Adaptive Coordinate Descent for High-Dimensional Logistic Regression: A Comparative Study

Zhicheng Wang[a]

[a]*University of California San Diego, 9500 Gilman Dr, La Jolla, 92093, CA, United States*

Abstract

In this project we introduces a coordinate descent optimization method for high-dimensional unconstrained optimization problems. Our approach iteratively updates one coordinate at a time to minimize a given cost function $L(w)$, where $w \in \mathbb{R}^d$. Our method employs an adaptive strategy for coordinate selection. This strategy prioritizes updates based on the potential reduction in the loss function, informed by the gradient of $L(w)$ with respect to each coordinate, enabling a more efficient path to convergence.

1. High Level Description

Problem a: Which Coordinate to Choose?

Instead of randomly picking a coordinate to update, our strategy evaluates the potential impact of updating each coordinate on the loss function, selecting the one that promises the most significant reduction in loss. This is achieved by examining the gradient of $L(w)$ with respect to each coordinate and taking the one with the highest absolute gradient.

Problem b: How to set the new value of w_i

$$\mathrm{w}_i := w_i - \left(\frac{\alpha}{1+\beta \cdot iteration}\right) \cdot gradient$$

- w_i is the weight for the i-th feature.
- α is the base learning rate
- β is the rate at which the learning rate decreases over iterations

The new value of w_i is determined by taking a step in the direction opposite to the gradient of the loss function with respect to that coordinate. The function adjusts the learning rate based on the iteration number. The learning rate decreases over time, following a simple adaptive scheme where it's inversely proportional to the iteration number.

Differentiability Requirement

Our method ideally requires the cost function $L(w)$ to be differentiable with respect to the coordinates of w since it relies on gradient information for both coordinate selection and update steps. However, for non-differentiable points or functions, we can adapt the method to use subgradients or other methods to extend the applicability of our coordinate descent method.

2. Convergence

Our adaptive coordinate descent method is likely to converge to the optimal loss under conditions where the cost function is convex and smooth, as these properties ensure a predictable and stable optimization path. The method's efficiency is further enhanced by appropriate feature scaling and the selection of an optimal step size, which prevents overshooting and ensures steady progress towards the minimum.

3. Experiment Results

3.1. Standard logistic regression solver

The final logistic loss (L^*) was calculated to be approximately $1.4917342015639051 \times 10^{-6}$, indicating an exceptionally close fit to the training data.

3.2. Adaptive vs Random Coordinate Descent

Fixed Learning Rate - α = 0.8

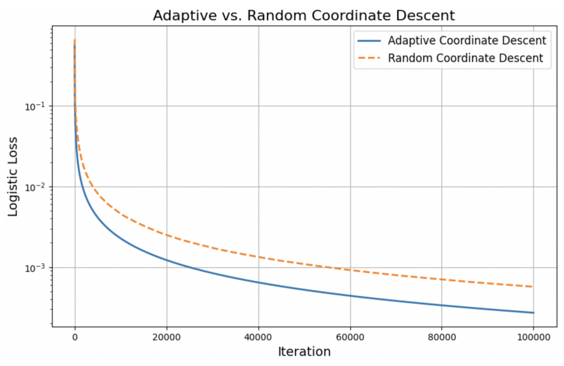

Figure 1: Adaptive vs Random Coordinate Descent

Method	Final Loss
Adaptive Coordinate Descent	0.00027124635451439977
Random Coordinate Descent	0.00057058079047400091O

Table 1: Final Loss Comparison between Adaptive and Random Coordinate Descent

Adaptive Learning Rate - $\alpha = 35, \beta = 0.001$

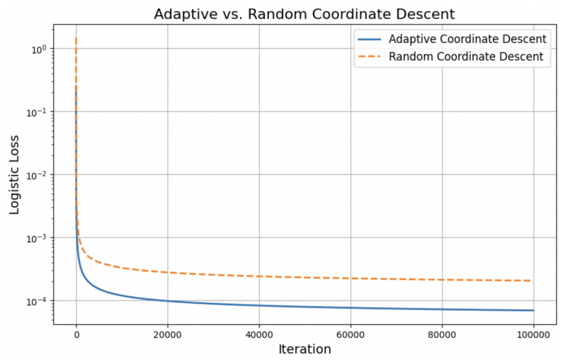

Figure 2: Adaptive vs Random Coordinate Descent

Method	Final Loss
Adaptive Coordinate Descent	$6.955304995670117 \times 10^{-6}$
Random Coordinate Descent	$2.0637610068404295 \times 10^{-4}$

Table 2: Final Loss Comparison between Adaptive and Random Coordinate Descent

The alpha and beta parameters were determined through a rigorous process of trial and error, ensuring optimal learning rate adjustments over the training iterations.

Based on the graphical analysis, it is evident that our adaptive method converges significantly faster than the approach of randomly selecting coordinates. The adaptive learning rate allows for a higher starting learning rate because it can adjust the learning rate dynamically based on the iteration number. This dynamic adjustment is particularly useful such that it allows faster convergence at the start, avoiding overshooting and a closer convergence to the optimal solution.

4. Critical Evaluation

There are several ways for improvement when focusing on optimizing performance and convergence rates. Including the following:

Dynamic Learning Rates: Adjust the learning rate dynamically for each coordinate based on its performance in previous iterations.

Better Coordinate Selection Strategies: Beyond random and adaptive selection, other strategies could be employed to choose the next coordinate to update.

Second-Order Methods: Incorporating information through second-order methods like Newton's method for the coordinate update step can significantly improve convergence rates. These methods adjust the step size using the second derivative of the loss function, which can lead to faster convergence but at the cost of increased computational complexity per iteration.

Sparsity Exploitation: If the dataset or the solution is sparse, techniques that exploit this sparsity can improve efficiency.

5. Sparse Coordinate Descent

Since we are searching within a restricted subset of the space. This constraint can make the problem non-convex in terms of the sparsity constraint, even if the original loss function is convex. Hence, the method may not always find the best k-parse solution.

k	Adaptive Coordinate Descent	Random Coordinate Descent
3	0.07238878875530039	0.07034434932359918
4	0.04224022855878221	0.0398680505968498
10	0.0013056294270390005	0.0002052885146153566
11	0.0002723661765523559	0.00018557808176622068

Table 3: Final Loss Values for Selected k using Adaptive and Random Coordinate Descent

6. Second Order Approximations

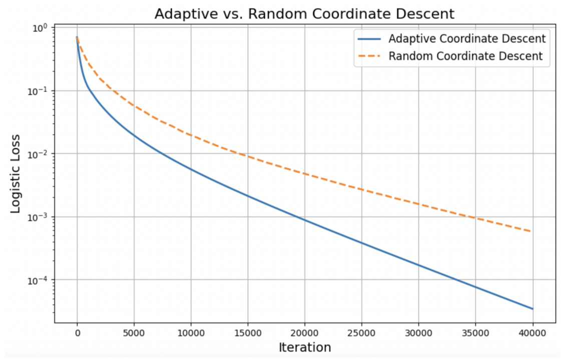

Figure 3: Adaptive vs Random Coordinate Descent

Method	Final Loss
Adaptive Coordinate Descent	$3.434152158460525 \times 10^{-5}$
Random Coordinate Descent	0.0005728804676173863

Table 4: Final Loss Comparison between Adaptive and Random Coordinate Descent