Matrix Factorization Techniques for Recommender Systems

Patrick Seemann, December 16th, 2014

TECHNISCHE UNIVERSITÄT DARMSTADT

Topics

- Intro
- New-User / New-Item Problem
- Matrix Factorization
  - Kernel Matrix Factorization
- Learning Matrix Factorization Models
- Online-Updating RKMF Models for Large-Scale RS
- Evaluation

Classification

Recommender Systems

- **Collaborative Filtering**
  - **Latent factor models**
    - **Matrix Factorization**
  - Neighborhood methods
- Content based approach

Rating Prediction

- How might a user rate a particular item?
- Problem can be seen as **matrix completion task** of a ratings matrix R

Columns: **Items**

$$R: |U| \times |I| \qquad R = \begin{pmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,i} \\ \vdots & \vdots & \ddots & r_{1,i} \\ r_{u,1} & r_{u,2} & \cdots & r_{u,i} \end{pmatrix}$$

Rows: **Users**

Note: R is **sparse**

an entry of R is the rating of user u for item i

New-User / New-Item Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Matrix Factorization**
  - model is learned in batch mode
  - captures the state of a system at **particular time**, but **doesn't update** itself (computation expensive)
  - But in the real world: users generate **new feedback**

- **Fast adaption** of the model is **crucial** when content changes frequently, e.g. a news website

- **Definition: New User-Problem**
  - A users profile **grows** from 0 to k ratings
- New-Item Problem defined symmetrically

Rating prediction

(Kernel) Matrix Factorization

Matrix Factorization (MF)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Task: **approximate** the true unobserved ratings matrix R by

$$\hat{R}:\ |U|\times|I|,\qquad \hat{R}=W\cdot H^{T}$$

- Where **W** and **H** are two **feature matrices**:

$$W:\ |U|\times k \quad H:\ |I|\times k$$

- A row of W contains the k features that describe user u.
- Similarly, each row of H describes a particular item

Matrix Factorization (MF) Contd.

Users:
- Joe
- Bob
- Ryan
- Josh
- ...

Items:
- Lord of the Rings 1
- Mission Impossible 1
- Mr. Bean
- The rise and the rise of Bitcoin
- ...

Factors:
- Action
- Comedy
- Fantasy
- Documentary
- ...

$$W = \begin{pmatrix} 0 & 10 & 0 & 10 & \dots \\ 0 & \boxed{10} & 0 & 0 & \dots \\ 8 & 0 & 0 & 5 & \dots \\ 10 & 5 & 9 & 0 & \dots \end{pmatrix} \qquad H = \begin{pmatrix} 7 & 1 & 10 & 0 & \dots \\ 10 & 4 & 0 & 0 & \dots \\ 0 & 9 & 0 & 0 & \dots \\ 0 & 0 & 0 & 10 & \dots \end{pmatrix}$$

Bob only likes comedy

Matrix Factorization (MF) Contd.

$$\hat{R}:\ |U| \times |I|, \qquad \hat{R} = W \cdot H^T$$

- Entries are denoted as: $\hat{r}_{u,i}$
- They approximate how the user “u” rates the item “i”

$$\hat{r}_{u,i} = \langle w_u, h_i \rangle = \sum_{f=1}^{k} w_{u,f} \cdot h_{i,f}$$

- Often a bias term is added (e.g. the global average rating)
  - then only residuals have to be learned

$$\hat{r}_{u,i} = b_{u,i} + \sum_{f=1}^{k} w_{u,f} \cdot h_{i,f}$$

Kernel Matrix Factorization (KMF)

- Like MF
- calculations between the feature vector of a user and the feature vector of an item are **kernelized**

$$\hat{r}_{u,i} = a + c \cdot K(w_u, h_i)$$

Where K is a Kernel defined as: $K: \mathbb{R}^k \times \mathbb{R}^k \rightarrow \mathbb{R}$

- Examples:
  - **Linear**: $K_l(w_u, h_i) = \langle w_u, h_i \rangle$
  - **Polynomial**: $K_p(w_u, h_i) = (1 + \langle w_u, h_i \rangle)^d$
  - **Logistic**: $K_s(w_u, h_i) = \Phi_s(b_{u,i} + \langle w_u, h_i \rangle)$

$$with \quad \Phi_s(x) := \frac{1}{1+e^{-x}}$$

Rating prediction

Learning Matrix Factorization Models

Learning MF Models

TECHNISCHE UNIVERSITÄT DARMSTADT

- High number of missing values in R
- Use only observed values S of R
  - S contains triples **(u, i, v)** of feedback

- Optimizing with regard to Root-Mean-Square-Error (**RMSE**)

$$\underset{W,H}{argmin} \; E(S, W, H)$$

$$E(S, \hat{R}) := E(S, W, H) = \sum_{r_{u,i} \in S} (r_{u,i} - \hat{r}_{u,i})^2$$

Learning MF Models Contd.

- Instead of learning optimal fit for $W \cdot H^T$, add a regularization term

- **Regularization**
  - To avoid overfitting
  - Here: **Tikhonov regularization** (“ridge regression”)

$$\underset{W,H}{argmin}\ Opt(S, W, H)$$

$$Opt(S, W, H) := E(S, W, H) + \underbrace{\lambda \cdot \left(\|W\|_F^2 + \|H\|_F^2\right)}_{regularization\ term}$$

- **Early stopping** is also used to work against overfitting

Learning MF Models Contd.

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Optimizing by Stochastic Gradient Descent**

Algo. 1

procedure OPTIMIZE(S, W, H)
  initialize W, H
  repeat
    for $r_{u,i} \in S$ **do**
      for $f \leftarrow 1, \ldots, k$ **do**
        $w_{u,f} \leftarrow w_{u,f} - \alpha \frac{\partial}{\partial w_{u,f}} \mathrm{Opt}(\{r_{u,i}\}, W, H)$
        $h_{i,f} \leftarrow h_{i,f} - \alpha \frac{\partial}{\partial h_{i,f}} \mathrm{Opt}(\{r_{u,i}\}, W, H)$
      end for
    end for
  until Stopping criteria met
  return (W, H)
end procedure

e.g. a fixed number

[1]

New-user / new-item problem

Online Updating RKMF Models

Complexity of Training KMF models

- Training a KMF models is **expensive**:

$$O\left(|S| \cdot k \cdot i\right)$$

- Where “i” is num of early stopping iterations
- In case of Netflix: k = 40, i = 120, cardinality(S) = 100 million
  - Leads to **480 billion** feature updates

Complexity of Training KMF models

- The paper [3] proposes an algorithm with complexity

$$O\left(\left|C\left(u,\cdot\right)\right|\cdot k\cdot i\right)$$

where C(u,*) is the current profile of the user

- Retraining single user requires **only 192k** updates (worst case)

Online updating MF Models

- To solve new-user / new-item problem
- Recalculating whole model **infeasible**

- We have the following scenario:
  - existing factorization (W,H) and a new user rating comes in

$\hat{R}_S$
- already calculated ratings matrix

$\hat{R}_{S \cup \{r_{u,i}\}}$
- can only be **approximated**, because
  - In stochastic gradient descent, the sequence of **how** ratings in S are visited is relevant
  - results between iterations **propagate** through the matrices

Online updating MF Models

- Algorithm 2: Goal
  - Update a **single** user/item feature vector when a new rating occurs

Online updating MF Models Contd.

TECHNISCHE UNIVERSITÄT DARMSTADT

Algo. 2

procedure UserUpdate$(S, W, H, r_{u,i})$
  $S \leftarrow S \cup \{r_{u,i}\}$
  return UserRetrain(S, W, H, u)
end procedure

procedure UserRetrain(S, W, H, u^*)
  initialize u^*-th row in W
  repeat
    for $r_{u,i} \in C(u^*, \cdot)$ **do**
      for $f \leftarrow 1, \ldots, f$ **do**
        $w_{u,f} \leftarrow w_{u,f} - \alpha \frac{\partial}{\partial w_{u,f}} \mathrm{Opt}(S, W, H)$
      end for
    end for
  until Stopping criteria met
  return (W, H)
end procedure

[2]

Online updating MF Models Contd.

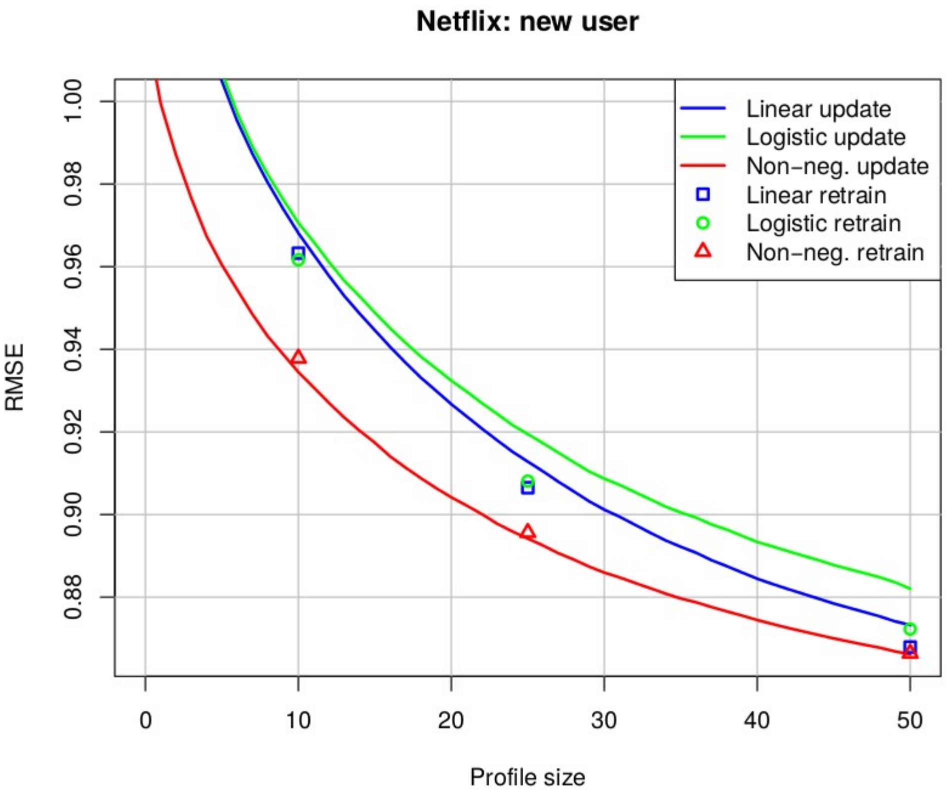
TECHNISCHE UNIVERSITÄT DARMSTADT

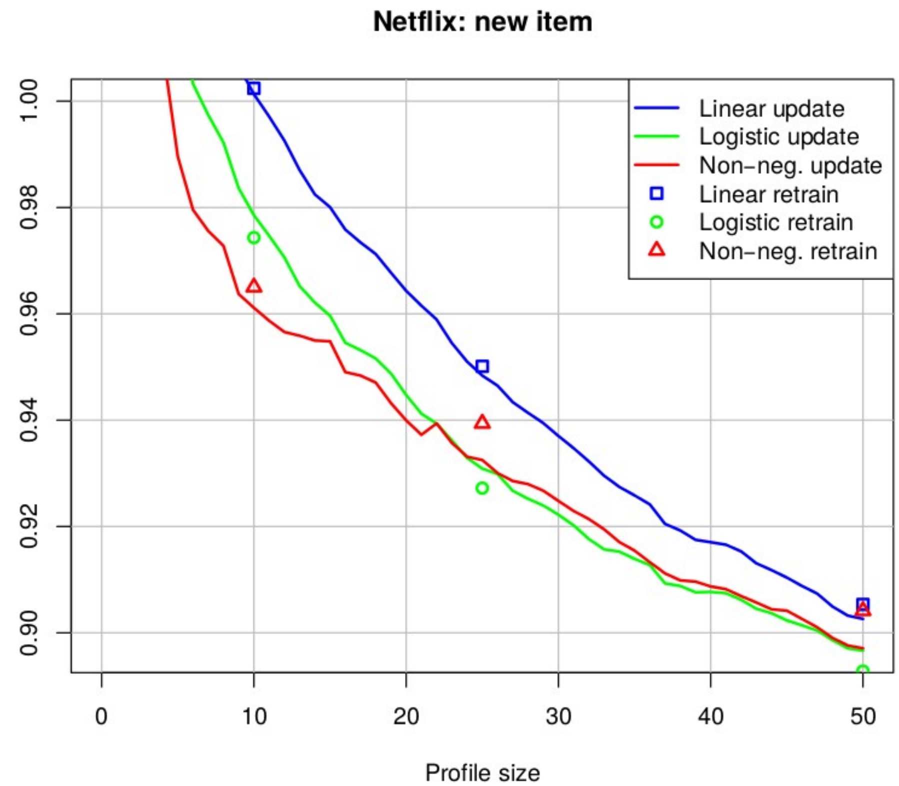
- Algorithm 2
  - Retrains a feature vector for a **single** user
  - Does **not** change matrix in other parts
  - Why does this work? **Assumptions**:
    - When new rating from user comes in, only that users feature vector will change much
    - the rest of the matrix won't change significantly → keep it **fixed**

New-user / new-item problem

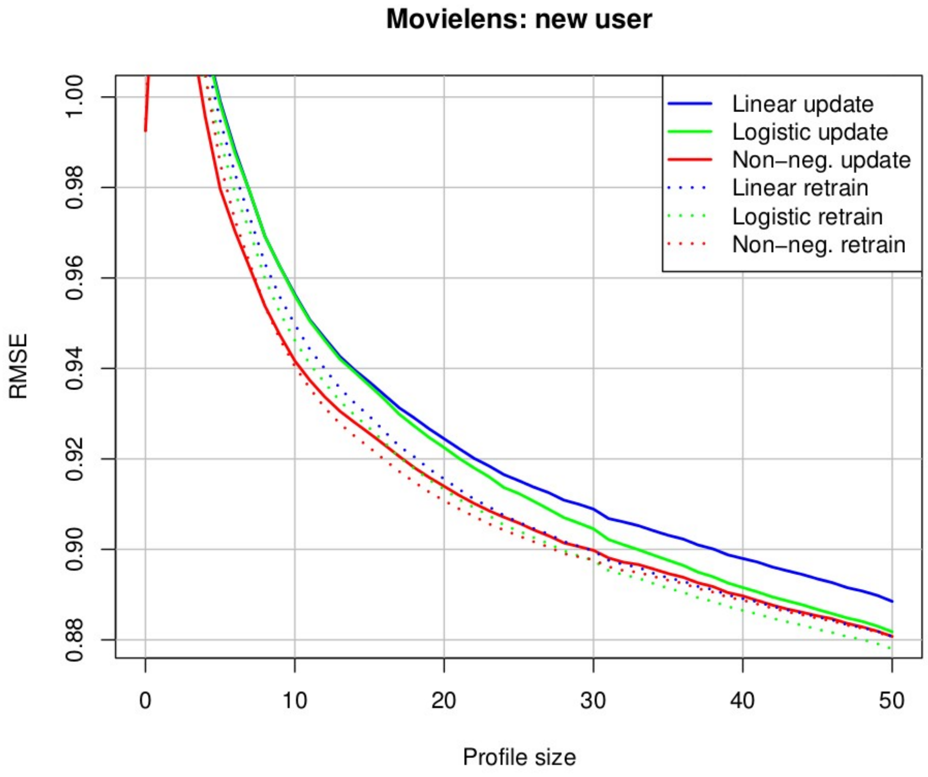
Evaluation

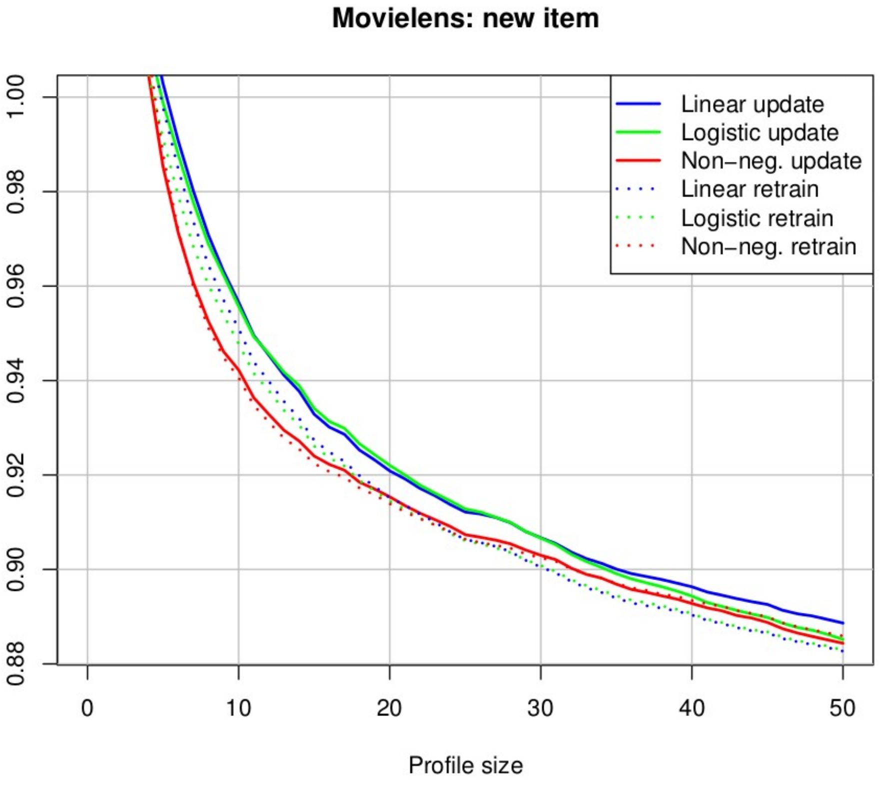
Evaluation on Netflix dataset





Evaluation on Movielens dataset





Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- RMF is very good for static rating prediction
  - Drawback: once model is computed it is static
- The proposed **online updating algorithm** overcomes this problem
  - Relatively low runtime complexity
  - Suitable for **large, dynamic** real-world applications (like Netflix)
- Results of update are **very close** to full retrain
  - While cost is substantially **smaller**!

Any questions?

References

- [3] Online-Updating Regularized Kernel Matrix Factorization Models for Large-Scale Recommender Systems
- [1] Online-Updating Regularized Kernel Matrix Factorization Models for Large-Scale Recommender Systems; Figure 2
- [2] Online-Updating Regularized Kernel Matrix Factorization Models for Large-Scale Recommender Systems; Figure 3
- Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights
- Item-Based Collaborative Filtering Recommendation Algorithms
- Matrix Factorization Techniques for Recommender Systems