CSE251A Project 2 Write-up

Junwei Chen [1]

Abstract

In this project we design a coordinate descent method that smartly (1) picks the coordinate to do the descent, and (2) descend quickly on this coordinate. We apply our method to logistic regression and compare it with the standard logistic regression from sklearn on the Wine training set. We further did an ablation study by choosing coordinate randomly to verify the effectiveness of our coordinate choice in (1). We also compared our (2) which uses backtracking line search to regular gradient descent.

1. Method Description

Throughout the discussion, we assume the loss function is convex and differentiable. When we compare the greatness of values, we are talking about their absolute value.

1.1. Intuitions for Coordinate Selection

There are several intuitions we experimented when choosing coordinates. First, we can choose the coordinate with the highest first-order partial derivative so that we can get the most improvement in each round. Second, we can also favor the coordinates with high second-order partial derivative (i.e. with strong convexity) so that we have fast convergence (this requires the loss function to have continuous second-order derivatives). Third, we can also choose the coordinate with the highest of Pearson correlation coefficients with the label, meaning that changing their values will have the most influence on the loss function.

1.2. Intuitions for Descent

Although doing regular gradient descent will work on the coordinate we pick, we seek to do better than that using the backtracking line search from lecture which gives a better stepsize each iteration so that we can converge quickly. We modify the terminating condition of backtracking line search to suit coordinate descent. The original terminating condition (when the descent direction is gradient) is

$$f(\mathbf{x}_t - \alpha \nabla f(\mathbf{x}_t)) \leq f(\mathbf{x}_t) - c\alpha ||\nabla f(\mathbf{x}_t)||_2^2$$

Our new terminating condition is

$$f(\mathbf{x}_t - \alpha \nabla_i f(\mathbf{x}_t)) \leq f(\mathbf{x}_t) - c\alpha \nabla_i f(\mathbf{x}_t)^2$$

where i is the coordinate we pick.

1.3. Our method

After experimenting on our intuitions, we design our coordinate descent method as follows:

At each iteration, pick the coordinate with the highest gradient in absolute value. Then, run backtracking line search with respect to this coordinate to determine a good stepsize. Finally, update the weight of the chosen coordinate by subtracting the partial gradient times derived stepsize.

We need the loss function to be at least differentiable for our method to work.

2. Convergence

Our method will converge to the optimal loss when the loss function is differentiable and convex like logistic regression. This is because (1) our method is essentially gradient descent that approaches the optimal loss along coordinates every iteration; (2) our method switch attention to a different coordinate when the gradient on the current coordinate gets less than some other coordinates, thereby decreasing the gradients of all coordinates; (3) we have a stepsize that is variant and good every iteration based on backtracking line search, so we will not encounter a situation that we stay in the peripheral of the optimal loss, circling it by taking steps only in the coordinates.

3. Experimental Results

3.1. Evaluation of Our Method

We compared our method with the standard logistic regression solver from sklearn on the normalized Wine dataset (first 2 labels). We marked its loss function on Figure 1.

We compared 4 versions of our methods for ablation study:

[1]UC San Diego, CSE Department. Correspondence to: Junwei Chen <juc005@ucsd.edu>, Taylor Berg-Kirkpatrick <tberg@ucsd.edu>.

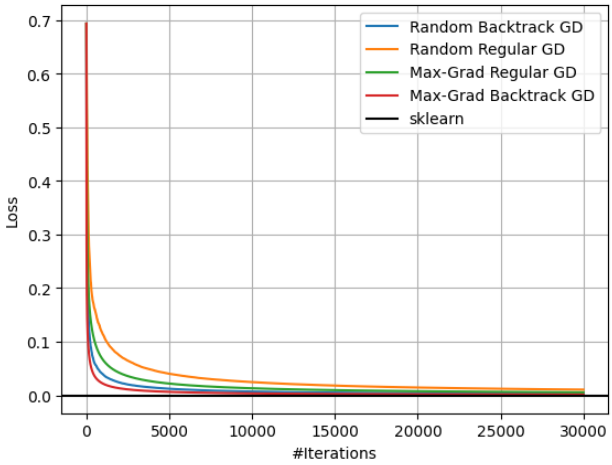

Figure 1. Ablation study of Random vs Max-Grad coordinate selection, Regular vs Backtracking line search based Gradient Descent

(1) a random-feature coordinate descent + regular gradient descent on that random coordinate "Random Regular GD", (2) a random coordinate descent + backtracking line search + gradient descent "Random Backtrack GD", (3) our selection of the coordinate with the greatest gradient + regular gradient descent "Max-Grad Regular GD", (4) our selection of the coordinate with the greatest gradient + backtracking line search + gradient descent "Max-Grad Backtrack GD".

For all methods, we perform 30000 iterations. We choose a stepsize of 0.1. For backtracking line search, we choose $c = 1e^8$ and $\rho = 0.8$

From Figure 1, we see that our complete version "Max-Grad Backtrack GD" achieves the fastest convergence and smallest loss. Comparing "Max-Grad Backtrack GD" with "Random Backtrack GD" and "Max-Grad Regular GD" with "Random Regular GD", we see that our coordinate selection method is better than random selection in terms of convergence speed. Comparing "Max-Grad Backtrack GD" with "Max-Grad Regular GD" and "Random Backtrack GD" with "Random Regular GD", we see that our backtracking line search also speeds up convergence.

3.2. Failed Attempts

Besides the methods we finally picked in Section 1.3, we also tried several other methods. For example, we tried our second intuition of coordinate selection (in Section 1.1) using the second-order partial derivative (Hessian) as an indicator of how fast the loss will converge. However, the Hessian of a coordinate seldom changes, so if we take the max, we end up always choosing one coordinate. To get around this, we tried to rank the coordinates by the value of Hessian and give a higher probability to picking coordinates with higher Hessian by a weighted random selection. However, it turned out that selecting this way is no different from selecting randomly.

Another attempt we tried is to fix the max-gradient coordinate that we do descent for some number of iterations before turning to the next max-gradient coordinate. We reasoned that in this way other coordinate will not influence whichever coordinate we are working with now. However, this is not different from selecting the max-gradient coordinate every iteration.

4. Critical Evaluation

There are three ways to improve our method that we can think of:

First, although we failed to use Hessian as an indicator for coordinate selection, we may be able to use it as an indicator of the stepsize and customize a stepsize for each coordinate.

Second, we have not tried our third intuition of coordinate selection (in Section 1.1) using Pearson correlation coefficients. If time permits, we can try to see if it can be combined with our max gradient to give quicker convergence.

Third, our method requires the loss function to be differentiable at least for the first order. However, if we encounter a function that is not differentiable and gradient descent cannot be used, we can still adapt our ideas to the Stochastic Three Points Method, which is to take a small step in two opposite directions of the weight and compare them to the current loss. For each coordinate, we can do such three points method by adding or subtracting a small amount (stepsize). Then we pick the coordinate with the most gain in loss. We can then even adapt the idea of backtracking line search, constantly multiplying a ρ to the stepsize, getting a sequence of losses, and pick the best stepsize with the best loss.