Statistical Properties of Alternating Least Squares Estimators of a Collaborative Filtering Model

Michael Curtis[1], Julia Baum[2], Cynthia Cook[3], Joshua Edgerton[4], and Scott Rabidoux[5]
Graduate assistant: Andrew M. Raim[1] Faculty mentor: Nagaraj K. Neerchal[1]
Client: Robert M. Bell[6]
University of Maryland, Baltimore County[1],
Worcester Polytechnic Institute[2] Catawba College[3] Cornell University[4] Wake Forest University[5]
AT&T Labs - Research[6]

Abstract

Recommender systems are emerging as important tools for improving customer satisfaction by mathematically predicting user preferences. Several major corporations including Amazon.com and Pandora use these types of systems to suggest additional options based on current or recent purchases. Netflix uses a recommender system to provide its customers with suggestions for movies that they may like, which are based on their previous ratings. In 2006, Netflix released a large data set to the public and offered one million dollars for significant improvements on their system. In 2009, BellKor's Pragmatic Chaos, a team of seven, won the prize by combining individual methods. Dr. Robert Bell, with whom we collaborated, was a member of the winning team and provided us with the data set used in this project. The database provided is a sparse matrix with rows of users and columns of movies. The entries of the matrix are ratings given to movies by certain users. The objective is to obtain a model that predicts future ratings a user might give for a specific movie. This model is known as a collaborative filtering model, which encompasses the average movie rating (mu), the rating bias of the user (b), the overall popularity of a movie (a), and the interaction between user preferences (p) and movie characteristics (q). The method of Alternating Least Squares was used to estimate each parameter of this non-linear regression model. In this paper, we take a look at the accuracy of our algorithm using simulated data from a known set of parameter values.

1 Introduction

Collaborative filtering systems have had growing interest recently due to the Netflix million dollar challenge to improve its existing algorithm for recommending movies. Through the contest, Netflix hoped to enhance the method with which they recommend movies to their users. This method would be based on prior movies the users have rated. The goal of the contest was to predict, as accurately as possible, what any particular user would rate a movie. For instance, this could be interpreted as a standard regression problem; fitting a model which relates the characteristics of movies with those preferred by users to develop an estimated rating of the user. However, a closer look into the problem reveals that it is more challenging. The Netflix data does not explicitly contain any information about movie or user characteristics, except for ratings that users have given to movies. The ratings data are also sparse, i.e. some users have rated many movies, others hardly any, and no user has rated a significant fraction let alone all of the movies. Collaborative filtering is a technique that makes use of the sparse data and obtains good estimates for the parameters of a regression model. The actual Netflix competition data is of the order of billions of entries and very sparse, and therefore presents a significant computational problem. Although the Netflix challenge has already been conquered, we believe that some of the iterative methods used for collaborative filtering can be modified and parallelized to become significantly more efficient.

In section 4 we describe the Netflix data, and a smaller testing dataset which we used to develop our implementation. We also describe the data structures we used to manipulate the data in code. We also describe the computing cluster available at UMBC which we used to run our experiments. In section 2 we describe the regression model for users' movie preferences. In section 3 we describe the Alternating Least Squares algorithm we developed and in section 6, we discuss our conclusions and opportunities for future work.

2 The Model

The model is a variation of a linear regression model using several parameters to predict each users rating for any particular movie. We denote the observed ratings as r_{ui}, the uth user's rating of the ith movie, for u in $\{1, \ldots, U\}$ and i in $\{1, \ldots, I\}$. The r_{ui} are allowed to be any real number. A higher rating represents a more favorable score for a movie, as usual. We also denote R for the $U \times I$ matrix of ratings, and $\hat{r}_{ui}$ for the predicted ratings.

The aim of the model is to have the smallest mean squared error between r and $\hat{r}$. The full model comes from Bell and Koren's 2009 paper [1] on their prize-winning methods for the Netflix Problem. The original base model is a latent factor model breaking the large matrix of ratings into three smaller ones, one of users, one of movies, and one of averages. The parameter a_i models the difference between a movie's rating and the average movie rating, for $i = 1, \ldots, I$. Similarly, b_u represents the difference between a user's rating and the average user's rating for a particular movie for $u = 1, \ldots, U$. The global average rating of all movies by all users is μ. A more advanced model takes into account the characteristic vectors p_u and q_i each of length d. Each element of p_u gives a rating of user's affinity for a certain characteristic, and the corresponding element in q_i quantifies the amount of this characteristic in the movie.

Consider a simple case scenario where a recommender system is to determine whether a given user $u = 1$ will enjoy movie $i = 1$. The user has rated several movies in the past generating a characteristic vector p_1, where $d = 3$. Suppose the vector measures comedy, action, and horror on a scale from -5 to 5. If u_1 likes comedy, loves action, but hates horror, his characteristic vector may be in the form $p_1 = (3.897, 4.875, -2.456)$. The characteristic vector for $i = 1$ is also of size $d = 3$ and is measured on comedy, action, and horror; suppose it is $q_1 = (4.132, 3.978, -4.978)$. By taking the dot product of the two characteristic vectors, the recommender system can measure if the movie would be a good fit. In our example, $p_1' q_1 = 47.61$, which means that movie $i = 1$ may be a good choice for user $u = 1$. Note that it is up to the recommender system to determine the d characteristics through the data, and that they may or may not have a simple interpretation as in this example. The full model cited in [1] is as follows:

$$\hat{r}_{ui} = \mu + a_i + b_u + p_u' q_i \tag{2.1}$$

The model appears fairly straightforward, but fitting it requires special algorithms such as the one discussed later. Our first task is to find an expression to minimize, with respect to the unknown parameters, which will result in reasonable parameter estimates. Let

$$\kappa = \{(u, i) : \text{there is a rating } r_{ui} \text{ in the dataset}\},$$

for whichever dataset is being considered. To fit the parameters of the model (2.1) we define the objective function as

$$\sum_{(u,i)\in\kappa} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left(\sum_{u=1}^{U} \|p_u\|^2 + \sum_{i=1}^{I} \|q_i\|^2 + \sum_{u=1}^{U} b_u^2 + \sum_{i=1}^{I} a_i^2 \right). \tag{2.2}$$

The left side of the objective represents the sum-squared error between the true ratings and the fitted ones. The portion with coefficient λ penalizes parameters with large magnitude.

3 Fitting the Model

Note that the problem presented in (2.1) and (2.2) is non-linear because of the presence of $p_u' q_i$ term. Also, the objective function is not a simple sum of squared errors, because it includes a penalty component. A further complication is that we have a minute fraction, perhaps in the order of one percent, of all the possible user/movie combinations available in the dataset. We use a method known as Alternating Least Squares (ALS) to fit this model for this data. As we shall see in the next section, ALS is effective in fitting this model, and also amenable to simplifications to increase computational efficiency.

The ALS method is a learning algorithm that finds a line to best fit given data by estimating unknown parameters for a collaborative filtering model of the form

$$y = X\beta + \epsilon. \tag{3.1}$$

In particular, ALS is used with non-linear regression models when solving for two dependent variables. The method fixes one of the dependent parameters, while solving for the other, creating a linear least squares problem. The method alternates solving for one parameter than the other. The least squares method obtains the best linear unbiased estimator

$$\hat{\beta} = \arg\min_{\beta} \|y - X\beta\|^2, \tag{3.2}$$

where the solution minimizes the sum of squared residuals [3]. The formulas for linear least squares fitting were independently derived by Gauss and Legendre [6]. The Gauss-Markov Theorem (GMT) [5] states that the best linear unbiased estimator for our least squares problem is given by

$$\hat{\beta} = (X'X + \lambda I)^{-1} X'y.$$

Recall our collaborative filtering model from (2.1); note that it is linear in μ, a_i, b_u, and p_u for a fixed q_i. Thus, for a fixed q_i the best linear unbiased estimate of μ, a_i, b_u, and p_u can be determined by the GMT. Similarly, the model is linear in μ, a_i, b_u, and q_i for a fixed p_u. Thus, for a fixed p_u the best linear unbiased estimate of μ, a_i, b_u, and q_i can be determined by the GMT. In either case, to use the GMT we let β be a vector of the unknown parameters, X be a corresponding design matrix containing of fixed quantities, and y be a vector of the ratings. The ALS method is an iterative algorithm that alternates between solving for p_u and q_i and updates the parameters until MSE converges within a given tolerance or the iteration count reaches the maximum iterations. We introduce some notation needed for the ALS algorithm:

- U = total number of users
- I= total number of movies
- K = total number of ratings
- N_u = number of ratings given by user u
- M_i = number of ratings given by item i

We begin the ALS algorithm by filling the characteristic vectors q_i with randomly generated real numbers ranging from -10 to 10. The quantities μ, a_i, b_u, and y_{ui} are initialized by the following equations using the ratings given in the test data set

$$\hat{\mu} = \frac{1}{K} \sum_U \sum_I r_{ui}$$

$$\hat{a}_i = \frac{1}{M_i} \sum_{u \in M_i} r_{ui} - \hat{\mu}$$

$$\hat{b}_u = \frac{1}{N_u} \sum_{i \in N_u} r_{ui} - \hat{\mu} - a_i$$

$$y_{ui} = r_{ui} - \hat{\mu} - \hat{a}_i - \hat{b}_u.$$

As stated above, the variable y_{ui} is created by subtracting a, b, and μ from the ratings. This is done as an attempt to adjust these parameters out of the data in order to simplify our set of regression equations. We then iterate between fixing q_i's and solving for p_u's, and fixing p_u's and solving for q_i's. Our algorithm continues to iterate solving for p_u and q_i on the training data set until each parameter converges or the tolerance and/or iteration maximum is met. We have not yet mentioned how λ is chosen. In practice a cross-validation approach is taken, fitting the model for many values of λ against the training dataset to obtain the best possible MSE on the test dataset.

Notice that the model simplifies when $d = 0$ and $\lambda = 0$. With these values, the p_u's and q_i's drop out, and the objective function (2.2) becomes a simple sum of squared errors. In this case the algorithm should converge in one step, although the fit will most likely not be adequate.

Algorithm 1 : ALS

```
Set up vectors yUsers, yItems, xq, and xp
Initialize $\mu, a_i, b_u, y_{ui}$
  while MSE has not converged and current iteration ¡ max iteration limit
        Create $q_i$ vectors
         Solve for $p_u$ vectors
        Create $p_u$ vectors
         Solve for $q_i$ vectors
        Rescale each $p_u$ and $q_i$ vector
        Update $\mu, a_i, b_u, y_{ui}$
        Find MSE
    End
```

4 The Simulation

While the Netflix data was available to us, it did not come with a set of actual values for our model. In order to verify the model, trials were run using simulated data sets. The data was generated from supplied values of model parameters, according to Table 1. The data set contained four users and four movies, treating the two characteristics "comedy" and "violence" as relevant. We selected four movies at different ends of these spectrums: *MacGruber*, *National Lampoon's Christmas Vacation*, *Citizen Kane*, and *Platoon*. For the purpose of this simulation we consider all four movies to be of the same overall quality, and let $a_1 = a_2 = a_3 = a_4 = 0.1$. *MacGruber* and *Christmas Vacation* are comedies, so we set $p_{11} = p_{21} = 1$ and use $p_{31} = p_{41} = -1$ for the others. For the violence attribute, *MacGruber* and *Platoon* should rate higher and so we set $p_{12} = p_{42} = 1$, and use $p_{22} = p_{32} = -1$ for the others. We also consider four users whose preferences lie in the same areas of the comedy/violence spectrum, and whose rating biases are all the same (so that $b_1 = b_2 = b_3 = b_4 = 0.1$). We select $\mu = 2.5$ as the overall average rating. From this scenario, the rating data is generated using

$$r_{ui} = \mu + a_i + b_u + p'_u q_i + \varepsilon_{ui} \tag{4.1}$$

for $i = 1, \ldots, 4$ and $u = 1, \ldots, 4$ where $\varepsilon_{ui} \sim \text{Normal}(0, 0.25^2)$. We selected several (u, i) combinations as missing, so that users have rated only two or three movies.

We considered fitting the collaborative filtering model with ALS using three dimensions for the characteristic vectors, $d = 0, 1, 2$ (where $d = 2$ is the true dimension). For each dimension, we tried seven settings of $\lambda = 0, 0.1, 0.25, 0.5, 1, 2.5, 5$. We refer to a particular (d, λ) combination as a "trial". Within each trial, 1000 randomly generated datasets are fitted with ALS; each fitting is referred to as a "subtrial". In each trial, the objective function, root mean squared error (RMSE) and mean squared error (MSE) were calculated for each parameter over 1000 subtrials. We expect that over many subtrials, the estimates for a good model will converge to the true parameters.

Movie	a_i	q_{i1}	q_{i2}	Movie Description
1. *MacGruber*	0.01	1	1	violent comedy
2. *National Lampoon's Christmas Vacation*	0.01	1	-1	non-violent comedy
3. *Citizen Kane*	0.01	-1	-1	non-violent serious
4. *Platoon*	0.01	-1	1	violent serious

User	b_u	p_{u1}	p_{u2}	Prefers
User 1	0.01	1	1	violent comedy
User 2	0.01	1	-1	non-violent comedy
User 3	0.01	-1	-1	non-violent serious
User 4	0.01	-1	1	violent serious

Table 1: Parameters used in simulation study

5 Results and Analysis

d	λ	Mean Objective	Mean RMSE	Mean μ
0	0	48.7843330675	1.73096991795	3.01058333333
	0.1	46.7860519979	1.6914636907	3.02332277178
	0.25	44.1894575468	1.64192204618	3.00700052275
	0.5	42.540685789	1.60602735548	3.00280910282
	1	40.0203143852	1.55313419535	3.00219719171
	2.5	36.6101262155	1.48377639945	2.99616445802
	5	35.4367768618	1.46322151755	3.01400997558
1	0	10452.8247662	6.80482261867	1.60749681235
	0.1	83.2385124012	2.1631521211	2.87724708008
	0.25	57.2037276259	1.78858656703	2.90859193791
	0.5	45.3393432814	1.54979883199	2.9354429455
	1	37.0654539134	1.320608357	2.97186741034
	2.5	36.5377653684	1.26168824043	2.98552998725
	5	35.5553635051	1.44833363009	3.02328137997
2	0	2180676.2075	31.9381276206	0.538571380193
	0.1	18.8149231344	0.89733579881	2.97667080541
	0.25	19.4429303548	0.883623869685	3.00881879105
	0.5	21.3728896677	0.877922362362	2.97914871683
	1	24.9975900034	0.892402383518	2.99989496699
	2.5	34.0362271225	1.13010450647	3.01049998491
	5	35.5495743718	1.44785066195	3.02405842536

When d is zero, as in the first set of trials, the average rating is based on a and b and the given rating. The true value in this case for all a's and b's is 0.01. As such, μ is not very likely to fluctuate. Also, remember that the goal of ALS is to minimize the objective function. The lowest objective function result in the $d = 0$ trial set was 35.43. Also for these trials, the ALS function is solving very similar, though randomly-generated, linear equations. Consequentially, the RMSE is quite stable, as well as inaccurate. Remember that RMSE is the square root of the average error.

The second set of trials used p and q vectors of length 1. In this set and the final set, the parameters fluctuate as a result of the p and q vectors, which are given random initial values. The objective function and RMSE also increase with λ, but not uniformly. The change between $\lambda = 0.5$ and $\lambda = 1$ is more apparent than $\lambda = 2.5$ and $\lambda = 5$. These parameters still are not quite accurate, as the objective function is always higher than its $d = 0$ counterpart. This is likely because that p and q are scalar values attempting to approximate a vector.

The final seven trials were ran at the true dimension of p and q, $d = 2$. These trials also reported a better objective function and RMSE for similar λ values than the other trial sets. The only case in which this is not true is $\lambda = 0$, where none of the results are accurate. As usual, objective function values and RMSE increase with λ. Note trial number 2 ($\lambda = 0.1$), which seemingly has the best balance between objective function and μ. It may be possible that this trial is a prime example of "overfitting" parameters to a dataset. At the same time, it is also possible the $\lambda = 0.1$ provides a better general approximation for the regression.

For each trial set, $\lambda = 0$ always reported the worst objective function. This is because of the nature of our objective function. Defined earlier, a quick look at equation 2.2 shows how $\lambda = 0$ cancels out a significant portion of the objective function. Something else to observe is the final trial for each set. At $\lambda = 5$, the accuracy of a and b is extremely great, and the MSE for p and q vectors remains the same throughout. At the same time, the increase in objective function value and RMSE is not too great from the previous trial. Lastly, the MSE of μ, the objective function and RMSE is the same for $\lambda = 5$, despite the change in d.

6 Conclusions

Through close work with Dr. Bell, a member of the winning team and the guidance of our mentors, we built a fairly accurate model. Constructing it completely from scratch made us familiar with the underlying framework, which made this study possible. From the results of these small data set trials, the Alternating Least Squares algorithm accurately approximates the parameters in our model. Using the different dimensions of d also confirmed that using the true dimension is important in fitting the model. For future work, we would like to test with a larger data set, as well as a randomized data set.

Acknowledgements

Within the High Performance Computing Center at UMBC, we have available a system called Tara to run our parallelized code. Completed and installed in 2009, Tara is a 86-node distributed-memory cluster with two quad-core Intel Nehalem processors: one front end node, one management node, two development nodes, and 82 compute nodes. We communicate directly with the front end node, which then sends our task to available processors on the compute nodes. Each node has 24 GB of memory, an InfiniBand interconnect, and 160 terabytes of central storage [2].

References

[1] R. Bell and Y. Koren, "Matrix Factorization Techniques for Recommender Systems," *IEEE International Conference on Data Mining Workshops*, 2009.

[2] `www.umbc.edu/hpcf`, 2010.

[3] Yehuda Koren, Robert Bell and Chris Volinsky, "Matrix Factorization Techniques for Recommender Systems," IEEE Vol. 0018-9162, pp.42-49, 2009.

[4] Robert Bell, "Matrix Factorization for Recommender Systems," presentation at UMBC, 2010.

[5] Jia Li, "Linear, Ridge Regression, and Principal Component Analysis," `www.stat.psu.edu/jiali`.

[6] Eric W. Weisstein, "Least Squares Fitting," From MathWorld–A Wolfram Web Resource. `www.mathworld.wolfram.com/LeastSquaresFitting.html`, 2010.