Enhancing Matrix Factorization Based Recommender Systems Using Co-purchase Networks

Walid Chaabene WALIDCH@VT.EDU
Virginia Tech
Blacksburg, VA 24061

Sirui Yao YSIRUI@VT.EDU
Virginia Tech
Blacksburg, VA 24061

Abstract

As E-commerce became increasingly popular in the latest decade, customers nowadays are overwhelmed by choices. Therefore, as tools that can provide suggestions to users according to their requirement, recommendation services have become a salient part of E-commerce websites. Presently, one of the most popular and most researched techniques of the recommendation systems is collaborative filtering. Algorithms like Matrix Factorization has been proved to be effective. However, this algorithm, like most collaborative filtering algorithms, is constantly hindered by two major drawbacks. On one hand, it faces the so called cold start problem as it fails when no usage data is available. And on the other hand, data sparsity limits its predictive performance. To work around these issues, we propose a hybrid model where we use co-purchase networks as a secondary source of information to increase the accuracy of the matrix factorization method. Such networks represent products as nodes and edges as a mean to capture a high likelihood of items being purchased together. This new approach is tested on Amazon and proved to outperform the classical matrix factorization approach.

Introduction

The sheer amount of information and data generated by users behavior on the internet has proliferated exponentially over the past few years, leading to an information overload. Successful online services have exploited this valuable data to create insights and uncover relevant patterns to adapt and personalize their services.
Companies like Netflix, Pandora and Amazon owe a part of their success and popularity to their efficient recommender systems (RS) (Langley, 2011). Recommender systems offer a high added value to online services. Their goal is to provide suggestions to users according to their profile. They learn hidden patterns of users behaviors and relations between offered items to help individuals identify the content of interest from a potentially overwhelming set of choices. The main issue is to scrape through the huge pile of data to mine what the user is actually looking for.
As this remains an open challenge for data science research, many approaches to design efficient recommender systems have been introduced in the recent years. One of the most popular and cost effective approach is matrix factorization (Koren et al., 2009). The main idea of this approach is to model ratings as dot products between user vectors and item vectors in a common latent space. The approach rely on existing data to learn those vectors and then uses them to predict missing ratings.
One of the major drawbacks of this approach is that it does not rely on any source of information other than existing ratings. This limits the predictive performance of the method as it can lead to over-fitting in highly sparse matrices.
We propose an modified version of matrix factorization where we exploit information from item co-purchase networks to reduce the effect of sparsity.
In the first section of this paper we present the different types of recommender systems. The matrix factorization method is presented in the second section. T

Virginia Tech CSx824/ECEx424 final project report. Copyright 2016 by the author(s).

1. Methods for building recommender systems

In this section we present the different types of recommender systems. We focus on three popular categories: Content-Base Filtering, Collaborative Filtering and Hybrid Recommendation Approaches (Thorat et al., 2015).

1.1. Content based filtering

Content-based recommender systems work with profiles of users that are created at the beginning. This kind of systems use information from user profiles and tastes. When creating a profile, recommender systems make a survey, to get initial information about a user in order to avoid the cold start problem. Based on the item that a given user rated positively, the system recommends the mostly similar to the user.

1.2. Collaborative filtering

Collaborative filtering is one of the most researched techniques of recommender systems. Such systems tend to cluster users in communities, where each community share almost the same taste. A user gets recommendations to items that have been positively rated by users in his neighborhood.

1.3. Hybrid methods

These techniques combine different techniques of collaborative approaches and content based approaches. It has been proven that using hybrid approaches can help avoid avoid some limitations and problems like the cold-start problems.

2. Matrix factorization techniques for recommender systems

Matrix factorization is a collaborative filtering method that became popular after the team that proposed it won the Netflix Prize competition in 2009. The key idea of this approach is to model ratings as dot products between user vectors and item vectors in a common latent space. The approach rely on existing data to learn those vectors and then uses them to predict missing ratings. Authors of (Koren et al., 2009) refer to two learning approaches. The first is the stochastic gradient descent which consist in minimizing the prediction's squared error over a large number of iterations. At each iteration one existing rating is randomly picked and used to update the corresponding user and item latent vectors. The second is alternating least squares. This is a stochastic approach as well where we pick one random rating and update the corresponding item and user vectors. The only difference is that for each iteration, we fix one vector and perform least square optimization on the other before rotating both vectors and repeating the same procedure.
Authors also present various ways of improving the basic model, such as adding biases, incorporating additional input sources, considering temporal dynamics and treating inputs with varying confidence levels. Although, these some of these improvement come at a considerable cost on both the infrastructure and the computation levels.

2.1. Basic form

We are given a rating matrix R that contains existing ratings of items by users. R is often highly sparse, where missing ratings are coded as 0. Matrix factorization models map both users and items to a joint latent factor space of dimensionality f, such that ratings are modeled as inner products in that space. Accordingly, each item i is associated with a vector q_i, and each user u is associated with a vector p_u. r_{ui}, the rating that user u gives to item i, is modeled as follows,

$$\hat{r}_{ui} = q^T p \tag{1}$$

For a given item i, the elements of q_i are hidden attributes that are not physically interpretable, however they are created in the as a result of item-user interactions. For a given user u, the elements of p_u measure the extent of interest the user has in items that are high on the corresponding hidden attributes (Koren et al., 2009). The major challenge is computing the mapping of each item and user to factor vectors q, p. After the recommender system completes this mapping, it can easily estimate the rating a user will give to any item by calculating the inner product

2.2. Biases

Considering the difference between different users or different items, it would be insufficient to explain the full rating value only by an interaction of the form $q^T p$. Instead, we try to identify the portion of these values that individual user or item biases can explain. Different users have different rating criteria, some of them tend to give higher ratings while some others tend to give lower. Some good products tend to get higher ratings while others get lower. So we use a bias term to capture this overall deviation of rating from the average.

$$\hat{r_{ui}} = p_u^T q_i + b_{ui} \tag{2}$$
$$b_{ui} = \mu + b_u + b_i \tag{3}$$

The overall average rating is denoted by μ; the parameters b_u and b_i indicate the observed deviations of user u and item i, respectively, from the average.

Correspondingly, we change the objective function and add a regularization quantity that is tuned by λ to control the complexity of the desired model.

$$F = \sum_{(u,i)\in S} (r_{ui} - (b_{ui} + p_u^T q_i))^2 + \lambda(||p_u||^2 + ||q_i||^2) \tag{4}$$

2.3. Confidence

For a given rating 'Votes' represent the number of users involved in the voting and 'Helpful' represents the number of people who voted helpful. Thus the number of 'helpful' is always less than or equal to the number of votes. We use this information to calculate the confidence in each rating. To account for ratings with no votes, we apply smoothing as follows.

$$c_{ui} = \frac{\#helpful(r_{ui}) + \alpha}{\#votes(r_{ui}) + 2\alpha} \tag{5}$$

Where α is a smoothing parameter.

Using the confidence matrix C, we correspondingly modify the objective function by adding c_{ui} for each given r_{ui}, so that we give more credit to the ratings that have higher confidence.

$$F = \sum_{(u,i)\in S} c_{ui}(r_{ui} - (b_{ui} + p_u^T q_i))^2 + \lambda(||p_u||^2 + ||q_i||^2) \tag{6}$$

3. Co-purchase networks

E-commerce companies, such as Amazon, keep track of their daily sales. One interesting insight to capture in such data is the list of items that are frequently purchased together by users.

3.1. Building a co-purchase network

Using this data a co-purchase network captured by an adjacency matrix A can be created.This network is an undirected graph that represents items as nodes. Edges capture the fact that items are frequently purchased together, i.e. if item i is linked to item j in the network then i and j are frequently purchased together. Thus the inputs of A are as follows,

$$a_{ij} = \begin{cases} 1 & \text{if } i,j \text{ are frequently purchased together} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

3.2. Mining the co-purchase network

Based on the data available in the co-purchase network we would like to infer the probability that a random user u would rate items i and j the same way. We assume that this probability depends on how close or equivalently, how well connected, i and j are to each other in the network. To this end we define a connectivity matrix M as described in (Michele Benzi, 2013):

$$M = \exp(A) = \sum_{k=0}^{\infty} \frac{A^k}{k!} \tag{8}$$

Where $M_{i,j}$ describes how well i and j are connected to each other.
The entry (i,j) in each $\frac{A^k}{k!}$ is the number of walks between i and j of length k penalized by the $\frac{1}{k!}$. This means that m_{ij} is the number of all possible walks between those two nodes were longer walks are penalized by a greater quantity.
M is row-wise normalized in order to transform it into a stochastic matrix where $m_{i,j}$ indicates the probability that a random user u would give the same rating to i and j.

4. Optimizaion problem

Using the information obtained from the co-purchase network, we can turn the objective function into:

$$F = \sum_{u,i:r_{u,i}\neq 0} \Big[c_{ui}(r_{ui} - b_{ui} - q_i^T p_u)^2 + \sum_{u,j:r_{u,j}=0} \left[m_{ij} c_{ui}(R_{ui} - b_{uj} - q_j^T p_u)^2 + \lambda ||q_j||^2 \right] + \lambda(||q_i||^2 + ||p_u||^2) \Big] \tag{9}$$

The computational complexity of the learning algorithm becomes the critical limiting factor when we are dealing with very large datasets. With its small memory footprint, robustness against noise, and rapid learning rates, Stochastic Gradient Descent (SGD) has proved to be well suited to data-intensive machine learning tasks (Bottou, 2011).

Similar to Batch Gradient Descent, the basic idea of Stochastic Gradient Descent is also to compute the gradient of the objective function and update the optimized parameters using a proportional quantity to the opposite direction of the gradient at each iteration until reaching a local minimum point. Instead of using all the dataset to perform every update SGD selects a random subset and

Algorithm 1 Solving the optimization problem

```
Input: Set the gradient step $\gamma$
repeat
    Pick $(u, i, j)$ s.t. $r_{ui} \neq 0$ and $r_{uj} = 0$
    Compute $\nabla_{q_i} F$, $\nabla_{q_j} F$ and $\nabla_{p_u} F$ (eq. 10)
    Update $q_i$, $q_j$ and $p_u$ (eq. 11)
until convergence
```

computes an empirical gradient of the objective function. This strategy hugely accelerates the learning process. With enough iterations, SGD approximate the optimal solution accurately and yields a reliable output in a much faster than Batch Gradient Descent.

In our case we select a random triplet (u, i, j) at each iteration such that $r_{ui} \neq 0$ and $r_{uj} = 0$.

The stochastic gradient update is performed as follows:

$$\begin{aligned} \nabla_{q_i} F &= -c_{ui}(r_{ui} - b_{ui} - q_i^T P_u)P_u + \lambda q_i \\ \nabla_{p_u} F &= -c_{ui}(r_{ui} - b_{ui} - q_i^T P_u)q_i \\ &\quad - m_{ij}c_{ui}(r_{ui} - b_{uj} - q_j^T P_u)q_j + \lambda p_u \\ \nabla_{q_j} F &= -m_{ij}c_{ui}(r_{ui} - b_{uj} - q_j^T P_u)p_u + \lambda q_j \end{aligned} \tag{10}$$

Each latent vector is updated as follows:

$$\begin{aligned} q_i &= q_i - \gamma \nabla_{q_i} F \\ p_u &= p_u - \gamma \nabla_{p_u} F \\ q_j &= q_j - \gamma \nabla_{q_j} F \end{aligned} \tag{11}$$

Where γ is the gradient step. Algorithm 1 summarizes the steps of the optimization problem.

5. Experiments

We tested the classical matrix factorization method on a publicly available Amazon data. Amazon bases its recommendation system on item to item collaboration filtering (Linden et al., 2003). Matrix factorization is more popular in streaming services like WebTV and WebRadios. We would like to see how our approach performs on the Amazon data. Fist we tested the method's ability to recover existing ratings. After validating that, we tested the method's ability at predicting missing ratings. Finally we tested the proposed method and compared it to the classical one.

5.1. Data Collection and pre-processing

5.1.1. Data description

The Amazon dataset was downloaded from the Stanfor Networ Analysis Project (SNAP) at (http://snap.stanford.edu/).
The dataset file is structured as follows:

- Id: Product id
- ASIN: Amazon Standard Identification Number title: Name/title of the product
- Group: Product group (Book, DVD, Video or Music)
- sales rank: Amazon Sales rank
- similar: ASINs of co-purchased products (people who buy X also buy Y)
- categories: Location in product category hierarchy to which the product belongs (separated by category id)
- reviews: Product review information: time, user id, rating, total number of votes on the review, total number of helpfulness votes (how many people found the review to be helpful)

5.1.2. Constructing the rating and confidence matrices

Numerical IDs were created for both items and users. We kept mapping vectors to keep track of the original IDs.

To build the matrices we first generated a five-column list L containing Product Id, Customer Id, Rating, Number of votes and Number of 'Helpful' endorsements. Every column has the same length. Using the first two columns of L as indices and the third one as inputs, a sparse matrix R was created. We create confidence coefficients as shown in section 3.3. Similar to to R we use the first two columns of L as indices and use the computed confidence coeficients as inputs to for C.

The original R has a dimension of 1555170×402724, and contains 7593244 rating records. It has a density of:

$$density = \frac{7593244}{1555170 \times 402724} = 0.00001212 \tag{12}$$

Running the method on such a large and highly sparse rating matrix will take a long time. Therefore, we select the most popular products and most active users.

6. Testing the classical matrix factorization method

We test the classical approach on a small subset of the rating matrix R containing 155 users and 2552 items.

6.1. Testing ratings reconstruction

We are first interested in seeing how well can this method recover existing ratings. To do so we define the bias error. We define $I = \{(u,i) : r_{ui} \neq 0\}$.

$$BiasError = \frac{1}{|I|} \sum_{(i,j)\in I} (r_{ij} - \hat{r}_{ij})^2 \tag{13}$$

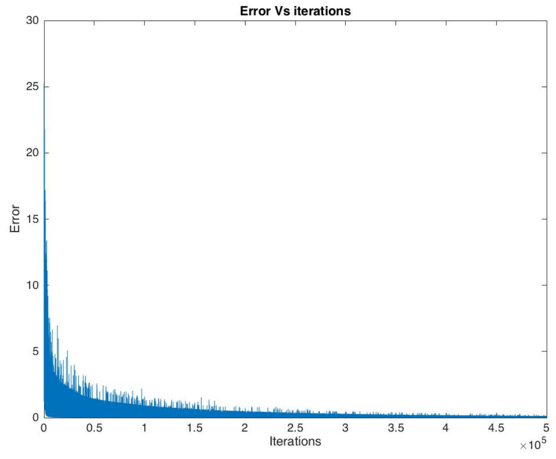

Figure 1. Experiment Results (No induced sparsity)

Figure 2 shows that the error decays to zero with the gradient iterations. At the end of the optimization problem we obtain:

$$BiasError = 0$$

This means that the classical method is able to recover existent ratings.

6.2. Testing ratings prediction

To test the method's predictive abilities, we first pick a random subset S of I. We then create a new matrix $\tilde{R}$ such that:

$$\tilde{R}_{ui} = \begin{cases} R_{ui} & (u,i) \notin S \\ 0 & otherwise \end{cases} \tag{14}$$

This means that we induced further sparsity in R and assume that we are missing some existing ratings. We train our model using $\tilde{R}$. Using the resulting $\hat{R}$, we define the prediction error as:

$$PredictionError = \frac{1}{|S|} \sum_{(i,j)\in S} (r_{ij} - \hat{r}_{ij})^2 \tag{15}$$

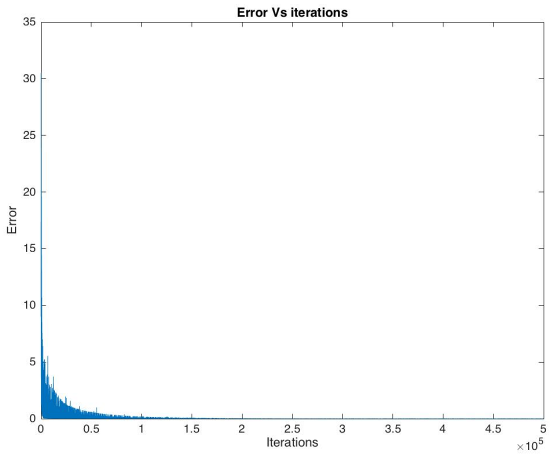

Figure 2. Experiment Results (Induced sparsity)

Figure 2 shows that the error decays to zero with gradient iterations. At the end of the optimization step we obtain:

$$PredictionError = 0$$

Hence the method has perfect predictive abilities on this small test dataset.

7. Testing the proposed method

Figure 3 shows the error trend versus the gradient iterations. We see that the error is decreasing. For this experiment we used a large rating matrix R capturing 12081 users and 46655 items. The use of the stochastic gradient approach was handy since the time complexity from an iteration to another remains the same. However we need to run the optimization program for a high number of iterations.

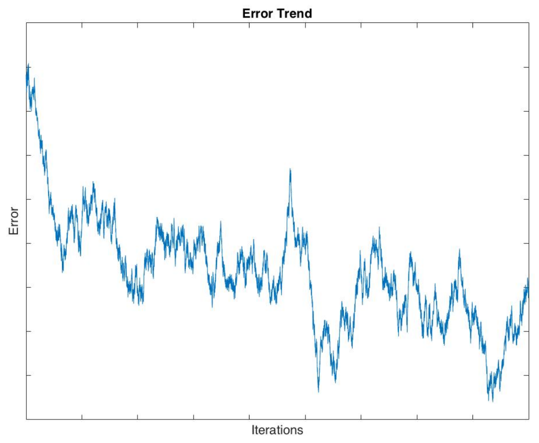

Figure 3. Experiment Results (proposed method)

The results obtained are satisfactory in the sense that the incorporated information from the network is not introducing noise but actual useful patterns. This conclusion is backed by the decaying trend of the error from an iteration to another.

Conclusion

In this work we proposed a modified version of matrix factorization where we incorporated information from Copurchase networks. The method is based on inferring a probability that computes the likelihood of two items getting the same rating from a random user. We proposed a modified objective function and stochastic gradient descend method to optimize. As shown in our result the classical version of matrix factorization is a robust and fast method. However we believe that the proposed method will have better performance since it incorporates the use of a rich source of information that is captured using copurchases networks. More data can be captured in the copurchase network as nodes contain rich local attributes that can be exploited to infer more accurate similarity probabilities in the future.

References

Bottou, Lon. Large-scale machine learning with stochastic gradient descent. *Computer*, 2011.

Koren, Yehuda, Bell, Robert, and Volinsky, Chris. Matrix factorization techniques for recommender systems. *Computer*, 2009.

Langley, P. Recommender systems handbook. In Francesco Ricci, Lior Rokach, Bracha Shapira Paul B. Kantor (ed.), *Springer*, pp. 1–29, 2011.

Linden, Greg, Smith, Brent, and York, Jeremy. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

Michele Benzi, Christine Klymk. Total communicability as a centrality measure. In *Journal of Complex Networks*, 2013.

Thorat, Poonam B, Goudar, RM, and Barve, Sunita. Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, 110(4), 2015.