 & 23.8 & 17.3 & 25.3 \end{bmatrix}$$

$$\mathbf{X}^T\mathbf{X} = \mathbf{S}\boldsymbol{\Lambda}\mathbf{S}^T$$

$$\mathbf{Z} := \sqrt{\boldsymbol{\Lambda}}\mathbf{S}^T$$

$$\mathbf{Z} = \begin{bmatrix} 0.44 & -0.15 & -0.78 & 0.70 & 0.16 \\ 1.08 & -0.49 & 0.52 & -0.06 & -0.62 \\ -0.53 & 1.03 & 0.04 & 1.00 & -1.79 \\ 1.4 & 2.17 & -1.42 & -1.94 & -0.26 \\ 2.94 & 5.59 & 5.15 & 3.96 & 4.65 \end{bmatrix}$$

IALS1

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 12.2 & 18.3 & 13.3 & 8.6 & 13.6 \\ 18.3 & 37.3 & 25.6 & 18.9 & 23.9 \\ 13.3 & 25.6 & 29.4 & 22.6 & 23.8 \\ 8.6 & 18.9 & 22.6 & 21.0 & 17.3 \\ 13.6 & 23.9 & 23.8 & 17.3 & 25.3 \end{bmatrix}$$

$$O(K^3 + K^2N)$$

$$O(K^2 + KN)$$

$$\mathbf{X}^T\mathbf{X} = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^T$$

$$\mathbf{Z} := \sqrt{\mathbf{\Lambda}}\mathbf{S}^T$$

$$\mathbf{Z} = \begin{bmatrix} 0.44 & -0.15 & -0.78 & 0.70 & 0.16 \\ 1.08 & -0.49 & 0.52 & -0.06 & -0.62 \\ -0.53 & 1.03 & 0.04 & 1.00 & -1.79 \\ 1.4 & 2.17 & -1.42 & -1.94 & -0.26 \\ 2.94 & 5.59 & 5.15 & 3.96 & 4.65 \end{bmatrix}$$

Experiments

- Netflix Dataset

Probe Set
1408395
Probe10 (Test) set
140840
Train Set
Probe - Probe10(Test) set

- Repetitive Implicit Feedback Dataset

Users	items	Feedback
215630	73863	9,617,414
Total Days		
182		
Train Days		
181 (9,439,863 events)		
Test Day		
1 (55,711 events)		

Results

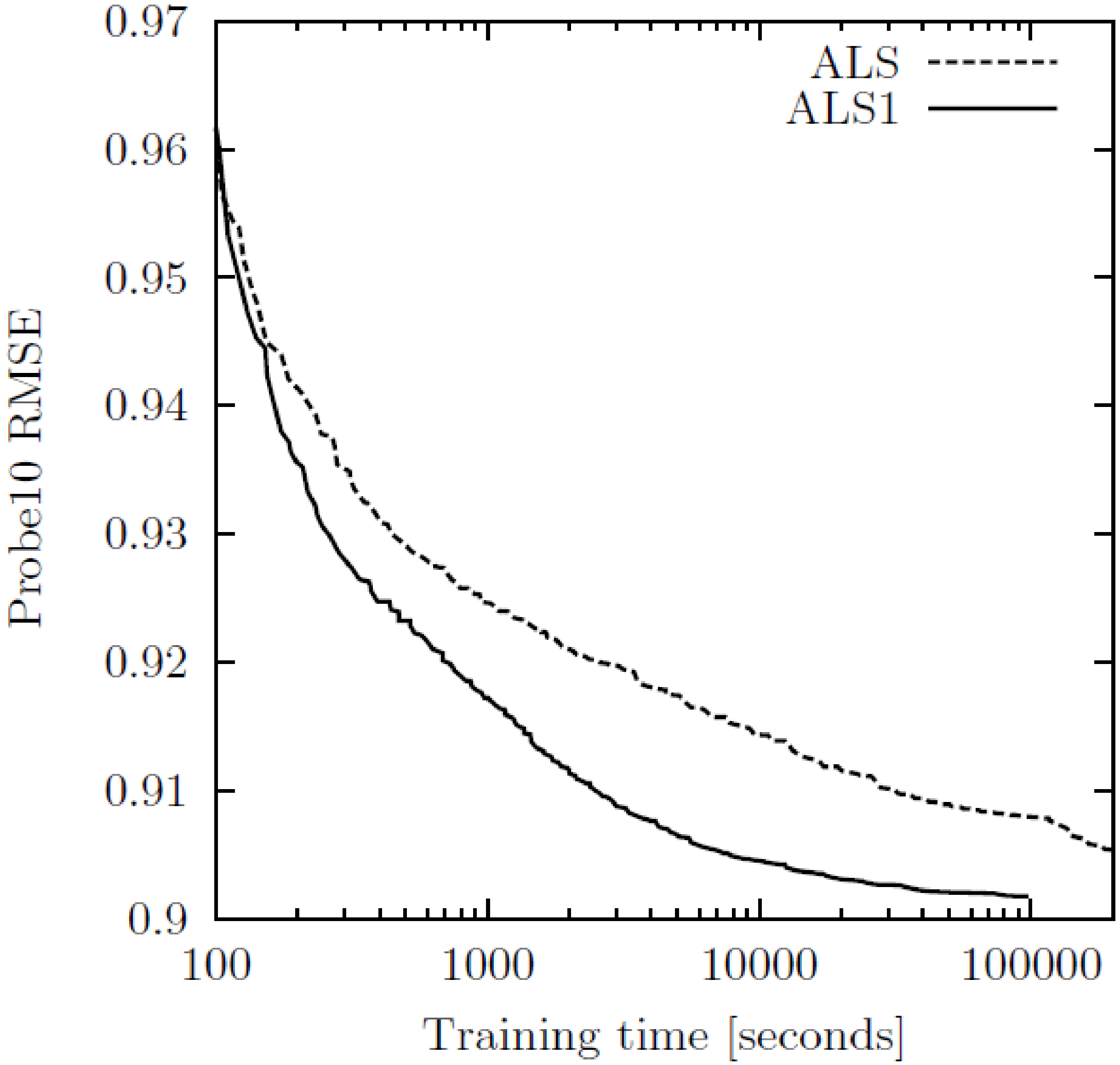
- Netflix Dataset 20 Epochs

K	ALS RMSE	ALS time	ALS1 RMSE	ALS1 time
5	0.9391	389	0.9394	305
10	0.9268	826	0.9281	437
20	0.9204	2288	0.9222	672
50	0.9146	10773	0.9154	1388
100	0.9091	45513	0.9098	2653
200	0.9050	228 981	0.9058	6308
500	0.9027	2 007 451	0.9032	22070
1000	N/A	N/A	0.9025	44345

- Netflix Dataset 25 Epochs

K	ALS RMSE	ALS time	ALS1 RMSE	ALS1 time
5	0.9386	980	0.9390	764
10	0.9259	2101	0.9262	1094
20	0.9192	5741	0.9196	1682
50	0.9130	27500	0.9134	3455
100	0.9078	115 827	0.9079	6622
200	0.9040	583 445	0.9041	15836
500	0.9022*	4 050 675*	0.9021	55054
1000	N/A	N/A	0.9018	110904

Results



ALS
ALS1
Probe10 RMSE
0.97
0.96
0.95
0.94
0.93
0.92
0.91
0.9
100
1000
10000
100000
Training time [seconds]

RIF

- Assume that the recommendable items are indexed by i ranging from 1 to *M*. denote whether an item is relevant to the user or not. Then the position for item i relevant to user u is defined as:

$$\mathrm{pos}_{ui} = |\{j: \quad r_{uj} = 0 \wedge \hat{r}_{uj} > \hat{r}_{ui}\}|$$

- Now we can say that the Relative position will be:

$$\mathrm{rpos}_{ui} = \frac{\mathrm{pos}_{ui}}{|\{j: \quad r_{uj} = 0\}|}$$

- And finally:

$$\mathrm{ARP} = \frac{\sum_{(u,i)\in\mathcal{R}} \mathrm{rpos}_{ui}}{|\mathcal{R}|},$$

Results

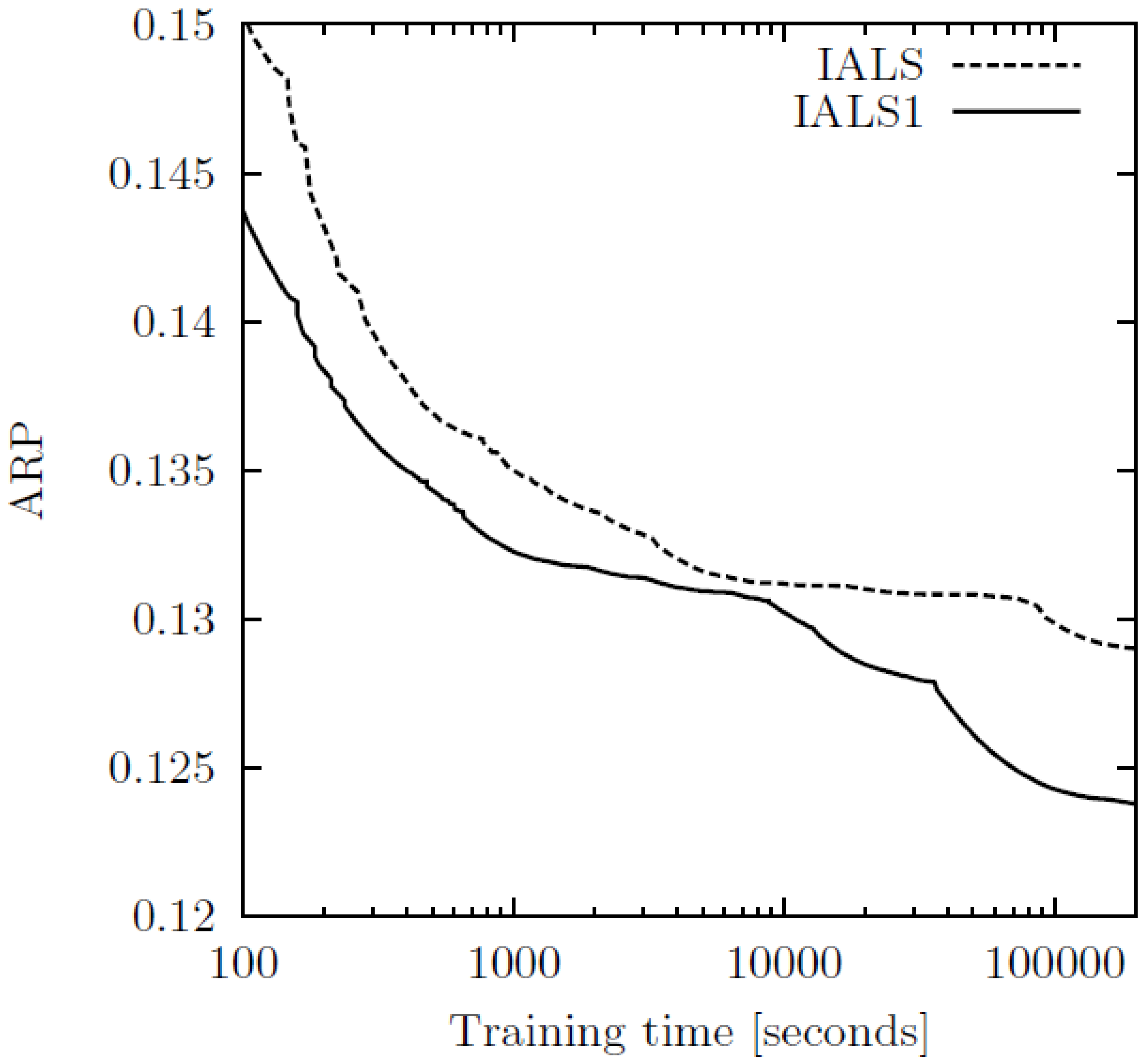
- RIF Dataset after 10 epochs

K	IALS		IALS1	
	ARP	time	ARP	time
5	0.1903	76	0.1890	56
10	0.1584	127	0.1598	67
20	0.1429	322	0.1453	104
50	0.1342	1431	0.1366	262
100	0.1328	5720	0.1348	680
250	0.1316	46472	0.1329	3325
500	0.1282	244 088	0.1298	12348
1000	N/A	N/A	0.1259	52305

- RIF Dataset after 20 epochs

K	IALS		IALS1	
	ARP	time	ARP	time
5	0.1903	153	0.1898	112
10	0.1578	254	0.1588	134
20	0.1427	644	0.1432	209
50	0.1334	2862	0.1344	525
100	0.1314	11441	0.1325	1361
250	0.1313	92944	0.1311	6651
500	N/A	N/A	0.1282	24697
1000	N/A	N/A	0.1242	104 611

Results


IALS
IALS1
ARP
0.15
0.145
0.14
0.135
0.13
0.125
0.12
100
1000
10000
100000
Training time [seconds]

Future Work

- Moving to other domains, different from collaborative filtering
- The proposed ALS1 and IALS1 store only the diagonal of the covariance matrix. We may relax this restriction and store data also in the box-diagonal. This leads to multivariate regression problems but with small number of variables.
- At IALS1 gradient descent method can replace RR1, offering the same time complexity.

Conclusion

- Vanilla ALS is computationally complex, restricts the number of latent factors thus compromising accuracy
- By using Fast ALS techniques, the training time can be lowered
- The depreciation of accuracy can be supplemented by increasing the number of latent factors
- Vanilla ALS needs a Linux cluster of 30 nodes to run for $K = 1000$, the method proposed here can compute that on a single station.

Winning Method

- Apples and Bananas
- Each Method has its advantages
- ALS1 and IALS1 are much better at speeding up the model learning time therefore the second paper wins over the first.

References

- Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM-08, 8*th *IEEE Int. Conf. on Data Mining*, pages 263-272 Pisa, Italy, 2008.
- R. M. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *ICDM-07, 7*th *IEEE Intl. Conf. on Data Mining*, pages 43-52, Omaha, Nebraska, USA, 2007.
- G. Takacs, I. Pilaszy, B. Nemeth, and D. Tikk.Scalable collaborative filtering approaches for large recommender systems. *Journal of Machine Learning Research*, 10:623{656, 2009.
- **Collaborative Filtering Recommender Systems** By Michael D. Ekstrand, John T. Riedl and Joseph A. Konstan
- https://jessesw.com/Rec-System/

Thank you.

Questions?

42