Logistic regression

Tutorial/Lab work: Week 6

Instructions: This is a coding exercise. Kindly stage+commit+push the `MATLAB` code to the "`week06`" directory on your INM431 Github repository. Grading: All attempts will get 1 point. **Please make sure that the document is pushed to Github by 24/11/2021**.

Background

Logistic regression seeks to model the relationship between a dependent variable and one or more independent variables: $x = \{x_1, \cdots, x_n\}$. As in the case of linear regression, logistic regression allows us to check the fit of the model as well as the significance of the relationships between dependent and independent variables by inspecting the learned coefficients θ such that $\theta^\top x = w_0 + w_1 x_1 + \cdots + w_n x_n$. However, while linear regression uses least squares to find a best fitting curve and come up with the coefficients that predict the change in the dependent variable given changes in the independent variables, logistic regression estimates the probability of a binary event occurring (e.g., the probability P of passing the exam or $1 - P$ of failing the exam given the number of hours that a person studied for that exam. The data in this case will include a target variable $y = \{0, 1\}$ for fail/pass, but the model is expected to predict the probability P)[1].

What one wants to predict from the data is not the numerical value of a dependent variable, but rather a probability of the event. In Logistic Regression, the natural log[2] of the odds ratio $\frac{P}{(1-P)}$ is assumed to be linear, that is, $\log\left(\frac{P}{(1-P)}\right) = \theta^\top x$. If P were known, this ratio could be predicted by estimating values for θ using linear regression. Odds ratio are defined as the ratio between the probabilities of an event occurring to it not occurring. For example, if the probability P of passing the exam is 0.8, the probability of not passing will be $Q = 1 - P = 0.2$. The odds of passing are then defined as:

$$\text{odds(passing)} = \frac{P}{Q} = 4(\text{or } 4 \text{ to } 1)$$

The odds of not passing would be:

$$\text{odds(not passing)} = \frac{Q}{P} = 0.25$$

that is, 1 to 4, or 4 to 1 against[3]. Although values of P are not present in the data, something very similar to linear regression will be done

[1] If, instead, the data contained the grades obtained by the students in the range $[0, 100]$ then a linear regression model could be used.

[2] log base e where $e = 2.718$

[3] The use of the term in statistics and gambling is not consistent (with the exception of horse racing). Gambling odds are expressed in the form "X to Y" and it is implied that the odds are odds against, i.e. with X possible outcomes in which the event will not take place.

here simply by applying a sigmoid function $\sigma(a) = \frac{1}{(1+e^{-a})}$ to $\theta^\top x$, which will return our estimated probability p of P. In the case of a single independent variable x, $\theta^\top x$ becomes $w_0 + w_1 x$ and therefore:

$$p = \sigma(w_0 + w_1 x).$$

The value of w_0 should tell us p when x is zero, while w_1 tells us how p should change with x (e.g. the number of hours studied). We are interested in finding the values for the parameters of the model w_0 and w_1 that minimize the difference between $\log\left(\frac{P}{(1-P)}\right)$ and $w_0 + w_1 x$. We need to define a suitable cost function. When $y = 1$, if $p = 1$ then our cost should be zero; if p = 0 then the cost should be very high. $\text{Cost}(p, y) = -\log(p)$ does this. The corollary of this, when $y = 0$, if $p = 1$ then our cost should be very high (∞); if $p = 0$ then the cost should be zero. In this case, $\text{Cost}(p, y) = -\log(1 - p)$. Bringing these together (you can check the value of the cost function below for $y = 0$ and $y = 1$):

$$\text{Cost}(p, y) = -y \log(p) - (1 - y) \log(1 - p)$$

To minimize the cost function, we try to make the sum of the partial derivatives of Cost w.r.t. w_0 and w_1 equal to zero for all the examples. Using gradient descent to approximate the above computation, we would start with a random set of values for θ and repeat the assignment below for each parameter and each example until convergence (where $\theta < \eta < 1$ is known as a learning rate):

$$w_j \leftarrow w_j - \eta \frac{\partial \text{Cost}(\theta)}{\partial w_j}$$

For those interested in the computation of the derivative with m training examples:

$$\frac{\partial \text{Cost}(\theta)}{\partial w_j} = \frac{1}{m} \sum_{i=1}^{m} ((\sigma(x^{(i)}) - y^{(i)}) x_j^{(i)})$$

Finally, notice that since $\frac{p}{(1-p)} = e^{\theta^\top x}$, we have that $p = \frac{e^{\theta^\top x}}{(1+e^{\theta^\top x})}$.

Setup:
We will run a simple example of logistic regression over a binomial data distribution.

- Unzip `logisticRegression.zip` into any folder of your choice;
- Open Matlab and inside it, locate the folder where the contents of `logisticRegression.zip` have been unzipped;
- Double click on `logisticRegression.m` inside `MATLAB`;
- Left-click the editor and press F5 to run the code.

You should see a logistic curve and the fitted points as circles, around the edges of the curve.

In the above example, the distribution is 'binomial' and the link function is 'probit'; the link function f does the following to the linear models seen in class: $y = f(w, \phi(x))$. Using link function 'identity' allows you to simulate the linear models seen in class.

Exercises

1. Read the description of the Matlab functions:

   - `glmfit` (http://uk.mathworks.com/help/stats/glmfit.html)
   - `glmval` (http://uk.mathworks.com/help/stats/glmval.html),

   these are used on lines 13 and 17 of `logisticRegression.m`.

2. Change the distribution/link functions used on the `.m` code (it was 'probit' for logistic regression; you can change the parameters of functions `glmfit` and `glmval` accordingly to each of a, b and c below):

   - normal/identity;
   - Poisson/log;
   - gamma/reciprocal.

   Notice that the binomial data used originally has an additional parameter, n. Therefore, changing the distribution used (which was 'binomial') requires several steps:

   - Line 9 needs to be commented out;
   - On line 13, [y n] becomes y;
   - On line 17, 'size', n needs to be deleted from `glmval`;
   - On line 18, ./n needs to be deleted from plot;

   Compare and contrast the use of the above 'probit' link function with the 'logit' function (log of odds ratio: $\text{logit}(P) = \log_e\left(\frac{P}{(1-P)}\right)$.

3. Change the data points used in lines 7 to 10 and redo exercise 2, using e.g.:

   - Linear points (write down similarly spaced x and y values)
   - Polynomial points (write down values for x and use any polynomial function e.g.$y = x^2 + e$, where e is a small integer of your choice)
   - Logarithmic points (write down values for x and use any logarithmic function e.g., $y = \log_e(x) + e$, where e is a small integer of your choice)

4. Open and inspect file `logisticRegressionNEW.m` before running it. Would you expect the choice of distribution/link functions to work well? Now run it and check the result