CSE 250B PA 2

Pin Tian PID:A53219987

Coordinate Descent

1. A short, high-level description of my coordinate descent method

   (a) Which coordinate to choose?

   The main idea of my coordinate descent algorithm is to update the coordinate which has the maximal absolute value of gradient in every epoch.

   For example, the input X has 3 dimensions. And we got the gradient of weights is $\frac{\partial L}{\partial w} = (-9.0, 7.0, 2.0)$ after first epoch. According to this result, my algorithm will choose and update the w_1. Then, after second epoch, the gradient becomes $\frac{\partial L}{\partial w} = (-5.0, 8.0, 1.0)$, w_2 will be chosen and updated. So on and so forth.

$$Index\ of[Max(\frac{\partial L}{\partial w_i}), \quad i = 1, 2, \cdots, d] \tag{1}$$

   I use the sigmoid activation function and cross entropy loss function in my algorithm. So the gradient function is:

$$\frac{\partial L}{\partial w_i} = (y - t)x_i, \quad i = 1, 2, \cdots, d \tag{2}$$

   where $y = \frac{1}{1+e^{-wx}}$ and d is the dimension of the input.

   (b) I used batch gradient decent to update the weight with maximal absolute value of gradient.

   First calculate the gradient. Then find the index of weight with maximal absolute value of gradient. Finally, update that weight.

   The loss function must be differentiable.

$$w^{(t+1)}[index] = w^{(t)}[index] - \eta \frac{\partial L}{\partial w_i^{(t)}} \tag{3}$$

2. Convergence

   The model is converged when the difference of losses between $L^{(t)}$ and $L^{(t-1)}$ is smaller than 5^{-6}.

   At this condition, my algorithm converges after 2761 epochs.

3. Experimental results

   (a) Logistic regression solver. I used Logistic Regression model in $sklearn.linear_model$

package. Called stochastic average gradient(SAG) algorithm , then got the loss and accuracy which are shown in Fg.2. WE can see that the tendency of loss curve of this optimizer is decreasing and converged at about $L^* = 0.16$.

(b) Coordinate descent solver. Fg.3 shows the loss and accuracy curve of my coordinate descent algorithm. The loss converged at about $L = 0.26$ which is higher than logistic model. We can clear see that it is asymptote to L^*.

(c) Uniformly random solver. The gradient descends very slow on this optimizer. It is reasonable because the model chooses coordinates uniformly at random. The result is shown in Fg.4.

(d) Fg.1 shows the loss comparison among three algorithms in 3000 iterations.

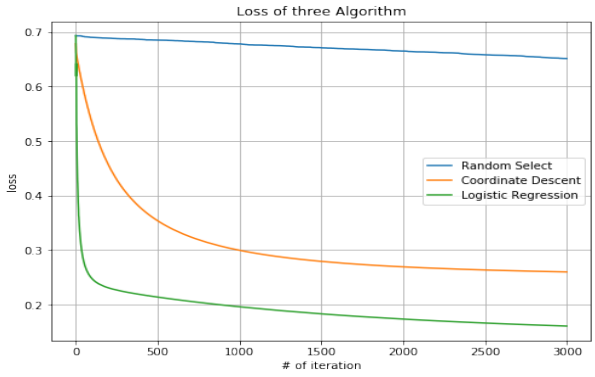

Figure 1

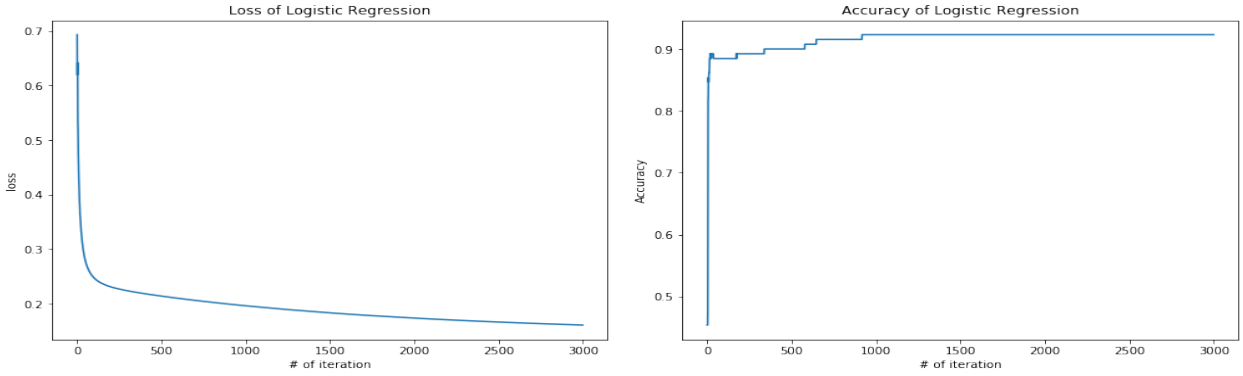

Figure 2

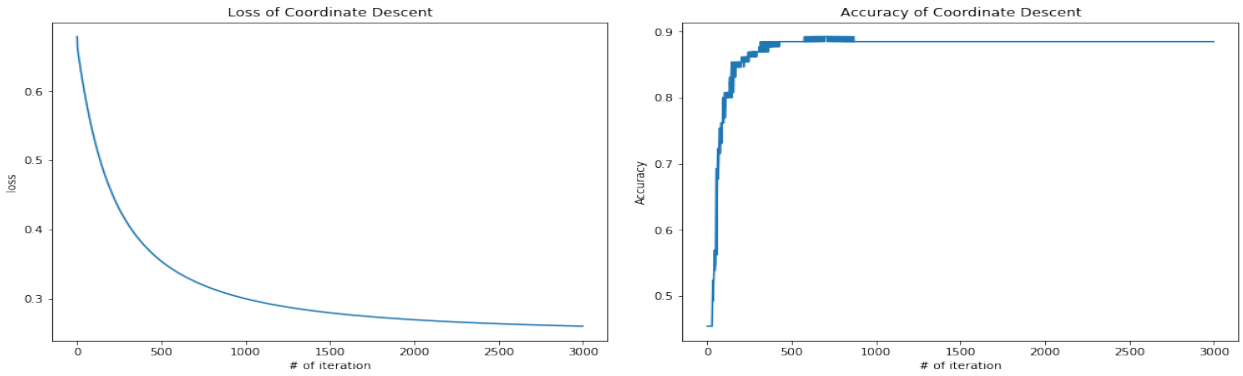

Figure 3

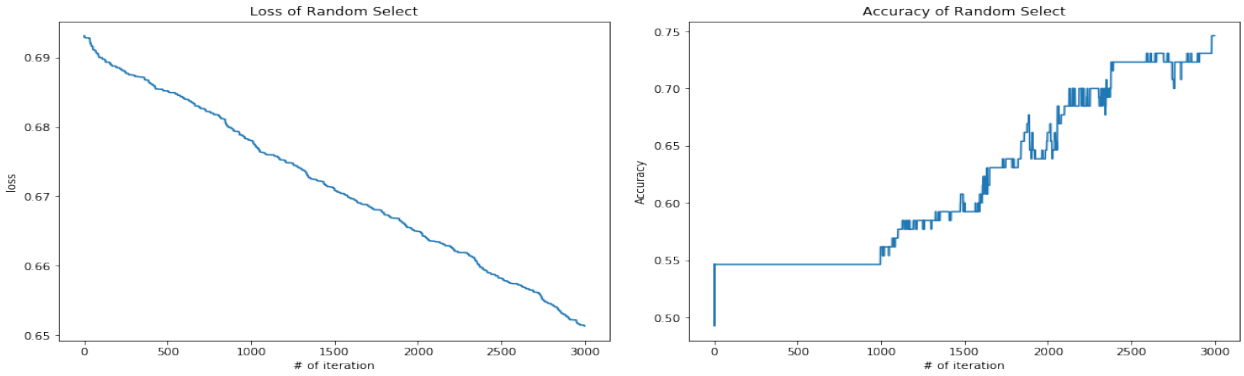

Figure 4

4. Critical evaluation.
   Do you think there is scope for further improvement in your coordinate descent scheme in (1); if so, how?
   Exactly. There are several other methods that can allow further improvements.
   First, Lasso linear optimizer is a coordinate descent algorithm which is based on L1 normalization. According to this idea, I can try to add normalization parameters in my algorithm. In my opinion, both L1 and L2 normalization can improve my algorithm. And the loss function and the gradient of loss becomes:

$$\begin{aligned} \mathbb{L} &= -\frac{1}{N}\sum_{n=1}^{N}[t^n \ln y^n + (1 - t^n \ln(1 - y^n)] + \lambda||w||_2^2 \quad (or \lambda||w||_1) \\ \frac{\partial L}{\partial w_i} &= (y - t)x_i + 2\lambda w_i \quad (or 1\, if w_i > 0, -1\, if w_i < 0) \end{aligned} \tag{4}$$

   The λ here is a penalizing parameter which can prevent model overfitting problem and improve the generalization of the model.