Lecture 4: Logistic Regression

Instructor: Prof. Shuai Huang

Industrial and Systems Engineering

University of Washington

Extend linear model for classification

- Need a mathematic transfer function to connect $\beta_0 + \sum_{i=1}^{p} \beta_i x_i$ with a binary outcome y
- How?
- Logistic regression chooses to use

$$\log \frac{p(\boldsymbol{x})}{1-p(\boldsymbol{x})} = \beta_0 + \sum_{i=1}^{p} \beta_i x_i.$$

- Why?

Justification for the logistic regression model

- It works in many applications
- It leads to analytical tractability (in some senses) and encourages in-depth theoretical investigation
- It has a strong tie with linear regression model. Therefore, methodologically there is much we can translate from linear regression to logistic regression. Conceptually, it inherits the aura of linear regression model and users can assume a similar degree of confidence of linear regression model onto the logistic regression model

Parameter estimation

- The likelihood function is:

$$L(\boldsymbol{\beta}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n)^{y_n}\big(1 - p(\boldsymbol{x}_n)\big)^{1-y_n}.$$

- We use the log-likelihood to turn products into sums:

$$l(\boldsymbol{\beta}) = \sum_{n=1}^{N}\big\{y_n \log p(\boldsymbol{x}_n) + (1-y_n)\log\big(1 - p(\boldsymbol{x}_n)\big)\big\}.$$

This could be further transformed into

$$l(\boldsymbol{\beta}) = \sum_{n=1}^{N} - \log\left(1 + e^{\beta_0 + \sum_{i=1}^{p}\beta_i x_{ni}}\right) - \sum_{n=1}^{N} y_n\big(\beta_0 + \sum_{i=1}^{p}\beta_i x_{ni}\big),$$

Then we can have

$$\sum_{n=1}^{N}\big\{y_n \log p(\boldsymbol{x}_n) + (1-y_n)\log\big(1 - p(\boldsymbol{x}_n)\big)\big\},$$
$$= \sum_{n=1}^{N}\log\big(1 - p(\boldsymbol{x}_n)\big) - \sum_{n=1}^{N} y_n \log\frac{p(\boldsymbol{x}_n)}{1-p(\boldsymbol{x}_n)},$$
$$= \sum_{n=1}^{N} - \log\left(1 + e^{\beta_0 + \sum_{i=1}^{p}\beta_i x_{ni}}\right) - \sum_{n=1}^{N} y_n\big(\beta_0 + \sum_{i=1}^{p}\beta_i x_{ni}\big).$$

Application of the Newton-Raphson algorithm

- The Newton-Raphson algorithm is an iterative algorithm that seeks updates of the current solution using the following formula:

$$\boldsymbol{\beta}^{new} = \boldsymbol{\beta}^{old} - \left(\frac{\partial^2 l(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T}\right)^{-1} \frac{\partial l(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}}.$$

- We can show that

$$\frac{\partial l(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} = \sum_{n=1}^{N} \boldsymbol{x}_n\big(y_n - p(\boldsymbol{x}_n)\big),$$

$$\frac{\partial^2 l(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T} = -\sum_{n=1}^{N} \boldsymbol{x}_n\boldsymbol{x}_n^T p(\boldsymbol{x}_n)\big(1 - p(\boldsymbol{x}_n)\big).$$

- A certain structure can then be revealed if we rewrite it in matrix form:

$$\frac{\partial l(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} = \mathbf{X}^T(\boldsymbol{y} - \boldsymbol{p}),$$

$$\frac{\partial^2 l(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T} = -\mathbf{X}^T\mathbf{W}\mathbf{X}.$$

where $\mathbf{X}$ is the $N \times (p+1)$ input matrix, $\boldsymbol{y}$ is the $N \times 1$ column vector of y_i, $\boldsymbol{p}$ is the $N \times 1$ column vector of $p(\boldsymbol{x}_n)$, and $\mathbf{W}$ is a $N \times N$ diagonal matrix of weights with the nth diagonal element as $p(\boldsymbol{x}_n)\big(1 - p(\boldsymbol{x}_n)\big)$.

The updating rule

Plugging this into the updating formula of the Newton-Raphson algorithm,

$$\boldsymbol{\beta}^{new} = \boldsymbol{\beta}^{old} - \left(\frac{\partial^2 l(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T}\right)^{-1} \frac{\partial l(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}},$$

we can derive that

$$\begin{aligned}\boldsymbol{\beta}^{new} &= \boldsymbol{\beta}^{old} + \left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{W}(\boldsymbol{y} - \boldsymbol{p}),\\ &= \left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{W}\left(\mathbf{X}\boldsymbol{\beta}^{old} + \mathbf{W}^{-1}(\boldsymbol{y} - \boldsymbol{p})\right),\\ &= \left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{W}\boldsymbol{z},\end{aligned}$$

where $\mathbf{z} = \mathbf{X}\boldsymbol{\beta}^{old} + \mathbf{W}^{-1}(\boldsymbol{y} - \boldsymbol{p})$.

Another look at the updating rule

- This resembles the generalized least squares (GLS) estimator of a regression model, where each data point $(\boldsymbol{x}_n, y_n)$ is associated with a weight w_n to reduce the influence of potential outliers in fitting the regression model.

$$\boldsymbol{\beta}^{new} \leftarrow \arg\min_{\boldsymbol{\beta}} (\boldsymbol{z} - \mathbf{X}\boldsymbol{\beta})^T \mathbf{W} (\boldsymbol{z} - \mathbf{X}\boldsymbol{\beta}).$$

- For this reason, this algorithm is also called the Iteratively Reweighted Least Square or IRLS algorithm. $\boldsymbol{z}$ is referred as the adjusted response.
- Why the weighting makes sense? Or, what are the implications of this?

A summary of the IRLS algorithm

Putting all these together, a complete flow of the IRLS is shown in below:

- Initialize $\boldsymbol{\beta}$.
- Compute $\boldsymbol{p}$ by its definition: $p(\boldsymbol{x}_n) = \frac{1}{1+e^{-\left(\beta_0+\sum_{i=1}^{p} \beta_i x_{ni}\right)}}$ for $n = 1,2,\ldots,N$.
- Compute the diagonal matrix $\mathbf{W}$, while the nth diagonal element as $p(\boldsymbol{x}_n)\big(1 - p(\boldsymbol{x}_n)\big)$ for $n = 1,2,\ldots,N$.
- Set $\boldsymbol{z}$ as $= \mathbf{X}\boldsymbol{\beta} + \mathbf{W}^{-1}(\boldsymbol{y} - \boldsymbol{p})$.
- Set $\boldsymbol{\beta}$ as $\left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{W}\boldsymbol{z}$.
- If the stopping criteria is met, stop; otherwise go back to step 2.

R lab

- Download the markdown code from course website
- Conduct the experiments
- Interpret the results
- Repeat the analysis on other datasets