Coordinate Descent Algorithm

Vivekanand Gyanchand Sahu
PID: A69027331
vsahu@ucsd.edu

Abstract

This paper investigates Coordinate Descent and Random-feature Coordinate Descent, models selecting and updating a single weight per iteration, comparing it with Gradient Descent in logistic regression on a wine dataset for binary classification. Evaluating their convergence, results reveal competitive performance of coordinate descent, shedding light on its suitability for various tasks. The study provides concise insights into the comparative effectiveness of Coordinate and Gradient Descent in optimizing logistic regression parameters for binary classification on the wine dataset.

1 Introduction

In the realm of optimization for machine learning models, this paper explores the intricacies of minimizing loss with respect to weights, elucidating a model akin to gradient descent known as coordinate descent. Unlike gradient descent, coordinate descent exhibits unique advantages, particularly in scenarios where the functions lack computable differentiations for the parameter update step. This versatility positions coordinate descent as an advantageous alternative, expanding its applicability to a broader range of functions. It only needs a loss to be a convex function.

Furthermore, the method's simplicity and adaptive hyper-parameter make it an attractive choice for optimization tasks. Unlike gradient descent, coordinate descent guarantees that the model will never overshoot or diverge, irrespective of hyper-parameter values. This inherent stability adds a layer of robustness to the optimization process, enhancing its suitability for a diverse array of optimization challenges. Through empirical analysis and comparisons with gradient descent, this paper aims to establish coordinate descent as a powerful and flexible methodology, shedding light on its unique advantages and practical implications in the context of machine learning model training.

2 Implementation Details

2.1 Coordinate Descent

Coordinate Descent distinguishes itself from Gradient Descent by updating weights individually, selecting the weight that minimizes loss at each iteration. This unique approach ensures targeted parameter adjustments, elevating optimization efficiency and adaptability. The method's iterative refinement enhances its applicability across diverse machine learning scenarios.

2.2 Coordinate Descent Steps

Coordinate Descent is an iterative method where weight parameters are altered at each iteration. The process continues until convergence, where the loss is minimized. This iterative refinement ensures gradual improvement, culminating in optimal parameter values for enhanced model performance.

Initialization: $w \leftarrow$ Randomly selected initial weights for features in the dataset.

Weight Selection: For each weight i in w, Loss $L[i]$ = min ($L(w[i] + alpha)$, $L(w[i] - alpha)$) are calculated, $alpha$ being a hyper parameter denoting the learning rate. The weight's index i is selected by calculating the losses of all weights and selecting the minimum $min(L)$ giving weight.

Update the weight: Update the weight that gives minimum loss by $w[i] \leftarrow w[i] + alpha$ if $L(w[i] + alpha) < L(w[i])$ and $w[i] \leftarrow w[i] - alpha$ if $L(w[i] - alpha) < L(w[i])$. When $L(w[i] + alpha)$ and $L(w[i] - alpha)$ both are > $L(w[i])$, then we restart this step with $alpha \leftarrow alpha/beta$, where $beta$ is another hyper-parameter that decreases the learning rate $alpha$ by a factor of $beta$ till the time one of the $L(w[i]+alpha)$ and $L(w[i]-alpha)$ drops below the $L(w[i])$.

The iterative nature of the algorithm ensures updating the weights towards the mist optimized values.

2.3 Pseudocode:

The BisectingKMeans function accepts input data and the desired size for the output sample. It then applies the bisecting k-means clustering algorithm to partition the input data, providing the clustered data and the distances of each sample from their respective centroids as outputs. This facilitates a comprehensive understanding of data distribution within clusters and serves as valuable information for further analysis and sampling considerations.

```
data = ImportWineData
function CORDINATEDESCENT ($data$, $alpha$, $beta$, $iterations$)
    $weights\ w \leftarrow randomValues$

    for each $j$ in $iterations$ do:

        for each $w[i]$ in $w$ do:

            $lp \leftarrow L(w[i] + alpha)$
            $lm \leftarrow L(w[i] - alpha)$

            while $min(L(w[i] + alpha),$
                $L(w[i] - alpha)) > L(w[i])$ do:

                $alpha \leftarrow alpha/beta$

        end for

        $select\ index\ i \leftarrow idx(minL(w))$
        $update\ weight\ w[i] \leftarrow w[i] + alpha$

    end for

    return $w$

end function
```

The data is imported, and the CoordinateDescent function takes the data, along with hyperparameters alpha and beta, while initializing weights randomly. In each iteration, the process unfolds as follows: 1) For each weight, both the addition and subtraction of alpha are considered, and the corresponding losses are computed. If both the losses for the alpha-added and alpha-subtracted weights surpass the loss of the raw weight, alpha is decreased by a factor of beta. This reduction continues until an alpha is found that, when added or subtracted to the raw weight, yields a loss lower than the loss of raw weight. The same steps are repeated for the given number of iterations. Finally the model returns the closest to optimal weights.

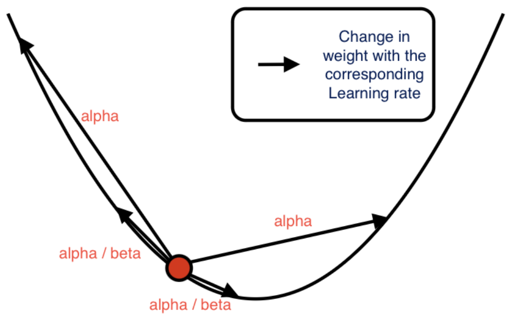

Figure 1: Prevention of divergence by reducing alpha by a factor of beta

2.4 Random-Feature Coordinate Descent:

n this approach, there's a departure from selecting features based on the minimum loss; instead, features are chosen randomly. The update step for the weights remain the same ensuring the model improves after each iteration. While this may not guarantee the absolute minimum loss at each step, it introduces a speed advantage to the optimization process. The randomness in feature selection allows for swift updates, providing a trade-off between accuracy and computational speed. This strategy becomes particularly valuable when the balance between achieving precise results and computational efficiency is a key consideration, enabling practitioners to tailor the optimization approach based on the specific accuracy vs speed trade-off required for a given application.

2.5 Hyper-parameter selection:

The selection of alpha and beta values depends on dataset variations. A higher alpha suggests larger weight steps towards the optimal weight, while a higher beta implies a more substantial reduction in the step size alpha. Alpha should generally be a large value. Beta should be bigger than 1. The reduction in value of alpha by a factor of beta when the change in weight does not reduce the loss on raw weight, prevents model overshooting and ensures that it never diverges. Notably, as weights are initialized and updated based on alpha and fine-tuned by beta in each iteration, there is no need to compute the differentiation of the loss function to determine the direction and magnitude of weight changes. This advantage is particularly significant when differentiation of the loss function is not always computationally feasible, as coordinate descent only requires the function to be convex for model convergence.

2.6 Convergence:

Model convergence is attained when an algorithm iteratively hones its parameters until they stabilize, indicating optimal values. It signifies that additional iterations yield marginal improvements in model performance or loss reduction. In Coordinate Descent, supplementing standard convergence checks, a while loop limitation, i.e after $beta^n$ operations, ensures a practical convergence criterion. Successful convergence ensures a trained model adept at capturing data patterns, achieving a delicate equilibrium between complexity and generalization for effective deployment in real-world scenarios. This robust convergence mechanism safeguards against over fitting or prolonged training, enhancing the model's practical utility.

3 Experiment Results

The comparison is made between the 3 models namely coordinate descent, gradient descent, and random-feature coordinate descent.

In the evaluation of three models coordinate descent, gradient descent, and random-feature coordinate descent an increase in the number of iterations consistently resulted in decreasing losses across all algorithms. Gradient descent exhibited the swiftest loss reduction, followed by coordinate descent, and finally, random-feature coordinate descent. The computational efficiency mirrored this trend, with gradient descent consuming the least time, succeeded by random-feature coordinate descent, and then coordinate descent. These findings underscore the varying trade-offs in convergence speed and computational demands among the considered optimization methods. The Gradient descent overshoots and diverges at learning rate of 1000 but there is no divergence of coordinate descent at any value of alpha.

Loss, the crucial performance metric, was computed using the formula:

$$L = \sum_{i=1}^{n} [y_i \log(p(x_i)) + (1 - y_i) \log(1 - p(x_i))]$$

# iterations	**Coordinate**	**Random-feature**
10	0.0035	0.2927
20	0.0007	0.0495
30	0.0001	0.0003
100	1.3e-05	1.5e-05

Table 1: Comparison of log likelihood loss of Coordinate descent and Random-feature coordinate descent for different iterations

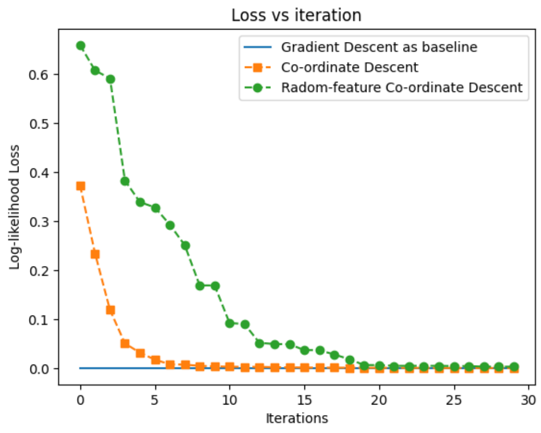

Figure 2: Comparison of Accuracies with Different Data Sampling Strategies

4 Evaluation and Inferences

This study systematically compares three optimization models: coordinate descent, gradient descent, and random-feature coordinate descent. While gradient descent operates with a single hyperparameter, coordinate descent methods utilize two, affording greater adaptability albeit with the added complexity of tuning two parameters. Gradient descent is prone to overshooting and divergence under high learning rates, a challenge mitigated by coordinate descent, which ensures stability through the dynamic adjustment of alpha with beta. Additionally, gradient descent mandates the derivative of the logistic loss function, a prerequisite absent in coordinate descent models. The gradient descent model showcases swifter convergence, empowered by its ability to determine weight changes without multiple iterations. Its efficiency is further accentuated as it simultaneously updates all weights, setting it apart from the coordinate descent methods. This comprehensive analysis sheds light on the nuanced trade-offs and performance distinctions among these optimization techniques.

5 Limitations and future scope

Limitations of the coordinate descent and random-feature coordinate descent methods are evident. Both approaches involve only one feature update per iteration, potentially limiting their efficiency. A more rapid and efficient algorithm could be achieved by allowing multiple features to be updated in each iteration. Additionally, both methods employ two hyper-parameters, adding complexity to the optimization process. To streamline and simplify, reducing these parameters to one could be considered, possibly by introducing a manageable alpha callable that consolidates the functionality of multiple parameters. Addressing these limitations could enhance the versatility and performance of the coordinate descent and random-feature coordinate descent algorithms.

6 Reference

1. Scikit-Learn Logistic Regression Documentation:
   https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

7 Code implementation

1. Code implementation GitHub:
   https://github.com/Vivekanand-Sahu/CSE_251_A2.git