A Hybrid Matrix Factorization Method with Isolation Forest for Recommendation System

Jia-kun ZHAO, Hui-min CHEN∗, Zhen LIU and Yi-fei XU
School of Software Engineering, Xi’an Jiaotong University, Xi’an, 710049, China
*Corresponding author

Keywords: Matrix factorization, Outlier detect, Isolation forest, Recommendation system.

Abstract. Matrix Factorization (MF), which is a traditional Collaborative Filtering (CF) technology, has been widely used in recommendation system. MF model relies on exiting user-item ratings, which maybe contains some noise because of intrusion attack, error of log system or mistake of artificial data. In order to detect these data noises and enhances the rating prediction accuracy, we propose a new method, a hybrid matrix factorization technique with Isolation Forest (IForest), which is shown to be highly effective in detecting anomalies with extremely high efficiency. IForest detects anomalies by builds an ensemble of iTrees for a given data set, then anomalies are those instances which have short average path lengths on the iTrees. Extensive experiment results on movielens (1M) datasets show that our hybrid model outperforms other methods in effectively utilizing side information and achieves performance improvement.

Introduction

Recently, recommendation system has become one of the effective methods to filtering information. Matrix factorization (MF) technique has become a dominant methodology within collaborative filtering recommendation algorithm[1], whose accuracy is better than other traditional methods.

However, due to the intrusion attack of bad merchants, the error of log system or the mistake of artificial data collection, the data of recommendation system will produce some noise, which will affect the effectiveness of recommendation algorithm. Therefore, it is necessary to integrate outlier detection technology and collaborative filtering to solve this problem.

Outlier detection (or anomaly detection) is a fundamental data analysis problem in machine learning and data mining field. Outlier detection algorithms can be divided into statistical-based methods[2-5], distance-based methods[6], density-based methods[7] and clustering-based methods[8,9]. Mehta et al. proposed principal component analysis (PCA), which can reduce the dimensionality of the data by finding a few orthogonal linear combinations of the original variables with the largest variance. However, only by clearing the size of attack profiles can PCA work well, otherwise it will affect the accuracy of the experiment[10]. Chakrabortya et al. applied a PAM based outlier detection algorithm to find attack profiles in large clusters that are not identified by the PAM algorithm[11]. Chung et al. proposed Beta-Protection to screen malicious attackers[12]. Liu et al. proposed isolation forest(iForest), which is a fast anomaly detection method based on Ensemble, with linear time complexity and high accuracy, is an accurate and efficient anomaly detector especially for large databases[13].

This paper proposes a different model, hybrid MF with iForest for recommendation system, which works well on movielens dataset, with 2.273% of RMSE being reduced than simply using MF.

Matrix Factorization

Some of the most successful realizations of latent factor models are based on matrix factorization. In its basic form, matrix factorization characterizes both items and users by vectors of factors inferred from item rating patterns[14]. High correspondence between item and user factors leads to
a vector \( q_i \in \mathbb{R}^f \) and each user \( u \) is associated with a vector \( p_u \in \mathbb{R}^f \). For a given item \( i \), the elements of \( q_i \) measure the extent to which the item possesses those factors, positive or negative. Figure 1 shows rating matrix decomposition process of \( M \) users and \( N \) movies with this idea.

\[
\text{** User size = } M, \text{ item size = } N, \text{ dimensions = } x
\]

For a given item \( i \), the elements of \( q_i \) measure the extent to which the item possesses those factors, positive or negative. The resulting dot product, \( q_i^T p_u \), captures the interaction between user \( u \) and item \( i \)—the user’s overall interest in the item’s characteristics. This approximates user \( u \)’s rating of item \( i \), which is denoted by \( r_{ui} \), leads to the estimate

\[
\hat{r}_{ui} = q_i^T p_u
\]

The major challenge is computing the mapping of each item and user to factor vectors \( q_i, p_u \in \mathbb{R}^f \). After completing this mapping through recommendation system, it can easily estimate the rating a user will give to any item by using Eq. 1.

In order to learn the factor vectors \( (p_u \text{ and } q_i) \), the system minimizes the regularized squared error on the set of known ratings:

\[
\min_{q_i, p_u} \sum_{(u,i) \in k} (r_{ui} - q_i^T p_u)^2 + \lambda (\| q_i \|^2 + \| p_u \|^2)
\]

Here, \( k \) is the set of the \((u,i)\) pairs for which \( r_{ui} \) is known (the training set). The constant \( \lambda \) controls the extent of regularization and is usually determined by cross-validation. The system learns the model by fitting the previously observed ratings. Two approaches to minimizing Eq. 2 are stochastic gradient descent (SGD)[15] and alternating least squares (ALS)[14].

### Isolation Forest

For high dimensional problems that contain a large number of irrelevant attributes, iForest can achieve high detection performance quickly with an additional attribute selector, whereas a distance-based method has poor detection performance and requires significantly more time. Isolation Forest is an accurate and efficient anomaly detector.

**Training Stage.** Given a \( d \)-variate database of \( n \) instances \((D = \{x(1), x(2), \ldots, x(n)\})\), iForest constructs \( t \) iTrees \((T_1, T_2, \ldots, T_t)\). Each \( T_i \) is constructed from a small random sub-sample \((D_i \mid D_i = \varnothing < n)\) by recursively dividing it into two nonempty nodes through a randomly selected attribute and split point. A branch stops splitting when the height reaches the maximum \((H_{max})\) or the number of instances in the node is less than \( MinPts \). The details of the producing an iTree can be found in Algorithms 1.
Algorithm 1: iTree(X,e,l)

Inputs: X-input data, e-current tree height, l-height limit

Output: an iTree

1: if e ≥ l or |X| ≤ 1 then
2: return exNode{Size←|X|}
3: else
4: let Q be a list of attributes in X
5: randomly select an attribute q ∈ Q
6: randomly select a split point p from max and min values of attribute q in X
7: X_l ← filter(X,q<p)
8: X_r → filter(X,q≥p)
9: return inNode{Left←iTree(X_l,e+1,l),
10: Right←iTree(X_r,e+1,l),
11: SplitAtt←q,
12: SplitValue←p}
13: end if

The default values used in iForest are $H_{\text{max}} = \log_2(\phi)$ and $\text{MinPts}=1$. In the end of training process, a collection of trees is returned and ready for the evaluating stage.

### Evaluating Stage

The anomaly score is estimated as the

$$ L(x) = \frac{1}{t} \sum_{i=1}^{t} I_i(x) $$

where $I(x)$ is the path length of $x$ in $T_i$. To find the top $m$ anomalies, we can simply sort the data using $s$ in descending order. The first $m$ instances are the top $m$ anomalies.

### A Hybrid Matrix Factorization with Isolation Forest

MF is a method to construct the implicit semantic model of recommendation system. By decomposing the sorted and extracted "user-item" score matrix, we can get a user implicit vector matrix and an item implicit vector matrix. However, in practical applications, some data noises are often caused by log system, malicious attacks, human errors and other reasons. Consequently MF is used to improve the recommendation performance through preprocessing the data by iForest and decomposing the user-item rating matrix.

#### Step 1. Filling Data

It could be supposed that movielens is a perfect dataset, noise data is added to user-item rating matrix at a filling rate of 10%. Details of filling data can be found in Algorithm 2.

Algorithm 2: dataFill(X,s,r)

Inputs: X-input user-item ratings, s-size of user-item ratings, r-filling rate

Output: ratings-filled

1: set filling number n = s * r
2: while i<n do
3: randomly select a userId $u \in U$
4: randomly select a movieId $m \in M$
5: if $(u,m)$ in X then
6: continue
7: randomly produce a integer r between 1 and 5
8: randomly produce a timestamp t
9: ratings-filled ← ratings-filled ∪ $(u,m,r,t)$
10: $i ← i+1$
11: end

#### Step 2. Preprocessing Data

Preprocessing the ratings-filled by iForest and filtering the top 10% noise data according the anomaly score.
Step 3 Decomposing Matrix with SGD and ALS. In order to minimizing Eq. 2, we used SGD and ALS optimization to decompose the matrix respectively.

- **Stochastic Gradient Descent (SGD)**

  For each given training case, the system predicts $r_{ui}$ and computes the associated prediction error

  \[ e_{ui} = r_{ui} - q_i^T p_u \]  

  Then it modifies the parameters by a magnitude proportional to $g$ in the opposite direction of the gradient, yielding:

  \[ q_i := q_i + \gamma (e_{ui} \cdot p_u - \lambda \cdot q_i) \]  

  \[ p_u := p_u + \gamma (e_{ui} \cdot q_i - \lambda \cdot p_u) \] 

- **Alternating Least Squares (ALS)**

  ALS techniques rotate between fixing the $q_i$’s and fixing the $p_u$’s. When all $p_u$’s are fixed, the system recomputes the $q_i$’s by solving a least-squares problem, and vice versa. This ensures that each step decreases Eq. 2 until convergence.

**Experiments**

**Dataset.** MovieLens is a web-based recommender system and virtual community, which recommends movies for its users to watch, based on their film preferences using collaborative filtering of members' movie ratings and movie reviews. MovieLens was created in 1997 by GroupLens Research, a research lab in the Department of Computer Science and Engineering at the University of Minnesota, to research data on personalized recommendations. In this experiment, we use the movielens-1m, which contains 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 MovieLens users who joined MovieLens in 2000[16].

**Model Evaluation Index.** We employ the root mean squared error (RMSE) as the evaluation index,

\[ RMSE = \sqrt{\frac{1}{|T|} \sum_{i \in T} (R_{ij} - \hat{R}_{ij})^2} \]  

Where $R_{ij}$ is the rating of user $i$ on item $j$, $\hat{R}_{ij}$ denotes the corresponding predicted rating, $T$ is the test set and $|T|$ is the total number of ratings in the test set.

**Result Analysis**

Figure 2 illustrates that the RMSE reaches the minimum in the third epoch. After that it slightly increases with model overfitting. As a result, we choose point in the third epoch for the analysis of the experimental results. In practical applications, we can also determine the number of epochs of model training by observing the RMSE of each epoch.

Figure 2 also shows that the RMSE reduction rate of iForest_ALS (2.273%, from 1.0877 to 1.063) is higher than iForest_SDG (1.264%, from 0.9256 to 0.9139), while the result of RMSE of iForest_SDG is better than iForest_ALS. It proves that preprocessing data by IForest outlier detect technique is more effective for ALS. It proves that preprocessing data by IForest outlier detect technique is more effective for ALS.

117
Table 1 shows the train and validation REMs of models, in which we can observe that iForest_MF achieves better result than MF. As a result, It demonstrates the effectiveness of pre-filtering data by iForest outlier detect.

Summary
This paper proposes a new method, a hybrid matrix factorization technique with iForest, which is shown to be highly effective in detecting anomalies with extremely high efficiency. The iForest has a linear time complexity with a low constant and a low memory requirement, which is ideal for high volume data sets. As a result, hybrid matrix factorization with iForest is effective for improving the accuracy of recommendation system. Our experimental results present that the hybrid model outperforms other two popular matrix factorization method. As part of the future work, we will continue to investigate the impact of other outlier detection methods and recommendation algorithms on the performance of recommended systems.

Acknowledgement
This work was supported by the Nature Science Foundation of China (NSFC) under Grant 61802300.

References
[1] Schafer J B, Dan F, Herlocker J, et al. Collaborative Filtering Recommender Systems[J]. 2007.
[2] P.J. Rousseeuw, K.V. Driessen. A fast algorithm for the minimum covariance determinant estimator. Technometrics, 41(3):212–223, 1999.
[3] Lewis T. Outliers in Statistical Data (3rd ed)[J]. 1994.
[4] Yamanishi, Kenji, Takeuchi, et al. Discovering outlier filtering rules from unlabeled data: combining a supervised learner with an unsupervised learner[J]. 2001.
[5] Ruts I, Rousseuw P J. Computing depth contours of bivariate point clouds[J]. Computational Statistics & Data Analysis, 1996, 23(1):153-168.
[6] Ramaswamy S, Rastogi R, Shim K. Efficient algorithms for mining outliers from large data sets[C]//ACM SIGMOD Record. ACM, 2000, 29(2): 427-438.
[7] Breunig M M, Kriegel H P, Ng R T, et al. LOF: identifying density-based local outliers[C]//ACM Sigmod Record. ACM, 2000, 29(2): 93-104.

[8] He Z, Xu X, Deng S. Discovering cluster-based local outliers[J]. Pattern Recognition Letters, 2003, 24(9): 1641-1650.

[9] Shi Y, Zhang L. Coid: A cluster–outlier iterative detection approach to multi-dimensional data analysis. Knowledge and information systems, 2011, 28(3): 709-733.

[10] Mehta B. Lies and propaganda: detecting spam users in collaborative filtering.[C]// International Conference on Intelligent User Interfaces. ACM, 2007.

[11] Chakraborty P, Karform a S. Detection of Profile-injection Attacks in Recommender Systems Using Outlier Analysis[C]//2013.

[12] Chung C Y, Hsu P Y, Huang S H. $\beta P\beta$ math Container Loading Mathjax: A novel approach to filter out malicious rating profiles from recommender systems[J]. Decision Support Systems, 2013, 55(1):314-325.

[13] Liu F T, Kai M T, Zhou Z H. Isolation Forest[C]// Eighth IEEE International Conference on Data Mining. 2009.

[14] Koren Y, Bell R, Volinsky C. Matrix Factorization Techniques for Recommender Systems[J]. Computer, 2009, 42(8):30-37.

[15] Ketkar N. Stochastic Gradient Descent[J]. Optimization, 2014.

[16] Information on https://grouplens.org/datasets/movielens/