SCHOOL OF PUBLIC HEALTH
UNIVERSITY OF MICHIGAN

Biostat 615 Final Project

User Manual

Maximum Likelihood Estimation of 2-Class Logistic Regression

Kaifeng Chen, Lu Zhang

Project Description:

We seek to compute Maximum Likelihood Estimation (MLE) of parameters in logistic regression through implementation of the Coordinate Descent algorithm in C++. We have included 5 test cases with 4 being simulated data (simulated from R - source code included in submission folder) and 1 real data (data included in submission folder).

Data Requirement:

To facilitate estimation, we scaled data to have mean 0 and variance 1. Please center and scale predictors when using our algorithm. We also assume that the dependent variable Y is in the first column of data. Moreover, in design matrix X a column of 1's is included in the second column to have intercept estimated.

Algorithm Parameter Selection:

- **Learning rate:** Learning rate determines the step size of each update of the parameter to reach a (local) maximum.The default value is 10/N (N: number of observations). We decide to choose the value for the parameter, because a false choice of learning rate can lead to divergence of the algorithm.
- **Tolerance:** Tolerance determines the stopping criterion of the algorithm. The default value is 10^{-6}, but users can input their choice of tolerance. We recommend the range to be between 10^{-8} and 10^{-2} in order to get reasonably good MLE coefficients.
- **Sliding window:** Sliding window serves for the purpose of smoothing mean statistics of the differences between $\widehat{\theta}_{t+1}$ and $\widehat{\theta}_t$. For reference, see a generally speaking sub-list that runs over an underlying collection.

Command Line Usage:

Option 1: 3 arguments from user: data file name, number of observations, number of columns

- g++ -I ~jiankang/Public/include -O -o f1 Coordinate_Descent.cpp
- ./f1 wine.txt 130 6

Option 2: 4 arguments from user: data file name, number of observations, number of columns, and tolerance

- g++ -I ~jiankang/Public/include -O -o f1 Coordinate_Descent.cpp
- ./f1 wine.txt 130 6 0.0001

Example Output:

Experiments of Synthetic Data
Example 1:

- **Input:** ./f1 df.txt 5000 10
- **Linear function of synthetic data:**
  $y_1 = 1 + 3 \times x_1 + 1 \times x_2 + 2 \times x_3 - 0.4 \times x_4 - 1.2 \times x_5 + 0.5 \times x_6 + 2 \times x_7 + 2 \times x_8$
- **C++ output:**

```
number of iterations is: 131
theta is: 1.11721 3.10379 0.968147 2.13778 -0.361228 -1.17864 0.512406 2.0963 1.99793
```

- **R output:**

```
             Estimate Std. Error
(Intercept)  1.1172632 0.06096755
x1           3.1039360 0.10616145
x2           0.9681922 0.06021692
x3           2.1378786 0.08121961
x4          -0.3612449 0.05366943
x5          -1.1786964 0.06178139
x6           0.5124271 0.05545961
x7           2.0964018 0.08136958
x8           1.9980210 0.07870742
```

Example 2:

- **Input:** ./f1 df2.txt 1000 5
- **Linear function of synthetic data:** $y_2 = 1 + 2 \times x_1 + 3 \times x_2 + 4 \times x_3$
- **C++ output:**

```
number of iterations is: 124
theta is: 1.05262 1.9216 2.78851 3.96822
standard error is: 0.136519 0.181656 0.226533 0.293335
```

- **R output:**

```
            Estimate Std. Error
(Intercept) 1.061652  0.1373236
x1          1.938300  0.1831515
x2          2.812598  0.2287150
x3          4.001614  0.2965097
```

Example 3:

- **Input:** ./f1 df3.txt 100000 6
- **Linear function of synthetic data:** $y_3 = 1 + 5 \times x_1 + 1 \times x_2 + 4 \times x_3 - 0.5 \times x_4$
- **C++ output:**

```
number of iterations is: 230
theta is: 0.997243 5.01083 1.00954 4.00558 -0.502369
```

- **R output:**

```
             Estimate
(Intercept)  0.9972553
x1           5.0108943
x2           1.0095497
x3           4.0056275
x4          -0.5023751
```

Example 4:

- **Input:** ./f1 df4.txt 1000000 6
- **Linear function of synthetic data:** $y_4 = 1 + 3 \times x_1 + 1 \times x_2 + 2 \times x_3 + 1.5 \times x_4$
- **C++ output:**

```
number of iterations is: 56
theta is: 1.00023 3.00591 0.99892 2.00273 1.49924
```

- **R output:**

```
             Estimate
(Intercept) 1.0002306
x1          3.0059142
x2          0.9989226
x3          2.0027373
x4          1.4992393
```

Experiments of Real Data

Example 5:

- **Input:** ./f1 wine.txt 130 6
- **C++ output:**

```
number of iterations is: 26
theta is: -0.353655 0.271106 -1.64285 0.249993 1.25987
standard error is: 0.234703 0.25243 0.344676 0.248507 0.299773
```

- **R output:**

```
             Estimate Std. Error
(Intercept) -0.3536465  0.2346999
V3           0.2710960  0.2524280
V4          -1.6428155  0.3446662
V5           0.2499850  0.2485045
V6           1.2598566  0.2997678
```

Error Message:

Example 6:

- **Input:** ./f1 df10.txt 1 1
- **C++ output:**

```
Incorrect filename: df10.txt
Abort (core dumped)
```

Example 7:

- **Input:** ./f1 df2.txt 1000 5 0
- **C++ output:** Algorithm cannot converge within 1000 iterations

Note:

- For data simulation, we observed that when true θ_j is as large as 4 or 5, the estimation in R may not converge, so we did not try those θ_j .
- We also note that, the calculation of standard errors for parameters $SE(\widehat{\theta}_j) = \sqrt{Diag_j((X^T W X)^{-1})}$ become infeasible with current computing capacity (server on scs.itd.umich.edu), so we restrict our calculation to data size of not bigger than 1,000. For our test cases, the standard errors are calculated for df.txt, df2.txt, and wine.txt, but not for df3.txt or df4.txt.
- We tested data of up to 1 million observations and the C++ implementation of our algorithm converges in a couple seconds. We did not test the cases with data size larger than 1 million.
- For our real dataset wine.txt, we do not have true parameters. But the results are close to what is given by the glm function in R.