CSE 250B – Programming assignment 2

1. A short, high-level description of coordinate descent method:
   My idea is to update the weight only with the maximum value of the gradient vector of the loss function at each iteration.
   Consider a dataset with m examples with each feature in d dimensions.
   The loss function that I considered here for logistic regression is,

$$L(w) = \frac{-1}{m}\sum_{i=1}^{m} y^i\left(\log\left(y^i_{pred}\right)\right) + (1-y^i)\log\left(1-y^i_{pred}\right)$$

$where,$

$$y_{pred} = P(y = 1/x) = \frac{1}{1+e^{-w^T x}}$$

$and\ the\ gradient\ is\ given\ by,$

$$\frac{\partial L}{\partial w_j} = \frac{1}{m}\sum_{i=1}^{m}\left(y^i_{pred} - y^i\right)x_j$$

$w = weight\ vector = [w_1, w_2 \ldots\ldots w_{d+1}] including\ the\ bias.$

$$Thus, we\ have\ \nabla L(w) = \left[\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \frac{\partial L}{\partial w_3} \ldots\ldots \frac{\partial L}{\partial w_{d+1}}\right].$$

$In\ general\ case, the\ update\ for\ each\ weight\ vector\ is\ given\ by,$

$$w = w - \eta\nabla L(w)$$

$However, In\ Coordinate\ update\ (which\ I\ call\ \max\ \_cordinate\ method\ for\ each\ iteration,$
$we\ are\ finding\ the\ maximum\ component\ of\ the\ gradient\ \nabla L(w),$

$$index_{update} = argmax\left(\mathrm{abs}(\nabla L(w))\right)$$

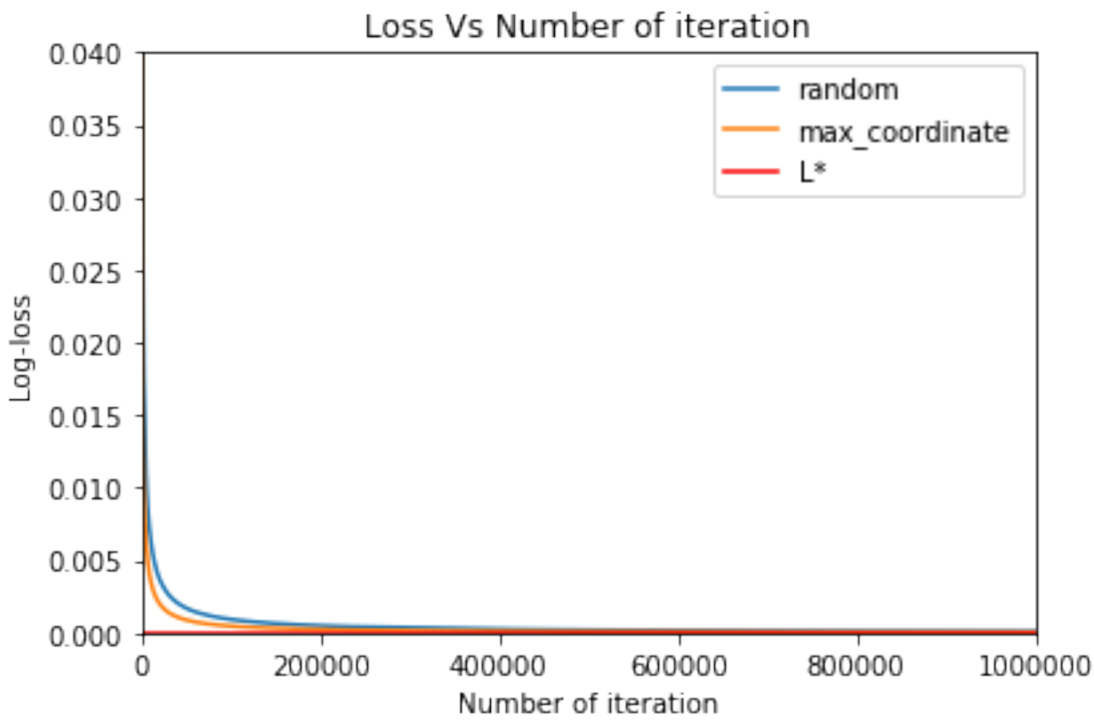
$and\ we\ are\ updating\ only\ that\ component\ of\ the\ weight\ vector\ for\ each\ iteration,$

$$w\left[index_{update}\right] = w\left[index_{update}\right] - \eta * \nabla L(w)\left[index_{update}\right]$$

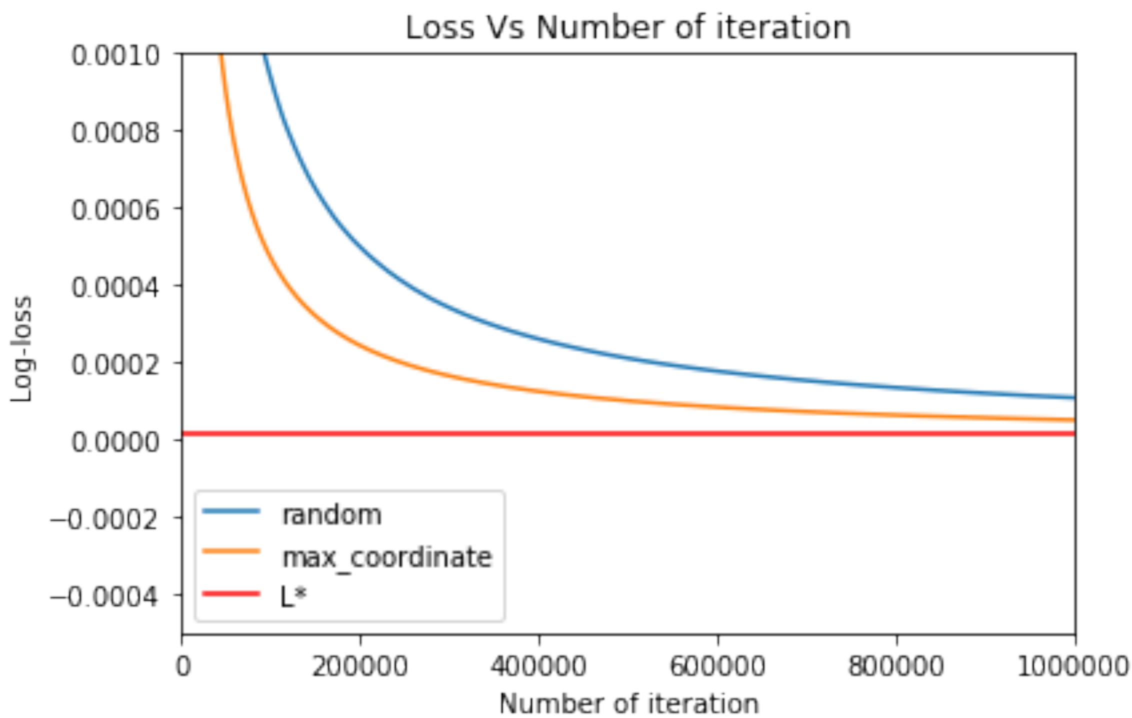
Yes, the loss function needs to be differentiable. The gradient gives the direction of descent. The descent thus will not work with a non-differentiable function.

2. My approach is very similar to general gradient descent update on logistic regression because of the same loss function. For convergence, apart from convexity, the step-size should be small enough so that the loss function does not over jump the minimum instead of approaching it.
3. In scikit-learn , I made regularization irrelevant by choosing a large C. The L^* by running the solver based on ‘liblinear’ was 5.5843851e-5.
   The following is the comparison between my coordinate descent algorithm and random-feature coordinate descent with L^* as horizontal asymptote.

Decrease in Loss for initial 1000000 iteration:



A Closer Look:



The loss after 1000000 iteration is,

$$Max\_coordinate\ method\ = 5.122385\ e - 5$$
$$Random\ method = 1.08\ e - 4$$

As we can see, the Max_coordinate method has reached the same order of loss as the scikit-learn method. Howerver, the random method is slow to reduce to that level and hence, we could see that Max_coordinate method is faster than random choice method.

4. Critical Analysis:
   The improvement will be to converge to a comparable loss with far less computation. One way to do this might be to pick n-large coordinate (like top 5 largest value in the gradient calculation) instead of maximum coordinate at each update. Thus, I expect the loss to converge faster because we are using multiple useful update at each iteration. I could also do stochastic update, that is, instead of going through all the example, I could go through specific examples to get the gradient update and thereby reducing computation cost.