Coordinate Descent Methods for Logistic Regression: A Comparative Study on the Wine Dataset

Allen Zou

February 26, 2024

Abstract

Logistic regression is a widely employed method for solving binary classification problems, often necessitating iterative optimization techniques. This study delves into coordinate descent methods, with a specific focus on the application of the greedy coordinate descent algorithm to logistic regression. A comparative analysis is conducted with a random-feature coordinate descent approach, exploring convergence properties. Experimental results are presented to showcase the superior efficiency and behavior of the greedy coordinate descent algorithm over random-feature coordinate descent. Additionally, we extend the greedy coordinate descent algorithm into a k-sparse setting and assess its performance across varying k values.

1 Introduction

Logistic regression stands as a fundamental tool for binary classification, frequently demanding iterative optimization strategies for model refinement. This study investigates coordinate descent methods, honing in on the application of the greedy coordinate descent algorithm to logistic regression. As an alternative to traditional optimization techniques, the simplicity of coordinate descent prompts a more detailed exploration.

2 Coordinate Descent Strategies

Coordinate descent involves updating one variable of the weight vector at a time while holding others fixed. In the context of logistic regression, the goal is to iteratively adjust the coefficients to minimize the cost function. This study compares the greedy and random-feature coordinate descent algorithms in logistic regression optimization. The greedy method selects the feature with the largest magnitude, while the random-feature method updates a randomly chosen feature. Both algorithms incorporate backtracking line search with the Armijo condition, ensuring a sufficient reduction in the objective function that justifies the chosen step size. The purpose of the line search is to dynamically adjust the step size (α) in each iteration, allowing for a more efficient and effective convergence towards the optimal solution. The Armijo condition serves as a criterion to ensure the chosen step size meets the desired reduction in the objective function. It is defined as:

$$f(w + \alpha \cdot d) \leq f(w) + c \cdot \alpha \cdot \nabla f(w)^T d$$

Here, w is the weight vector, d is the search direction, and c is a constant.

2.1 Assumptions

The coordinate descent algorithms presented here assume the differentiability of the loss function $L(w)$. This assumption allows for the computation of gradients necessary for weight updates. However, further assumptions regarding continuous second-order derivatives are not made in this study.

2.2 Pseudocode

Algorithm 1 Random-Feature Coordinate Descent

Require: Initial weights w, Dataset X, Labels y, Learning rate γ, Number of iterations T

for $t = 1$ to T **do**
  $dw \leftarrow$ Compute $\frac{\partial L}{\partial w}$ using w
  $i \leftarrow$ Randomly choose from $[0, d)$
  $\alpha \leftarrow$ line_search(w, dw, i) ▷ Step size
  $w_i \leftarrow w_i - \alpha \cdot dw_i$
end for
return w

Algorithm 2 Greedy Coordinate Descent

Require: Initial weights w, Dataset X, Labels y, Learning rate γ, Number of iterations T

for $t = 1$ to T **do**
  $dw \leftarrow$ Compute $\frac{\partial L}{\partial w}$ using w
  $i \leftarrow \text{argsort}(|dw|)$
  $\alpha \leftarrow \text{line\_search}(w, dw, i)$ ▷ Step size
  $w_i \leftarrow w_i - \alpha \cdot dw_i$
end for
return w

Algorithm 3 Backtracking Line Search

Require: Weights w, Gradients dw, Coordinate i

$\alpha \leftarrow 1.0$
$c \leftarrow 10^{-4}$ ▷ constant for Armijo condition
while True **do**
  $old\_loss \leftarrow$ Compute log loss using w
  $w' \leftarrow$ deep copy w
  $w'_i \leftarrow w'_i - \alpha \cdot dw_i$
  $new\_loss \leftarrow$ Compute log loss using w'
  $armijo \leftarrow c \cdot \alpha \cdot ||dw||^2$
  if $new\_loss \leq old\_loss - armijo$ **then**
    break
  end if
end while
return α

3 Experimental Setup

For this study, the Wine dataset was employed, initially consisting of 178 data points across 13 dimensions and belonging to 3 classes. To create a binary classification problem, the dataset was narrowed down to include only the first two classes, resulting in 130 data points—59 from the first class and 71 from the second. An 8:2 split was applied for training and testing.

Both greedy and random-feature coordinate descent algorithms were executed with 1,000 runs each, where each run encompassed 50 iterations. The choice of a relatively low number of iterations was motivated by the observed rapid convergence of the algorithms. The resulting plots showcase the average function evaluations over iterations, providing a robust overview of algorithm behavior.

Hyper-parameters were selected based on empiricism to ensure meaningful comparisons. For coordinate descent, a learning rate of 0.01 was employed. In the context of backtracking line search, the constant in the Armijo condition was set to 10^{-4}, and the beta parameter, responsible for scaling the predicted step size, was chosen as 0.8.

4 Experimental Results

We present experimental results for the greedy and random-feature coordinate descent algorithms. The analysis focuses on the first two classes of the Wine dataset, where the mean function evaluation and associated variability for each iteration are depicted in the figure below. The shaded regions around each line represent one standard deviation from the mean function evaluation.

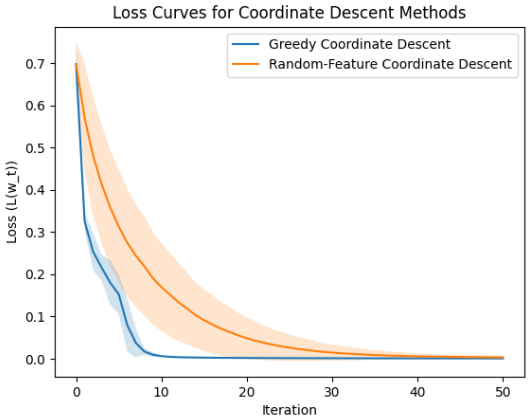

Figure 1: Comparison Between Greedy Coordinate Descent and Random-Feature Coordinate Descent

5 Discussion

In this section, we analyze and interpret the experimental results obtained from the greedy and random-feature coordinate descent algorithms applied to logistic regression on the first two classes of the Wine dataset.

Overall, the results demonstrate that the greedy coordinate descent method converges to the minimum faster compared to the random-feature coordinate descent. Notably, the mean function evaluation for the greedy method shows less variance, evident by the relatively thinner shading around the line. In contrast, the random-feature method exhibits a larger shaded region, indicating greater variance in function evaluations.

This aligns with our expectations, as the greedy coordinate descent strategy involves selecting the feature with the largest magnitude at each iteration.

This approach intuitively focuses on updating the coordinate that contributes the most to the gradient, potentially leading to a more efficient convergence towards the minimum.

The observed differences in convergence speed and variability between the two methods have implications for their practical use in logistic regression. The minimal variance in the greedy method suggests a more stable and consistent convergence, potentially making it a preferable choice in scenarios where efficiency and reliability are critical.

6 K-sparse Coordinate Descent

In this section, we extend our analysis to explore the performance of the greedy coordinate descent algorithm under sparsity constraints. The algorithm is modified to enforce a k-sparse solution, where the weight vector has at most k nonzero entries. We investigate the impact of varying sparsity levels on convergence behavior and final loss values.

6.1 Experimental Setup

For the k-sparse experiments, we utilized the same Wine dataset with binary classification between the first two classes as in the previous section. Hyperparameters such as the learning rate (0.01) and backtracking line search with an Armijo condition remained consistent with our earlier experiments.

The key modifications include a reduction in the number of iterations to 5 per run due to the notably rapid convergence observed in the k-sparse setting. Unlike the previous experiments, where a single coordinate was updated in each iteration, the k-sparse experiments involved updating the k largest features of the weight vector.

Additionally, the method for evaluating and plotting values entailed the following steps: after each iteration of updating the weight vector, we extracted its k-sparse version. The loss was subsequently computed based on this k-sparse variant of the current weight vector. This approach allowed us to gain insights into the model's behavior throughout the optimization process.

We conducted experiments for various sparsity levels, specifically testing for k values of 1, 3, 7, 10, and 13, to explore the effects of different degrees of sparsity on the convergence behavior.

6.2 Experimental Results

In this section, we present experimental results for the greedy coordinate descent algorithm, considering various k-sparsity values (k = 1, 3, 7, 10, and 13). As in the previous experiments, the figure below illustrates the mean function evaluation along with the associated variability for each iteration. The shaded regions surrounding each line indicate one standard deviation from the mean function evaluation.

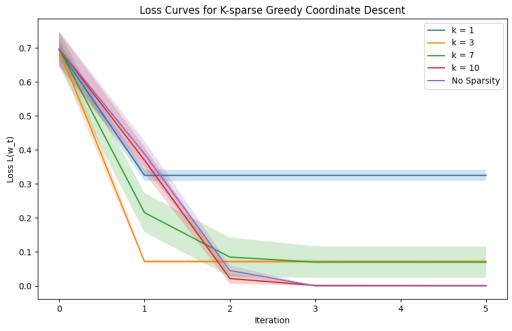

Figure 2: K-sparse Greedy Coordinate Descent for Different k Values

6.3 Discussion

k Value	Final Loss
1	0.325148
3	0.071640
7	0.069859
10	0.001087
13	0.000004

Table 1: Final Loss for Different k Values in k-sparse Coordinate Descent

In the k-sparse setting, it is evident that all runs converge significantly faster, typically stabilizing within no more than three iterations. A notable trend emerges regarding the relationship between the sparsity level (k) and the optimization process. As anticipated, higher values of k lead to lower final loss values, albeit at the expense of a slower convergence rate.

An interesting observation is the presence of groups of k values that exhibit similar performance. For instance, k=1 has its own distinct high loss value, while k=3 and k=7 demonstrate remarkably

similar loss values. This pattern persists in the comparison between k=10 and k=13, with the larger of the two k values having a slightly lower final loss. The observed behavior suggests that, for certain applications, choosing an intermediate sparsity level might be preferable. This decision could be based on a trade-off analysis, balancing the desire for lower loss values with the need for faster convergence.

7 Limitations

While our study focuses on a binary classification setting, one avenue for future research involves extending our methodology to scenarios with a higher number of classes. The decision to restrict our analysis to classes 1 and 2 was driven by the available dataset, resulting in 59 and 71 points, respectively. Recognizing the potential impact of training on a limited number of points, it is essential to acknowledge that generalization may be constrained. Future work should explore datasets with more extensive class distributions to assess the scalability and robustness of the proposed coordinate descent methods. While our study provides insights into the greedy coordinate descent method, the aforementioned considerations serve as avenues for refinement and expansion in subsequent research efforts.

8 Conclusion

This study has delved into coordinate descent methods, specifically focusing on the application of the greedy coordinate descent algorithm to logistic regression. Through a comparative analysis with a random-feature coordinate descent approach, we've unraveled the superior efficiency and dynamic behavior inherent in the greedy method. Additionally, our exploration has extended the algorithm's utility into a k-sparse setting, shedding light on its adaptability across varying sparsity levels.

Our experimental findings underscore the significant advantage of the greedy coordinate descent method in terms of convergence speed and stability when compared to its random-feature counterpart. The method's ability to efficiently update the coordinate with the largest magnitude at each iteration proves instrumental in achieving a rapid convergence towards the minimum.

Venturing into the k-sparse realm, we've uncovered intriguing patterns. Higher sparsity levels, as indicated by larger k values, lead to lower final loss values, albeit with a trade-off of slower convergence. Notably, certain groups of k values exhibit similar performance, hinting at the importance of selecting an intermediate sparsity level that balances the desire for low loss values with the need for faster convergence.

However, it's crucial to acknowledge the study's limitations. The binary setting chosen for logistic regression restricted the scope, and future work could explore extensions to scenarios with a higher number of classes. Additionally, the relatively low number of data points in classes 1 and 2 (59 and 71, respectively) raises concerns about generalizability, emphasizing the potential benefits of incorporating more extensive datasets.

Our contributions to understanding coordinate descent methods in logistic regression provide insights into their nuanced behavior on the Wine dataset. As we chart the course for future endeavors, we envision further exploration across diverse datasets, meticulous parameter tuning, and the investigation of alternative selection strategies. These efforts aim to refine and broaden the applicability of coordinate descent methods, addressing limitations and bringing us closer to an optimized and versatile toolbox for logistic regression optimization.