4033/5033: Assignment 4

Your Name

Due: Oct 28 (by 11:59pm)

In this assignment, we will implement logistic regression using two numerical optimization techniques, namely, gradient descent and Newton-Raphson. We will implement the following logistic model, which is different from but equivalent to the lectured one. (Models for $y = 0$ and $y = 1$ are swapped.)

$$\Pr(y_i = 0 \mid x_i) = \frac{1}{1 + \exp(-x_i^T \beta)} \tag{1}$$

and

$$\Pr(y_i = 1 \mid x_i) = \frac{\exp(-x_i^T \beta)}{1 + \exp(-x_i^T \beta)}. \tag{2}$$

Let the log likelihood function be

$$L(\beta) = \sum_{i=1}^{n} \log \Pr(y_i \mid x_i). \tag{3}$$

Task 1. Derive the following equation based on (1) (2) and (3).

$$L(\beta) = \sum_{i=1}^{n} (1 - y_i) x_i^T \beta - \log[1 + \exp(x_i^T \beta)]. \tag{4}$$

Task 2. Derive the following equations based on (4).

$$\frac{\partial L(\beta)}{\partial \beta} = \sum_{i=1}^{n} (y_i - \Pr(y_i = 1 \mid x_i)) \cdot x_i = X^T (Y - P_1), \tag{5}$$

where P_1 is an n-dimensional vector with the i_{th} element being $\Pr(y_i = 1 \mid x_i)$ and X is an n-by-p matrix with the i_{th} rowing being x_i^T. (Here, x_i is a p-dimensional feature (column) vector.)

Task 3. Implement the gradient descent algorithm to optimize β. In this algorithm, we first randomly pick a β and then start a loop: in each iteration, we update β based on

$$\beta = \beta - \lambda \frac{\partial L(\beta)}{\partial \beta}, \tag{6}$$

where λ is a hyper-parameter we manually pick (called 'learning rate').

Draw a curve of the testing error of the updated β versus the number of updates. For example, the first point on this curve should be the error of the randomly initialized β, and the second point on this curve should be the error of β after one round of update i.e., one round of applying (6). Pick a proper λ yourself so we can observe a somewhat smooth and convergent curve.

Task 4. Implement the Newton-Raphson algorithm to optimize β. In this algorithm, we first randomly pick a β and then start a loop: in each iteration, we find the optimal point of the 2nd-order Tylor approximation of $L(\beta)$ at the current point β_*. The Tylor approximation has the form

$$J(\beta) = L(\beta_*) + \left(\frac{\partial L(\beta_*)}{\partial \beta}\right)^T (\beta - \beta_*) + \frac{1}{2} (\beta - \beta_*)^T \left(\frac{\partial L(\beta_*)}{\partial \beta \partial \beta^T}\right) (\beta - \beta_*), \tag{7}$$

where $\frac{\partial L(\beta_*)}{\partial \beta}$ is a p-dimensional vector and $\frac{\partial L(\beta_*)}{\partial \beta \partial \beta^T}$ is a p-by-p dimensional matrix called Hessian matrix.

We can evaluate $\frac{\partial L(\beta_*)}{\partial \beta}$ based on (5) – just plug in β_*.

We can evaluate $\frac{\partial L(\beta_*)}{\partial \beta \partial \beta^T}$ based on the following – again, just plug β_* into the two probabilities:

$$\frac{\partial L(\beta)}{\partial \beta \partial \beta^T} = \sum_{i=1}^{n} [-\Pr(y_i = 1 \mid x_i) \Pr(y_i = 0 \mid x_i)] \cdot x_i x_i^T = -X^T W X, \tag{8}$$

where W is an n-by-n diagonal matrix with the i_{th} diagonal element being $\Pr(y_i = 1 \mid x_i) \Pr(y_i = 0 \mid x_i)$.

Task 4.1. Derive the optimal β that minimizes $J(\beta)$ and show it has the following form:

$$\beta = \beta_* - \left(\frac{\partial L(\beta_*)}{\partial \beta \partial \beta^T} \right)^{-1} \frac{\partial L(\beta_*)}{\partial \beta}. \tag{9}$$

Task 4.2. Implement the algorithm and draw a curve of the testing error of the updated β versus the number of updates. For example, the first point on this curve should be the error of the randomly initialized β, and the second point should be the error of β after one round of update i.e., one round of applying (9).

Tip: In general, we should expect this curve to converge faster than the curve of gradient descent.

<u>More Instructions on Experiments</u>

1. Let us experiment on the diabetes data set. (Given in the hw4 zip file.)

2. Let us use the first 75% of data for training and the rest for testing in all experiments. (No need to randomly split data or report average performance.)

3. It is recommended to initialize β using Gaussian distribution.

<u>Submission Instruction</u>

Please submit three files to Canvas. (Do not zip them. Upload them separately.)

(i) All your mathematical and experimental results should be presented in a single pdf file named as ‘hw4.pdf’.

(ii) A Python source code for the implementation of gradient descent named ‘hw4_gra.py’

(iii) A Python source code for the implementation of Newton-Raphson named ‘hw4_new.py’