Journal of Machine Learning Research 13 (2012) 3619-3622 Submitted 10/11; Published 12/12

SVDFeature: A Toolkit for Feature-based Collaborative Filtering

Tianqi Chen TQCHEN@APEX.SJTU.EDU.CN
Weinan Zhang WNZHANG@APEX.SJTU.EDU.CN
Qiuxia Lu LUQIUXIA@APEX.SJTU.EDU.CN
Kailong Chen CHENKL@APEX.SJTU.EDU.CN
Zhao Zheng ZHENGZHAO@APEX.SJTU.EDU.CN
Yong Yu YYU@APEX.SJTU.EDU.CN
Apex Data and Knowledge Management Lab
Shanghai Jiao Tong University
800 Dongchuan Road
Shanghai 200240 China

Editor: Mikio Braun

Abstract

In this paper we introduce SVDFeature, a machine learning toolkit for feature-based collaborative filtering. SVDFeature is designed to efficiently solve the feature-based matrix factorization. The feature-based setting allows us to build factorization models incorporating side information such as temporal dynamics, neighborhood relationship, and hierarchical information. The toolkit is capable of both rate prediction and collaborative ranking, and is carefully designed for efficient training on large-scale data set. Using this toolkit, we built solutions to win KDD Cup for two consecutive years.

Keywords: large-scale collaborative filtering, context-aware recommendation, ranking

1. Introduction

Recommender system, which recommends items based on users' interests, has become more and more popular in many real-world situations. Collaborative filtering (CF) techniques, as the main thrust behind recommender systems, have been developed for many years and keep to be a hot area in both academia and industry. In this paper, we focus on building collaborative filtering based recommendation toolkit which can effectively leverage the rich information of data collected and naturally scale up to very large data set.

Matrix factorization (MF) is one of the most popular CF methods, and variants of it have been proposed in specific settings. However, traditional approaches design specific models for each problem, demanding great efforts in engineering. Fortunately the majority of factorization models share many common patterns, which enables us to summarize them into a single model and implement a unified toolkit, called SVDFeature.[1] SVDFeature enables incorporating auxiliary information via feature engineering. This helps save the efforts of engineering for specific models and allows users to focus on model design. Meanwhile, SVDFeature is specifically designed to handle large data sets. With SVDFeature, we have won the two most recent KDD Cups. Moreover, it costs less than

1. Project page can be found at `http://svdfeature.apexlab.org`.

©2012 Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng and Yong Yu.

2GB memory to get an excellent performance of 22.16 on RMSE on Yahoo! Music data set with 200 million ratings records. This has manifested SVDFeature great modeling power and scalability.

2. Model of SVDFeature

There are three important factors in most CF problems: users' interests, items' properties and other factors that directly affect users' preferences over items. Various kinds of information can be used to model these factors. For example, users' browsing history over movie reviews may correlate with users' taste over movies; the information of the director and actors of a movie can be used to predict its properties; the rating history over similar movies directly affect whether a user will favor the current one. Our model summarizes the three factors as feature vectors (denoted by $\alpha \in \mathbb{R}^m, \beta \in \mathbb{R}^n, \gamma \in \mathbb{R}^s$) and predicts the preference score $\hat{y}$ as

$$\hat{y}(\alpha,\beta,\gamma) = \left(\sum_{j=1}^{s} \gamma_j b_j^{(g)} + \sum_{j=1}^{n} \alpha_j b_j^{(u)} + \sum_{j=1}^{m} \beta_j b_j^{(i)}\right) + \left(\sum_{j=1}^{n} \alpha_j \mathbf{p}_j\right)^T \left(\sum_{j=1}^{m} \beta_j \mathbf{q}_j\right).$$

Here the model parameter set is defined as $\Theta = \{b^{(g)}, b^{(u)}, b^{(i)}, \mathbf{p}, \mathbf{q}\}$. $\mathbf{p}_j \in \mathbb{R}^d$ and $\mathbf{q}_j \in \mathbb{R}^d$ are d dimensional latent factors associated with each feature. $b_j^{(u)}$, $b_j^{(i)}$ and $b_j^{(g)}$ are bias parameters that directly influence the prediction. We call α user feature, β item feature, and γ global feature. This model is described as part of solution to KDD Cup (Chen et al., 2011, 2012). We call it feature-based matrix factorization. Intuitively, we use a linear model to construct user and item latent factors from features. The parameters are trained efficiently by minimizing a loss function using stochastic gradient descent. The supported loss functions include square-loss, negative logistic log-likelihood loss and smoothed hinge loss.

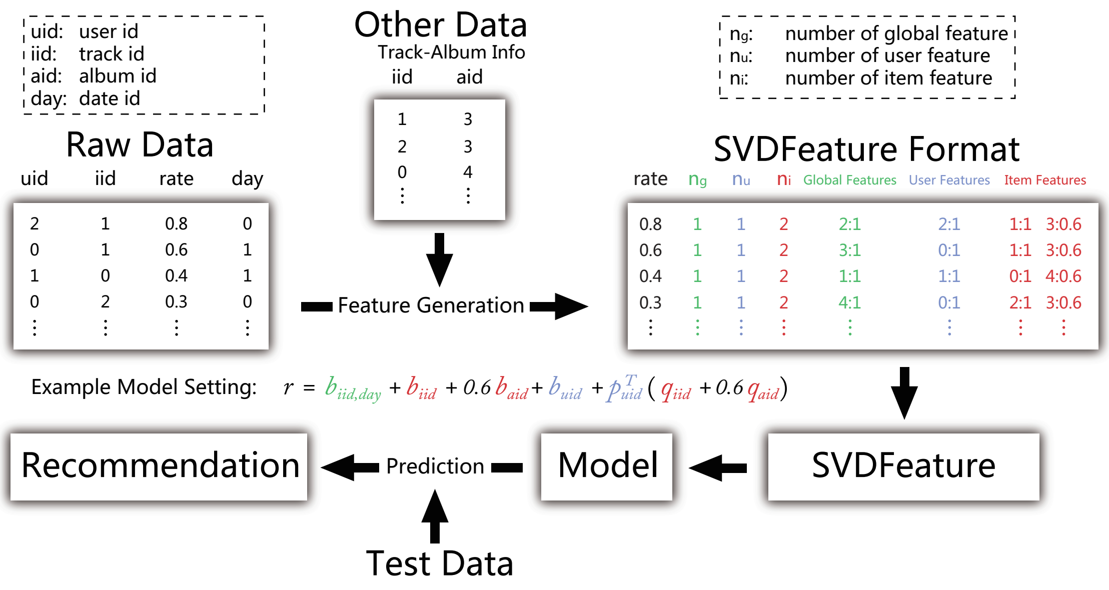
Many state-of-the-art collaborative filtering algorithms can be implemented using SVDFeature. Let us suppose we want to recommend music tracks for users using the rating history, and we have known the albums of tracks and the timestamps of the ratings as auxiliary information. As we know, the album information can be used to better represent the content of tracks, and the temporal information can be used to detect the changes of item popularity. All these auxiliary information can help improve the performance of a recommender system. Taking them into consideration, we can represent a user u's preference towards track i at time t as follows

$$\hat{y}(u,i,t) = b_{i,bin(t)} + b_i + b_{p(i)} + b_u + \mathbf{p}_u^T(\mathbf{q}_i + \mathbf{q}_{al(i)})$$

where $al(i)$ is the album of a track i. We segment time into consecutive bins and define $bin(t)$ to map timestamp to corresponding bin index. We use time localized bias $b_{i,bin(t)}$ to model items' popularity change over time. To incorporate the taxonomical information, we introduce $\mathbf{q}_{al(i)}$ to make the prediction towards track i depend on the latent factor of corresponding album. To implement this model using SVDFeature, we can simply define the features as follows

$$\alpha_h = \begin{cases} 1 & h = u \\ 0 & h \neq u \end{cases}, \quad \beta_h = \begin{cases} 1 & h = i \text{ or } h = al(i) \\ 0 & otherwise \end{cases}, \quad \gamma_h = \begin{cases} 1 & h = \#bins \times i + bin(t) \\ 0 & otherwise \end{cases}.$$

The item feature β is defined to be indicator of whether the record is related to the track and its corresponding album. We also define a global feature γ_h to encode the temporally varying item bias $b_{i,bin(t)}$. We can further incorporate more information by defining more features.



uid	iid	rate	day
2	1	0.8	0
0	1	0.6	1
1	0	0.4	1
0	2	0.3	0
⋮	⋮	⋮	⋮

iid	aid
1	3
2	3
0	4
⋮	⋮

rate	n_g	n_u	n_i	Global Features	User Features	Item Features	
0.8	1	1	2	2:1	2:1	1:1	3:0.6
0.6	1	1	2	3:1	0:1	1:1	3:0.6
0.4	1	1	2	1:1	1:1	0:1	4:0.6
0.3	1	1	2	4:1	0:1	2:1	3:0.6
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Example Model Setting: $r = b_{iid,day} + b_{iid} + 0.6\, b_{aid} + b_{uid} + p_{uid}^T (q_{iid} + 0.6\, q_{aid})$

Figure 1: Usage flow example of SVDFeature

3. Using SVDFeature

Figure 1 gives a usage example of SVDFeature to implement the model introduced in previous section. We encode the album information as item features and item day biases as global features. The number 0.6 associated with album id is an empirical parameter chosen by the user to control the influence of album information in prediction. Assuming there are only two days' records in the data, the global feature index is defined as $gid = 2 \times iid + day$,where the number of item day biases is twice the number of items. SVDFeature will learn a feature-based matrix factorization model with the given training data and make predictions on supplied test feature files. We provide a manual to give more details about the usage of SVDFeature.

4. Handling Big Data

Recommendation algorithms often have to deal with problems with large scale data set in real world, which has been taken into consideration in designing SVDFeature. In our approach, we store the data into a buffer file in hard disk. The data is shuffled before storing, and then the training program linearly iterates over the buffer and updates the model with each training sample. This approach allows us to do training as long as the model fits into memory. To reduce the additional cost of I/O introduced by data loading, we use a pre-fetch strategy. An independent thread is created to fetch the data from hard disk into a memory queue. At the same time, the training thread reads the data from memory queue and updates the model. This pipeline style of execution releases the burden of I/O from training thread. As long as I/O speed is similar to (or faster than) training speed, the cost of I/O is negligible. Figure 2 shows the procedure of pipeline execution.

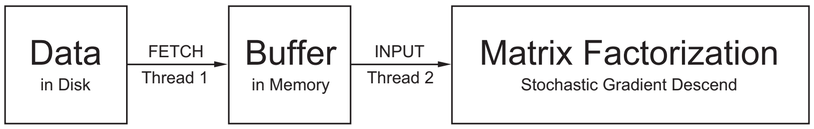


Figure 2: Pipeline training

5. Extra Features

SVDFeature also provides several other extra features for better modeling capability. We list notable features here: (1) Efficient speedup training for user feedback information; (2) Supporting collaborative ranking training; (3) Different kinds of regularization options; (4) Separated feature table extension for user and item features. The details are described in the project page and manual.

6. Availability and Documentation

The source code of SVDFeature is implemented in C++ and can be compiled under both Linux and Windows. The project is self-contained and only depends on standard libraries. SVDFeature is released under Apache License, Version 2.0. We provide a technical document introducing the algorithm and a user manual describing the usage details of the toolkit. To help users get started, we also provide a demo folder with example shell scripts that show the procedures from feature generation to training and prediction.

Acknowledgments

Yong Yu is supported by grants from NSFC-RGC joint research project 60931160445. We thank the reviewers and editor for their helpful comments. We greatly appreciate the discussions with Zhengdong Lu, Diyi Yang and Li'ang Yin.

References

T. Chen, Z. Zheng, Q. Lu, X. Jiang, Y. Chen, W. Zhang, K. Chen, Y. Yu, N. Liu, B. Cao, L. He, and Q. Yang. Informative ensemble of multi-resolution dynamic factorization models. In *KDD-Cup Workshop*, 2011.

T. Chen, L. Tang, Q. Liu, D. Yang, S. Xie, X. Cao, C. Wu, E. Yao, Z. Liu, Z. Jiang, C. Chen, W. Kong, and Y. Yu. Combining factorization model and additive forest for collaborative followee recommendation. In *KDD-Cup Workshop*, 2012.