Online-Updating Regularized Kernel Matrix Factorization Models for Large-Scale Recommender Systems.

Steffen Rendle, Lars Schmidt-Thieme

University of Hildesheim, Germany

February 10th, 2012

Outline

- Motivation
- Related work
- Matrix Factorization (MF)
- Kernel Matrix Factorization (KMF)
- Learning Matrix Factorization Models
- SVD versus Regularized KMF
- Online Updates
- Evaluation

Motivation

Recommenders predict how much a user likes a given item.
A matrix completition task, where a matrix $R : |U| \times |I|$ should be completed.

- The entry $r_{u,i}$ represents a rating of a user u for item i.
- A set S of observed ratings contains triples (u, i, v).
- A matrix factorization estimates R with $\hat{R}$ and in a such way new ratings are predicted.

Motivation

Dynamics of recommender systems require often recomputation of prediction models e.g., when new user enters a system.
Models for large-scale recommenders are static and do not reflect after training users' ratings – *new-user problem*.
The profile $C(u, .)$ of a user u grows from 0 to k ratings where:

$$C(u, i) := \{r_{u',i'} \in S | u' = u \wedge i' = i\}$$

Related work

Different approaches for rating prediction:

- Collaborative filtering based on k-nearest-neighbor method (kNN) [SKKR01].
- Latent semantic models [Hof04], classifiers [ST05].
- Models based on matrix factorization (MF) [Wu07].

Matrix factorization (MF)

The goal is to approximate the true unobserved ratings-matrix R by $\hat{R} : |U| \times |I|$.

$$\hat{R} = W \cdot H^t$$

where $W : |U| \times k$ and $H : |I| \times k$

- w_u represents k features that describe user u.
- h_i represents k features that describe item i.

$$\hat{r_{u,i}} =< w_u, h_i >= \sum_{f=1}^{k} w_{u,f} \cdot h_{i,f}$$

Often can be added a bias term $b_{u,i}$ which centers the approximation.

Kernel Matrix Factorization

Interactions between the feature vector w_u and h_i are kernelized:

$$\hat{r_{u,i}} =< w_u, h_i >= a + c \cdot K(w_u, h_i)$$

- Terms a, c allow rescaling the approximation.
- Kernel function $K : \mathcal{R}^k \times \mathcal{R}^k \rightarrow \mathcal{R}$ can utilize one of the following well-known kernels:

linear:	$K_l(w_u, h_i) =< w_u, h_i >$
polynomial:	$K_p(w_u, h_i) = (1+ < w_u, h_i >)^d$
RBF:	$K_r(w_u, h_i) = exp(-\frac{\|\|w_u, h_i\|\|^2}{2\sigma^2})$
logistic:	$K_s(w_u, h_i) = \phi_s(b_{u,i}+ < w_u, h_i >)$
	where $\phi_s(x) = \frac{1}{1+\exp^{-x}}$

Kernel Matrix Factorization

Benefits of using kernels:

- Using kernel like logistic one results into bounded values of the ratings to the application domain.
- Model non-linear correlations between users and items.
- Kernels lead to different models that can be combined in an ensemble.

Non-negative matrix factorization

Additional constraints on feature matrices W and H such that each entry has to be non-negative.
The motivation is to eliminate interactions between negative correlations (commonly used in CF algorithms).

Learning Matrix Factorization Models.

Minimize an error between an approximated matrix $\hat{R}$ and original matrix R.
The optimization task is defined as: $\underset{W,H}{\operatorname{argmin}} E(S, W, H)$ where:

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

Overfitting:
Netflix dataset contains 480000 users and 17000 items with 100 million ratings. It leads to estimating 50 million of parameters when $k = 100$ which results into overfitting.
Two strategies:

- Regularization
- Early stopping criterion

Regularization

A regularization term is added to the optimization task.
Tikhonov regularization is used and a parameter λ controls regularization.
The final optimization task is: $\underset{W,H}{\operatorname{argmin}} Opt(S, W, H)$ where:

$$Opt(S, W, H) := E(S, W, H) + \lambda(||W||_F^2 + ||H||_F^2)$$

Optimization by Gradient Descent

Gradient descent is used for MF and also for KMF:

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**

Figure: Generic learning algorithm for KMF.

SVD versus Regularized KMF

Singular Value Decomposition (SVD) decomposes a matrix into 3 matrices:

$$R = W'\Sigma H', \text{where:} W' : |U| \times |U|, \Sigma : |U| \times |I|, H' : |I| \times |I|$$

SVD is not suitable for recommender systems because of:

- The huge number of missing values that has to be estimated e.g, sparsity rate is 99% for Netflix dataset.
- Lack of regularization leads to overfitting.

	SVD	Regularized MF		
		linear	logistic	lin. non-neg.
Netflix RMSE	0.946 [8]	0.915	0.918	0.914

Figure: RMSE results on Netflix probe for RKMF and k-rank SVD.

Online Updates

Retraining the whole KMF when a new rating arrives is not suitable e.g., For Netflix dataset when $k = 40$, $i = 120$ and $S = 100000000$ results into 480 billion feature updates.
Online Updates technique:

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

Further Speedup

Retraining a user u on a new rating is extremely important for a user with small profile.
Proposed rules that allow to skip some online updates when user's profile size is large enough:

$$P_u(train|r_{u,i}) = \gamma^{|C_{u,.}|}, \gamma \in (0,1)$$

$$P_u(train|r_{u,i}) = \max\left(1, \frac{m}{|C(u,.)|}\right); m \in N^+$$

Evaluation

1. Create a new-user scenario.
   - Pick n% of the users and put them in U_t.
   - For each user $u \in U_t$ do
     - Split the ratings in $C(u,.)$ in 2 disjoint sets T_u and V_u where $|T_u| = \min(m, \frac{|C(u,.)|}{2})$ and $V_u = C(u,.) \setminus T_u$
     - Remove all of the ratings $C(u,.)$ from S
2. Train the model on S: $(W, H) \leftarrow Opt(S, W, H)$.
3. Evaluate the new-user scenario for $j = 1 \ldots m$ do:
   - For each user $u \in U_t$ do
     - add one rating $r_{u,i} \in T_u$ to S.
     - update the model: $(W, H) \leftarrow USERUPDATE(S, W, H, r_{u,i})$
     - calculate error $se_u^j = E(V_u, W, H)$.
   - calculate $RMSE^j = \sqrt{\frac{1}{\sum_{u:|T_u| \geq j} |V_u|} \cdot \sum_{u:|T_u| \geq j} se_u^j}$.

Evaluation settings

Evaluation on two movie recommendation datasets:

Dataset	Users	Items	Ratings
Neflix	480000	17000	100 million
Movielens	6040	3706	1 million

Quality of recommendations

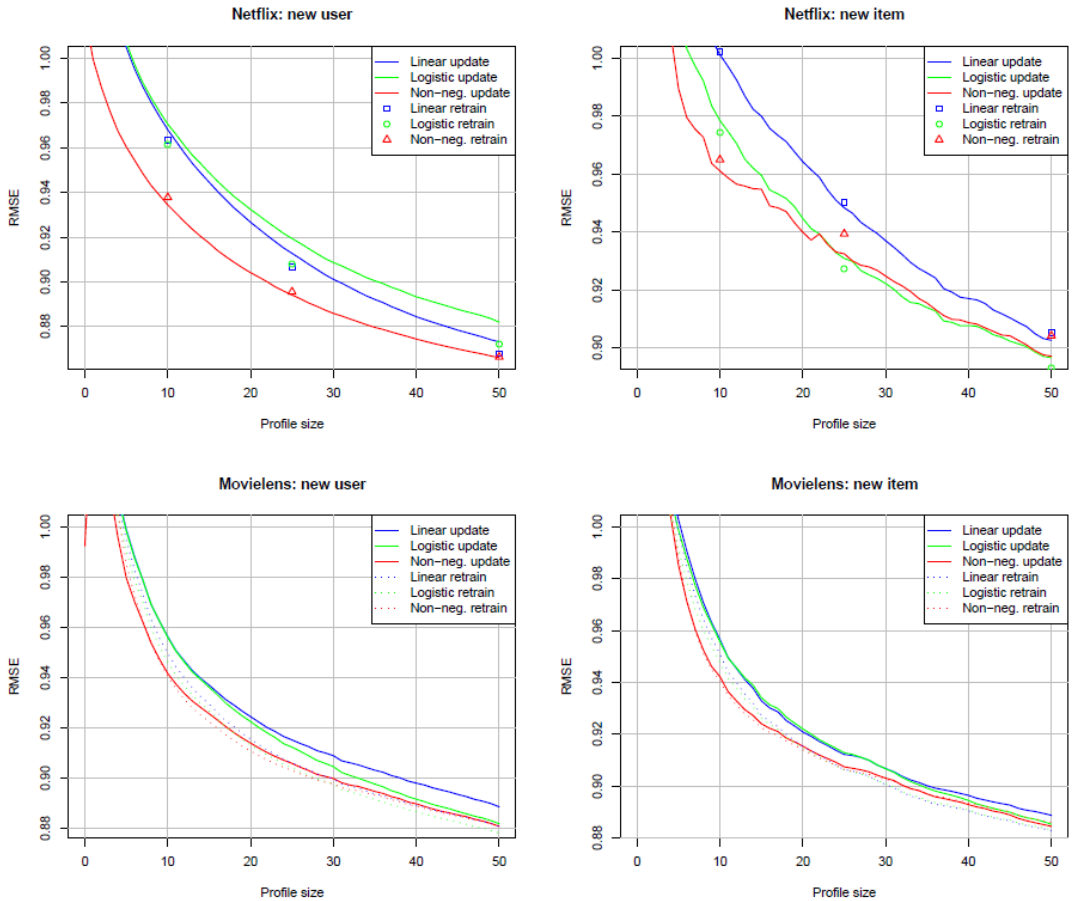

Figure: New-user/ new-item problem on Movielens and Netflix. Curves show the RMSE of online-updates (see protocol) compared to a full retrain.

Speedup

Dataset	Training	Regularized MF		
		linear	logistic	non-neg.
Movielens	Online Update	0-1 ms	0-10 ms	0-1 ms
	Retrain	18 s	3.5 min	25 s
	Features k	10	10	10
	# Iterations	30	270	50
Netflix	Online Update	0-15 ms	0-15 ms	0-18 ms
	Retrain	11.6 h	10.2 h	13.8 h
	Features k	40	40	40
	# Iterations	120	120	120

Figure: Runtime of full retrain and runtime of proposed online updates wrt to profile size C(u; .) (for 'new-user' problem).

Conclusion

- Generic online-update methods for RKMF models.
- A precise approximation without a whole retraining process.
- Runtime complexity makes online-updates feasible for a real world datasets.

References

T. Hofmann.

Latent semantic models for collaborative filtering.

ACM Transactions on Information Systems, 22(1):89–115, 2004.

B. Sarwar, G. Karypis, J. Konstan, and J. Reidl.

Item-based collaborative filtering recommendation algorithms.

In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

L. Schmidt-Thieme.

Compound classification models for recommender systems.

In *Data Mining, Fifth IEEE International Conference on*, pages 8–pp. IEEE, 2005.

M. Wu.

Collaborative filtering via ensembles of matrix factorizations.

In *Proceedings of KDD Cup and Workshop*, 2007.