Logistic Regression

General summary:

I write my own logistic regression classifier to solve the Titanic problem on Kaggle, deciding which one should be survived based on the available information. I first did the whole mathematical deduction to make sure my fully understanding about the method. Then I re-wrote the logistic regression in matrix form, because it is more efficient and easier to be implemented in code using the manipulation of matrix provided by numpy and pandas package. Last, I discuss the difference between the normal Gradient Descent and Stochastic Gradient Descent.

Mathematical Deduction:

Different from linear regression, which uses $Y = \beta^T X$ to classify samples, logistic regression use sigmoid function, $Y = \frac{1}{1+e^{-\beta^T X}}$, as classifier.

We make

$$\log\left(\frac{\Pr(G = 1|X = x)}{\Pr(G = 2|X = x)}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

where $\Pr(G = 2|X = x) = 1 - \Pr(G = 1|X = x)$

Denote

$$\Pr(G = 1|X = x) = p_1(x; \beta)$$
$$\Pr(G = 2|X = x) = p_2(x; \beta)$$

We have

$$p_1(x; \beta) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p)}}$$

Now we want to maximize the probability

$$\text{Likelyhood} = \prod_{G=1} p_1(x_i;\beta) \prod_{G=2} p_2(x_i;\beta)$$

$$\log(\text{Likelyhood}) = l(\beta)$$

$$= \sum_{G=1} \log\left(p_1(x_i;\beta)\right) + \sum_{G=2} \log\left(p_2(x_i;\beta)\right)$$

$$= \sum_{i=1}^{N} y_i \log\left(p_1(x_i;\beta)\right) + (1-y_i)\log\left(1-p_1(x_i;\beta)\right)$$

$$= \sum_{i=1}^{N} y_i\left(-\log\left(1+e^{-\beta^T x_i}\right) + (1-y_i)\log\left(\frac{e^{-\beta^T x_i}}{1+e^{-\beta^T x_i}}\right)\right.$$

$$= \sum_{i=1}^{N} -y_i \log\left(\frac{1+e^{\beta^T x_i}}{e^{\beta^T x_i}}\right) + (1-y_i)\log\left(\frac{1}{1+e^{\beta^T x_i}}\right)$$

$$= \sum_{i=1}^{N} -y_i \log\left(1+e^{\beta^T x_i}\right) + y_i e^{\beta^T x_i} - \log\left(1+e^{\beta^T x_i}\right) + y_i \log\left(1+e^{\beta^T x_i}\right)$$

$$= \sum_{i=1}^{N} y_i e^{\beta^T x_i} - \log\left(1+e^{\beta^T x_i}\right)$$

We use Gradient Descent to find out the β that make $l(\beta)$ maximized. That is, find out the β that make $-l(\beta)$ minimized.

$$\frac{\partial l(\beta)}{\partial \beta} = \frac{\partial l(\beta)}{\partial \beta_0} i + \frac{\partial l(\beta)}{\partial \beta_0} j + \cdots + \frac{\partial l(\beta)}{\partial \beta_p} p = \begin{bmatrix} \frac{\partial l(\beta)}{\partial \beta_0} \\ \frac{\partial l(\beta)}{\partial \beta_0} \\ \vdots \\ \frac{\partial l(\beta)}{\partial \beta_p} \end{bmatrix}$$

where $\frac{\partial l(\beta)}{\partial \beta_0} = \sum_{i=1}^{N} y_i - \frac{1}{1+e^{-\beta^T x_i}} = \sum_{i=1}^{N} y_i - p_1(x_i;\beta)$,

$$\frac{\partial l(\beta)}{\partial \beta_j} = \sum_{i=1}^{N} (y_i - \frac{1}{1+e^{-\beta^T x_i}}) x_{ij} = \sum_{i=1}^{N} (y_i - p_1(x_i;\beta))\, x_{ij}$$

We have

$$\beta_i^{new} = \beta_i^{old} - \alpha \frac{\partial l(\beta)}{\partial \beta_i}$$

Re-write all the above in matrix form:

$$Y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ \vdots \\ 0 \end{bmatrix} N*1 \qquad X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & \cdots & x_{Np} \end{bmatrix} N*(p+1)$$

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix} (p+1)*1 \qquad \frac{\partial l(\beta)}{\partial \beta} = \begin{bmatrix} \frac{\partial l(\beta)}{\partial \beta_0} \\ \frac{\partial l(\beta)}{\partial \beta_0} \\ \vdots \\ \frac{\partial l(\beta)}{\partial \beta_p} \end{bmatrix} (p+1)*1$$

$$P = \begin{bmatrix} p_1(x_1; \beta) \\ p_1(x_2; \beta) \\ \vdots \\ p_1(x_N; \beta) \end{bmatrix} N*1$$

To minimize $-l(\beta) = -Y^T \log(P) - (1-Y)^T \log(1-P)$:

$$-\frac{\partial l(\beta)}{\partial \beta} = X^T(P - Y)$$

$$\beta^{new} = \beta^{old} - \alpha\left(-\frac{\partial l(\beta)}{\partial \beta}\right) = \beta^{old} - \alpha(X^T(P - Y))$$

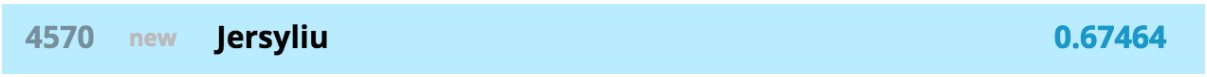
Code: There is a LogisticRegression.py in the attachment.

Result:

I used the data from Kaggle to train the β. When preparing the training data, I drop "Name", "Ticket", "Cabin", "Embarked", and "PassengerId". And I used

the mean of age and fare to fill in the missing data of “Age” and “Fare” respectively.

When testing the model using the test data also from Kaggle, I got 0.67464 as correct rate.



I think the reason of the not high correct rate is that I didn’t fully use the information of training data. I drop too many variables that maybe relevant. However, if I included those information, there would have been a lot more work to do during the preprocessing data process. Since the whole point of this exercise is to make us familiar with Logistic Regression, I think I should just leave the correct rate there.

Insight:

Neither Newton-Raphson method nor Gradient Decent method is a good method in industry. Although theoretically they are correct, they are not efficient when they encounter a tremendous dataset. Therefore, in industry, the most often used method is Stochastic Gradient Decent, which update β every time using only one individual sample. SGD is faster and has its advantage when dealing with non-convex optimization problem because SGD vacillate stronger than normal GD and thus more likely to jump out of local optimal and find the global optimal.