Logistic Regression and Applications using R Language

1 Synopsis

In order to complete this project successfully, the student should be familiar with many basic notions of statistics, statistical learning, and the **R** language that can be found in the following references

1. *An Introduction to Statistical Learning with Applications in R*, G. James, D. Witten, T. Hastie, and R. Tibshirani, Springer, 2015.
   (available online at: `http://faculty.marshall.usc.edu/gareth-james/ISL/`)
2. The **R** Project for Statistical Computing (at: `https://www.r-project.org/`)
3. *Probability & Statistics for Engineers & Scientists*, R. E. Walpole et al., 9th Edition, Prentice Hall, 2012.

2 Logistic Regression and Classification

Logistic regression is a machine learning classification algorithm used to assign observations to a discrete set of classes. Some of the well known examples of this kind of classification are Email spam or not spam, Online transactions Fraud or not Fraud, Tumor Malignant or Benign. Given a feature vector X and a qualitative response Y taking values in the set $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$, a classification task is to build a function $f : X \to Y$ (classifier) that takes as input the feature vector X and predicts the value for Y, i.e. $Y \in \mathcal{C}$. The model or function or classifier f is built using a set of training observations $(x_1, y_1), \ldots, (x_n, y_n)$ for a given n. In order to avoid crisp decisions and give to the user flexibility in the classification process, often we are more interested in estimating the probabilities that Y belongs to each category in $\mathcal{C}$ than estimating directly the value of Y.

Unlike linear regression where the target variable y is related to the features via the linear relationship:

$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k + \epsilon,$$

in logistic regression, we relate $p(y)$, the probability of Y belonging to a certain class (which ranges between 0 and 1), to the features $x_1, \ldots, x_k$ via the logistic (or logit) transformation given by

$$\log \frac{p(y)}{1-p(y)} = \beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k.$$

(a) Show that

$$p(y) = S\left(\beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k\right)$$

where $S(w)$ is the logistic sigmoid function given by

$$S(w) = \frac{1}{1+e^{-w}} = \frac{e^w}{1+e^w}.$$

(b) Display a graph of $S(w)$ for $-\infty < w < +\infty$ and show that no matter what values $\beta_0, \beta_1, \ldots, \beta_k$ or $x_1, \ldots, x_k$ take, $p(y)$ will always have values between 0 and 1.

To summarize, the logistic regression starts with the idea of linear regression and transforms its output using the sigmoid function to return a probability value.

2.1 Maximum Likelihood Estimation (MLE) of the Model

In logistic regression, our goal is to learn a set of parameters $\beta^T = (\beta_0, \beta_1, \ldots, \beta_n)$ using the available training data. For linear regression, the typical method used is the least squares estimation. Although we could use (non-linear) least squares to fit the logistic regression model, the more general method of maximum likelihood estimation (MLE) is preferred, since it has better statistical properties. The idea behind MLE is to choose the most likely values of the parameters $\beta_0, \ldots, \beta_n$ given the observed sample

$$\{(x_1^{(i)}, \ldots, x_k^{(i)}, y_i), 1 \leq i \leq n\}.$$

In logistic regression, the probability model is based on the binomial distributions:

$$f(x_i, p_i) = f(y_i, p_i) = \begin{cases} p_i & \text{if } y_i = 1 \\ 1-p_i & \text{if } y_i = 0, \end{cases}$$

where $x_i = (x_1^{(i)}, \ldots, x_k^{(i)})$ is the vector of features and $0 < p_i < 1$ are the probabilities associated to the binomials in the model. In other words, the probability of the feature vector x_i specifying the class $y_i = 1$ occurs with probability p_i, that is

$$p(y_i = 1) = p_i = \frac{e^{\beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_k x_k^{(i)}}}{1 + e^{\beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_k x_k^{(i)}}} = \frac{e^{x_i^T \beta}}{1 + e^{x_i^T \beta}}$$

where $\beta = (\beta_0, \ldots, \beta_k)^T$ and $x_i = (1, x_1^{(i)}, x_2^{(i)}, \ldots, x_k^{(i)})$ for $1 \leq i \leq n$.

(1) Assuming you are given a dataset with n training examples and k features, write down a formula for the conditional likelihood $L(\beta)$ and the log likelihood $\ell(\beta) = \log L(\beta)$ of the training data in terms of the class labels y_i, the features $x_1^{(i)}, \ldots, x_k^{(i)}$, and the parameters $\beta_0, \beta_1, \ldots, \beta_k$, where the superscript (i) denotes the sample index. This will be your `objective function` (or `cost function`).

(2) Compute the partial derivative of the objective function with respect to β_j, i.e. derive $\partial\ell/\partial\beta_j$, for $0 \leq j \leq k$ where ℓ is the objective function that you provided above. Show particularly that

$$\frac{\partial \ell}{\partial \beta_j} = \sum_{i=1}^{n} (y_i - p_i) x_j^{(i)}, \; 0 \leq j \leq k, \tag{1}$$

where

$$p_i = p(y_i = 1) = \frac{e^{\beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_k x_k^{(i)}}}{1 + e^{\beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_k x_k^{(i)}}}.$$

We recall that for $j = 0$, we set $x_0^{(i)} = 1$ for every i.

(3) To maximize the log-likelihood, we set the partial derivatives (1) to zero. Show that when $j = 0$, the score equation is equivalent to

$$\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} p_i.$$

Does this equation have any statistical interpretation?

(4) The obtained equations are known to be transcendental and not to have closed form solutions. Explain this statement by studying the equations and by researching relevant literature on this subject.

(5) Typically, numerical approximations and optimization procedures are used to find the (best) vector β^* satisfying

$$\beta^* = \underset{\beta}{\operatorname{argmax}}(\ell(\beta)).$$

There are many known techniques and **R** has built-in methods to achieve a solution. However, your task here is to implement your own procedure which is a logistic regression classifier using gradient ascent as illustrated in the following algorithm

Algorithm 1 [Algorithm for finding β^*]

1: `Set` $\eta \in [0, 1]$ `(learning coefficient)`
2: `Set` $\epsilon > 0$ `(tolerance term)`
3: $\beta^{(0)} \leftarrow initial\_value$
4: **for** `t = 0, 1, ...` **do**
5: `Compute the gradient:` $g_t = \nabla\ell(\beta^{(t)})$
6: `Update the coefficients:` $\beta^{(t+1)} \leftarrow \beta^{(t)} + \eta g_t$
7: `Iterate until:` $||\beta^{(t+1)} - \beta^{(t)}|| < \epsilon$.
8: **end for**
9: `Return the final coefficients:` $\beta^{(t^{\text{final}})}$

You are asked to provide an implementation in the simple case where there is only one feature variable x_1. For this part of the assignment, you have to submit **R** code where the following procedures are clearly identified:

- Normalize the feature variable x_1 (see the **Nota-Bene** below).
- Calculate the value of the objective function $\ell(\beta_0, \beta_1)$.
- Choose a value of the learning rate η (you should try different values).
- Initialize the parameter value and calculate the gradient $\nabla\ell(\beta_0, \beta_1)$.
- Update the parameter value.
- Check whether gradient ascent has converged.

  Here, it is much better to look at the convergence of the values of $\ell(\beta_0, \beta_1)$ than the convergence of the parameters themselves.
- Complete the implementation of gradient ascent.
- Predict the labels for a set of test examples.

  I am including the data set `SAheart.data` in which you should predict the value of the variable `chd` (response, coronary heart disease diagnosis) from the feature value `ldl` (low density lipoprotein cholesterol). You may use the first 100 rows for training and any values in the remaining rows for testing.

Nota-Bene. In machine learning, it is a standard routine to normalize or scale the feature variables to speed up the convergence of the learning algorithms and to ensure that the features contribute equally to the learning task. One way to achieve the normalization is by making the values of each feature in the data have zero mean (when subtracting the mean in the numerator) and unit variance, i.e., we replace the variable x_1 with $\frac{x_1 - \mu}{\sigma}$, where μ and σ are respectively the mean and the standard deviation of the values of x_1 in the training data.