Implementing Logistic Regression using Numpy

1 Write a function to generate an m+1 dimensional data set, of size n, consisting of m continuous independent variables (X) and one dependent binary variable (Y) defined as

$$Y = \begin{cases} 1 & \text{if } p(y=1|\vec{x}) = \frac{1}{1+\exp^{-\vec{x}.\vec{\beta}}} > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

where,

- β is a random vector of dimensionality $m+1$, representing the coefficients of the linear relationship between X and Y, and
- $\forall i \in [1,n], x_{i0} = 1$

To add noise to the labels (Y) generated, we assume a Bernoulli distribution with probability of success, θ, that determines whether or not the label generated, as above, is to be flipped. The larger the value of θ, the greater is the noise.

The function should take the following parameters:

- θ: The probability of flipping the label, Y
- n: The size of the data set
- m: The number of indepedent variables

Output from the function should be:

- X: An $n \times m$ numpy array of independent variable values (with a 1 in the first column)
- Y: The $n \times 1$ binary numpy array of output values
- β: The random coefficients used to generate Y from X

2 Write a function that learns the parameters of a logistic regression function given inputs

- X: An $n \times m$ numpy array of independent variable values
- Y: The $n \times 1$ binary numpy array of output values
- k: the number of iteractions (epochs)
- τ: the threshold on change in Cost function value from the previous to current iteration
- λ: the learning rate for Gradient Descent

The function should implement the Gradient Descent algorithm as discussed in class that initialises β with random values and then updates these values in each iteraction by moving in the the direction defined by the partial derivative of the cost function with respect to each of the coefficients. The function should use only one loop that ends after a number of iterations (k) or a threshold on the change in cost function value (τ).

The output should be a $m+1$ dimensional vector of coefficients and the final cost function value.

3 Create a report investigating how differen values of n and θ impact the ability for your logistic regression function to learn the coefficients, β, used to generate the output vector Y. Also include your derivation of the partial derivative of the cost function with respect to the parameters of the model.

4 Add L1 and L2 regularization to the Logistic Regression cost function. How does this impact the models learnt? How does the choice of regularization constant impact the β vector learned?

5 Merge the linear regression code base created in Exercise 1 and the logistic regression code base created in this Excercise and create an object oriented code base that maximises reuse of code across the algorithms.