Coordinate Descent

Joshua Chen
UC San Diego
CSE 251A
jpc005@ucsd.edu

Abstract

This project explores the implementation of the coordinate descent method in optimizing a standard unconstrained optimization problem, where the objective is to minimize a cost function. This method simplifies the gradient descent process by iteratively updating one coordinate at a time to reduce the loss. Key aspects of this method include determining which coordinate to update and how to set the new value for this coordinate. My approach involves an adaptive strategy for selecting the coordinate, rather than random selection. I demonstrate the effectiveness of my approach through experiments on the Wine dataset from the UCI Machine Learning Repository [1].

1 Introduction

The main goal is to develop and evaluate a coordinate descent optimization method for solving a cost function L(w). Since we are using logistic regression, the cost function is the logistic loss function, defined as:

$$L(w) = \frac{1}{N}\sum_{i=1}^{N}\log(1+\exp(-y_i w^T x_i)) \qquad (1)$$

where:

- N is the number of observations in the dataset.
- y_i is the actual label of the i-th observation (1 or 0).
- x_i is the feature vector of the i-th observation.
- w is the vector of weights or coefficients.

To optimize the cost function effectively, I have developed four distinct approaches for coordinate descent, encompassing two methods for selecting the coordinate and two strategies for updating w_i. This results in four unique combinations of coordinate selection and update mechanisms:

- Max Gradient Method + Simple Update: Coordinate is chosen based on the largest gradient. The feature corresponding to the maximum absolute value of the gradients for each feature given the current weights and labels is selected for updating. w_i is then updated based on the gradient descent step specific to that feature.
- Random Method + Simple Update: Coordinate is chosen randomly. w_i is then updated based on the gradient descent step specific to that feature.
- Max Gradient Method + Backtracking: Coordinate is chosen based on the largest gradient. The feature corresponding to the maximum absolute value of the gradients for each feature given the current weights and labels is selected for updating. w_i is then updated with the backtracking line search method.
- Random Method + Backtracking: Coordinate is chosen randomly. w_i is then updated with the backtracking line search method

I found that the best method was the Max Gradient Method + Backtracking, as it resulted in the lowest loss.

2 Methods

My study commenced with the implementation of the 4 approaches listed above and compared them with the standard logistic regression loss. In the next sections, I will discuss the various approaches in detail along with the baseline method of standard logistic regression.

2.1 Standard Logistic Regression

In standard logistic regression, the model selects all coordinates (features) for updating in each iteration, employing gradient descent without the inclusion of a regularizer. Throughout the training phase, the algorithm comprehensively adjusts the weights for all features in unison during each epoch. This adjustment follows the following gradient descent step:

$$w_i \leftarrow w_i - \alpha\frac{\partial L}{\partial w_i} \qquad (2)$$

where w_i is the weight corresponding to the i-th feature, α is the learning rate, and $\frac{\partial L}{\partial w_i}$ represents the partial derivative of the logistic loss function L with respect to w_i:

$$\frac{\partial L}{\partial w_i} = \frac{y_i x_i}{1+\exp(y_i w^T x_i)} \qquad (3)$$

2.2 Random Coordinate Descent

2.2.1 Simple Update

In this method, the coordinate is chosen uniformly at random then the stochastic gradient descent method is used to update the weights w_i at each step. This strategy uses the same equation as equation (2) for updating the weight, but only the weight corresponding to the randomly chosen coordinate i is being updated.

2.2.2 Backtracking

In this method, the weight is updated with a backtracking line search that follows the algorithm.

Algorithm 1 Backtracking Line Search Algorithm

Initialize step size $\alpha = 1$
Choose reduction factor β (0 ¡ β ¡ 1)
Choose a small constant c (0 ¡ c ¡ 1)
$f_{current} \leftarrow f(\mathbf{w})$
while True **do**
    $\mathbf{w}_{new} \leftarrow \mathbf{w} - \alpha \nabla f(\mathbf{w})$
    if $f(\mathbf{w}_{new}) \leq f_{current} - c\alpha \|\nabla f(\mathbf{w})\|^2$ **then**
        break
    end if
    $\alpha \leftarrow \beta \cdot \alpha$ ▷ Reduce step size
end while
return α

Backtracking line search is advantageous because it adaptively adjusts the learning rate, ensuring each step in the optimization process is sufficiently large to make progress yet small enough to avoid overshooting the minimum.

2.2.3 Discussion

In random coordinate descent with simple update, the cost function must indeed be differentiable. This is because the updating step relies on computing gradients, which are derived from the derivatives of the cost function for the coordinate being optimized.

In random coordinate descent with backtracking, the cost function does not necessarily need to be differentiable. This is because the backtracking line search adjusts the step size based on function evaluations rather than gradients, allowing for the optimization of non-differentiable functions.

2.3 Max Gradient Coordinate Descent

2.3.1 Simple Update

In this method, we calculate the gradient corresponding to each weight and then find the index of the weight that has the maximum absolute gradient value. The algorithm is shown in algorithm 2.

2.3.2 Backtracking

In this method, the weight is updated with a backtracking line search (algorithm 1). The process is essentially the same as algorithm 2, except that instead of initializing a constant learning rate alpha, each epoch will undergo the backtracking process and find the optimal learning rate.

Algorithm 2 Max Gradient Coordinate Descent

Require: Training Dataset X
Initialize weight vector w to zeros
Initialize bias b
Initialize Learning Rate α
for each epoch **do**
    for each feature i in X **do**
        Calculate gradient g_i for feature i
    end for
    Find the index i_{max} of the maximum absolute gradient
    Update the weight: W^i_{t+1} = W^i_t - $\alpha * g_i$
end for
return Updated weight vector w and bias b

2.3.3 Discussion

Differentiability: In both weight update approaches, the cost function would need to be differentiable because gradients are used to determine the feature with the maximum gradient. While backtracking line search itself does not necessarily require a differentiable function, the selection of the feature with the maximum gradient implies the need for differentiability, as gradients are computed from the derivatives of the cost function.
Convergence: This coordinate selection method relies on the fact that for each iteration, we are trying to select a weight where we see a max change in gradient. Therefore, if we see that there is a steepest downfall for a particular coordinate, it makes sense to go in that direction for the best result.

3 Results

For all 5 cases, I each model for a total epoch of 100 and evaluated their performances.

3.1 Standard Logistic Regression

The final test loss was 0.002609. This final loss is taken as the best-case loss baseline to compare with the proposed approaches in the project.

3.2 Random Coordinate Descent with Simple Update

Since this method has the potential for variation due to the random selection of features, I ran this experiment 100 times and the final average test loss was 0.34907.

3.3 Random Coordinate Descent with Backtracking

Since this method has the potential for variation due to the random selection of features, I ran this experiment 100 times and the final average test loss was 0.023788.

3.4 Max Gradient Coordinate Descent with Simple Update

The final test loss was 0.19888

3.5 Max Gradient Coordinate Descent with Backtracking

The final test loss was 0.018810

3.6 Discussion

Table 1: Loss for All Approaches

Strategy	Loss
Standard Logistic Regression	0.02601
Random CD + Simple	0.34933
Random CD + Backtrack	0.02379
Max Gradient CD + Simple	0.19888
Max Gradient CD + Backtrack	0.01881

Note: The loss values presented for the Random-Feature Coordinate Descent strategy are derived from the average of 100 trials, reflecting the inherent variability due to the stochastic nature of the method. In contrast, the loss values for the Standard Logistic Regression and Max Gradient Coordinate Descent are consistent across all trials, as these methods do not incorporate randomness.

The outcomes indicated in the table highlight a noteworthy trend in the performance of coordinate descent strategies. Strategies employing random feature selection are associated with higher loss values, which may be attributed to the stochastic nature of their selection process. This randomness can lead to sub-optimal descent trajectories and, consequently, slower progress toward the minimum loss.

In contrast, the Max Gradient approaches exhibit substantially lower loss figures. This improvement demonstrates the advantage of a deterministic selection criterion based on the gradient, which systematically targets the most promising direction for each iteration, thereby accelerating the convergence.

Furthermore, it is apparent that the incorporation of a backtracking line search in both random and max gradient strategies significantly reduces the loss. The use of backtracking enhances the precision of the step size in the descent, ensuring that each iteration makes meaningful progress toward the objective. This adaptive step-sizing contributes to more efficient optimization and robustness in the face of varying landscape complexities.

3.6.1 Loss Plot

The graph in Figure 1 illustrates the convergence behavior of different coordinate descent strategies over 100 iterations. Each approach is on a trajectory toward the optimal loss value, denoted as 'Final Loss L*', which was established by standard logistic regression.

From the graph, it is observable that the 'Max Gradient + Backtrack' strategy converges to the vicinity of L* the quickest, showing a steep decline in loss before plateauing. This indicates an efficient optimization process, likely due to the precise direction and step size adjustments provided by the gradient information and backtracking.

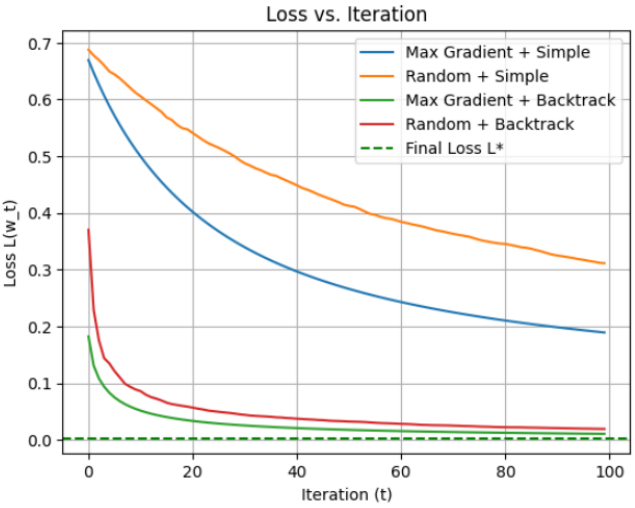

Figure 1: Loss

The 'Random + Backtrack' approach also shows a marked improvement over 'Random + Simple', suggesting that backtracking significantly enhances the effectiveness of the random coordinate selection method. However, it does not converge as rapidly as the 'Max Gradient + Backtrack', which emphasizes the superiority of using gradient information for feature selection in coordinate descent.

Both 'Max Gradient + Simple' and 'Random + Simple' strategies appear to converge more slowly, implying that without backtracking, the optimization process is less efficient. Nonetheless, these approaches are still making progress towards the optimal loss, albeit at a slower rate.

4 Critical Evaluation

From my experiments, it can be said that different feature selection techniques that follow some sort of guide can yield better results than randomly selecting the feature. In this project, I've shown that selecting the max gradient during each iteration can get close to the best-case standard logistic regression model. This can further be improved upon by normalizing the max gradient method [2]. Furthermore, one can explore other coordinate descent feature selection techniques as shown in Coordinate Descent Algorithms by Stephen J. Wright [3]

In addition to improving the feature selection technique, further works can incorporate a better way of updating the weights. From my experiment, I've shown that backtracking significantly improves the loss compared to simply subtracting the gradient. To enhance the weight updating process, an approach worth exploring

is the Adaptive Moment Estimation (Adam) method. Adam optimizes the learning rate for the individual feature by combining the advantages of AdaGrad and RMSProp, two extensions of stochastic gradient descent. This method adapts the learning rates based on the first and second moments of the gradients, effectively balancing the benefits of both momentum and adaptive learning rates [3].

References

[1] Aeberhard, Stefan and Forina, M.. (1991). Wine. UCI Machine Learning Repository.

[2] Nutini, J., Schmidt. M., Laradji, I., Friedlander, M., & Koepke, H., (2015). "Coordinate Descent Converges Faster with the Gauss-Southwell Rule than Random Selection."

[3] Wright, S., (2015). "Coordinate Descent Algorithms."

[4] Kingma, D. P., & Ba, J. (2015). "Adam: A Method for Stochastic Optimization."