Solving Matrix Completion Problem via Low Rank and Low Dimensional Model

Banghua Zhu, 2014011218
Electronic Department, Tsinghua University

Abstract
Content filtering and collaborative filtering are two common methods in recommender systems. In the field of collaborative filtering, latent factor models remain an actively researched topic. This model can be formulated as a matrix completion problem. In this project, we focus on the comparison between two matrix completion methods based on low rank model and low dimensional manifold model. Because of the limitation of potential bias in the single test data, instead of trying to show superiority of any of the models, my work focuses on illustrating the analysis of the principles and theories of the two models.

1 Introduction

Recommender systems are important components in contemporary e-commerce platforms, and were popularized by the Netflix challenge. Detailed surveys of this field can be found in [1] [2]. Recommendation algorithms are commonly based on collaborative filtering. Matrix completion is originated from the model-based methods which assume that the entire user-by-item rating matrix is low-rank [3] or low-dimensional [4], and the goal is to predict the missing ratings given the observed ratings.

Matrix completion problem has a widespread application that is not restricted to recommender systems, but also includes haplotype assembly [5] and image blind inpainting [4]. In this report, we focus on the test of recommender system and evaluate our algorithms' performance on a given dataset.

Popular model-based methods such as regularized SVD [3] minimize the sum-of-squares error over all the observed ratings. When evaluating the predictive accuracy of these algorithms, we often divide the whole data set into a training set and a testing set. After obtaining a model on the training set, we evaluate the accuracy of the model's prediction on the testing set in order to see how well it generalizes to unseen data.

2 Algorithms

In recommender system, each element $R(i,j) \in \mathbb{R}^{mn}$ of the rating matrix R represents the rating of movie j by customer i if customer i has watched movie j and is otherwise missing. In reality, the movies one customer has watched are only meager part of the sum, which makes the rating matrix R quite sparse. our goal is to predict the remaining entries in order to make good recommendations to customers on what to watch next, i.e. recover a completed matrix M that gives prediction on the missing position of R.

A traditional but efficient assumption is that the users who show similar ratings for the same movie tend to have similar flavor in terms of movies, and will give similar ratings for other movies. And traditional method interprets this assumption as the low-rank property, thus large numbers of low-rank method has been put forward, such as alternative updating or Iterative Reweighted Algorithms. Recently, another model called low dimensional model is put forward as an alternative to low-rank model. In the following part, we will give brief introduction to the two different methods.

2.1 Low rank model

As stated above, the low rank model can be formulated as:

$$\begin{array}{ll} \min & rank(M) \\ s.t. & P(M) = R \end{array} \tag{1}$$

Here operator P is defined as:

$$P(M_{ij}) = \begin{cases} M_{ij}, & if\ R_{ij}\ is\ observed \\ 0 & else \end{cases} \tag{2}$$

This operator can also be seen as a point wise product with index matrix W, with $W(i,j) = 1$ if the user i has given a rating for movie j and 0 otherwise.

In this paper, we apply two of the low rank methods, alternating least square (ALS) and alternating gradient descent (AGD). By changing the constraint into penalty function, we can get the following target function:

$$\min \quad \|W * (R - M)\|_F^2 + \lambda \, rank(M) \tag{3}$$

In order to force the property of low rank, we introduce latent variables $U \in \mathbb{R}^{mk}$ and $V \in \mathbb{R}^{nk}$ by assuming $M = UV^T$, where $M_{ij} = U(i,:)V(j,:)^T$. And the rank of M can thus be approximated by

$$\min_{U,V} \quad \frac{1}{2}(\|U\|_F^2 + \|V\|_F^2) \tag{4}$$

So finally our target function is in the form of:

$$\min_{U,V} \quad f(U,V) = \left\|W * (R - UV^T)\right\|_F^2 + \frac{\lambda}{2}(\|U\|_F^2 + \|V\|_F^2) \tag{5}$$

When we fix one of U and V, the optimization of other factor degrades into a least square problem. By observation, the cost function can be decoupled as

$$\begin{aligned} f(U,V) &= \textstyle\sum_{i=1}^m f(u_i, V) \\ &= \textstyle\sum_{i=1}^m ((r_i - u_i V^T) W_i (r_i - u_i V^T)^T + \frac{\lambda}{2} u_i u_i^T) + \frac{\lambda}{2} \|V\|_F^2 \end{aligned} \tag{6}$$

Here $r_i = R(i,:)$, $u_i = U(i,:)$, $W_i = diag(W(i,:))$. When V is fixed, for each i, this is a normal least square problem, by solving the equation that the gradient is equal to zero, the optimal value is found:

$$u_i = (V^T W_i V + \lambda I)^{-1} V^T W_i r_i \tag{7}$$

Following similar procedure, we can get the optimal value of v_j when we fix U and apply least square.

$$v_j = (U^T W_j U + \lambda I)^{-1} U^T W_j r_j \tag{8}$$

Note here $r_j = R(:,j)$, $v_j = V(:,j)$, $W_i = diag(W(:,j))$. So We iteratively update U and V according to 7 and 8 until convergence criterion is satisfied.

Since the overall function $f(U,V)$ is quite nonlinear and not convex. Even the existence of a minimum is general not ensured. We need some further assumptions on the sequence from ALS method. We can prove that every accumulation point of U or V is a critical point under some assumptions. One may refer to [6] for more details.

Another more efficient way is alternating gradient descent (AGD). The merit of this method lies in that the gradient of the target function of overall U and V can be derived directly, thus we do not need to update each row of U and each column of V separately. There will be no matrix inverse calculation. the update of U and V follows the procedure as below:

$$U_{new} = U_{old} - \alpha(-2(W * (R - U_{old}V^T))^T V + \lambda U_{old}) \tag{9}$$

$$V_{new} = V_{old} - \alpha(-2(W * (R - UV_{old}^T))^T U + \lambda V_{old}) \tag{10}$$

On the convergence of AGD, we give without proof the following theorem. For proof we refer to [5].

Theorem 2.1. *Let the step size α in (8) be selected as*

$$\alpha = \frac{C \left\| \Delta f(V_t) \right\|_F^2}{\| P(U_t \Delta f(V_t)^T)_F^2 \|}, \tag{11}$$

where $C \in (0, 1)$ is a constant. Then the solution (U^*, V^*) found by AGD is a critical point of f.

Furthermore, a global minimum can be achieved under alternating gradient descent setting. [7] and [8] give detailed analysis on this problem. The basic theorem can be summarized as follows.

Theorem 2.2. *Assume M is μ-incoherent. Suppose the condition number of M is κ. Then, there exists numerical constants $C_0, C_1 > 0$ such that if Ω is uniformly generated at random, and*

$$mp > max\{C_0 \sqrt[3]{\mu^4 k^{14} \kappa^{12} np}, \frac{p_e k^2 \kappa^6}{2C_1^2}\} \tag{12}$$

with probability at least $1 - \frac{1}{m^3}$, the solution (U^*, V^*) found by AGD satisfies

$$\frac{1}{\sqrt{mn}} \| M - U^* {V^*}^\top \|_F \leq C_1 \kappa^2 \sqrt{\frac{p_e k}{2mnp}}. \tag{13}$$

2.2 Low Dimensional Manifold Model

Recently, [4] has proposed a model based on low dimensional manifold. Without loss of generality, we assume that there exists a user manifold, denoted as $\mathcal{M}$, which consists of an infinite number of users. In a user-by-movie rating system with n movies, each user is identified by an n-dimensional vector that consists of the ratings to n items. Thus, the user manifold $\mathcal{M}$ is a sub-manifold embedded in $\mathbb{R}^n$. For the i-th item, we define the rating function $f_i : M \to R$ that maps a user into the rating of this item.

In the Low dimensional manifold model (LDMM), by viewing the point cloud as a regular manifold, the dimension of this manifold is then introduced as a regularization term in an optimization problem. Briefly the low-rank model thinks the small part of the manifold should be a low-dimensional subspace, while the low dimensional model prefers to explain that there just exists a homeomorphism between the small part and the low-dimensional subspace.

The general optimization problem can be formulated as

$$\begin{aligned} \underset{u}{\text{argmin}} \;\; & R(u) \\ s.t. \;\; & ||\Phi u - f||_p \leq c \end{aligned}$$

where $R(u)$ represents the regularization term, i.e. the prior information we can utilize to recover the matrix. And the constraints mean that the impact of noise is bounded by c in a suitable norm $|| \cdot ||_p$.

A similar formulation can be obtained by introducing a multiplier λ and treat the constraints as a penalty term

$$\underset{u}{\text{argmin}} \;\; R(u) + \lambda ||\Phi u - f||_p$$

For the low dimensional manifold model, $R(u) = \dim\mathcal{M}$ and the structure of $\mathcal{M}$ is closely and complexly related to the decision variable u. Given a u, we could construct $\mathcal{M}$ thus get $\dim\mathcal{M}$ in a proper way (up to the definition of the dimension), so the optimization problem makes sense. From a beautiful formula in differential geometry the problem writes

$$\begin{aligned} \underset{u,\alpha_i}{\operatorname{argmin}} \ & \sum_{i=1}^{n} ||\nabla_{\mathcal{M}}\alpha_i||_2^2 + \lambda||\Phi u - f||_p \\ s.t. \ & \alpha_i(\mathcal{P}u(x)) = \mathcal{P}_i(u(x)) \end{aligned}$$

where the operator $\mathcal{P}$ transforms the users rating u (the row of the matrix M) into a point cloud, and $\mathcal{P}u(x)$ stands for the point in R^Nassociated with the patch whose top-left pixel is at x. And α_i is the coordinate function of $\mathcal{M}$.

To solve this optimization problem, we consider a two-term strategy, which optimizes the objective function with a fixed point set manifold first followed by a updating procedure. The kth iteration reads

$$\begin{aligned} \underset{u,\alpha_i}{\operatorname{argmin}} \ & \sum_{i=1}^{n} ||\nabla_{\mathcal{M}^k}\alpha_i||_2^2 + ||\Phi u - f||_p \\ s.t. \ & \alpha_i(\mathcal{P}u^k(x)) = \mathcal{P}_i(u(x)) \end{aligned}$$

and the updating procedure is simple

$$\mathcal{M}^{k+1} = \{(\alpha_1(x), ..., \alpha_d(x)) \mid x \in \mathcal{M}^k\}$$

When the norm is l_0,l^1 and l^2 then ADMM method can be applied to solve this problem, for in this case the objective function enjoys a separate structure and we have a closed form of the proximal operator of $||\cdot||_0, ||\cdot||_1$ and $||\cdot||_2$. For matrix completion problem, we only need to solve this following subproblem:

$$\underset{u}{\operatorname{argmin}} \ ||\nabla_{\mathcal{M}^k} u||_2^2 + \lambda \sum_x |W * (u(x) - v(x))|_2^2$$

where u can stand for any α_i and v is the related point set associated with the noisy matrix.

There are two ways to discretize $\nabla_{\mathcal{M}}$. can be based on point integral method or weighted nonlocal laplacian method, both of which approximate the operator only using the information of the point set and have good approximation order when the function is smooth. The latter has a symmetric form so the linear equation is easier to solve.

A general formula for solving the optimization problem utilizes a bregman iteration as listed below.

1. $u^{k+1} = \operatorname{argmin}_u ||\nabla_{\mathcal{M}} u||_2^2 + \mu||u - w^k + d^k||_2^2$
2. $w^{k+1} = \operatorname{argmin}_w \mu||u^{k+1} - w + d^k||_2^2 + \gamma \sum_x |w(y) - v(y)|$
3. $d^{k+1} = d^k + u^{k+1} - w^{k+1}$

The first step can be solved in a discretized laplacian method. The step2 has a closed form

$$w_i^{k+1} = \begin{cases} u_i^{k+1} + d_i^k - \dfrac{\gamma}{2\mu} \text{ if } u_i^{k+1} + d_i^k - v_i > \dfrac{\gamma}{2\mu} \\ 0 \text{ if } |u_i^{k+1} + d_i^k - v_i| <= \dfrac{\gamma}{2\mu} \\ u_i^{k+1} + d_i^k + \dfrac{\gamma}{2\mu} \text{ if } u_i^{k+1} + d_i^k - v_i < -\dfrac{\gamma}{2\mu} \end{cases}$$

In this implementation, I only use the first step to update u and assign the known value to u in every iteration for simplicity.

3 Implementation and Simulation Details

3.1 Results

We have implemented 3 different algorithms in Matlab for solving matrix completion problem, ALS, AGD and LDMM, and test them on the given dataset. For LDMM, we don't implement the bregman iteration step, but only minimizes its manifold dimension for simplicity. The initial value is set either the outcome of SVD or pure random value. In fact, the difference is not so much as to the two choices of initialization.

There are basically two important hyper-parameters for AGD and ALS, λ and k. Here k means the dimension of latent variables for U and V, i.e. $U \in \mathbb{R}^{mk}$ and $V \in \mathbb{R}^{nk}$. We apply a grid search for all the hyper-parameters of the algorithms and select the parameters that lead to the minimum MSE. Here we choose $k = 2$. A higher k may not lead to a better performance but higher degree of freedom will loose the constraint and make the problem much slower. Basically, a higher λ for ALS and AGD means the low rank property is more dominant. And a higher λ for LDMM means the low dimension property is less dominant.

To test the performance of the three algorithms, we adopt a ten-fold cross validation. We found that for ALS, the minimum mean square error on validation set (testing set) can reach 7.59 when $\lambda = 1.6$. The minimum mean square error can reach 7.95 for AGD when $\lambda = 1.9$. The minimum mean square error can reach 8.91. for LDMM when $\lambda = 0.3$. The overall hyper-parameter, MSE and running time are listed below. Here overall MSE refers to the MSE calculated from all the given data.

Table 1: Outcome summarize of three algorithms

Algorithm	λ	MSE (on testing set)	overall MSE	running time
ALS	1.6	7.5881	6.7915	473.34s
AGD	1.9	7.9453	7.2279	7.53s
LDMM	0.3	8.9074	7.8661	93.07s

3.2 Discussion

As analyzed above, AGD is the fastest algorithm because no matrix inverse is needed. And it happens to have the best performance (which may benefit from its speed because it can run large amounts of iteration to better ensure the convergence.)

One interesting thing is, for ALS, when I feed the outcome of one parameter settings to another λ-based ALS, the MSE goes down. When I iterate λ from 1 to 30 and set the outcome of each iteration as the initial value of next iteration, the MSE can reaches 8.3 (though it will take a much longer time). This is somewhat similar to the idea of gradient boosting. But in my report, I only include the case with random valued (SVD) initial value for single λ.

I'd like to point out that a low MSE for a single dataset may not prove the superiority of any of these algorithms. And there're large amounts of parameters in low dimensional manifold model, including the choice of weight matrix w, the inner loop and outer loop times etc. So it could be hard to tune the algorithm to the best in a short time. In my implementation, the weight matrix w is defined as a gaussian kernel, while in [4], a cosine based weight matrix is put forward. Furthermore, the bregman iteration with penalty term can definitely boost the performance. For the limitation of time, I do not implement these properties. There's much more work can be done in order to further improve the performance of LDMM.

We can see overfitting is inevitable during the procedure of training. But basically, a low MSE on training set guarantees a low MSE on testing set.

Another consideration is the low rank property of the data might be designed purposely so applying a low dimensional manifold model may introduce useless degree of freedom, a more general dataset like movie-lens may show the superiority of LDMM model.

4 Conclusion

In this paper, I introduce and implement three different methods for solving matrix completion problem. Of all the algorithms, AGD is shown to be the fastest and ALS has the best performance with MSE = 7.59 on given dataset. The source code is provided in the zip file and on https://github.com/13aeon/Statistical-Signal-Processing-Project2-Matrix-Completion. Also the predicted matrix X is saved in ALSOut.mat, AGDOut.mat and LDMMOut.mat separately. The implementation of LDMM is based on VLFeat toolbox in matlab. For more detailed information on usage, please refer to README.md file on github or as attached.

References

[1] Joonseok Lee, Mingxuan Sun, and Guy Lebanon. A comparative study of collaborative filtering algorithms. *Computer Science*, 2012.

[2] Xia Ning and George Karypis. *Recent Advances in Recommender Systems and Future Directions*. Springer International Publishing, 2015.

[3] Kazuhiro Miyahara and Toshio Okamoto. Collaborative information filtering towards distributed learning environment : Concept of collaborative filtering and its realization. *Ieice Technical Report Education Technology*, 96:39–44, 1996.

[4] Da Kuang, Zuoqiang Shi, Stanley Osher, and Andrea Bertozzi. A harmonic extension approach for collaborative ranking. 2016.

[5] Changxiao Cai, Sujay Sanghavi, and Haris Vikalo. Structured low-rank matrix factorization for haplotype assembly. *IEEE Journal of Selected Topics in Signal Processing*, 10(4):647–657, 2016.

[6] Mike Espig, Wolfgang Hackbusch, and Aram Khachatryan. On the convergence of alternating least squares optimisation in tensor format representations. *Mathematics*, 72(22):4303, 2015.

[7] Cand, S Emmanuel, and Benjamin Recht. *Exact matrix completion via convex optimization*. ACM, 2012.

[8] Keshavan, H Raghunandan, Montanari, Andrea, and Sewoong. Matrix completion from noisy entries. *Journal of Machine Learning Research*, 11(3):2057–2078, 2009.