A Coordinate Descent Method for Robust Matrix Factorization and Applications

Spencer Sheen [*]

Abstract

Matrix factorization methods are widely used for extracting latent factors for low rank matrix completion and rating prediction problems arising in recommender systems of on-line retailers. Most of the existing models are based on L2 fidelity (quadratic functions of factorization error). In this work, a coordinate descent (CD) method is developed for matrix factorization under L1 fidelity so that the related minimization is done one variable at a time and the factorization error is sparsely distributed. In low rank random matrix completion and rating prediction of MovieLens-100k datasets, the CDL1 method shows remarkable stability and accuracy under gross corruption of training (observation) data while the L2 fidelity based methods rapidly deteriorate. A closed form analytical solution is found for the one-dimensional L1-fidelity subproblem, and is used as a building block of CDL1 algorithm whose convergence is analyzed. The connection with the well-known convex method, the robust principal component analysis (RPCA), is made. A comparison with RPCA on recovering low rank Gaussian matrices under sparse and independent Gaussian noise shows that CDL1 maintains accuracy at much lower sampling ratios (from much fewer observed entries) than that for RPCA.

Keywords. L1 fidelity, coordinate descent, regularized weighted median, closed form solution, convergence, robust matrix factorization.

AMS subject classifications: 49M27, 65K10, 90C26, 90C30.

[*]Woodbridge High School, 2 Meadowbrook, Irvine, CA 92604, USA. Email: spsheen97@gmail.com. Mentor: Prof. Hongkai Zhao, Department of Mathematics, University of California, Irvine, CA, 92697, USA. E-mail: zhao@math.uci.edu.

1 Introduction

On-line shopping has been part of our daily lives in the digital age. In constrast to traditional shopping in a store (a limited physical environment), on-line shoppers can be easily inundated in a sea of choices. Matching products with shoppers' tastes and preferences arises as a key issue for enhancing market efficiency and customer satisfaction. So there is a natural need for a recommender system to anticipate a buyer's next move and make recommendations based on the past preference or rating data [7]. E-commerce giants Amazon, Netflix, Spotify among many others, have embedded recommender systems in their websites. The available data from the past (also called training data) can be tabulated as a matrix (utility matrix) with row numbers representing the user/buyer identification (IDs) and the column number representing the item/product IDs. The value of the (i, j)-th entry is the rating of the i-th user to the j-th item.

The utility matrix is however incomplete since almost no users rate all items and no items receive ratings from all users. On the other hand, there are many people with similar tastes, and many items with similar attributes (e.g. genres, casts, popularities of movies). So there is a lot of redundancy in the ratings. If we categorize users and movies into classes, the number of classes is in the range of 20's to 100 while the size of the utility matrix can be over thousands. Mathematically, one can form a few groups of rows or columns. Inside each group, the rows or columns are nearly aligned (or approximately rank-1). In between the groups, there is clear distinction. This pattern leads to the low rank property of the utility matrix, see [10] and its Fig. 2 for an illustration of the low rank phenomenon. The *first research problem* is to complete the utility matrix as a low rank matrix, the so called *matrix completion* which also comes up in other applications such as face recognition and video restoration [3] where occlusions and defects occur. The *second research problem* is to *predict future ratings* based on the training data, or find a low rank matrix to capture the essential features of users and items and serve as a predictor.

In this paper, we shall study and compare optimization based matrix factorization methods for both types of problems above. Linear algebra tells us that a size $m \times n$ low rank matrix is a product of an $m \times r$ matrix and a $r \times n$ matrix, with $r \ll m(n)$ or a sum of r rank-1 matrices. The new phenomenon here is that the factorization is to be approximately found and computed under the constraint that the utility matrix (denoted by A) is low rank and incomplete. Matrix factorization techniques have been found successful for large scale recommendation systems of NetFlix and Spotify in recent years [10]. Let each user u (item i) be associated with a row vector $p_u \in \mathbb{R}^r$ (a column vector $q_i \in \mathbb{R}^r$). The dot product $p_u \cdot q_i$ models the interaction/preference of item i with/to user u. The basic idea is to find these so called latent factors (or computer's way to categorize the users and items) so that $A(u, i) \approx p_u \cdot q_i$. To avoid overfitting, especially in prediction problems, it is standard to use the L^2 regularization (penalty) to formulate a matrix factorization optimization problem [10]:

$$\min_{p_u, q_i} \sum_{(u,i)\in\Omega} (A(u, i) - p_u \cdot q_i)^2 + \lambda \, (\sum_{u=1}^{m} \|p_u\|^2 + \sum_{i=1}^{n} \|q_i\|^2), \tag{1.1}$$

where Ω is the index set of the observed ratings, λ is a positive parameter. Efficient computational methods for (1.1) include stochastic gradient descent, alternating least squares, and coordinate descent methods [10, 18, 21]. We notice that the error term of model (1.1) is L2 squared.

The main contribution in this paper is to develop the coordinate descent method (CD) for the following matrix factorization model with L^1 error term (L^1 fidelity):

$$\min_{p_u, q_i} \sum_{(u,i)\in\Omega} |A(u, i) - p_u \cdot q_i| + \lambda \, (\sum_{u=1}^{m} \|p_u\|^2 + \sum_{i=1}^{n} \|q_i\|^2). \tag{1.2}$$

For high dimensional optimization problems such as (1.1) and (1.2), the CD method minimizes one variable at a time while fixing the other variables. The one variable L2 model reduces to minimizing a sum of quadratic functions and is solvable (see equation (4.2) of section 4.1). The one variable L1 model involves a sum of absolute value terms and is only piecewise quadratic. We found *a closed-form analytical solution for each one-dimensional sub-problem* of the L1 model (1.2). The solution is a regularized weighted median function of a sequence of numbers. The CD algorithm is built on top of the regularized weighted median function via convergent iterations. We conducted numerical experiments on low rank random matrix completion and MovieLens 100k prediction, in comparison with existing results and methods in the literature. We examined the effect of gross corruptions (large and localized errors) of training data on the recovery and prediction accuracies. The CD method of L1 model (1.2) is found to be amazingly stable while all methods solving the L2 model (1.1) deteriorate considerably. The L1 model (1.2) and its CD algorithm (CDL1) are hence desirable in designing a stable recommender system in the event of a malicious hacking attack.

The rest of the paper is organized as follows. In section 2, we derive the closed form regularized weighted median function as a solution to a class of convex programs arising in CD method of L1 model (1.2). In section 3, we introduce the L2 and L1 matrix factorization models and draw connections to the nuclear norm regularized convex models in the literature. The L1 model (1.2) is a non-convex version of robust principal component analysis (RPCA) or the low rank plus sparse decomposition of matrices [3]. In section 4, we present CDL2 and CDL1 algorithms, their update sequences and convergence properties. In section 5, we show computational results on robustness of CDL1 algorithm in matrix completion and MovieLens 100k prediction problems, in comparison with CDL2 and other L2 fidelity based methods. We compared CDL1 with RPCA [3] on recovering incomplete low rank matrices under sparse Gaussian noise. Though both methods are designed to be robust under sparse noise, CDL1 maintains its stable and accurate recovery when sampling ratio (percentage of observed entries) is as low as 5% for 1000 by 1000 Gaussian matrices while robust PCA performs well only when sampling ratio is 80% and above. The conclusions are in section 6. An algebraic identity of CDL2 iterations for convergence analysis is proved in the Appendix.

2 Medians and Optimization Problems

In this section, we discuss "median" of a finite sequence as a solution of a class of convex optimization problems, to set the stage for coordinate descent method of matrix completion model in the next section.

The standard median of an increasing sequence $a = \{a_j\}$, $j = 1, 2 \cdots, J$, is denoted by median(a). Consider the convex optimization problem:

$$\xi_* = \mathrm{argmin}_{\xi \in \mathbb{R}} \sum_{j=1}^{J} |\xi - a_j| := \mathrm{argmin}_{\xi \in \mathbb{R}}\, g(\xi), \tag{2.1}$$

where the objective function is a continuous (piecewise linear) function with corners. We claim that $\xi_* = \mathrm{median}(a)$.

To see that the median is a solution, let us observe that the sum is a decreasing (increasing) linear function at $\xi < a_1$ ($\xi > a_J$). So the minimal point is located in (a_1, a_J). If $\xi \in (a_j, a_{j+1})$, the derivative: $g' = j - (J - j) = 2j - J$. So if J is an even number, the function g is *flat* in the interval (a_{j_*}, a_{j_*+1}), $j_* = J/2$. Since g is continuous, the minimum is attained at any point in the interval $[a_{j_*}, a_{j_*+1}]$, in particular $\xi = (a_{j_*} + a_{j_*+1})/2$ or median(a).

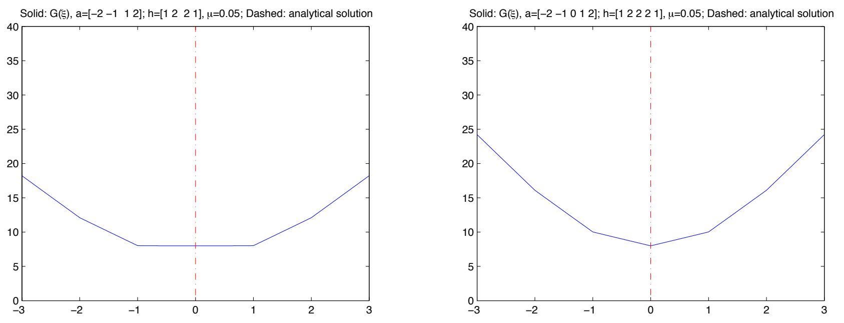

Figure 1: Objective function G of (2.4) at $\mu = 0.05$: $a = [-2, -1, 1, 2]$, $h = [1, 2, 2, 1]$, $J = 4$ (solid, left); $a = [-2, -1, 0, 1, 2]$, $h = [1, 2, 2, 2, 1]$, $J = 5$ (solid, right). Dashed lines indicate minimum locations by formula (2.6).

If J is odd, $g' = 1$ ($g' = -1$) at $\xi > a_{j_*}$ ($\xi < a_{j_*}$), $j_* = (J+1)/2$, so the minimal point is a_{j_*} or median(a).

The weighted median problem is:

$$\xi_* = \text{argmin}_{\xi \in \mathbb{R}} \sum_{j=1}^{J} h_j \, |\xi - a_j| \tag{2.2}$$

where h_j is a positive finite sequence (the weights). A minimal point (minimizer) can be found similarly as:

$$\xi_* = a_m, \quad m = \min \left\{ k \in [1, J] : \sum_{j=1}^{J} h_j \; < \; 2 \sum_{j=1}^{k} h_j \right\}. \tag{2.3}$$

The formula (2.3) is a limiting case of the following quadratically regularized weighted median problem (QRWM):

$$\xi_* = \text{argmin}_{\xi \in \mathbb{R}} \frac{\mu}{2} \xi^2 + \sum_{j=1}^{J} h_j \, |\xi - a_j| := \text{argmin}_{\xi \in \mathbb{R}} G(\xi), \tag{2.4}$$

where $\mu > 0$ is a regularization parameter.

The QRWM problem will be used in the coordinate descent method of our matrix completion model later. We derive a closed form solution below. The objective function G is convex as it is a linear combination of convex functions. To find the solution of (2.4), we examine where G changes monotonicity from decreasing to increasing. The function G is differentiable except at corner points a_j:

$$G'(\xi) = \mu\,\xi + \sum_{j=1}^{J} h_j \frac{\xi - a_j}{|\xi - a_j|}, \quad \xi \neq a_j, \tag{2.5}$$

so the criterion for a minimum occuring at a point $\xi = p$ is that $G'(p) = 0$ if $p \neq a_j$; $G'(p-) < 0$ and $G'(p+) > 0$ if $p = a_j$.

Consider $\xi < a_1$, and $G' = \mu\xi - \sum_{j=1}^{J} h_j$. If $a_1 > \mu^{-1}\sum_{j=1}^{J} h_j$, the critical point $\mu^{-1}\sum_{j=1}^{J} h_j = \xi_*$. If $a_1 < \mu^{-1}\sum_{j=1}^{J} h_j$, G is strictly decreasing over $\xi < a_1$, so we look for a minimal point on the next interval $[a_1, a_2)$ where $G' = \mu\,\xi + h_1 - \sum_{j=2}^{J} h_j$ for $\xi > a_1$. If $a_1 > \mu^{-1}(-h_1 + \sum_{j=2}^{J} h_j)$, $G'(a_1+) > 0$, so $\xi_* = a_1$. On the other hand, if $a_1 < \mu^{-1}(-h_1 + \sum_{j=2}^{J} h_j) < a_2$, then $G'(a_1+) < 0$, $G'(a_2-) > 0$, $G'(x_*) = 0$, where $\xi_* = \mu^{-1}(-h_1 + \sum_{j=2}^{J} h_j)$. The argument repeats over $\xi > a_2$. The full solution is:

$$\begin{array}{ll}
\xi_* = \mu^{-1}\sum_{j=1}^{J} h_j, & \text{if } a_1 > \mu^{-1}\sum_{j=1}^{J} h_j \\
= a_1, & \text{if } a_1 \in \mu^{-1}(-h_1 + \sum_{j=2}^{J} h_j, \sum_{j=1}^{J} h_j) \\
= \mu^{-1}(-h_1 + \sum_{j=2}^{J} h_j), & \text{if } \mu^{-1}(-h_1 + \sum_{j=2}^{J} h_j) \in (a_1, a_2) \\
= a_2 & \text{if } a_2 \in \mu^{-1}(-h_1 - h_2 + \sum_{j=3}^{J} h_j, -h_1 + \sum_{j=2}^{J} h_j) \\
= \mu^{-1}(-h_1 - h_2 + \sum_{j=3}^{J} h_j) & \text{if } \mu^{-1}(-h_1 - h_2 + \sum_{j=3}^{J} h_j) \in (a_2, a_3) \\
= a_3 & \text{if } a_3 \in \mu^{-1}(-\sum_{j=1}^{3} h_j + \sum_{j=4}^{J} h_j, -h_1 - h_2 + \sum_{j=3}^{J} h_j) \\
\cdots & \cdots \\
\cdots & \cdots \\
\cdots & \cdots \\
= \mu^{-1}(h_J - \sum_{j=1}^{J-1} h_j) & \text{if } \mu^{-1}(h_J - \sum_{j=1}^{J-1} h_j) \in (a_{J-1}, a_J) \\
= a_J, & \text{if } a_J \in \mu^{-1}(-\sum_{j=1}^{J} h_j, h_J - \sum_{j=1}^{J-1} h_j) \\
= -\mu^{-1}\sum_{j=1}^{J} h_j, & \text{if } a_J < -\mu^{-1}\sum_{j=1}^{J} h_j
\end{array} \tag{2.6}$$

It is instructive to write down two special cases of the solution (2.6) for $J = 2$ and $J = 3$. The $J = 2$ solution is:

$$\xi_* = \begin{cases} \mu^{-1}(h_1 + h_2) & \text{if } a_1 > \mu^{-1}(h_1 + h_2) \\ a_1 & \text{if } a_1 \in \mu^{-1}(h_2 - h_1, h_1 + h_2] \\ \mu^{-1}(h_2 - h_1) & \text{if } \mu^{-1}(h_2 - h_1) \in [a_1, a_2] \\ a_2 & \text{if } a_2 \in \mu^{-1}[-h_1 - h_2, h_2 - h_1) \\ -\mu^{-1}(h_1 + h_2) & \text{if } a_2 < -\mu^{-1}(h_1 + h_2) \end{cases}$$

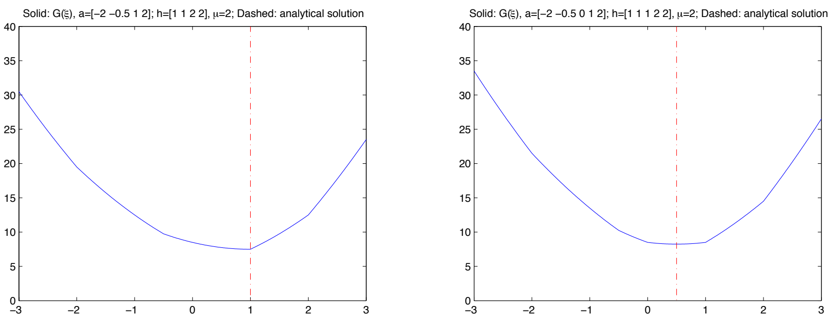

Figure 2: Objective function G of (2.4) at $\mu = 2$: $a = [-2, -0.5, 1, 2]$, $h = [1, 1, 2, 2]$, $J = 4$ (solid, left); $a = [-2, -0.5, 0, 1, 2]$, $h = [1, 1, 1, 2, 2]$, $J = 5$ (solid, right). Dashed lines indicate minimum locations by formula (2.6).

The $J = 3$ solution is:

$$\xi_* = \begin{cases} \mu^{-1}(h_1 + h_2 + h_3) & \text{if } a_1 > \mu^{-1}(h_1 + h_2 + h_3) \\ a_1 & \text{if } a_1 \in \mu^{-1}(h_2 + h_3 - h_1, h_1 + h_2 + h_3] \\ \mu^{-1}(h_2 + h_3 - h_1) & \text{if } \mu^{-1}(h_2 + h_3 - h_1) \in [a_1, a_2] \\ a_2 & \text{if } a_2 \in \mu^{-1}[h_3 - h_1 - h_2, h_2 + h_3 - h_1) \\ \mu^{-1}(h_3 - h_1 - h_2) & \text{if } \mu^{-1}(h_3 - h_1 - h_2) \in (a_2, a_3] \\ a_3 & \text{if } a_3 \in \mu^{-1}[-h_1 - h_2 - h_3, -h_1 - h_2 + h_3) \\ -\mu^{-1}(h_1 + h_2 + h_3) & \text{if } a_3 < -\mu^{-1}(h_1 + h_2 + h_3) \end{cases}$$

We see that $\xi_* \to 0$ if $\mu \uparrow \infty$, also it is not hard to check that ξ_* converges to a minimal point of (2.2) as $\mu \downarrow 0$.

Formula (2.6) and objective function G in (2.4) are illustrated in Fig. 1 at $\mu = 0.05$ and in Fig. 2 at $\mu = 2$. To summarize, we have:

Proposition 2.1 *The quadratically regularized weighted median problem (QRWM) has a closed form solution given by (2.6).*

Next, due to strong convexity of G, we have:

Proposition 2.2 *There exists a positive constant $c = c(\mu, \min(h_1, \cdots, h_J))$ such that:*

$$G(\xi) - G(\xi_*) \geq c\,|\xi - \xi_*|^2, \quad \forall\, \xi \in \mathbb{R}. \tag{2.7}$$

Proof: If ξ_* occurs away from any a_j, G is locally smooth near ξ_*, and (2.7) follows from Taylor expansion and strong convexity of G in a small neighborhood of ξ_*, with $c < \mu/4$, which also works if ξ is large enough as $G \geq \mu\xi^2/2$.

Reducing c properly to accomodate the intermediate ξ yields (2.7). If $\xi_* = a_j$ for some j, then for ξ near ξ_*, the function G behaves as $c_1|\xi - a_j| \geq c_1|\xi - a_j|^2$, for some positive constant $c_1 = 1/2 \min(h_1, \cdots, h_J)$. Arguing similarly for ξ large and intermdiate as above, we see that (2.7) holds.

3 Low Rank Matrix Factorization Models

3.1 L2 Models

In a recommender system, one aims to learn a model from past incomplete rating data (training data) then use the model to estimate current user's preference over all items. Since there are normally a few dozen intrinsic degrees of freedom in terms of user and item categories, an efficient way for computer to learn the user or item features is to approximate the training data (or equivalently the utility matrix) by a low rank matrix. Let $A \in \mathbb{R}^{m\times n}$ be a utility matrix whose row/column numbers are user/item ID's, and the (i, j)-th entry if available has the rating value of the i-th user to j-th item. The number of users (items) is m (n). Let Ω denote the set of indices where the ratings are available. The low rank matrix factorization problem is to find $W \in \mathbb{R}^{m\times r}$ and $H \times \mathbb{R}^{r\times m}$, for a proper integer $r \ll m$ or n, so that

$$A \approx W \ \cdot H, \quad \text{for all entries in } \Omega. \tag{3.1}$$

The factorization problem (3.1) can be made more precise as an optimization problem of the form:

$$\begin{aligned} [W, H] &= \text{argmin } F_2(W, H) \\ &:= \text{argmin} \sum_{(i,j)\in\Omega} (A_{ij} - w_i^T \cdot h_j)^2 + \lambda \left(\|W\|_F^2 + \|H\|_F^2\right), \end{aligned} \tag{3.2}$$

where w_i^T is the i-th row vector of W, h_j is the j-th column vector of H, $\|\cdot\|_F$ is the Frobenius matrix norm. Commonly used methods for (3.2) include stochastic gradient descent (SDG) [10, 18], alternative least squares (ALS) [19, 11] and coordinate descent (CD) [21]. A variant of (3.2) without the quadratic regularization is solved by a weighted alternating method LMaFit [20].

Since all norms in (3.2) are 2-norms, we shall call it the L2 model. If matrix A is fully observed, the L2 model is related to the following model involving the nuclear norm and rank constraint:

$$\min_{rank(Z)\leq r} \|A - Z\|_F^2 + \lambda \|Z\|_*, \tag{3.3}$$

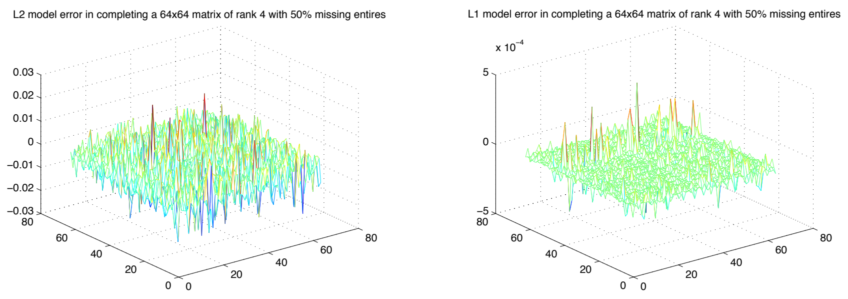

Figure 3: Illustration of factorization error of L2 (left) and L1 (right) fidelity models in completing a 64×64 matrix of rank 4 with 50% missing entries.

where the nuclear norm $\| \cdot \|_*$ is the sum of singular values, $r \leq \min(m, n)$. In fact, by the identity [2, 19]:

$$\|Z\|_* = \min_{Z=W\,H} \frac{1}{2} (\|W\|_F^2 + \|H\|_F^2), \tag{3.4}$$

one has:

$$\min_{W_{m\times r}, H_{r\times n}} \|A - W\,H\|_F^2 + \frac{\lambda}{2} (\|W\|_F^2 + \|H\|_F^2) = \min_{rank(Z)\leq r} \|A - Z\|_F^2 + \lambda \, \|Z\|_*.$$

If the rank constraint is dropped from (3.3), one arrives at the relaxed convex program:

$$\min_Z \|A - Z\|_F^2 + \lambda \, \|Z\|_*, \tag{3.5}$$

for a low rank approximation of A. If the nuclear term is dropped, (3.3) reduces to the classical principal component analysis (PCA,[8]):

$$\min_{rank(Z)\leq r} \|A - Z\|_F^2, \tag{3.6}$$

If A contains missing entries, the connection between (3.2) and (3.5) remains [11], only with $A - Z$ restricted to Ω.

3.2 L1 Models

The L2 error (fidelity) term in (3.5) is known to cause error to spread and is sensitive to outliers in the data matrix A, see the left plot of Fig. 3, similar to classical least squares estimator. In the spirit of robust statistics [6] and robust principal component analysis [3], we replace the Frobenius norm in (3.5) by L1 norm or sum of absolute value of all entries. The resulting error distribution is more sparse with a few large peaks, see the right plot of Fig. 3. In view of the connection of (3.2) and (3.5), the robust factorization model for partially observed matrix A is then:

$$\begin{aligned} [W,H] &= \operatorname{argmin} F_1(W,H) \\ &:= \operatorname{argmin} \sum_{(i,j)\in\Omega} |A_{ij} - w_i^T \cdot h_j| \,+\, \lambda\,(\|W\|_F^2 + \|H\|_F^2), \end{aligned} \tag{3.7}$$

which we shall call the L1 model. The L1 fidelity renders the error of low rank approximation sparse. If we denote $S := A - W \cdot H$, then the corresponding convex nuclear norm relaxation of (3.7) is:

$$\min_{S,Z} \|S\|_1 + \lambda\,\|Z\|_*, \text{ subject to } Z + S = A, \text{ over } \Omega, \tag{3.8}$$

which is to decompose A into a sum of low rank and sparse matrices. In other words, the L1 model (3.7) is akin to a robust PCA (a.k.a principal component pursuit) for incomplete matrix A, see [3] for analysis and computation of (3.8) and [4] for a related convex model.

Our work is to study the coordinate descent (CD) method for the L1 model (3.7) and compare it with L2 model (3.2) in matrix factorization problems based on synthetic and real world recommender system data. The CD method is simple in algebra, and is recently found [21] to be parallelizable and competitive with ALS and SGD methods on large scale Netflex ($m = 2,649,429, n = 17,770$) and Yahoo music ($m = 1,000,990$, $n = 624,961$) data.

4 Coordinate Descent Method

Both L1 and L2 models concern high dimensional non-convex optimization. The idea of CD method is to minimize one variable at a time while fixing all the other variables, and iterate this procedure till convergence [1]. We shall first review CD algorithm [21] for L2 model (3.2), then present our CD algorithm for the L1 model (3.7).

4.1 CD Algorithm of L2 Model (CDL2)

The single variable sub-problem is:

$$\min_z f_2(z) := \sum_{j:(i,j)\in\Omega} (A_{ij} - w_i^T \cdot h_j + (w_{it} - z)h_{tj})^2 + \lambda\, z^2, \tag{4.1}$$

where w_{it}, the (i,t)-th entry of W, is to be updated, and so the t-th term in $w_i^T h_j$ of the fidelity term is now $z h_{tj}$. The function f_2 is a quadratic polynomial in z, its minimum is attained at:

$$z_* = \frac{\sum_{j:(i,j)\in\Omega} \left(A_{ij} - w_i^T \cdot h_j + w_{it} h_{tj}\right) h_{tj}}{\lambda + \sum_{j:(i,j)\in\Omega} h_{jt}^2}. \tag{4.2}$$

Denote the residual matrix as:

$$R_{ij} = A_{ij} - w_i^T \cdot h_j, \quad \forall (i,j) \in \Omega, \tag{4.3}$$

through which we write:

$$z_* = \frac{\sum_{j:(i,j)\in\Omega} \left(R_{ij} + w_{it} h_{tj}\right) h_{jt}}{\lambda + \sum_{j:(i,j)\in\Omega} h_{tj}^2}. \tag{4.4}$$

Computationally, it is cost effective to maintain and update the residual. Once z_* is available, the update is: $w_{it} \leftarrow z_*$. Such update loops over $i = 1, 2, \cdots, m$. Similarly, the update on h_{tj} is (with current w_{it}): $h_{tj} \leftarrow \zeta_*$, where:

$$\zeta_* = \frac{\sum_{i:(i,j)\in\Omega} \left(R_{ij} + w_{it} h_{tj}\right) w_{it}}{\lambda + \sum_{i:(i,j)\in\Omega} w_{it}^2}, \tag{4.5}$$

and the loop is over $j = 1, 2, \cdots, n$. The update sequence is:

$$w_{1,t}, w_{2,t}, \cdots, w_{m,t}, h_{t,1}, h_{t,2}, \cdots, h_{t,n}.$$

The sequence is iterated a number of times (so called cyclic CD) to produce a pair of column vectors (w_t^*, h_t^*). The residual matrix R_{ij} is updated as $R_{ij} \leftarrow R_{ij} + w_{it}^o h_{tj}^o - w_{it}^* h_{tj}^*$, where superscript 'o' denotes the previous value before the alternating iterations. During the cyclic CD, the R_{ij} remains unchanged. Doing this for $t = 1, 2, \cdots, k$ forms one outer iteration. After a fixed number of outer iterations (at the end of each, assign '*' to 'o'), we end up with a rank-k completed or prediction matrix:

$$\sum_{t=1}^{k} w_t^* \, h_t^{*,T} \tag{4.6}$$

We initialize w_t^o as one vector, h_t^0 as zero vector, for each t.

To summarize, the update sequence for one outer iteration is:

$$w_1^o, h_1^o, w_2^o, h_2^o, \cdots, w_k^o, h_k^o \longrightarrow w_1^*, h_1^*, w_2^*, h_2^*, \cdots, w_k^*, h_k^* \tag{4.7}$$

To arrive at each vector pair w_t^*, h_t^* in (4.7), the update sequence of a number I of inner iterations is:

$$\begin{aligned}
u_0 &= w_t^o,\; v_0 = h_t^o, \\
u_1 &= \mathrm{argmin}_{u\in\mathbb{R}^m} \|R + w_t^o\, h_t^{o,T} - u\, v_0^T\|_{F,\Omega}^2 + \lambda\|u\|^2 \quad (4.8) \\
v_1 &= \mathrm{argmin}_{v\in\mathbb{R}^n} \|R + w_t^o\, h_t^{o,T} - u_1\, v^T\|_{F,\Omega}^2 + \lambda\|v\|^2 \quad (4.9) \\
\text{alternating} &\to u_2, v_2, \cdots, u_I, v_I \\
(w_t^*, h_t^*) &\leftarrow (u_I, v_I) \\
R &\leftarrow R + w_t^o\, h_t^{o,T} - u_I v_I^T, \quad \text{on } \Omega, \quad (4.10)
\end{aligned}$$

where $\|\cdot\|_{F,\Omega}$ denotes the restriction of the Frobenius norm to matrix indices in Ω.

The alternating minimization in (4.8)-(4.9) is a rank-one case of alternating least squares: the components of u in (4.8) or v in (4.9) can be solved for independently in closed analytical form. Let:

$$E(u,v) := \|R + w_t^o\, h_t^{o,T} - u\, v^T\|_{F,\Omega}^2 + \lambda(\|u\|^2 + \|v\|^2), \quad (4.11)$$

we show:

Proposition 4.1 *The alternating updates of the CDL2 inner iterations satisfy the descent property:*

$$E(u_k, v_k) \geq E(u_{k+1}, v_k) \geq E(u_{k+1}, v_{k+1}), \quad (4.12)$$

and the identity:

$$\begin{aligned}
E(u_k, v_k) - E(u_{k+1}, v_{k+1}) &= \|(u_k - u_{k+1})v_k^T\|_{F,\Omega}^2 + \|u_{k+1}(v_{k+1} - v_k)^T\|_{F,\Omega}^2 \\
&+ \lambda(\|u_k - u_{k+1}\|^2 + \|v_k - v_{k+1}\|^2), \quad (4.13)
\end{aligned}$$

implying the inequality:

$$\|(u_k, v_k) - (u_{k+1}, v_{k+1})\|^2 \;\leq\; \lambda^{-1}\,|E(u_k, v_k) - E(u_{k+1}, v_{k+1})|. \quad (4.14)$$

Proof: The descent property (4.12) is clear by construction. The identity (4.13) is proved in the appendix.

4.2 CD Algorithm for L1 Model (CDL1)

The CDL1 algorithm follows the steps of CDL2 outlined above, except that the formulas for the minimizers z_* and ζ_* are replaced by the quadratically regularized weighted median formulas (2.6). To this end, the single variable sub-problem:

$$\min_z f_1(z) := \sum_{j:(i,j)\in\Omega} |A_{ij} - w_i^T h_j + (w_{it} - z)h_{tj}| + \lambda\, z^2, \tag{4.15}$$

is written as:

$$\min_z \sum_{j:(i,j)\in\Omega, h_{tj}\neq 0} |h_{tj}||z - a_j| + \lambda\, z^2, \tag{4.16}$$

where:

$$a_j = \frac{A_{ij} - w_i^T h_j + w_{it}h_{tj}}{h_{tj}}.$$

In the algorithm, we set a threshold 10^{-9}, below which $|h_{tj}|$ is regarded as zero. If h_{tj} is zero in this sense for all j such that $(i,j) \in \Omega$, the model is nearly singular. We opt to not update, and keep w_{it} as is for preserving stability and non-increasing property of the objective function, instead of taking $z = 0$ and zero out a term in the factorization. If $|h_{tj}|$ is above the threshold for some j, we call the function in (2.6) to define z_*. The update on h_{jt} is similar.

Proposition 4.2 *Let us denote by (W^l, H^l) the approximate factors after the l-th CD outer iteration of either CDL2 or CDL1. By construction, the descent property:*

$$F_i(W^{l+1}, H^{l+1}) \leq F_i(W^l, H^l), \quad i = 1, 2, \tag{4.17}$$

holds.

We recap the CD algorithms as a pseudo code below.

- Input: $R = A$, k, $W = zeros(m,k)$, $H = ones(k,n)$, λ, O, I.
- for iter $= 1, 2, \cdots, O$, do
- for $t = 1, 2, ..., k$, do
- solve in closed-form (4.8)-(4.9) or the L1 version alternately I times.
- end for
- save vectors w_t^*, h_t^*, and update R by (4.10).
- end for
- Output: rank k matrix (4.6).

4.3 Convergence

We study convergence properties of CDL2 and CDL1. The regularization terms of L2 and L1 models (3.2)-(3.7) imply the uniform bound

$$\|(W^l, H^l)\|_F \leq \lambda^{-1} \max(F_1(W^0, H^0), F_2(W^0, H^0)), \;\; \forall\, l \geq 1,$$

where (W^0, H^0) is the initialization. Hence (W^l, H^l) converges to $(\bar{W}, \bar{H})$ up to a subsequence l_j, and $\lim_{l_j \to \infty} F_i(W^{l_j}, H^{l_j}) = F_i(\bar{W}, \bar{H})$, $i = 1$ or 2.

Next, recall that during the each single variable update on way from (W^{l-1}, H^{l-1}) to (W^l, H^l), the objective function is non-increasing, in fact, the change of objective function is bounded by $F_i(W^{l-1}, H^{l-1}) - F_i(W^l, H^l) := \delta_l$ which tends to zero as $l \to +\infty$. For CDL1, by inequality (2.7) on updating $\bar{W}$ or $\bar{H}$, the difference of the previous value and the new value is bounded by $O(\delta_l)$. Note that the constant c in (2.7) depends on λ and the cut-off threshold (10^{-9}) we introduced in the algorithm, but independent of the iteration number l. Since the total number of updates from $l-1$ to l in the outer iteration is order $O(mn)$, we have

$$\|W^{l-1} - W^l\|_F^2 + \|H^{l-1} - H^l\|_F^2 \leq O(\delta_l). \tag{4.18}$$

By the inequality (4.14), the estimate (4.18) holds for CDL2 as well, except that the constant appearing in the upper bound is global (λ^{-1}). It follows that the critical point equations at each update can be passed to the limit $(l_j \to \infty)$, and together they imply:

Theorem 4.1 *The limit point $(\bar{W}, \bar{H})$ is a first order stationay point for the L2 or L1 model. For L2 model:*

$$\nabla_W \, F_2(\bar{W}, \bar{H}) = 0, \;\; \nabla_H \, F_2(\bar{W}, \bar{H}) = 0, \tag{4.19}$$

and for the L1 model:

$$0 \in \partial_W \, F_1(\bar{W}, \bar{H}), \;\; 0 \in \partial_H \, F_1(\bar{W}, \bar{H}). \tag{4.20}$$

The partial denotes sub-gradient.

4.4 CD Limits and Convex Programs

Recall the inequality [14] for $m \times r$ matrix W and $r \times n$ matrix H:

$$\|WH\|_* \leq \frac{1}{2}(\|W\|_F^2 + \|H\|_F^2),$$

with equality for a particular representation of WH based on singular value decomposition (SVD):

$$WH = UDV^T, \; W = UD^{1/2}, \; H = D^{1/2}V^T. \tag{4.21}$$

Applying the representation (4.21) to the CDL2 or CDL1 descending sequence (W^k, H^k) to get $(\tilde{W}^k, \tilde{H}^k)$, we find that:

$$F_l(W^k, H^k) \geq F_l(\tilde{W}^k, \tilde{H}^k) = E_l(\tilde{W}^k \tilde{H}^k), \;\; l = 1, 2, \tag{4.22}$$

where:

$$\begin{aligned} E_2 &= E_2(Z) := \|A - Z\|_{F,\Omega}^2 + \lambda \, \|Z\|_*, \\ E_1 &= E_1(Z) := \|A - Z\|_{1,\Omega} + \lambda \, \|Z\|_*. \end{aligned} \tag{4.23}$$

If the outer iteration sequence is modified to:

$$(W^1, H^1) \to (\tilde{W}^1, \tilde{H}^1) \to (W^2, H^2) \to (\tilde{W}^2, \tilde{H}^2) \to \cdots, \tag{4.24}$$

then we generate a sequence of rank-r matrices $Z_k = \tilde{W}^k \tilde{H}^k$ so that:

$$E_l(Z_k) = F_l(\tilde{W}^k, \tilde{H}^k) \geq F_l(\tilde{W}^{k+1}, \tilde{H}^{k+1}) = E_l(Z_{k+1}). \tag{4.25}$$

The $\lim_{k \to \infty} E_l(Z_k)$ may not be the minimum of convex objectives due to the rank-r constraint (when r is less than the rank of the optimal solution of the corresponding convex program). A necessary condition is $r \geq r^*$, where r^* is the rank of an optimal solution of the convex program. Additional global condition is needed to ensure that the matrix $Z_* := \lim Z_k$ is optimal for the convex objectives (4.23). For the L2 model, an additional condition [11] is that Z_* is an optimal solution of the convex problem:

$$Z_* = \operatorname{argmin}_Z \|P_\Omega(A) + P_\Omega^\perp(Z) - Z\|_F^2 + \lambda \, \|Z\|_*, \tag{4.26}$$

where P_Ω is the projection onto the observed entries in Ω. Similar condition works for L1 model as well, with $\| \cdot \|_F$ in (4.26) replaced by $\| \cdot \|_1$.

5 Numerical Experiments

In this section, we present numerical results of CDL2 and CDL1 on matrix completion and rating prediction problems, and compare them with other representative methods in the literature. We shall demonstrate the robustness of CDL1 when the observed data contains gross corruptions.

The methods in comparison include: FPCA [13], sIRLs-q [16], IRucL-q [12], LMaFit [20], for matrix completion; sIRLs-q [16], Optspace [9], and Iterative Hard Thresholding (IHT) [5, 15] for rating prediction. The $q \in [0, 1)$. LMAFit solves a constrained low-rank factorization model without computing SVD similar to CD methods. The other methods in comparison compute SVD, and are non-convex in nature except the convex FPCA based on a relaxed nuclear norm model. Default parameter values are used in the comparison codes.

5.1 Low Rank Matrix Completion

Given an incomplete matrix with missing entries, the goal is to extend it to a low rank complete matrix so that the observed entries are preserved or minimally changed.

We generated random matrices $M = M_L M_R^T \in \mathbb{R}_{m\times n}$, where matrices $M_L \in \mathbb{R}_{m\times r}$ and $M_R \in \mathbb{R}_{n\times r}$ are generated from independent unit Gaussian distribution. Setting r small, we obtain M at low rank (generically r). Then M is normalized as $M \leftarrow M/\|M\|_2$. After this step, we uniformly random-sampled a subset Ω of p entries from M to produce $A = M|_\Omega$. The sampling ratio sr is $p/(mn)$. In our experiment, we set $m = n = 100$, and rank equal to $1, 2, \cdots, 5$. The recovered matrix $\tilde{A}$ at convergence is compared with the ground truth M in relative Frobenius norm: rel.err $:= \|\tilde{A} - M\|_F/\|M\|_F$. For CDL2 (CDL1), $\lambda = 10^{-10}$ (10^{-2}); inner and outer iteration numbers are 24 and 32 respectively. The λ values are so chosen that the completion error is under 10^{-6} for sample Gaussian matrices in Table 1. The values are the same later when corruptions of A are considered. Ten random samples of A are computed and the average accuracy is reported in Table 1. We see that CDL1 is most accurate at bottom rank values. Table 2 shows computational times at convergence of each method under comparison. The CD methods are much slower at this size of matrices. The speedup of CDL2 via parallel computation and the good scalability property for large matrices have been shown in [21]. Doing so for CDL1 will be left for a future project. The computation reported here is done in Matlab on a single processor (Intel-Core i7-4770) with 3.4 GHz cpu speed and 16G RAM.

Next we introduce a corrupted value at a randomly selected entry of A by magnifying the observed value by 10 times. Table 3 shows that CDL1 is remarkably robust and accurate while the other methods under-perform drastically by several orders of magnitude. Table 5 makes a comparison when the

Table 1: Accuracy comparison of CDL2, CDL1, sIRLS-0, IRucL-1/2 and LMaFit on completing Gaussian matrices at known rank, $m = n = 100$, $sr = 0.4$.

Problem	CDL2	CDL1	FPCA	sIRLS-0	IRucL-1/2	LMaFit
rank	rel.err	rel.err	rel.err	rel.err	rel.err	rel. err
1	4.54e-06	**1.09e-16**	2.94e-06	6.08 e-06	6.92e-06	1.83e-06
2	4.14e-06	**1.35e-08**	3.47e-06	2.19e-06	7.16e-06	1.56e-06
3	5.03e-06	**6.28e-08**	4.06e-06	3.79e-06	6.73e-06	1.68e-06
4	4.28e-06	1.79e-06	4.70e-06	4.25e-06	6.71e-06	**1.72e-06**
5	4.46e-06	5.13e-05	5.34e-06	4.56e-06	7.40e-06	**1.93e-06**

Table 2: Runtime comparison of CDL2, CDL1, sIRLS-0, IRucL-1/2 and LMaFit on completing Gaussian matrices at known rank, $m = n = 100$, $sr = 0.4$. Runtime is the average time of all trials in seconds.

Problem	CDL2	CDL1	FPCA	sIRLS-0	IRucL-1/2	LMaFit
rank	time	time	time	time	time	time
1	11.16	16.92	0.401	0.32	1.37	0.005
2	17.34	32.88	0.410	0.39	1.52	0.005
3	23.79	50.89	0.400	0.39	1.85	0.006
4	37.74	68.49	0.402	0.45	2.23	0.007
5	40.16	81.43	0.428	0.49	2.49	0.008

Table 3: Accuracy comparison of CDL2, CDL1, sIRLS-0, IRucL-1/2 and LMaFit on completing Gaussian matrices **under corruption by magnification of an observed entry by 10 times**; $m = n = 100$, $sr = 0.4$.

Problem	CDL2	CDL1	FPCA	sIRLS-0	IRucL-1/2	LMaFit
rank	rel.err	rel.err	rel.err	rel.err	rel.err	rel. err
1	4.15e-02	**1.10e-16**	1.47e-02	3.12e-02	3.29e-02	1.08e-02
2	5.18e-02	**3.41e-09**	4.18e-02	6.51e-02	7.78e-02	3.92e-02
3	3.06e-02	**1.69e-07**	6.65e-02	1.04e-01	1.35e-01	7.62e-02
4	5.79e-02	**8.78e-07**	2.72e-02	3.30e-02	4.71e-02	2.35e-02
5	6.94e-02	**3.40e-05**	5.67e-02	5.97e-02	8.07e-02	5.08e-02

Table 4: Runtime (in seconds) comparison of CDL2, CDL1, sIRLS-0, IRucL-1/2 and LMaFit on completing Gaussian matrices **under corruption by magnification of an observed entry by 10 times**; $m = n = 100$, $sr = 0.4$.

Problem	CDL2	CDL1	FPCA	sIRLS-0	IRucL-1/2	LMaFit
rank	time	time	time	time	time	time
1	8.71	16.42	0.396	15.09	0.69	0.003
2	19.65	32.30	0.423	19.02	0.84	0.003
3	29.38	48.76	0.409	19.12	0.97	0.004
4	39.15	64.55	0.407	16.94	1.09	0.005
5	40.87	81.30	0.410	20.18	1.29	0.007

corrupted value is 100 times the observed one. CDL1 remains robust and accurate at the same order while the others deteriorate. Comparing computation times in Tables 2, 4 and 6, we see that the CD methods and the convex FPCA method appear most stable under increased corruptions.

5.2 Prediction on MovieLens Data

We study the MovieLens 100k data set [17] which consists of 100,000 ratings (1 to 5) from 943 users on 1682 movies. Each user has rated at least 20 movies. Users and items are numbered consecutatively from 1. To compare with results in the literature, we consider 4 different 80/20 splits of the full data set u into training (**ui.base**) and test sets (**ui.test**), $i = 1, 2, 3, 4$. Each of the four training/test dataset is mapped to matrix A/T with rows/columns representing users/movies. The index set of available entries of T is denoted by Ω_T. A matrix factorization algorithm is applied to A to generate a prediction

Table 5: Accuracy comparison of CDL2, CDL1, sIRLS-0, IRucL-1/2 and LMaFit on completing Gaussian matrices **under corruption by magnification of an observed entry by 100 times**; $m = n = 100$, $sr = 0.4$.

Problem	CDL2	CDL1	FPCA	sIRLS-0	IRucL-1/2	LMaFit
rank	rel.err	rel.err	rel.err	rel.err	rel.err	rel. err
1	4.98e-01	**1.09e-16**	2.72e-01	1.91	4.63e-01	4.66e-01
2	3.07	**4.27e-08**	1.99e-01	1.80	5.42e-01	1.34
3	10.94	**2.85e-08**	4.76e-01	3.03	7.79e-01	2.73
4	5.33	**6.57e-07**	1.90e-01	1.93	6.83e-01	1.91
5	5.73	**3.42e-05**	5.47e-01	2.62	7.93e-01	2.37

Table 6: Runtime (in seconds) comparison of CDL2, CDL1, sIRLS-0, IRucL-1/2 and LMaFit on completing Gaussian matrices **under corruption by magnification of an observed entry by 100 times**; $m = n = 100$, $sr = 0.4$.

Problem	CDL2	CDL1	FPCA	sIRLS-0	IRucL-1/2	LMaFit
rank	time	time	time	time	time	time
1	9.91	16.25	0.388	18.54	0.84	0.006
2	18.52	33.34	0.437	19.29	1.76	0.029
3	26.12	48.86	0.416	19.42	2.79	0.134
4	40.05	67.07	0.423	19.75	2.64	0.084
5	46.80	83.66	0.420	21.10	3.96	0.205

matrix P which is compared with the test matrix T on Ω_T. The performace metric is Normalized Mean Absolute Error (NMAE) defined as:

$$\text{NMAE} := (r_{max} - r_{min})^{-1} \, |\Omega_T|^{-1} \sum_{(i,j)\in\Omega_T} |P_{ij} - T_{ij}|,$$

where r_{max}/r_{min} are maxium/minimum ratings, and $|\Omega_T|$ is the number of elements in Ω_T. In the three methods (sIRLS,IHT,Optspace) under comparison, the rank of P was set to five, see [16]. As suggested in [10], we extract the leading order information from the training set by a first order base model:

$$\bar{A}_{ij} = \bar{r} + u_i + m_j, \tag{5.1}$$

where $\bar{r}$ is the mean of all rating entries in A; u_i is the average rating by user i in A minus $\bar{r}$; m_j is the average rating received by movie j minus $\bar{r}$. The u_i and m_j terms are called bias [10]. The deviation from the base model $\hat{A}_{ij} = A_{ij} - \bar{A}_{ij}$ is factorized by CDL2 and CDL1 at rank equal to one. Computation indicated that there is no significant gain beyond rank one, partly because the prediction matrix from the first order model reaches a rank about 20. The predicted ratings are finally truncated to the range $[1, 5]$.

A major difference of the rating prediction problem from the low rank matrix completion problem is that we do not want P to be too close to A because this would cause overfitting instead of extracting robust features to be more predicative on the test matrix T. For this reason, the regularization parameter is chosen much larger: $\lambda = 150$ for CDL2 and $\lambda = 60$ for CDL1. The range of λ in CDL2 is drawn from that of the alternating least squares method of [11], which is in the 100's. The specific value 150 comes from minimizing the prediction error during cross validation on the 80/20 split of training/testing data. The λ value of 60 for CDL1 is chosen for keeping its prediction error no more than that of CDL2 on the four splits in Table 7. The inner and outer iteration numbers are same as before (24 and 32). Minimal improvement is gained from more iterations.

In Table 7, CDL1 and CDL2 perform the best with CDL1 slightly ahead of CDL2. Next, we introduce corruptions of three types in the training sets: (1) randomly select and switch 1000 lowest ratings (1's) to highest ratings (5's); (2) randomly select 200 lowest ratings (1's) and magnify by a factor of 10; (3) randomly select 10 lowest ratings (1's) and magnify by a factor of 100. In the 4 training sets, the total number of lowest ratings is in the range of 4700 to 4900, and number of the highest ratings is from 16700 to 17100. The type (2) corruption requires modifying the rating of about 8 users, while type (3) corruption involves only 1 user's ratings, a much easier task for a hacker.

Table 7: Comparison of CDL2, CDL1, sIRLS, IHT and Optspace on 100k MovieLens data.

Problem	CDL2	CDL1	sIRLS	IHT	Optspace
Problem	NMAE	NMAE	NMAE	NMAE	NMAE
split-1	0.1836	**0.1835**	0.1919	0.1925	0.1887
split-2	0.1810	**0.1808**	0.1878	0.1883	0.1878
split-3	0.1811	**0.1808**	0.1870	0.1872	0.1881
split-4	0.1809	**0.1808**	0.1899	0.1896	0.1882

Table 8: Comparison of CDL2 and CDL1 on 100k MovieLens data under 3 types of corruptions indexed by the last number: (1) switch 1000 lowest ratings to highest ratings; (2) magnify 200 lowest ratings by a factor of 10; (3) magnify 10 lowest ratings by a factor of 100.

Problem	CDL2-1	CDL1-1	CDL2-2	CDL1-2	CDL2-3	CDL1-3
Problem	NMAE	NMAE	NMAE	NMAE	NMAE	NMAE
split-1	0.1890	0.1890	0.2027	0.1890	0.3221	0.1857
split-2	0.1829	0.1828	0.2112	0.1838	0.3104	0.1826
split-3	0.1823	0.1821	0.2130	0.1825	0.2777	0.1825
split-4	0.1821	0.1820	0.2122	0.1818	0.2481	0.1966

The results from CDL2 and CDL1 on the three types of corruptions for the 4 splits are in Table 8. We see that the difference between CDL2 and CDL1 is small (relative change about 0.1%) for type (1) corruption, though CDL1 is consistently a shade better. This is partly due to the limited range of ratings (i.e. [1,5]), and the outliers are tamer than in the experiment on Gaussian matrices. The difference reaches nearly 14% for type (2) corruption and as much as 43% for type (3) corruption. As in the matrix completion problem, the CDL1 method is much more stable than CDL2 under a small number of large size (gross) corruptions.

5.3 Comparison with Robust PCA

Following [3], a low rank square matrix L_0 is generated and randomly sampled to form matrix L at sampling ratio sr. An independent Gaussian matrix is generated and randomly sampled at sparse ratio $spr < sr$ to form S (the sparse noise). The input to CDL1 and robust PCA (RPCA [3]) is $L + S$. Let the low rank output be matrix $\underline{L}$. We compare its relative Frobenius norm

Table 9: Comparison of RPCA and CDL1 on recovering low rank 100 by 100 Gaussian matrices under sparse Gaussian noise with 0.5 % sparse ratio. Each reported data point is the average of lowest 7 relative Frobenius errors over 10 trials, as sampling ratio (sr) varies.

Methods	sr	rank-1	rank-2	rank-3	rank-4	rank-5
RPCA	90%	9.01e-6	1.18e-5	1.64e-5	1.15e-5	1.31e-5
CDL1	90%	1.03e-16	2.40e-8	3.12e-8	1.10e-8	1.40e-9
RPCA	80%	3.08e-6	2.12e-6	2.29e-6	7.60e-3	2.02e-2
CDL1	80%	1.05e-16	1.19e-7	5.53e-9	1.93e-8	4.50e-8
RPCA	**70%**	2.00e-1	2.52e-1	3.57e-1	3.93e-1	3.98e-1
CDL1	70%	1.22e-16	1.57e-8	6.40e-10	1.38e-8	1.71e-8
CDL1	40%	1.08e-16	9.43e-10	1.44e-8	2.98e-6	3.47e-5
CDL1	30%	1.10e-16	2.17 e-8	1.47e-6	2.69e-4	4.26e-3
CDL1	**20%**	1.10e-16	6.49e-6	2.23e-3	1.47e-1	5.79e-1

Table 10: CDL1 recovering larger low rank square Gaussian matrices under sparse Gaussian noise with 0.5 % sparse ratio. Each reported data point is the average of lowest 7 relative Frobenius errors over 10 trials, as sampling ratio (sr) and matrix size vary.

sr	size	rank-1	rank-2	rank-3	rank-4	rank-5
10%	400×400	1.09e-16	3.21e-9	1.57e-7	4.55e-6	3.52e-4
10%	800×800	1.06e-16	1.13e-12	7.64e-12	6.17e-10	4.17e-9
5%	1000×1000	1.08e-16	1.35e-9	6.47e-9	2.19e-7	1.36e-5
3%	1000×1000	1.11e-16	6.30e-8	3.02e-5	2.6e-3	7.65e-2

error from L_0, with CDL1 parameters in subsection 5.1 and default setting in RPCA. Table 9 lists the average of lowest seven relative Frobenius norm errors in 10 trials on 100 by 100 Gaussian matrices with low ranks. At high sampling ratios (80% and above), both methods perform well. However, at 70% sampling ratio, RPCA starts to incur large errors (over 20%), while CDL1 mainstains its accuracy down to 30% (20% up to rank-3). Taking 7 lowest errors removes some fluctuations in CDL1 due to its non-convex nature. As matrix size increases to 1000 by 1000, CDL1 performs even better. Table 10 shows that CDL1 maintains accuracy at 3 % to 5 % sampling ratio, which is a typical range for Movielens and Netflix data [11]. The RPCA performs well still at or above 80% sampling ratio.

6 Conclusions

We studied the coordinate descent method for matrix factorization model with L1 fidelity and applied its algorithm (CDL1) for low rank matrix completion and recommender system rating prediction problems. We discovered a closed form analytical solution for the one-dimensional sub-problem and applied it to CDL1 iterations. In comparison with CDL2 and other L2 fidelity based methods, CDL1 stands out as a robust and stable method in the presence of large and sparse corruptions as possibly encountered in a hacker attack. In comparison with RPCA, CDL1 recovers low rank matrices at much lower sampling ratios in the presence of sparse noise. A line of future work is to speed up the CDL1 algorithm on parallel machines and apply it to large scale data sets to further verify the scalability of its robustness. A theoretical question is to study the convergence to a local minimum.

7 Acknowledgements

I would like to thank my mentor Prof. Hongkai Zhao at UC Irvine for his encouragement and advice throughout the project. I thank Mr. Shuai Zhang for helpful communications on matrix completion research.

8 Appendix

We show the identity (4.13). Let $\hat{R} = R + w_t^o h_t^o$ over the index set Ω where R is known. Direct calculation gives:

$$E(u_k, v_k) - E(u_{k+1}, v_{k+1}) = 2 \sum_{(i,j)\in\Omega} \hat{R}_{ij}(u_{k+1,i}v_{k+1,i} - u_{k,i}v_{k,i}) \tag{8.1}$$

$$+ \sum_{(i,j)\in\Omega} (u_{k,i}v_{k,j})^2 - (u_{k+1,i}v_{k+1,j})^2 + \lambda\,(\|u_k\|^2 + \|v_k\|^2 - \|u_{k+1}\|^2 - \|v_{k+1}\|^2).$$

Since $u_{k+1} = \operatorname{argmin}_u E(u, v_k)$, it satisfies $E_u(u_{k+1}, v_k) = 0$, or:

$$\sum_{j\in\Omega_i} [-2\hat{R}_{ij}\, v_{k,j} + 2u_{k+1,i}(v_{k,j})^2] + 2\lambda\, u_{k+1,i} = 0, \quad \forall\ i. \tag{8.2}$$

Similarly, $v_{k+1} = \operatorname{argmin}_v E(u_{k+1}, v)$, and so $E_v(u_{k+1}, v_{k+1}) = 0$, or:

$$\sum_{i\in\Omega_j} [(-2)\hat{R}_{ij}u_{k+1,i} + 2(u_{k+1,i})^2 v_{k+1,j}] + 2\lambda\, v_{k+1,j} = 0, \quad \forall\ j. \tag{8.3}$$

Multiplying (8.2) by $u_{k,i}$ and summing over i, likewise multiplying (8.3) by $v_{k+1,j}$ and summing over j, then subtracting the two resulting equalities to get:

$$\begin{aligned} 0 &= \sum_{(i,j)\in\Omega} -2\hat{R}_{ij}u_{k,i}v_{k,j} + 2\hat{R}_{ij}u_{k+1,i}v_{k+1,j} + 2\sum_{(i,j)\in\Omega} u_{k,i}u_{k+1,i}(v_{k,j})^2 \\ &- 2\sum_{(i,j)\in\Omega}(u_{k+1,i}v_{k+1,j})^2 + 2\lambda\sum_i u_{k+1,i}u_{k,i} - 2\lambda\sum_j (v_{k+1,j})^2. \end{aligned} \tag{8.4}$$

It follows from (8.1) and (8.4) that:

$$\begin{aligned} E(u_k, v_k) - E(u_{k+1}, v_{k+1}) &= \sum_{(i,j)\in\Omega} (u_{k,i}v_{k,j})^2 + (u_{k+1,i}v_{k+1,j})^2 - 2u_{k,i}u_{k+1,i}(v_{k,j})^2 \\ &- 2\lambda\sum_i u_{k+1,i}u_{k,i} + 2\lambda\sum_i (v_{k+1,i})^2 \\ &+ \lambda(\|u_k\|^2 + \|v_k\|^2 - \|u_{k+1}\|^2 - \|v_{k+1}\|^2). \end{aligned} \tag{8.5}$$

The 3 terms in the first sum of (8.5) can be written as:

$$(u_{k+1,i}v_{k+1,j})^2 - (u_{k+1,i}v_{k,j})^2 + (u_{k+1,i}v_{k,j})^2 - 2u_{k,i}u_{k+1,j}(v_{k,j})^2 + (u_{k,i}v_{k,j})^2,$$

which is:

$$(v_{k,j})^2(u_{k+1,i} - u_{k,i})^2 + (u_{k+1,i}v_{k+1,j})^2 - (u_{k+1,i}v_{k,j})^2. \tag{8.6}$$

The remaining terms of (8.5) with λ in front combine to give:

$$\sum_i (u_{k,i})^2 - 2u_{k,i}u_{k+1,i} - (u_{k+1,i})^2 + (v_{k,i})^2 + (v_{k+1,i})^2. \tag{8.7}$$

Multiplying (8.2) by $u_{k+1,i}$ and summing over i; multiplying (8.3) by $v_{k,j}$ and summing over j, subtracting the two resulting expressions, we find:

$$0 = 2\sum_{(i,j)\in\Omega} [(u_{k+1,i})^2(v_{k,j})^2 - v_{k,j}v_{k+1,j}(u_{k+1,i})^2] + 2\lambda\sum_i [(u_{k+1,i})^2 - v_{k,i}v_{k+1,i}]. \tag{8.8}$$

Adding (8.8) to (8.5), we see that (8.6) becomes:

$$(v_{k,j})^2(u_{k+1,i} - u_{k,i})^2 + (u_{k+1,i}v_{k+1,j})^2 + (u_{k+1,i})^2(v_{k,j})^2 - 2v_{k,j}v_{k+1,j}(u_{k+1,i})^2$$

$$= (v_{k,j})^2(u_{k+1,i} - u_{k,i})^2 + (u_{k+1,i})^2(v_{k+1,j} - v_{k,j})^2, \tag{8.9}$$

leading up to the first two square terms on the right hand side of (4.13). Similarly, (8.7) becomes:

$$\sum_i (u_{k,i})^2 - 2u_{k,i}u_{k+1,i} + (u_{k+1,i})^2 + (v_{k,i})^2 + (v_{k+1,i})^2 - 2v_{k,i}v_{k+1,i},$$

producing the sum of squares term with λ prefactor on the right hand side of (4.13). Proof is complete.

References

[1] D. Bertsekas, “Nonlinear Programming”, Belmont, MA, Athena Scientific, 2nd ed, 1999.

[2] S. Burer, R. Monteiro, *A nonlinear programming algorithm for solving semidefinite programs via low rank factorization*, Math. Programming, Ser. B, 95(2003), pp. 329-357.

[3] E. J. Candès, X. Li, Y. Ma, and J. Wright, *Robust Principal Component Analysis?*, Journal of ACM, 58(1), pp. 1-37, 2009.

[4] Y. Chen, A. Jalali, S. Sanghavi, and C. Caramanis, *Low-rank matrix recovery from errors and erasures*, IEEE Transactions on Information Theory, 59(7):4324–4337, 2013.

[5] D. Goldfarb and S. Ma, *Convergence of fixed point continuation algorithms for matrix rank minimization*, Foundations of Computational Mathematics, 11(2), 183–210, 2011.

[6] P. Huber, E. Ronchetti, “Robust Statistics”, Wiley, New York, 2009.

[7] D. Jannach, M. Zanker, A. Felfernig, G. Friedrich, “Recommender Systems: An Introduction”, Cambridge Univ. Press, 2012.

[8] I. Jolliffe, “Principal Component Analysis”, Springer-Verlag, 1986.

[9] R. Keshavan, A. Montanari, S. Oh, *Matrix completion from a few entries*, IEEE Trans. Info. Theory, 56 (6), 2980-2998, 2010.

[10] Y. Koren, R. Bell, C. Volinsky, *Matrix factorization techniques for recommender system*, Computer, vol. 42, pp. 30-37, 2009.

[11] T. Hastie, R. Mazumder, J. Lee, R. Zadeh, *Matrix Completion and Low-Rank SVD via fast Alternating Least Squares*, J. Machine Learning Research, 2015, to appear.

[12] M. Lai, Y. Xu, and W. Yin, *Improved iteratively reweighted least squares for unconstrained smoothed l_q minimization*, SIAM Journal on Numerical Analysis, 51(2):927–957, 2013.

[13] S. Ma, D. Goldfarb, and L. Chen, *Fixed point and bregman iterative methods for matrix rank minimization*, Mathematical Programming, 128(1-2):321–353, 2011.

[14] R. Mazumder, T. Hastie, R. Tibshirani, *Spectral regularization algorithms for learning large incomplete matrices*, J. Machine Learning Research, 11: 2287–2322, 2010.

[15] R. Meka, P. Jain, and I. S. Dhillon, *Guaranteed rank minimization via singular value projection*, In Proc. of Neural Information Processing Systems (NIPS), 2010.

[16] K. Mohan and M. Fazel, *Iterative reweighted algorithms for matrix rank minimization*, Journal of Machine Learning Research, 13(1):3441–3473, 2012.

[17] MovieLens Dataset: groupens.org/datasets/movielens.

[18] B. Recht and C. Ré, *Parallel stochastic gradient algorithms for large-scale matrix completion*, Mathematical Programming Computation, 5(2):201–226, 2013.

[19] N. Srebo, J. Rennie, T. Jaakkola, *Maximum margin matrix factorization*, Advances in Neural Info Processing Systems, 17, pp. 1329–1336, 2005.

[20] Z. Wen, W. Yin, and Y. Zhang, *Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm*, Mathematical Programming Computation, 4(4):333–361, 2012.

[21] H. F. Yu, C.-J. Hsieh, S. Si, and I. S. Dhillon, *Scalable Coordinate Descent Approaches to Parallel Matrix Factorization for Recommender Systems*, Proceedings of the IEEE International Conference on Data Mining(ICDM), pages 765-774, December 2012.