Final Project Report

MATH-124: Optimization

Zarina Lin

linx@brandeis.edu

Dec 10, 2024

Contents

1 Introduction

In this report, we explore Coordinate Descent (CD), a method designed to solve high-dimensional optimization problems more efficiently. By simplifying a multivariate problem into a series of univariate problems, CD optimizes one coordinate (variable) at a time. This approach reduces the computational complexity of each iteration, making it both time-efficient and practical for real-world applications involving large datasets. Various variants of CD have been developed, including Cyclic Coordinate Descent (CCD), Greedy Coordinate Descent (GCD), Proximal Coordinate Descent (PCD), Block Coordinate Descent (BCD) and Accelerated Coordinate Descent, etc.

The report is structured as follows. In section 2, we will discuss the convergence theorem of CD, its computational complexity, and some other properties. In section 3, we discuss the convergence and iteration complexity of CD under some example objective functions. In section 4, we will provide detailed explanations of the experiments we conducted for some variants of CD and their implications.

2 Convergence Theorem

Coordinate Descent (CD): let $x^{(0)} \in \mathbb{R}^n$, and repeat:

$$x_i^{(k)} = \arg\min_{x_i} f\left(x_1^{(k)}, \ldots, x_{i-1}^{(k)}, x_i, x_{i+1}^{(k-1)}, \ldots, x_n^{(k-1)}\right), \quad \text{for } i = 1, \ldots, n \text{ and } k = 1, 2, 3, \ldots$$

Theorem 1:

For $f(x) = g(x) + \sum_{i=1}^{n} h_i(x_i)$, where $g(x)$ convex, differentiable, and each h_i convex, coordinate descent (CD) leads to global minimization.

Proof: Using convexity of g and subgradient optimality, we have:

$$\begin{aligned}
0 &\in \nabla_i g(x) + \partial h_i(x_i) \\
&\iff -\nabla_i g(x) \in \partial h_i(x_i) \\
&\iff h_i(y_i) \geq h_i(x_i) - \nabla_i g(x)(y_i - x_i) \\
&\iff \nabla_i g(x)(y_i - x_i) + h_i(y_i) - h_i(x_i) \geq 0
\end{aligned}$$

and by convexity of f, using the first-order characterization:

$$\begin{aligned} f(y) - f(x) &= g(x) - g(y) + \sum_{i=1}^{n} \left[h_i(y_i) - h_i(x_i) \right] \\ &\geq \sum_{i=1}^{n} \left[\nabla_i g(x)(y_i - x_i) + h_i(y_i) - h_i(x_i) \right] \geq 0 \end{aligned}$$

Building upon this result, we analyze the descent property and convergence of coordinate descent. For functions $f : \mathbb{R}^n \to \mathbb{R}$ that are coordinate β-smooth and α-strongly convex, the Descent Lemma ensures descent of CD: $f(x_{k+1}) - f(x_k) \leq -\frac{1}{2\beta} \left[\frac{\partial f(x_k)}{\partial x_j} \right]$ when using a step size $t = \frac{1}{\beta}$. Greedy Coordinate Descent (GCD), which selects the coordinate with the largest potential decrease at each iteration, converges with the bound:

$$f(x_{k+1}) - f(x_*) \leq (1 - \frac{\alpha}{\beta n})^k (f(x_0) - f(x_*))$$

indicating linear convergence. To achieve an error ε, GCD requires at least $k \geq \frac{\beta n}{\alpha} \log \left(\frac{f(x_0) - f(x_*)}{\varepsilon} \right)$ iterations, while Gradient Descent (GD) requires $k \geq \frac{\beta}{\alpha} \log \left(\frac{f(x_0) - f(x_*)}{\varepsilon} \right)$.This indicates that GCD needs n times more iterations than GD to achieve the same error, but takes n times less computation per iteration.

CD guaranteeing convergence for separable objectives makes CD a suitable and efficient method for solving problems like the Lasso regression, and this property will be further explored in our experiments.

3 Example Objectives

In this section, we examine the iteration complexity of coordinate descent (CD) under the quadratic objective and the Lasso objective, by comparing it with gradient descent (GD).

3.1 Quadratic

$$\min_x \frac{1}{2} x^T Q x + b^T x + c \tag{1}$$

Assume Q is positive definite with dimension $n \times n$, then $f(x)$ is strongly convex. Using Coordinate Descent (CD), the update rule for the j-th coordinate is given by:

$$x_j^{k+1} = x_j^k - \frac{1}{Q_{jj}} (\sum_{i \neq j} Q_{ji} x_i^k + b_j)$$

For each iteration, it takes $O(n)$ flops, and for a cycle of coordinate, it takes $O(n^2)$ flops. When Q is sparse, CD reduces memory usage. In contrast, Gradient Descent (GD) updates all n coordinates simultaneously at each iteration, requiring a full matrix-vector product with Q. This operation takes $O(n^2)$ flops per iteration.

3.2 LASSO

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{2}\|X\beta - y\|_2^2 + \lambda \|\beta\|_1 \tag{2}$$

Let $X \in \mathbb{R}^{n \times p}$ and $\beta \in \mathbb{R}^p$. To minimize this objective, Coordinate Descent (CD) updates each coefficient β_i, while keeping all other coefficients fixed. For the i-th coefficient, the update solves:

$$\beta_i = \arg\min_{\beta_i} \frac{1}{2}\|y - X\beta\|_2^2.$$

Expanding the terms and setting the derivative to zero gives:

$$0 = \nabla_i f(\beta) = X_i^T (X\beta - y) = X_i^T (X_i\beta_i + X_{-i}\beta_{-i} - y)$$

where X_i is the i-th column of X, and X_{-i} and β_{-i} are the matrix and vector with the i-th column and coefficient removed, respectively. Solving for β_i, we get:

$$\beta_i = \frac{X_i^T (y - X_{-i}\beta_{-i})}{X_i^T X_i}.$$

Thus, the update rule is as:

$$\beta_i^{k+1} = S\left(\frac{X_i^T \left(y - X_{-i}\beta_{-i}^k\right)}{X_i^T X_i}, \lambda\right) \tag{3}$$

where the soft-thresholding operator $S(\hat{\beta}_i, \lambda)$ is defined as:

$$S(\hat{\beta}_i, \lambda) = \begin{cases} \hat{\beta}_i + \lambda & \text{if } \hat{\beta}_i < -\lambda, \\ 0 & \text{if } |\hat{\beta}_i| \leq \lambda, \\ \hat{\beta}_i - \lambda & \text{if } \hat{\beta}_i > \lambda. \end{cases}$$

Each coordinate costs $O(n)$ to update $y - X_{-i}\beta_{-i}$, and $O(n)$ to compute $X_i^T (y - X_{-i}\beta_{-i})$. So the computational cost for one cycle of CD is $O(np)$ which is the same as GD (the number of coefficients p is the dominant term in the computation).

4 Experiments

In this section, we will present three experiments conducted for different variants of Coordinate Descent (CD), covering topics ranging from basic convergence analysis to applications in empirical studies.

4.1 Comparing the convergence rate of cyclic coordinate descent and gradient descent

Objective Function

The least squares objective for linear regression:

$$F(x) = \frac{1}{n}\|Ax-b\|^2 = \frac{1}{n}\sum_{i=1}^{n}\left(a_i^T x - b_i\right)^2$$

where:

- A is the $n \times m$ feature matrix,
- x is the m-dimensional parameter vector we are optimizing,
- b is the n-dimensional vector of observed target values,
- a_i^T represents the i-th row of A.

Gradient of $F(x)$:

The gradient with respect to x is:

$$\nabla F(x) = \frac{2}{n}A^\top(Ax-b)$$

Gradient Descent (GD) Rule:

The GD update rule should be:

$$x^{(k)} \leftarrow x^{(k-1)} - t\nabla F(x^{(k-1)})$$

Substituting $\nabla F(x)$, we get:

$$x^{(k)} \leftarrow x^{(k-1)} - t \cdot \frac{2}{n}A^\top(Ax^{(k-1)} - b)$$

Cyclic Coordinate Descent (CCD):

For CCD, the objective is minimized one coordinate x_j at a time, fixing all others. The partial derivative with respect to x_j is:

$$\frac{\partial F(x)}{\partial x_j} = \frac{2}{n}\sum_{i=1}^{n} a_{ij}\left((a_i^\top x) - b_i\right)$$

The update rule for CCD would then be:

$$x_j^{(k)} \leftarrow x_j^{(k-1)} - t\frac{2}{n}\sum_{i=1}^{n} a_{ij}\left((a_i^\top x^{(k-1)}) - b_i\right)$$

Generated Data Set

The data set for the experiment is generated in the following way with $n = 5000, m = 1000, z = 200$:

- The entries of the observation matrix $A \in \mathbb{R}^{n\times m}$are independently and identically distributed (i.i.d.) samples drawn from a standard Gaussian distribution $N(0,1)$.
- The true coefficient vector $x \in \mathbb{R}^m$ is generated such that it contains z zero entries and the rest of $m-z$ entries are ones.
- The error term $\varepsilon \in \mathbb{R}^n$ consists of i.i.d. samples drawn from $N(0,1)$.
- The response vector $b \in \mathbb{R}^n$ is computed as: $b = Ax + \varepsilon$
- We chose a fixed step size $t = \frac{1}{\lambda_{max}(\frac{A^T A}{n})}$

Comparison of Convergence Rate

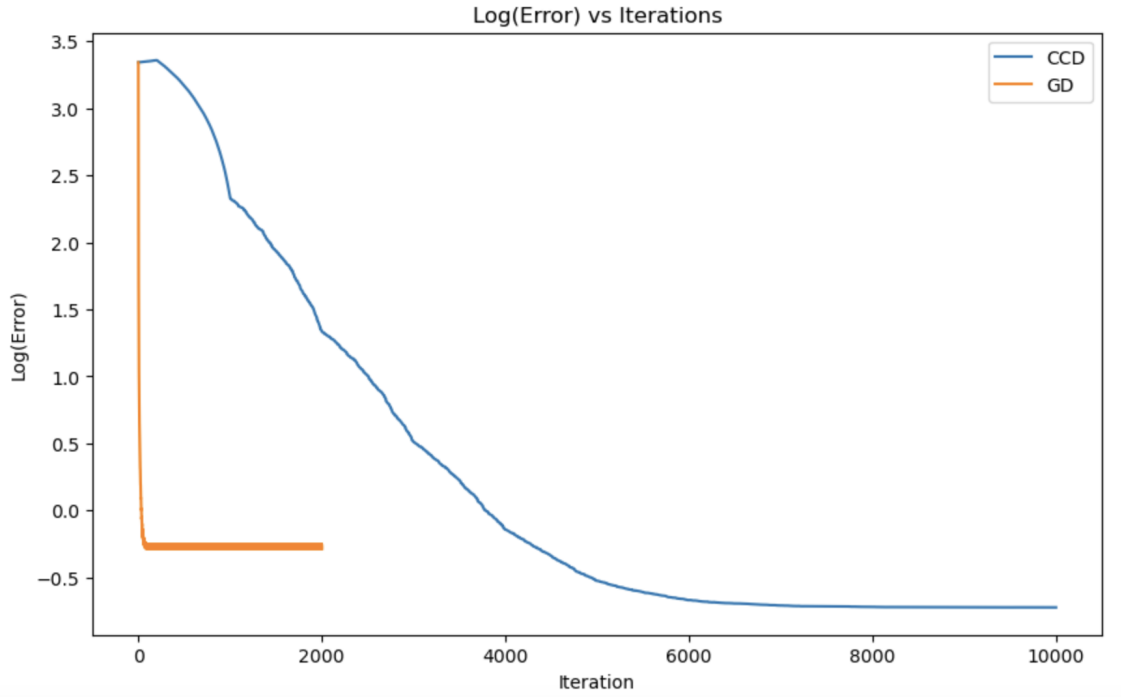

Figure 1: Comparison of GD and CCD in terms of convergence rate

Analysis for Convergence Rate

Figure 1 compares the convergence of Cyclic Coordinate Descent (CCD) and Gradient Descent (GD). In the figure, GD is represented by an orange line and converges much faster, in approximately 100 iterations, becuase it simultaneously updates all coordinates in each iteration. However, CCD, represented by the blue line, requires significantly more iterations to converge, approximately 4,000 iterations, to achieve a comparable level of error as GD. This aligns with the theoretical expectations that coordinate descent methods typically require more iterations to reach the same error levels as gradient descent.

As discussed in Section 2, Greedy Coordinate Descent (GCD) generally needs n times more iterations than GD to achieve the same error, where n is the dimension of the objective vector x. In our experiments, since $x \in \mathbb{R}^{1000}$, it would theoretically be expected that GCD would require approximately 1000×100 more iterations than GD. It is important to distinguish here that CCD, which updates each coordinate in a cyclic order without selecting the most effective coordinate, could potentially require even more iterations than GCD. This suggests that CCD may need more than 100,000 iterations to achieve the same level of error as GD. However, in our experiment, the true coefficient vector x is sparse, and this sparsity leads CCD to reach the same error level as GD

in just 4,000 iterations. This result illustrates that the specific structure of the problem, such as sparsity, can enhance the convergence of CCD.

4.2 Path-wise Coordinate Descent for LASSO

The LASSO model, characterized by an l_1-penalty on the least squares objective, promotes sparsity in the solution by shrinking some coefficients exactly to zero. A key challenge in applying LASSO is selecting the optimal parameter λ, which balances the trade-off between prediction accuracy and solution sparsity. Larger λ values produce sparser solutions but may compromise predictive accuracy, while smaller λ can lead to dense solutions and potential overfitting. Efficiently determining the optimal λ is especially important when working with large datasets. In this experiment, we explore the application of path-wise coordinate descent, an efficient algorithm that leverages warm starts and sparsity, to identify the optimal λ in practice.

In this experiment, we solved the LASSO objective Eq.(2) over a path of λ values, where $[\lambda_{\max} > \lambda_2 > \lambda_3 > \cdots > \lambda_{\min}]$, using Cyclic Proximal Coordinate Descent (CPCD). The update rule is described in section 3 Eq.(3)

For each λ in the parameter path, we update $\hat{\beta}$ by cycling through its coordinates until convergence. To ensure a warm start, the solution $\hat{\beta}_\lambda$ is used as the initial value for solving $\hat{\beta}_{\lambda+1}$. This approach reduces the number of iterations required for convergence compared to initializing with a zero vector, which significantly improves efficiency, especially when fitting the model over a long path of λ for a large data set.
We conducted the experiment on a generated dataset with the following setup, where $n = 300, p = 100$, and $m = 20$:

- The entries of the observation matrix $X \in \mathbb{R}^{n\times p}$are independently and identically distributed (i.i.d.) samples drawn from a standard Gaussian distribution $N(0,1)$.
- The true coefficient vector $\beta \in \mathbb{R}^p$ is generated such that it contains m zero entries and $p-m$ nonzero entries. The nonzero entries are i.i.d. samples drawn from a Gaussian distribution $N(1,1)$ (mean 1 and variance 1). After generating β, we randomly permute its entries.
- The error term $\varepsilon \in \mathbb{R}^n$ consists of i.i.d. samples drawn from $N(0,1)$.
- The response vector $y \in \mathbb{R}^n$ is computed as:$y = X\beta + \varepsilon$

The λ path is selected in the following way. It starts with $\lambda_{\max} = \frac{2\cdot\max(|X^T y|)}{n}$, where $\lambda_{\max}$ is the smallest value of λ that shrinks all coefficients to zero. A sequence of 10 values is then generated logarithmically from $\lambda_{\max}$ down to $0.001 \cdot \lambda_{\max}$. This approach ensures a smooth transition from strong regularization to weak regularization.

Figure 2 illustrates the solution $\hat{\beta}$ over the λ path. Starting from the right side of the axis (corresponds to a large λ value), all coefficients $\hat{\beta}_i$ are initially shrunk to zero. As λ decreases, more coefficients become non-zero, demonstrating that larger values of λ promote sparser solutions. By shrinking most coefficients to zero and gradually introducing nonzero entries, the method reduces memory usage. The coefficient path can also be used to assess feature importance: at larger λ values, only the most important features with strong correlations to the response are nonzero, while weaker predictors enter the model as λ decreases.

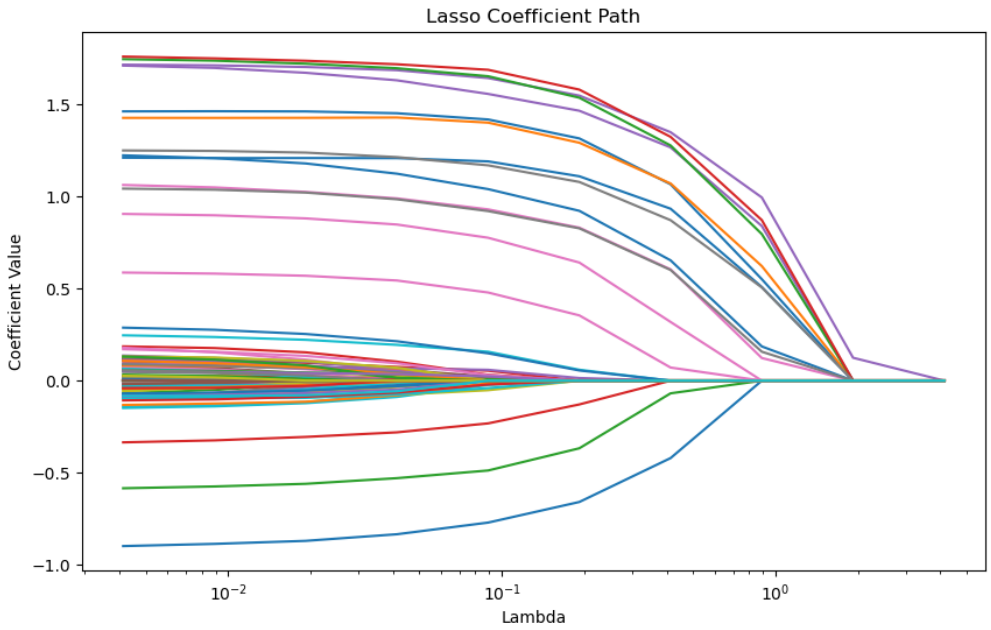

Figure 2: Lasso coefficient path

Figures 3 and 4 illustrate the convergence of CPCD across the λ path, comparing its performance with and without warm starts. In Figure 3, warm starts initialize $\hat{\beta}$ using the solution from the previous (larger) λ, significantly reducing the number of iterations required for convergence. For most λ values, CPCD converges in approximately 100 iterations. In contrast, Figure 4 shows that initializing $\hat{\beta}$ to zero for each λ increases the number of iterations required for convergence, with up to 400 iterations needed for the smallest λ. This underscores the computational efficiency of warm starts, particularly for data sets with large n and p, as each iteration of CPCD costs $O(n)$ flops and one cycle costs $O(np)$.

Figures 3 and 4 also illustrate the behavior of the estimator $\hat{\beta}$ under varying λ. Larger λ values promote sparsity in the solution, but the estimation error is high. As λ decreases, the error reduces until reaching an optimal λ that balances the trade-off between sparsity and accuracy. Beyond this point, further reductions in λ lead to overfitting, causing the error to increase. This is evident in Figure 3, where the pink curve ($\lambda \approx 0.04$) represents the optimal λ, balancing regularization and

model fit.

In conclusion, Path-wise Coordinate Descent is an efficient method for solving the LASSO problem and identifying the optimal regularization parameter λ, as it leverages the problem structure to avoid redundant computations.

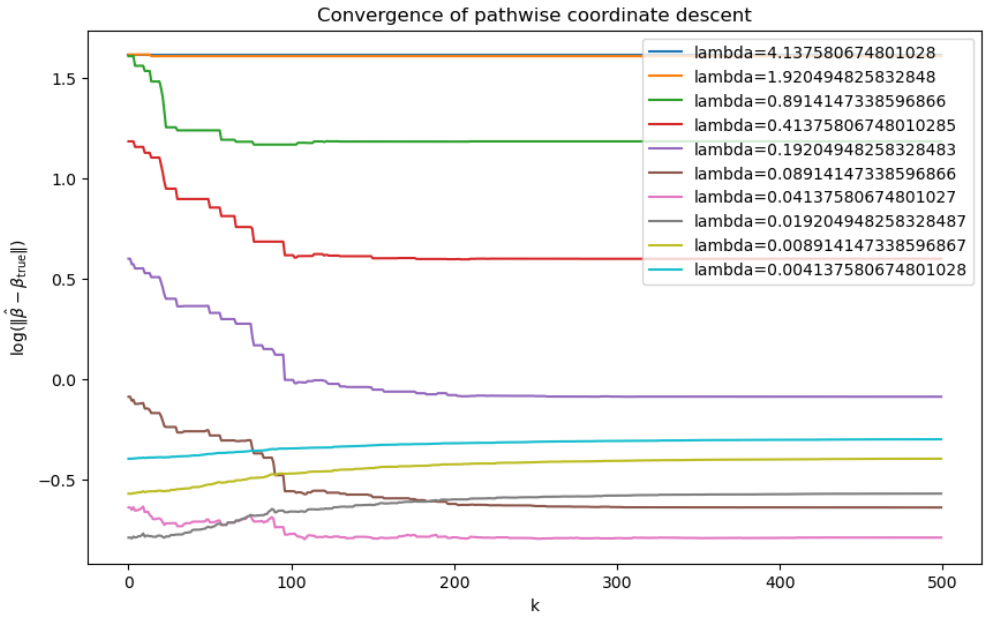

Figure 3: Convergence of CPCD using warm starts

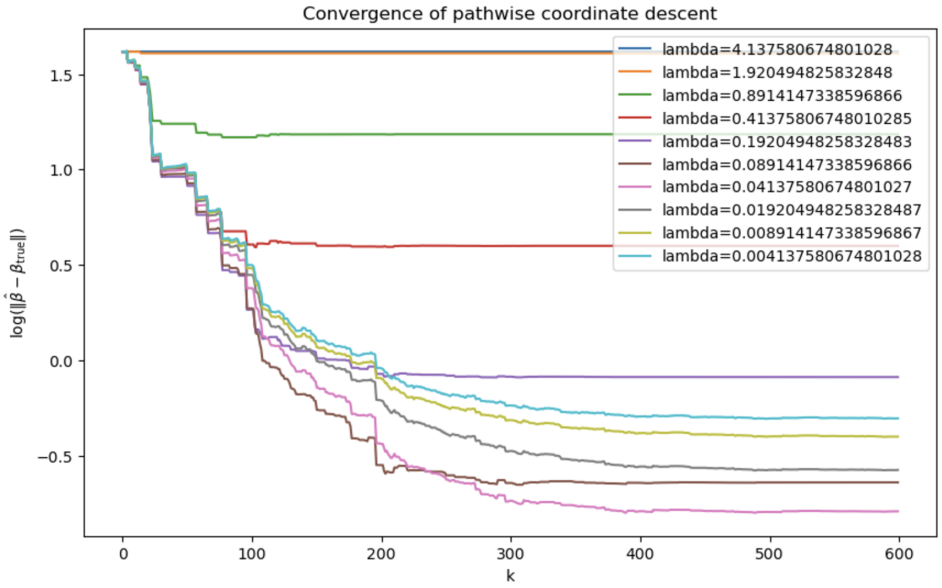

Figure 4: Convergence of CPCD without warm starts

4.3 Cyclic Coordinate Descent for Matrix Completion

This experiment investigates the application of coordinate descent to practical problems, focusing specifically on matrix completion, a common task in online user recommendation systems. Matrix completion involves predicting missing entries in a user-item rating matrix based on observed ratings. For this study, we utilized the MovieLens dataset , which contains 610 users, 9724 movies, and a subset of known user-movie ratings a_{ij} for user i and movie j. The goal is to predict the unknown ratings using the available data.

The rating matrix $A \in \mathbb{R}^{m\times n}$ can be filled through a low-rank approximation $A \approx WH^T$, where $W \in \mathbb{R}^{m\times k}$ represents the user matrix, and $H \in \mathbb{R}^{n\times k}$ represents the movie matrix. The optimization problem for this matrix completion task is formulated as:

$$\min_{W,H} \sum_{(i,j)\in\Omega} \left(A_{ij} - \mathbf{w}_i \mathbf{h}_j^T\right)^2 + \lambda \left(\|W\|_F^2 + \|H\|_F^2\right), \tag{4}$$

where Ω is the set of index pairs corresponding to observed ratings, and λ is a regularization parameter to prevent overfitting.

There are several alternative ways for solving the above problem. In this experiment, we applied Cyclic Coordinate Descent (CCD) and the Alternating Least Squares Method (ALS) to solve for W and H and compare the convergence and running time.

- Update rule for ALS:alternately optimizing W and H keeping the other fixed

$$w_i^* = (H_{\Omega_i}^T H_{\Omega_i} + \lambda I)^{-1} H_{\Omega_i} a_{\Omega_i}, \;\; h_j^* = (W_{\Omega_j}^T W_{\Omega_j} + \lambda I)^{-1} W_{\Omega_j} a_{\Omega_j} \tag{5}$$

- Update rule for CCD: fixing all other variables except for only one variable w_{it}

$$w_{it}^* = \frac{\sum_{j\in\Omega_i} \left(R_{ij} + w_{it}h_{jt}\right) h_{jt}}{\lambda + \sum_{j\in\Omega_i} h_{jt}^2}, \;\; h_{jt}^* = \frac{\sum_{i\in\Omega_j} \left(R_{ij} + w_{it}h_{jt}\right) w_{it}}{\lambda + \sum_{i\in\Omega_j} w_{it}^2} \tag{6}$$

where $R_{ij} = A_{ij} - w_i h_j^T$ and the residual after updating w_{it} and h_{jt} will be updated as:

$$R_{ij} \leftarrow R_{ij} - (w_{it}^* - w_{it})\, h_{jt}, \; \forall j \in \Omega_i, \quad R_{ij} \leftarrow R_{ij} - \left(h_{jt}^* - h_{jt}\right) w_{it}, \; \forall i \in \Omega_j$$

In theory, ALS requires less iterations to converge but has high computational costs because its update rule Eq.(5) involves matrix multiplication and inverse operations, which take $O(|\Omega_i|k^2)$ and $O(k^3)$ respectively. So one iteration to update all vraibles in W and H costs $O(|\Omega|k^2 + (m+n)k^3)$ flops. However, CCD requires less computation for each iteration than ALS. Its update rule for a single variable Eq.(6) costs $O(|\Omega_i|)$ and updating all variables of W and H in one iteration takes

$O(|\Omega|k)$, but it converges slower than ALS.

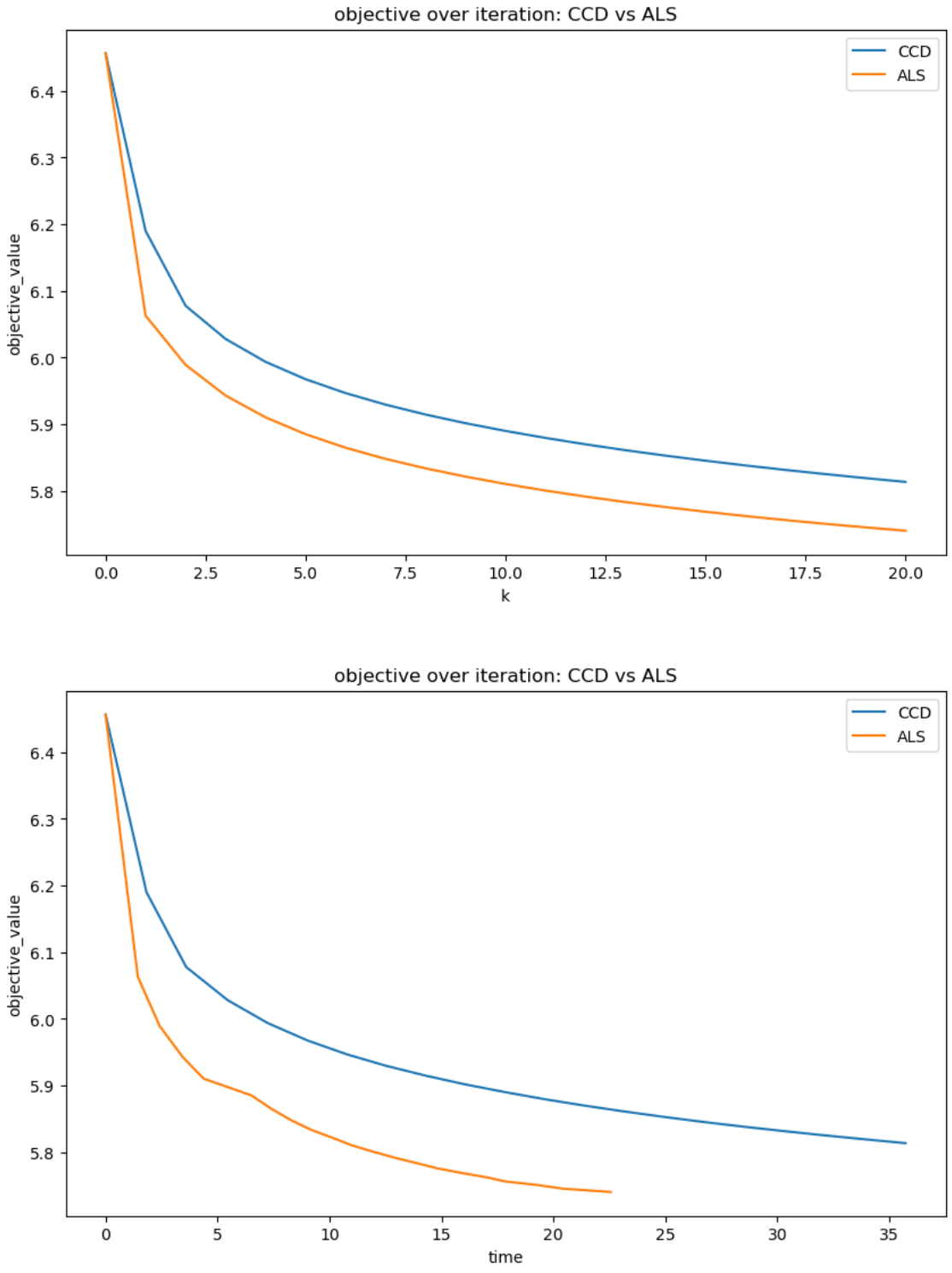

Figure 5: Comparison of CCD and ALS in terms of convergence and run time.

In our experiment, we selected a rank of $k = 15$ and a regularization parameter of $\lambda = 0.1$. Both W and H were randomly initialized, and we ran the algorithms for 20 iterations. The objective function value (Eq. (4)) was plotted on a log scale, both over iterations and time.

Figure 5 presents the results of our experiment. We observed that ALS showed a faster convergence trend, as its objective value at each iteration was lower than that of CCD, aligning with the theoretical expectations. However, ALS also outperformed CCD in terms of runtime, completing 20 iterations in 22.56 seconds compared to 35.78 seconds for CCD. This contradicts the theory that CCD is computationally cheaper. The discrepancy could be due to the additional computational cost of updating the residual matrix R. In CCD, each variable in W and H is updated sequentially, and the residual matrix R is updated after every variable update, which increases the computation for each iteration. Other possible factors might be the condition or sparsity of the rating matrix A. The exact reason for this discrepancy requires further investigation, but it highlights the importance of implementation and dataset characteristics in determining runtime efficiency.

Despite ALS converging faster and taking less time in this experiment, CCD achieved more accurate predictions for our dataset,given the chosen rank and regularization parameter. The dataset was divided into training and test sets: the training set was used to fit the model by solving for W and H, while the test set was used to evaluate performance using the root mean squared error (RMSE). After 20 iterations, CCD achieved an RMSE of 1.3149, which is lower than the 1.4184 achieved by ALS. But during our experiment, we found that different choice of rank k and the regularization parameter λ largely affect the performance, reflecting that choosing the right parameter and rank is important for matrix completion in practice.

5 References

1. Convergence Theorem for Coordinate Descent:
   https://www.stat.cmu.edu/ ryantibs/convexopt/scribes/coord-desc-scribed.pdf

2. Pathwise Coordinate Descent:
   https://arxiv.org/pdf/0803.3876

3. Data source for Matrix Completion:
   https://aigamer28100.github.io/Movie-Lens-Dataset-Visualization-and-Prediction/

4. Cyclic Coordinate Descent for Matrix Completion:
   https://www.sciencedirect.com/science/article/abs/pii/S0020025519309284

5. Coordinate Descent Algorithms:
   https://link.springer.com/article/10.1007/s10107-015-0892-3

6. Anderson Acceleration of Coordinate Descent:
   https://proceedings.mlr.press/v130/bertrand21a

7. JAXopt Fixed Point Resolution:
   https://arxiv.org/abs/2105.15183