CME 323: Distributed Algorithms and Optimization, Spring 2015
`http://stanford.edu/~rezab/dao`.
Instructor: Reza Zadeh, Databricks and Stanford.

Lecture 14, 5/26/2015. Scribed by Dinesh Bharadia .

Introduction

This note summarizes the paper [1], which presents an algorithm for recommendation systems used by most web-pages like amazon, Netflix, Alibaba, etc. This paper focuses on Netflix problem to demonstrate the improvements of the algorithm. Before we describe the algorithm, let's understand what is the problem?

The problem is for a collection I of items, and a collection S of users, recommend an item from I to user $i \in S$, which it would likely rate better.

There are two basic approaches to solving this problem. The first approach popularly known as content-based. This approach uses the past choices of user i and not on choices of other users, but it uses information about the items to recommend user new items. In this approach, the hard part is the right way to quantify information about items (one simple mechanism is to use cross-correlation in the items to predict the new items). The second approach to solving this problem is collaborative filtering. This approach uses the ratings of users $j \in S$ for all the items, and no information about the items, to recommend user i new items. This is purely based on collaboration between users. The key idea is there is sparsity in groups or topics, as compared to the number of users and or the number of items. For example, a certain set of users say C like "electronics", meaning have high rating for electronics , now a certain set of items say B belongs to "electronics". It's very likely that users of C would like items in B more than the rest of the community. The paper [1] summarized presents an algorithm which achieves collaborative filtering(CF) which limits the over-fitting to data.

To state this formally, say there is matrix R, whose entries R_{ij} represents rating of user i for an item j. The problem is that all the entries of R are not known. We would like to complete matrix R based on the entries known, say $\tilde{R}$ is the output of collaborative filtering which is best estimate of R under RMSE error.

We define performance based on minimizing RMSE error, which is defined as following,

$$RMSE = \sum_{i,j} \frac{1}{k} ||R_{ij} - \tilde{R}_{ij}||_2 \tag{1}$$

where k is the number of known values of R. Cross-validation strategy is used to see the performance, generally 20 percent of data is left out of training and then predicting RMSE for that 20 percent.

Notation: $\tilde{R}$ is the output of collaborative filtering(CF), further cardinality of users $|S| = m$,

and items $|I| = n$, X is $m \times r$ and Y is $n \times r$. This CF output matrix is defined as following,

$$\tilde{R}^{m \times n} = X * Y' \tag{2}$$

$$\tilde{R}_{ij} = x_i^T * y_j \tag{3}$$

Where x_i, y_j are $r \times 1$ and $r \times 1$ vectors. This predicted $\tilde{R}$, is reduction of ratings to low rank r, matrix product. The idea is to project R matrix, to $X * Y'$, where X is $m \times r$ and Y is $n \times r$. Here $r << m, n$, which represents the feature sets, like in electronics is one feature set. **Why should this work with low feature set ?** The intuition for this is best explained by the Fig. 1, where the columns of matrix X represents the affinity towards the a particular topic or feature set, for example, "electronics" and rows of Y' represents the belongingness (affinity) of items to the electronics feature. The number of feature set or topics is much lower than the users and items, for example, genres of movies, classification of items into categories.

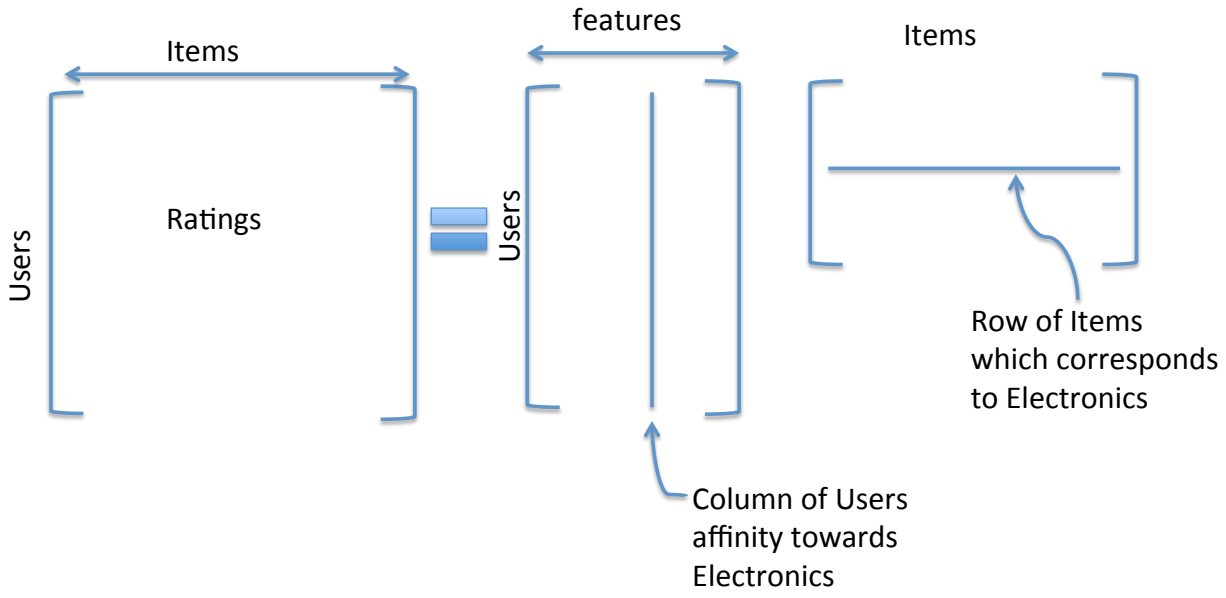

Figure 1: On the left-hand side we have ratings $\tilde{R}$ matrix, which has users as columns and items as rows. This example shows that a small size of feature set projection on both user and items space. This technique clearly shows the example of users who like electronics has higher weight in the column and items which are popular within electronics would higher weight on the rows of Y.

This completes the basic description of CF idea for solving the problem. We will next examine the algorithm proposed in the paper and other choices.

Algorithm

In this section, we will present algorithm described by the paper and also easier algorithm. Basically, we are trying to get a low-rank approximation of the matrix R. The simplest idea that comes to

mind is use SVD to get low-rank approximation, Let's say that SVD is represented by,

$$R = U * \Sigma * V^T = \sum_{i=i}^{rankR} u_i \sigma_i v_i^T \tag{4}$$

The r rank approximation of R matrix is achieved by following take only r singular values into account. These correspond to features, or topics.

$$\tilde{R} = U_r * \Sigma_r * V_r^T = \sum_{i=i}^{r} u_i \sigma_i v_i^T \tag{5}$$

$$\tilde{R} = \underbrace{(U_r * \Sigma_r^{\frac{1}{2}})}_{X} * \underbrace{(\Sigma_r^{\frac{1}{2}} * V_r^T)}_{Y'} \tag{6}$$

Issues with SVD r rank approximation: Intuitively, SVD makes X and Y orthogonal matrices, which is too strong and is often not true in practice. To see this, consider following example columns of U are the affinity of users for a topic, now two column being orthogonal means that two topics are mutually exclusive which may not be true. For example, kitchen supplies (Microwave, Coffee machine) and electronics are two topics/features, which are not mutually exclusive.

Second, SVD overfits to the available data, hence end up not doing well in cross-validation. This paper proposes a technique which can deal with these two issues. Let's formulate what we ideally would like: No orthogonality imposed, no overfitting.

Algorithm in [1]: The idea is simple, let's write the problem as a least squares, with no orthogonality constraint. Hence X or Y need not be an orthogonal matrix. We can add rank constraint to it, which would make problem hard, hence we simplify the problem[1].

$$minimize\ ||R - X * Y'||^2, \tag{7}$$

$$X \in \mathbf{R}^{m \times r}, Y \in \mathbf{R}^{n \times r}, \tag{8}$$

This still doesn't solve the problem of overfitting, hence let's apply Tikhonov regularization in the simplest form[2].

$$minimize\ (||R - X * Y'||^2 + ||\Gamma_X X||^2 + ||\Gamma_Y Y||^2) \tag{9}$$

$$X \in \mathbf{R}^{m \times r}, Y \in \mathbf{R}^{n \times r}, \Gamma_X \in \mathbf{R}^{m \times m}, \Gamma_Y \in \mathbf{R}^{n \times n}, \tag{10}$$

$$\tag{11}$$

To simplify we use,

$$minimize\ (||R - X * Y'||^2 + \lambda * (\sum_{i=1}^{m} n_{u_i} ||x_i||^2 + \sum_{j=1}^{n} n_{r_j} ||y_j||^2)) \tag{12}$$

$$x_i \in \mathbf{R}^{1 \times r}, y_j \in \mathbf{R}^{1 \times r}, \lambda > 0 \tag{13}$$

[1] This doesn't matter since one should iterate over r to find the right number of features.

[2] The more general form of Tikhonov regularization is use a matrix. We will use fixed value in this case.

where n_{u_i} and n_{r_j} denote the number of ratings of user i and movie j respectively.

Now that we have addressed the two issues with SVD and we have an optimization problem to solve. However, this problem is not easy to solve. We apply a simple trick, solve for X first and then solve for Y and keep on doing this iteratively until we meet the objective. Each of these problems is a simple least squares problem. Let's take a look, Assume we know Y, solve for X,

$$minimize(||R - X * Y'||^2 + \lambda(\sum_{i=1}^{m} n_{u_i}||x_i||^2)) \tag{14}$$

$$x_i \in \mathbf{R}^{1\times r}, y_j \in \mathbf{R}^{1\times r}, \lambda > 0 \tag{15}$$

This is simple least squares problem, would lead to solution for X. The solution can be referred in paper is following,

$$x_i = A^{-1}V_j, \tag{16}$$

$$A = Y_{I_i^M} Y_{I_i^M}^T + \lambda n_{r_j} E, \tag{17}$$

$$V_j = Y_{I_j^M} R(I_i^M, j) \tag{18}$$

E is identity matrix, $Y_{I_i^M}$ presents the submatrix of Y where columns $i \in I_i^M$ are selected, $R(I_i^M, j)$ is the column vector where rows $i \in I_i^M$ of the jth column of R is taken. Use this solution to solve for Y and so on.

To summarize algorithm from the paper [1],
Step1: Initialize X, Y random matrix
Step2: While RMSE is reducing
Step3: Fix Y and solve X for least squares problem.
Step4: Fix X and solve Y for least squares problem.
end while

Issues with this algorithm: Need to tweak with λ to minimize overfitting. Choosing an initial point for X and Y, determines the number of iteration it takes to converge.

Improvement over the algorithm: Initial point for X and Y can be chosen using SVD. This would significantly improve the convergence time.

Parallel Algorithm

The equation 16 is intuitively suggested how to design a distributed algorithm. As long as a node has either one copy of either row of R or column of R, it can do one update as seen in equation 16. However to compute an update on X or Y, the node needs Y or X respectively. For example, to update Y, we require a copy of X, local to each node. We use rating data distributed by movies (columns). Divide to each node equal number of of movies. The node that stores the ratings of movie j updates the corresponding column of Y, which is the feature vector of the movie. These columns are collected later, and we have a copy of new Y.

The gather operation (shuffle) here would consume here either $n*r$ or $m*r$, which is very small as r is very small.

Performance

With 1000 features, the algorithm performs RMSE error of .8985, the number of iteration is proportional to the order of feature vectors. This improvement corresponds to 6 % improvement over existing algorithms.

References

[1] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. 5034:337–348, 2008.