Coordinate descent

Qimin Chen
A53284263
qic003@ucsd.edu

1 Coordinate descent method

1.1 Description

After computing the derivative of logistic loss with respect to w_i, pick the k largest absolute value of dw_i where $i \in \{1, 2, 3, ..., 13\}$ (for problem a and b $k = 1$), update the corresponding value of w_i by subtracting its $dw_i \times \alpha$ where α is the step size and stop iterations when the difference between current loss and last loss is smaller than 10^{-4}.

1.2 Pseudocode

Algorithm 1 Coordinate descent

```
normalize the dataset and split data into training set and test set
randomly initialize weights w
compute initial loss l_prev
set current loss l_curr = ∞
while l_curr − l_prev > 10^-4 do
    compute dE/dw = (Y_pred − Y_true)X
    sort and find index of K-largest dw
    for k=0,1,...,K do
        w[k] -= α × dw[k]
    end for
    l_prev = l_curr
    recompute the current loss l_curr
end while
```

2 Convergence

The trainging loss stops decreasing and remains roughly the same for some epoche when reducing the learning rate then the model converges to the optimal loss.

3 Experimental results

Figure 1 shows the training loss on my coordinate descent(red line), random coordinate descent(yellow line) and logistic regression solver from scikit-learn(blue line) over 1000 iterations. It can be seen from Figure 1 that my coordinate descent performs better than random one because random coordinate descent merely picks coordinates at random and update corresponding weights which causes unstability on training phase.

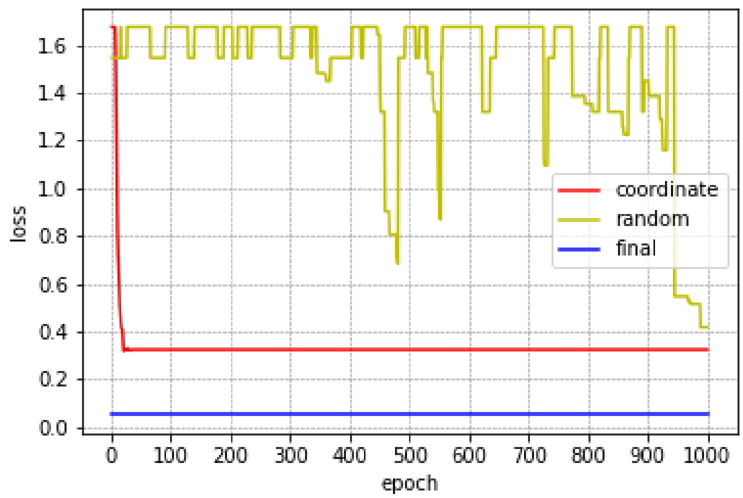

Figure 1: training loss on my coordinate descent, random coordinate descent and logistic regression solver over 1000 iterations.

4 Critical evaluation

I think there is scope for further improvement of my coordinate descent scheme. Since I only update k largest value of dw, it might not be appropriate to update w by just subtracting its corresponding dw. One way to update the k largest weights is to calculate Hessian matrix:

$$H_{\alpha\beta} = -\sum_t \sigma'(w \cdot x_t) x_{\beta t} x_{\alpha t}$$
$$w \leftarrow w - H^{-1} dw$$

where $\alpha, \beta \in \{1, 2, 3, ..., 13\}$ and $\sigma(z)$ is the sigmoid function.

5 Sparse coordinate descent

As the pseudocode described above, pick the k largest absolute value of dw_i where $i \in \{1, 2, 3, ..., 13\}$, update the corresponding value of w_i by subtracting its $dw_i \times \alpha$

Table 1 shows the loss values for different values of k on my coordinate descent and random descent over 1000 iterations. I do not think my method for sparse coordinate descent always find the best k-sparse solution when $L(\cdot)$ is convex.

Experiment	My coordinate descent	Random coordinate descent
k = 1	0.3256	0.4191
k = 3	0.2606	0.3868
k = 5	0.3242	0.3837
k = 7	0.3201	0.5803
k = 9	0.3002	0.4190
k = 11	0.2907	0.5480
k = 13	0.2873	0.3868

Table 1: training loss for different values of k on my coordinate descent and random descent over 1000 iterations.