1. Mathematical derivation of cyclical coordinate descent algorithm of Logistic Regression with NO PENALTY

Let $\mathbf{y} = (y_1, \cdots, y_n)^{'}$, $\mathbf{X} = (\mathbf{1}, \mathbf{x}_1, \cdots, \mathbf{x}_p)$, $\mathbf{x}_j = (x_{1j}, \cdots, x_{nj})^{'}$, $\boldsymbol{\beta} = (\beta_0, \cdots, \beta_p)$, $\sigma(x) = [1 + exp(-x)]^{-1}$, $p_i = \sigma(\boldsymbol{\beta}^{'}\mathbf{x}_i)$. Then, the log likelihood of logistic regression is

$$L(\boldsymbol{\beta} \mid \mathbf{x}) = \prod_{i=1}^{n} p_i^{y_i}(1 - p_i)^{1-y_i} \tag{1}$$

Maximizing (1) is equivalent to minimize

$$\mathcal{L} = -log\, L(\boldsymbol{\beta} \mid \mathbf{x}) \tag{2}$$

If $\mathcal{L}$ in (2) is convex function, we could apply Newton's method to get numerical solution of (2). Let's verify this step by step

Deriving $\nabla_{\boldsymbol{\beta}}\mathcal{L}$

- $\dfrac{\partial}{\partial \beta_j} logp = \dfrac{\partial}{\partial \beta_j}\left(log \dfrac{1}{1 + exp(-\boldsymbol{\beta}^{'}\mathbf{x})}\right) = \dfrac{x_j exp(-\boldsymbol{\beta}^{'}\mathbf{x})}{1 + exp(-\boldsymbol{\beta}^{'}\mathbf{x})} = \dfrac{x_j}{1 + exp(\boldsymbol{\beta}^{'}\mathbf{x})} = x_j(1 - p)$
- $\dfrac{\partial}{\partial \beta_j} log(1 - p) = \dfrac{\partial}{\partial \beta_j}\left\{-\boldsymbol{\beta}^{'}\mathbf{x} - log(1 + exp(-\boldsymbol{\beta}^{'}\mathbf{x}))\right\} = -x_j + x_j(1 - p) = -px_j$

$$\begin{aligned} \therefore \frac{\partial}{\partial \beta_j}\mathcal{L} &= -\sum_{i=1}^{n}\{y_i x_{ij}(1 - p_i) + (1 - y_i)(-p_i x_{ij})\} \\ &= -\sum_{i=1}^{n}\{y_i x_{ij} - y_i x_{ij} p_i - p_i x_{ij} + y_i p_i x_{ij}\} \\ &= \sum_{i=1}^{n} x_{ij}(p_i - y_i) \end{aligned}$$

$$\therefore \nabla_{\boldsymbol{\beta}}\mathcal{L} = \mathbf{X}^{'}(\mathbf{p} - \mathbf{y}) \tag{3}$$

Because gradient w.r.t. $\boldsymbol{\beta}$ is not linear in $\boldsymbol{\beta}$, we need to apply Newton's method.

Deriving $\nabla^2_{\boldsymbol{\beta}}\mathcal{L}$ (Hessian Matrix)

If we show that the Hessian matrix is positive semi-definite, $\mathcal{L}$ is convex function and we can apply Newton's method to get numerical solution in (3). Note that

$$\frac{\partial^2}{\partial \beta_j \partial \beta_k}\mathcal{L} = \sum_{i=1}^{n} x_{ij}\frac{\partial}{\partial \beta_k}p_i$$

Because $\partial logp = \frac{1}{p}\partial p \Longleftrightarrow \partial p = p\,\partial logp = px_j(1-p)$, it is obviout that $\frac{\partial}{\partial \beta_k}p_i = x_{ik}p_i(1-p_i)$. Therefore,

$$
\begin{aligned}
\sum_{i=1}^{n} x_{ij}\frac{\partial}{\partial \beta_k}p_i &= \sum_{i=1}^{n} x_{ij}x_{ik}p_i(1-p_i) \\
&= \mathbf{x}_j^{'}\mathbf{W}\mathbf{x}_k
\end{aligned}
$$

$$
where\ \mathbf{W} = \begin{bmatrix} p_1(1-p_1) & & & \\ & \ddots & & \\ & & \ddots & \\ & & & p_n(1-p_n) \end{bmatrix}
$$

$$
\therefore \nabla^2_{\boldsymbol{\beta}}\mathcal{L} = \mathbf{X}^{'}\mathbf{W}\mathbf{X} = \left(\mathbf{W}^{1/2}\mathbf{X}\right)^{'}\left(\mathbf{W}^{1/2}\mathbf{X}\right) \tag{4}
$$

The eigen values of $\nabla^2_{\boldsymbol{\beta}}\mathcal{L}$ are non-negative, so $\nabla^2_{\boldsymbol{\beta}}\mathcal{L}$ is p.s.d. matrix. Therefore, $\mathcal{L}$ is convex function w.r.t. $\boldsymbol{\beta}$.

Newton's method

The general iterative equation for getting numerical solution via Newton's method is

$$
\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \mathbf{H}^{-1}\mathbf{g}
$$

where $\mathbf{H}$ is Hessian matrix and $\mathbf{g}$ is gradient w.r.t. $\boldsymbol{\beta}$. Using (3), (4) results, we get

$$
\begin{aligned}
\boldsymbol{\beta}_{t+1} &= \boldsymbol{\beta}_t - \mathbf{H}^{-1}\mathbf{g} \\
&= \boldsymbol{\beta}_t - \left(\mathbf{X}^{'}\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^{'}(\mathbf{p}-\mathbf{y}) \\
&= \left(\mathbf{X}^{'}\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^{'}\mathbf{W}\left(\mathbf{X}\boldsymbol{\beta}_t - \mathbf{W}^{-1}(\mathbf{p}-\mathbf{y})\right) \\
&= \left(\mathbf{X}^{'}\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^{'}\mathbf{W}\mathbf{z}_t
\end{aligned}
$$

$$
where\ z_t = \mathbf{X}\boldsymbol{\beta}_t - \mathbf{W}^{-1}(\mathbf{p}-\mathbf{y}) \tag{5}
$$

From (5), we can see that $\boldsymbol{\beta}_{t+1}$ is solution of weighted least squares, where we minimize following quantity.

$$
argmin_{\boldsymbol{\beta}} \sum_{i=1}^{n} b_i(z_i - \boldsymbol{\beta}^{'}\mathbf{x}_i)^2,\ b_i = p_i(1-p_i) \tag{6}
$$

In conclusion, to get mle, minimizing quantity (2) is equivalent to minimizing quantity (6).

2. Adding Lasso Penalty to Logistic Regression.

Note that the objective function of logistic regression with lasso penalty will be

$$Q(\boldsymbol{\beta}) = -log\, L(\boldsymbol{\beta} \mid \mathbf{x}) + \lambda \sum_{j=1}^{p} \mid \beta_j \mid \tag{7}$$

However, we show that minimizing $Q(\boldsymbol{\beta})$ is equal to minimizing

$$Q(\boldsymbol{\beta})^N = \sum_{i=1}^{n} b_i (z_i - \boldsymbol{\beta}^{'} \mathbf{x}_i)^2 + \lambda \sum_{j=1}^{p} \mid \beta_j \mid \tag{8}$$

Now, we can derive cyclical coordinate descent using gradient and subgradient w.r.t. $\boldsymbol{\beta}$.

$$\begin{aligned}
\frac{\partial}{\partial \beta_j} \left(\sum_{i=1}^{n} b_i (z_i - \boldsymbol{\beta}^{'} \mathbf{x}_i)^2 \right) &= -2 \sum_{i=1}^{n} b_i x_{ij} \left(z_i - \sum_{j=0}^{p} \beta_j x_{ij} \right) \\
&= -2 \sum_{i=1}^{n} b_i x_{ij} \left(z_i - \sum_{k \neq j}^{p} \beta_j x_{ij} \right) + 2\beta_j \sum_{i=1}^{n} b_i x_{ij}^2 \\
&= -2\rho_j + 2\beta_j q_j
\end{aligned}$$

The subgradient of $\lambda \mid \beta_j \mid$ is

$$\lambda\, \partial_{\beta_j} \mid \beta_j \mid = \begin{cases} -\lambda & \beta_j < 0 \\ [-\lambda, \lambda] & \beta_j = 0 \\ \lambda & \beta_j > 0 \end{cases}$$

Therefore,

$$\begin{aligned}
\partial_{\beta_j} (Lasso\, Cost) &= -\, 2\rho_j + 2\beta_j q_j + \begin{cases} -\lambda & \beta_j < 0 \\ [-\lambda, \lambda] & \beta_j = 0 \\ \lambda & \beta_j > 0 \end{cases} \\
&= \begin{cases} -2\rho_j + 2\beta_j q_j - \lambda & \beta_j < 0 \\ [-2\rho_j - \lambda, -2\rho_j + \lambda] & \beta_j = 0 \\ -2\rho_j + 2\beta_j q_j + \lambda & \beta_j > 0 \end{cases}
\end{aligned}$$

Finally, setting subgradient to zero, we get the soft thresholding as

$$\hat{\beta}_j = \begin{cases} \dfrac{\rho_j + \lambda/2}{q_j} & \rho_j < -\lambda/2 \\ 0 & -\lambda/2 \leq \rho_j \leq \lambda/2 \\ \dfrac{\rho_j - \lambda/2}{q_j} & \lambda/2 < \rho_j \end{cases}$$

Algorithm 1 Cyclical Coordinate Descent for Penalized Logistic Regression

Input : $\mathbf{y}, \mathbf{X}, \lambda$
Output : $\hat{\boldsymbol{\beta}}$
Initialize $\hat{\beta}_j$
Precompute q_j
while not converged **do**
    Compute ρ_j
    Compute $\hat{\beta}_j$
end while

The pseudo code for CCD is summarized in Algorithm 1.