Matrix Factorization Techniques for Recommender Systems

Patrick Seemann, December 16th, 2014
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

Content based approach

Neighborhood methods
Rating Prediction

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

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

Columns: **Items**

Rows: **Users**

Note: $R$ is **sparse**

an entry of $R$ is the rating of user $u$ for item $i$
New-User / New-Item Problem

- **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)

- Task: approximate the true unobserved ratings matrix \( R \) by
  
  \[ \hat{R} : |U| \times |I|, \quad \hat{R} = W \cdot H^T \]

- 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
- ...

\[ W = \begin{pmatrix} 0 & 10 & 0 & 10 & \ldots \\ 0 & 10 & 0 & 0 & \ldots \\ 8 & 0 & 0 & 5 & \ldots \\ 10 & 5 & 9 & 0 & \ldots \end{pmatrix} \]

\[ H = \begin{pmatrix} 7 & 1 & 10 & 0 & \ldots \\ 10 & 4 & 0 & 0 & \ldots \\ 0 & 9 & 0 & 0 & \ldots \\ 0 & 0 & 0 & 10 & \ldots \end{pmatrix} \]

Bob only likes comedy
Matrix Factorization (MF) Contd.

\[ \hat{R} : |U| \times |I|, \quad \hat{R} = W \cdot H^T \]

- 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 \textbf{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:
  - \textbf{Linear}: \( K_l(w_u, h_i) = \langle w_u, h_i \rangle \)
  - \textbf{Polynomial}: \( K_p(w_u, h_i) = (1+\langle w_u, h_i \rangle)^d \)
  - \textbf{Logistic}: \( K_s(w_u, h_i) = \Phi_s(b_{u,i} + \langle w_u, h_i \rangle) \) with \( \Phi_s(x) := \frac{1}{1+e^{-x}} \)
Rating prediction

Learning Matrix Factorization Models
Learning MF Models

- 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)

$$\argmin_{W,H} E(S,W,H)$$

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

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

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

\[
\arg\min_{\mathbf{W}, \mathbf{H}} \text{Opt}(\mathbf{S}, \mathbf{W}, \mathbf{H})
\]

\[
\text{Opt}(\mathbf{S}, \mathbf{W}, \mathbf{H}) := E(\mathbf{S}, \mathbf{W}, \mathbf{H}) + \lambda \cdot (\|\mathbf{W}\|_F^2 + \|\mathbf{H}\|_F^2)
\]

- **Early stopping** is also used to work against overfitting
Learning MF Models Contd.

• Optimizing by Stochastic Gradient Descent

Algo. 1 1: procedure OPTIMIZE(S, W, H)
2: initialize W, H
3: repeat
4: for \( r_{u,i} \in S \) do
5: for \( f \leftarrow 1, \ldots, k \) do
6: \( w_{u,f} \leftarrow w_{u,f} - \alpha \frac{\partial}{\partial w_{u,f}} \text{Opt}(\{r_{u,i}\}, W, H) \)
7: \( h_{i,f} \leftarrow h_{i,f} - \alpha \frac{\partial}{\partial h_{i,f}} \text{Opt}(\{r_{u,i}\}, W, H) \)
8: end for
9: end for
10: until Stopping criteria met
11: return (W, H)
12: end procedure

[1] e.g. a fixed number
New-user / new-item problem

Online Updating RKMF Models
Complexity of Training KMF models

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

\[ O(|S| \cdot k \cdot i) \]

- 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

\[ O(|C(u, \cdot)| \cdot k \cdot i) \]

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

• Retraining single user requires \textbf{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

\[
\hat{R}_S \quad \text{• already calculated ratings matrix}
\]
\[
\hat{R}_{S \cup \{r_{u,i}\}} \quad \text{• can only be \textit{approximated}, because}
\]
- In stochastic gradient descent, the sequence of how ratings in \(S\) are visited is relevant
- results between iterations \textit{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.

Algo. 2

1: **procedure** USERUPDATE($S, W, H, r_{u,i}$)
2: 
\[ S \leftarrow S \cup \{r_{u,i}\} \]
3: **return** USERRETRAIN($S, W, H, u$)
4: **end procedure**

5: **procedure** USERRETRAIN($S, W, H, u^*$)
6: initialize $u^*$-th row in $W$
7: **repeat**
8: \[ \text{for } r_{u,i} \in C(u^*, \cdot) \text{ do} \]
9: \[ \text{for } f \leftarrow 1, \ldots, f \text{ do} \]
10: 
\[ w_{u,f} \leftarrow w_{u,f} - \alpha \frac{\partial}{\partial w_{u,f}} \text{Opt} (S, W, H) \]
11: \[ \text{end for} \]
12: \[ \text{end for} \]
13: **until** Stopping criteria met
14: **return** $(W, H)$
15: **end procedure**

[2]
Online updating MF Models Contd.

- 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

Netflix: new user

Netflix: new item

RMSE

Profile size
Evaluation on Movielens dataset

Movielens: new user

Movielens: new item
Summary

- 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