CSE 250B: Machine Learning

Jiazhou Gao, A53278947

February 23, 2019

1.Description of your coordinate descent method

a. Which coordinate to choose:

Choose the coordinates in order.
Let k be the coordinate we choose to update the w_k, for $k \in 1, 2, 3, .., 13$ choose k in order, which means starting from $i = 1$, if last time we choose $i = j$, next time we would choose

$$i = \begin{cases} j+1, & j < 13 \\ 1, & j = 13 \end{cases} \tag{1}$$

b. How to set the new value of ω_i ?:

We update the w_i by the following equation:

$$w_i^{t+1} = argmin_{wi} L(w^t) \tag{2}$$

The idea is that for each step, we find the w_i that minimize the loss function $L(w^t)$
To achieve this goal, we use newton method to find w_i. To be specific, for each update, our goal is to find w_i, such that $\frac{\partial L(w)}{\partial w_i} = 0$
For each step in the newton method, the update would be:

$$w_i^t \quad = w_i^t - \frac{\partial L(w^t)/\partial w_i^t}{\partial^2 L(w^t)/\partial^2 w_i^t} \tag{3}$$

After the newton method converges, we find w_i^t that minimize the loss function, assign this value to w_i^{t+1} and one update ends.
where $L(w)$ is the cost function of the Logistic Regression:

$$L(w) == -\frac{1}{m}\sum_{i=1}^{m} y^i \log(h_w(x^i)) + (1-y^i)\log(1-h_w(x^i)) \tag{4}$$

$$h_w(x) = g(w^T x) \tag{5}$$

$$g(z) = \frac{1}{1+e^{-z}} \tag{6}$$

c. Do you need the function L(·) to be differentiable?

Yes, I need $L(w)$ to be differentiable and also have continues second-order derivatives, because the newton method requires the first and second derivatives of $L(w)$

2.Convergence

My method converges if the change of loss during one update for all ω is less than 1e-4, and the absolute value of $loss - L_*$ is less than 1e-4, where L_* is the loss of the standard logistic regression solver from scikit-learn.

3.Experimental results

First, we run a standard logistic regression solver. I use $LogisticRegression(C = 1e10, solver = ' liblinear')$
Notice that C is the inverse of regularization strength, therefore, C should be as large as could be. The final loss L calculated by the loss function $L(w)$ I defined in the previous section.
Then we run random-feature coordinate descent and our method of coordinate descent, make comparison between them.
The result is shown in figures below.

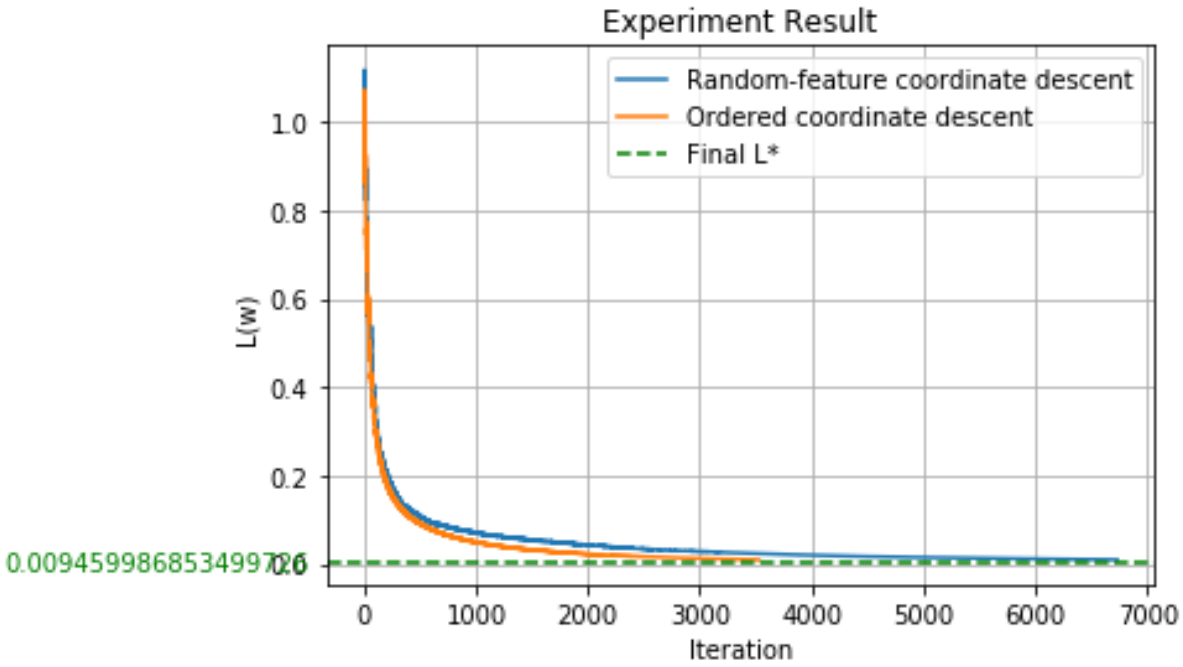

figure1

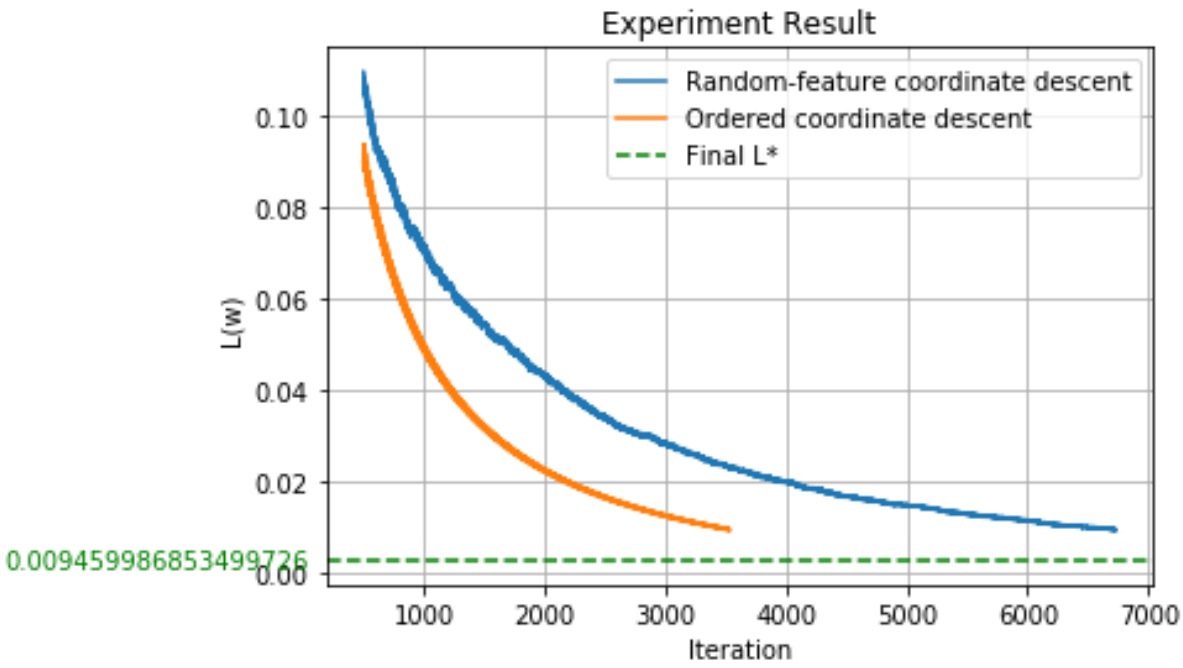

figure2

In figure1, we can see both algorithms make loss drop greatly in the first 1000 iteration, then the decreasing slows down.

Figure2 is a partial image of figure1 with iterations greater than 1000. In figure2, we can see more clearly that our method has larger decreasing rate than random feature coordinate descent and also converges earlier. In the experiment the random feature coordinate descent converges at around 6500 iterations but our method converges at around 3500 iterations.

As both algorithms use newton method to update w_i. The performance difference is because of the different methods to choose coordinates.

The final loss L_* is 0.009459986853499726

4.Critical evaluation

I think there is scope for further improvement in my coordinate descent scheme in (1).

Currently I just choose the coordinates in the order of one after one. This method works better than randomly choose coordinates but still can be improved. For example, we can choose the coordinate which decreases the $L(w)$ most. To be specific, given the update method, we choose the coordinate i which decreases the $L(w)$ most after the update.

Using this method, we can expect that the decreasing rate will be larger and hopefully we can achieve earlier convergence. However, the running time would also increase in this method because we need to try every coordinate to check which one decreases the loss most.