An Alternative DCA-based Approach for Reduced-Rank Multitask Linear Regression with Covariance Estimation

Hoai An Le Thi

Computer science and Applications Dept., LGIPM
University of Lorraine
Metz, France
hoai-an.le-thi@univ-lorraine.fr

Tao Pham Dinh

Laboratory of Mathematics, INSA - Rouen
University of Normandie
76801 Saint-Etienne-du-Rouvray Cedex, France
pham@insa-rouen.fr

Vinh Thanh Ho

Computer science and Applications Dept., LGIPM
University of Lorraine
Metz, France
vinh-thanh.ho@univ-lorraine.fr

Abstract—We investigate a nonconvex, nonsmooth optimization approach based on DC (Difference of Convex functions) programming and DCA (DC Algorithm) for the reduced-rank multitask linear regression problem with covariance estimation. The objective is to model the linear relationship between a multitask response and more explanatory variables by estimating a low-rank coefficient matrix and a covariance matrix. The problem is formulated as minimizing the constrained negative log-likelihood function as follows.

\[
\min \left[ \frac{1}{n} \sum_{i=1}^{n} (z_i - X\phi_i)^\top \Theta (z_i - X\phi_i) \right] - \log \det(\Theta)
\]

where \(\mathcal{X} = \{X \in \mathbb{R}^{m \times d} : \text{rank}(X) = r\}\) represents the low-rank constraint set and \(\mathcal{Y} = \{\Theta \in \mathbb{R}^{m \times m} : \Theta \succeq 0\}\) is the set of positive semi-definite matrices.

This problem has many real-world applications ranging from chemometrics (see e.g. [2]) to imaging neuroscience (see e.g. [3]), to quantitative finance and risk management (see e.g. [4]), to bioinformatics (see e.g. [5]), to robotics (see e.g. [6], [7]), to cite a few. For instance, in robotics, multivariate regression analysis is applied to evaluate the impact of robotic technique and high surgical volume on the cost of radical prostatectomy [6]. In another robotics application [7], linear regression analysis is performed to quantify the effect of surgeon experience on the operating time for each surgical step in the robotic-assisted laparoscopic prostatectomy procedure. In bioinformatics, the multitask regression algorithms are developed to solve the genomic selection problem in the fields of plant/animal breeding and genetic epidemiology (see [5] for more details).

In general, it is very hard to search globally optimal solutions to the problem (2) due to a double difficulty: first, the objective function of (2) is nonconvex in the variable \((X, \Theta)\), and, second, the rank function in the constraint set \(\mathcal{X}\) is discontinuous and nonconvex.

There are some existing approaches for solving the problem (2) which use an alternating optimization procedure on the variable \((X, \Theta)\). In particular, a classic Alternating Method (AM) will alternate between computing two variables \(X\) and

I. INTRODUCTION

In this paper, we consider the reduced-rank multitask linear regression problem with covariance estimation (see, e.g., [1]). Given \(m\) different tasks with the \(d\)-dimensional feature vector denoted \(\phi_i \in \mathbb{R}^d\), the corresponding response denoted \(z_i \in \mathbb{R}^m\) is generated using the linear model

\[
z_i = X\phi_i + \epsilon_i,
\]

where \(X \in \mathbb{R}^{m \times d}\) is an unknown matrix whose rows represent the coefficient vector for each task; the error \(\epsilon_i \in \mathbb{R}^m\) is assumed from a centered multivariate normal distribution with an unknown covariance matrix \(\text{Cov}(\epsilon_i) = (\Theta)^{-1}, \Theta \in \mathbb{R}^{m \times m}\).

The objective is to find the matrices \(X\) and \(\Theta\) from \(n\) data points \(\{(z_i, \phi_i)\}_{i=1,...,n}\). In the high-dimensional setting, the problem aims to minimize the constrained negative log-likelihood function as follows.

\[
\min \left[ \frac{1}{n} \sum_{i=1}^{n} (z_i - X\phi_i)^\top \Theta (z_i - X\phi_i) \right] - \log \det(\Theta)
\]

s.t. \(X \in \mathcal{X}, \Theta \in \mathcal{Y}\),

where \(\mathcal{X} = \{X \in \mathbb{R}^{m \times d} : \text{rank}(X) = r\}\) represents the low-rank constraint set and \(\mathcal{Y} = \{\Theta \in \mathbb{R}^{m \times m} : \Theta \succeq 0\}\) is the set of positive semi-definite matrices.
The standard DCA scheme is described below. Its convergence properties are given completely in [10], [12], [15].

**Standard DCA scheme**
Initialization: Let \( x^0 \in \mathbb{R}^p \) be a best guess. Set \( k = 0 \).
repeat
1. Calculate \( \bar{x}^k \in \partial h(x^k) \).
2. Calculate \( x^{k+1} = \text{argmin}\{g(x) - \langle x, \bar{x}^k \rangle : x \in \mathbb{R}^p\} \).
3. \( k = k + 1 \).
until convergence of \( \{x^k\} \).

Next, we briefly introduce partial DC programming and Alternative DCA [16]. A partial DC program takes the form
\[
\min F(x, y) := G(x, y) - H(x, y) \quad \text{s.t. } (x, y) \in \mathbb{R}^p \times \mathbb{R}^q,
\]
where \( G \) and \( H \) are partial convex functions in the sense that they are convex in each variable when fixing all other variables. Such a function \( F \) is called a partial DC function.

An alternative version of DCA for solving (3) consists in, at the iteration \( k \), alternatively computing \( x^{k+1} \) and \( y^{k+1} \) by performing one iteration of standard DCA for solving the following DC programs in variable \( x \) and \( y \), respectively:
\[
\min F(x, y^k) := G(x, y^k) - H(x, y^k) \quad \text{s.t. } x \in \mathbb{R}^p,
\]
and
\[
\min F(x^{k+1}, y) := G(x^{k+1}, y) - H(x^{k+1}, y) \quad \text{s.t. } y \in \mathbb{R}^q.
\]
This version, named Alternative DCA, is described as follows.

**Alternative DCA scheme**
Initialization: Let \( (x^0, y^0) \in \mathbb{R}^p \times \mathbb{R}^q \) be a best guess. Set \( k = 0 \).
repeat
1. Calculate \( \bar{x}^k \in \partial_x H(x^k, y^k) \).
2. Calculate \( x^{k+1} = \text{argmin}\{G(x, y^k) - \langle x, \bar{x}^k \rangle : x \in \mathbb{R}^p\} \).
3. Calculate \( \bar{y}^k \in \partial_y H(x^{k+1}, y^k) \).
4. Calculate \( y^{k+1} = \text{argmin}\{G(x^{k+1}, y) - \langle y, \bar{y}^k \rangle : y \in \mathbb{R}^q\} \).
5. \( k = k + 1 \).
until convergence of \( \{(x^k, y^k)\} \).

In the sequel, we present a reformulation of (2) and then show that it takes the form of a partial DC program for which the Alternative DCA scheme can be investigated.

**A. A Brief Introduction to Partial DC Programming and Alternative DCA**

DC programming and DCA were introduced by Pham Dinh Tao in a preliminary form in 1985 and have been extensively developed by Le Thi Hoai An and Pham Dinh Tao since 1994. DCA is well-known as an efficient approach in the non-convex programming framework. In recent years, numerous DCA-based algorithms have been developed for successfully solving large-scale nonsmooth/nonconvex programs in several application areas (see the list of references in [11], [14]). For a comprehensible survey on thirty years of development of DCA, the reader is referred to the recent work [11].

The standard DCA scheme is described below. Its convergence properties are given completely in [10], [12], [15].
Note that if \((X^*, \Theta^*)\) is a globally optimal solution to the problem (4) and \((X^*, \Theta^*) \in X \times Y\), then \((X^*, \Theta^*)\) is also a globally optimal solution to the problem (2).

It is easy to see that the function \(d_X^2\) is a DC function with DC decomposition
\[
d_X^2(X) = \min_{Y \in \mathcal{X}} \|X - Y\|_F^2 = \|X\|^2_F - \max_{Y \in \mathcal{X}} (2\langle X, Y \rangle - \|Y\|^2_F).
\]
As a result, the problem (4) can be expressed as a partial DC program
\[
\min F(X, \Theta) := G(X, \Theta) - H(X, \Theta)
\]
where
\[
G(X, \Theta) := \frac{1}{n} \sum_{i=1}^{n} (z_i - X \phi_i)^\top \Theta (z_i - X \phi_i) - \log \det(\Theta) + \chi_{\Theta \succeq 0}(\Theta),
\]
\[
H(X, \Theta) := \max_{Y \in \mathcal{X}} \left(2\langle X, Y \rangle - \|Y\|^2_F\right).
\]

Obviously, the functions \(G\) and \(H\) are partially convex.

C. Alternative DCA for solving the problem (5)

According to the Alternative DCA scheme in Section II-A, we need to construct two sequences \(\{(X^k, \Theta^k)\}\) and \(\{(U^k, V^k)\}\) such that
\[
U^k \in \partial H(X^k, \Theta^k),
\]
\[
X^{k+1} \in \operatorname{argmin}\{G(X, \Theta^k) - \langle X, U^k \rangle : X \in \mathbb{R}^{m \times d}\},
\]
and
\[
V^k \in \partial \Theta H(X^{k+1}, \Theta^k),
\]
\[
\Theta^{k+1} \in \operatorname{argmin}\{G(X^{k+1}, \Theta) - \langle \Theta, V^k \rangle : \Theta \in \mathbb{R}^{m \times m}\}.
\]

From the definition of the function \(H\), we compute the partial subdifferentials of \(H\) as follows:
\[
\partial H(X, \Theta) = 2\alpha \Theta \Omega\{\text{Proj}_\Theta(X)\} \text{ and } \partial \Theta H(X, \Theta) = \{0\}.
\]

Here \(\text{Proj}_\Theta\) and \(\text{co}(\mathcal{C})\) denote, respectively, the projection operator on the set \(\mathcal{C}\) and the convex hull of \(\mathcal{C}\).

We can choose the subgradients \(U^k \in \partial H(X^k, \Theta^k)\) and \(V^k \in \partial \Theta H(X^{k+1}, \Theta^k)\) as follows:
\[
U^k = 2\alpha W^k, \quad W^k \in \text{Proj}_\Theta(X^k), \quad \text{and } V^k = 0.
\]

Solving the convex subproblem (6) amounts to solving the problem
\[
\min_{X \in \mathbb{R}^{m \times d}} \left[\frac{1}{n} \sum_{i=1}^{n} (z_i - X \phi_i)^\top \Theta (z_i - X \phi_i) + \alpha \|X\|^2_F - \langle U^k, X \rangle\right].
\]
By setting the derivative of the objective function of the last problem (9) to zero, we can see that its optimal solution \(X^{k+1}\) satisfies the Sylvester equation
\[
A^k X + XB^k = C^k,
\]
where the matrices \(A^k \in \mathbb{R}^{m \times m}\), \(B^k \in \mathbb{R}^{d \times d}\), and \(C^k \in \mathbb{R}^{m \times d}\) are defined as
\[
A^k = \alpha(\Theta^k)^{-1}, \quad B^k = \frac{1}{n} \sum_{i=1}^{n} (\phi_i \phi_i^\top), \quad C^k = \alpha(\Theta^k)^{-1} W^k + \frac{1}{n} \sum_{i=1}^{n} (z_i \phi_i^\top).
\]

From (7) and the definition of \(G\), \(\Theta^{k+1}\) is an optimal solution to the convex program
\[
\min_{\Theta \geq 0} \left[\frac{1}{n} \sum_{i=1}^{n} (z_i - X^{k+1} \phi_i)^\top \Theta (z_i - X^{k+1} \phi_i) - \log \det(\Theta)\right].
\]

It is easy to check that the problem (11) has a closed-form optimal solution (see, e.g., [1]) as follows.
\[
\Theta^{k+1} = \left(\frac{1}{n} \sum_{i=1}^{n} (z_i - X^{k+1} \phi_i)(z_i - X^{k+1} \phi_i)^\top\right)^{-1}.
\]

Here \(Z^{-1}\) denotes an inverse of a matrix \(Z\).

Finally, the Alternative DCA scheme applied to (5) can be summarized in Algorithm 1 (ADCA).

**Algorithm 1 ADCA: Alternative DCA for solving (5)**

**Initialization:** Let \(\epsilon\) be a sufficiently small positive number. Let \(X^0 \in \mathbb{R}^{m \times d}\), \(\Theta^0 \in \mathbb{R}^{m \times m}\), \(\Theta^0 \geq 0\), \(\alpha > 0\). Set \(k = 0\).

**repeat**

1. Compute \(W^k \in \text{Proj}_\Theta(X^k)\).
2. Compute \(X^{k+1}\) by solving the Sylvester equation (10).
3. Compute \(\Theta^{k+1}\) using (12).
4. \(k = k + 1\).

**until** Stopping criteria are satisfied.

**Remark 1:** In numerical experiments, \(X^\star\) obtained by ADCA does often not belong to \(\mathcal{X}\). Thus, after stopping ADCA, we propose performing one projection step: projecting \(X^\star\) into the set \(\mathcal{X}\) and then updating \(\Theta^\star\) by (12).

III. NUMERICAL EXPERIMENTS

Our experiments aim to compare the proposed alternative algorithm ADCA with other alternating/joint algorithms for the multitask linear regression problem (2).

**Comparative algorithms.** As listed in Section I, we consider three alternating/joint algorithms for solving the problem (2): classic alternating method (AM), alternating method using gradient descent method (AGD) [1], and joint gradient method (JGD) [1] (see the Appendix for more details).

**Datasets.** We test the four algorithms ADCA, AGD, JGD, and AM on six synthetic datasets and eight real datasets.

We generate synthetic datasets using the linear model (1) similarly to the works, e.g., [1], [17]–[19]. Specifically, the feature vector \(\phi_i\) is drawn independently from a multivariate normal distribution \(N(0, \Sigma_\phi)\) where each element \(\Sigma_\phi(i, j) = 0.5^{i-j}\). Similarly, the error \(\epsilon_i\) is also generated.
from $N(0, \sigma^2 \Sigma_e)$ where $\sigma^2$ is chosen such that the corresponding signal-to-noise is equal to 1 (see, e.g., [1], [17]) and $\Sigma_e$ is defined by the following type: $AR(1)$, denoted $ar(\rho_e)$, with $\Sigma_{\phi}(i, j) = (\rho_e)^{|i-j|}$. Here, $\rho_e$ represents a correlation parameter; the larger its value is, the more the degree of dependence of errors would be. The coefficient matrix $X$ is computed as $X = AB$ where the orthonormal matrices $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times d}$ are generated form $N(0, 1)$. Finally, the respond vector $z_i \in \mathbb{R}^n$ is computed using (1). By setting $r = 3$, $m \in \{10, 20, 60\}$, $d \in \{10, 20, 40\}$, $\rho_e \in \{0, 0.5\}$, we have six synthetic datasets which are summarized in Table I. For each synthetic dataset, we generate 50 training samples and 1000 test samples in each run time, and we repeat the whole process 30 times.

As for real datasets, we test on eight benchmark multi-task regression datasets\(^1\). These datasets are collected from various interesting applications and can be found in the recent work [20] (see the references therein). The parameters of these datasets and the given values of $r$ are provided in Table III. We split each real dataset into a training set containing the first 75% of dataset and a test set containing the rest of dataset.

**Comparison criteria and stopping criteria.** We are interested in the following aspects: prediction error and CPU time (in seconds) for training the solution $(X^*, \Theta^*)$. As for synthetic datasets, the prediction error is defined by the mean squared error (MSE) [17]

$$\text{MSE} = \frac{\sum_{i=1}^{n} \|X \phi_i - AB \phi_i\|^2}{nm},$$

(13)

while the relative root mean squared error (RRMSE) on real datasets is used to measure the prediction error of the algorithm on each task and defined as [20]

$$\text{RRMSE} = \sqrt{\frac{\sum_{i=1}^{n} \|\tilde{z}_i - z_i\|^2}{\sum_{i=1}^{n} \|z_i\|^2}},$$

(14)

where $\tilde{z}_i$ is a respond vector estimated by the algorithm and $z_i$ is the mean value of the respond vectors on the training set. We stop the algorithms if the relative difference between two consecutive points $(X^{k-1}, \Theta^{k-1})$ and $(X^k, \Theta^k)$ or between two corresponding objective function values is less than or equal to $\varepsilon$.

**Set up parameters.** Our experiment is performed in MATLAB R2016b on a PC Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz of 8GB RAM. The MATLAB’s *sylvestre* function is used for solving Sylvester equation (10). All algorithms start with the same point $(X^0, \Theta^0)$. The starting point $X^0$ is set to a zero matrix in $\mathbb{R}^{n \times d}$, and the matrix $\Theta^0$ is computed using (12). To validate the performance of the algorithms on all synthetic/real datasets, we consider the following validation procedure: first we run the algorithm with the different parameters on the training set, then choose the solution $(X^*, \Theta^*)$ that furnishes the best objective function value $F(X^*, \Theta^*)$, and finally evaluate the obtained model using MSE (13) or RRMSE (14) on the test set. The ranges of parameters $\eta_X$, $\eta_{\Theta}$, and $\alpha$ are defined as: $\alpha \in \{5, 10, 100\}$, $\eta_X \in \{10^{-3}, 10^{-4}, \ldots, 10^2\}$, $\eta_{\Theta}$ belongs a geometric sequence from 5 to 400 [1]. The default tolerance is $\varepsilon = 10^{-3}$.

**Descriptions of result tables.** The average MSE and its standard deviation obtained by all comparative algorithms on six synthetic datasets over 30 run times are reported in Table I. The average results of training time of the algorithms on synthetic datasets are given in Table I. Table III shows the experimental results on real datasets in terms of RRMSE and training time.

**Comments on numerical results.**

**Synthetic datasets.** We observe from Table I that, in terms of MSE, ADCA is more efficient than AGD, JGD, and AM. To be specific, ADCA is the best on 5/6 datasets – the ratio of gain of ADCA versus AGD, JGD, and AM varies from 0.32% to 39.6%, from 1.60% to 75.1% and from 4.96% to 97.3%, respectively. Moreover, ADCA well performs for two model errors (independent, moderately correlated). In terms of

<table>
<thead>
<tr>
<th>$d$</th>
<th>$m$</th>
<th>$\Sigma_e$</th>
<th>ADCA</th>
<th>AGD</th>
<th>JGD</th>
<th>AM</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>ar(0.0)</td>
<td>2.44e-02</td>
<td>2.65e-02</td>
<td>2.14e-02</td>
<td>5.13e-01</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.47e-03</td>
<td>5.78e-03</td>
<td>5.71e-03</td>
<td>8.94e-01</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>ar(0.5)</td>
<td>2.00e-02</td>
<td>2.42e-02</td>
<td>2.40e-02</td>
<td>4.34e-01</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.15e-03</td>
<td>4.64e-03</td>
<td>4.81e-03</td>
<td>6.04e-01</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$d$</th>
<th>$m$</th>
<th>$\Sigma_e$</th>
<th>ADCA</th>
<th>AGD</th>
<th>JGD</th>
<th>AM</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>ar(0.0)</td>
<td>4.79e-02</td>
<td>5.83e-02</td>
<td>1.37e-01</td>
<td>5.04e-02</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.67e-02</td>
<td>2.39e-02</td>
<td>2.24e-02</td>
<td>1.65e-02</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>ar(0.5)</td>
<td>3.30e-02</td>
<td>5.47e-02</td>
<td>1.33e-01</td>
<td>3.56e-02</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.24e-02</td>
<td>2.87e-02</td>
<td>2.01e-02</td>
<td>1.10e-02</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>ar(0.0)</td>
<td>6.13e-02</td>
<td>6.15e-02</td>
<td>6.23e-02</td>
<td>2.34e+00</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8.01e-03</td>
<td>8.07e-03</td>
<td>8.28e-03</td>
<td>9.66e+00</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>ar(0.5)</td>
<td>6.35e-02</td>
<td>6.27e-02</td>
<td>6.31e-02</td>
<td>5.72e-01</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8.76e-03</td>
<td>8.54e-03</td>
<td>8.42e-03</td>
<td>5.96e-01</td>
<td></td>
</tr>
</tbody>
</table>

\(^1\)For the detailed descriptions of all datasets, the reader is referred to [20] and the website http://mulan.sourceforge.net/datasets-mtr.html.
training time, all four algorithms run very fast (less than 0.1 seconds).

**Real datasets.** The error RRMSE obtained by ADCA is the best on 6/8 datasets, especially the rf2 dataset with more than 7000 samples. In particular, as for the rf2 dataset, ADCA significantly outperforms AGD, JGD and AM with the ratio of gain of 92.6%, 92.6% and 85.8%, respectively. On other datasets, the ratio of gain varies from 1.18% to 77.5%, from 4.07% to 77.5% and from 22.5% to 99.9%. Comparing with AM, ADCA is worse on 2/8 datasets with the ratio from 5.36% to 9.19%. In Table III, training times of ADCA are reasonable (less than 1 seconds on 6/8 datasets and 25 seconds on the atp7d and rf2 datasets).

**IV. Conclusions**

We have investigated a new approach based on DC programming and DCA for solving the reduced-rank multitask linear regression problem with covariance estimation. An Alternative version of DCA, ADCA, has been developed. Numerical results on synthetic/real datasets have turned out that the ADCA is more efficient than exiting alternating/joint methods in terms of the prediction error and runs within a reasonable consuming time. In the future, we plan to extend this work in the future to study the convergence properties of ADCA and show the efficiency of ADCA on many other synthetic/real datasets with different model errors as well as various applications.

**APPENDIX**

**Comparative Algorithms for Solving the Problem (2)**

The **AM method** alternates between computing the variable \( X \) and \( \Theta \) at every iteration. In particular, at iteration \( k \), for fixed \( \Theta \), we need to compute \( X_{k+1} \), an optimal solution to the following problem (see, e.g., [9])

\[
\min_{X} \frac{1}{n} \sum_{i=1}^{n} (z_i - X \phi_i)^{\top} \Theta^k (z_i - X \phi_i) \text{ s.t. rank}(X) = r. \tag{15}
\]

Let us denote by \( Z \) (resp. \( \Phi \)) a matrix in \( \mathbb{R}^{m \times n} \) (resp. \( \mathbb{R}^{d \times n} \)) whose each column is a vector \( z_i \) (resp. \( \phi_i \)); and define \( D^k := (\Phi \Phi^{\top})^{(-1/2)} (\Phi Z^{\top}) (\Theta^k)^{(1/2)} \). A reduced-rank regression estimate \( X_{k+1} \) of (15) is given by

\[
X_{k+1} = \sum_{i=1}^{r} \lambda_i \left[ (1/n) \Phi \Phi^{\top} \right]^{(-1/2)} u_i v_i^{\top} (\Theta^k)^{(1/2)}, \tag{16}
\]

where the sequence \( \{ \lambda_i \} \) is the singular values of matrix \( D^k \); \( \{ u_i \} \) and \( \{ v_i \} \) are the left-hand and right-hand singular vectors of \( D^k \). For fixed \( X \), the AM computes the point \( \Theta_{k+1} \) using (12) at \( X_{k+1} \). Note that the AM method does not have any parameters.

**AGD: Alternating method using Gradient Descent method for solving (2)**

Initialization: Let \( \varepsilon \) be a sufficiently small positive number. Let \( X^0 \in \mathbb{R}^{m \times d}, \Theta^0 \in \mathbb{R}^{m \times m}, \Theta^0 \geq 0 \). Set \( k = 0 \).

repeat

1. Compute \( X^{k+1} \) using (17).
2. Compute \( \Theta^{k+1} \) using (12).
3. \( k = k + 1 \).

until Stopping criteria are satisfied.

**JGD: Joint Gradient Descent method for solving (2)**

Initialization: Let \( \varepsilon \) be a sufficiently small positive number. Let \( X^0 \in \mathbb{R}^{m \times d}, \Theta^0 \in \mathbb{R}^{m \times m}, \Theta^0 \geq 0 \). Set \( k = 0 \).

repeat

1. Compute \( X^{k+1} \) using (17).
2. Compute \( \Theta^{k+1} \) using (12).
3. \( k = k + 1 \).

until Stopping criteria are satisfied.

**References**


### TABLE III

Comparative results of ADCA, AGD, JGD, and AM in terms of the relative root-mean-squared error RRMSE defined by (14) (upper row) and training time in seconds (lower row) on eight real datasets. Bold values indicate the best result.

<table>
<thead>
<tr>
<th>Name</th>
<th>n</th>
<th>d</th>
<th>m</th>
<th>r</th>
<th>ADCA</th>
<th>AGD</th>
<th>JGD</th>
<th>AM</th>
</tr>
</thead>
<tbody>
<tr>
<td>andro</td>
<td>49</td>
<td>30</td>
<td>6</td>
<td>4</td>
<td>8.42e-01</td>
<td>3.75e+00</td>
<td>3.75e+00</td>
<td>1.22e+00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.17e-01</td>
<td>1.21e-03</td>
<td>6.32e-04</td>
<td>4.06e-03</td>
</tr>
<tr>
<td>atp7d</td>
<td>296</td>
<td>411</td>
<td>64</td>
<td>4</td>
<td>3.63e+00</td>
<td>4.60e+00</td>
<td>4.83e+00</td>
<td>2.74e+04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.45e+01</td>
<td>7.93e-03</td>
<td>3.05e-03</td>
<td>1.12e-01</td>
</tr>
<tr>
<td>oes10</td>
<td>403</td>
<td>298</td>
<td>16</td>
<td>4</td>
<td>5.18e-01</td>
<td>1.18e+00</td>
<td>1.26e+00</td>
<td>1.95e+01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.41e-01</td>
<td>2.82e-02</td>
<td>3.56e-03</td>
<td>7.53e-02</td>
</tr>
<tr>
<td>osales</td>
<td>639</td>
<td>376</td>
<td>12</td>
<td>4</td>
<td>9.89e-01</td>
<td>1.05e+00</td>
<td>1.05e+00</td>
<td>1.09e+03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.71e-02</td>
<td>4.61e-03</td>
<td>4.50e-03</td>
<td>1.34e-01</td>
</tr>
<tr>
<td>rf2</td>
<td>7679</td>
<td>576</td>
<td>8</td>
<td>6</td>
<td>1.09e-01</td>
<td>1.48e+00</td>
<td>1.48e+00</td>
<td>7.70e-01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.67e+00</td>
<td>2.54e-02</td>
<td>2.52e-02</td>
<td>3.48e-01</td>
</tr>
<tr>
<td>scpf</td>
<td>1137</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>9.88e-01</td>
<td>1.03e+00</td>
<td>1.03e+00</td>
<td>9.35e-01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6.48e-03</td>
<td>6.22e-04</td>
<td>6.66e-04</td>
<td>1.65e-03</td>
</tr>
<tr>
<td>sf1</td>
<td>323</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>9.61e-01</td>
<td>1.02e+00</td>
<td>1.07e+00</td>
<td>1.24e+00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.49e-03</td>
<td>1.40e-03</td>
<td>4.12e-04</td>
<td>1.51e-03</td>
</tr>
<tr>
<td>wq</td>
<td>1060</td>
<td>16</td>
<td>14</td>
<td>4</td>
<td>9.98e-01</td>
<td>1.01e+00</td>
<td>1.62e+00</td>
<td>9.14e-01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.48e-02</td>
<td>1.45e-02</td>
<td>9.50e-04</td>
<td>2.85e-03</td>
</tr>
</tbody>
</table>