# STAMATICS
# Mini Project 1

Dravya Marwaha(190322)
Twinkle Arora(190919)
Deeksha Vijay(190257)

June 5, 2021

## 1 Problem Statement

For $i = 1, \ldots, n$, let $x_i = (1, x_{i2}, \ldots, x_{ip})$ be the vector of covariates for the $i^{th}$ observation and $\beta \in \mathbb{R}_p$ be the corresponding vector of regression coefficients. Suppose response $y_i$ is a realization of $Y_i$ with

$$Y_i \sim Bern(\phi(x_i^T \beta)),$$

where $\phi(.)$ is the CDF of a standard normal distribution. We need to write an algorithm to find the Maximum Likelihood Estimator (MLE) for $\beta$.

## 2 Likelihood Function

According to the statement, given a data point $x_i$, the probability of $y_i$ being 1 (positive class) is $\phi(x_i^T \beta))$ (call it $p_i$). Clearly, the probability of $y_i$ being 0 (negative class) is $(1 - p_i)$. Thus, the likelihood for a single data point $x_i$ can be written as

$$l_i = p_i^{y_i}(1 - p_i)^{1-y_i}$$

Since the data points have been sampled from a Bernoulli distribution, the combined likelihood for them is just the product of the individual likelihoods. Thus,

$$\begin{aligned} L &= \prod_i l_i \\ &= \prod_i p_i^{y_i}(1 - p_i)^{1-y_i} \end{aligned}$$

This is our likelihood function. We need to find parameter $\beta$ which would maximize the value of this function. But the value of the above expression can reach very small values, making it hard to work with this function directly. Instead we can take the log of this function, and

maximize that instead. Since log is a monotonically increasing function, the value of $\beta$ that maximizes L will also maximize log(L).

$$\mathcal{L} = log(L) = \sum_i y_i log(p_i) + (1 - y_i) log(1 - p_i)$$

# 3 Maximizing the Log Likelihood

$\mathcal{L}$ is a function in the parameter $\beta$. We need to find a value of $\beta$ that would maximize this function for the given data points. From elementary calculus, we know that a function achieves maxima (local or global) at the point where its derivative (gradient) is zero and double derivative is negative. Fortunately, the given function happens to be a concave function[1] , so we don't need to worry about local optima or the sign of the double derivative. We only need to find a root for the derivative of $\mathcal{L}$, which would be the required value of $\beta$. We chose the **Newton Raphson method**[2] for finding the root of a non-linear equation to find the required value.

To use the Newton Raphson method, we would need to find the gradient ($\nabla\mathcal{L}$) and the hessian ($\nabla^2\mathcal{L}$). To calculate those, we will use some properties of the $\phi(.)$ (CDF) and $\varphi(.)$ (PDF) of the standard normal distribution and define some new variables.

$$\begin{aligned}
\phi(x) &= 1 - \phi(-x) \quad \forall x \in \mathbb{R} \\
\varphi(x) &= \varphi(-x) \qquad \forall x \in \mathbb{R} \\
q_i &= 2y_i - 1 \\
\lambda_i &= \frac{q_i \varphi(q_i x_i^T \beta)}{\phi(q_i x_i^T \beta)}
\end{aligned}$$

We now calculate the partial derivative of $\mathcal{L}$ wrt $\beta_j$, where $\beta_j$ is the $j^{th}$ element of the $\beta$ vector.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \beta_j} &= \frac{\partial}{\partial \beta_j} \left( \sum_i y_i log(\phi(x_i^T \beta)) + (1 - y_i) log(1 - \phi(x_i^T \beta)) \right) \\
&= \frac{\partial}{\partial \beta_j} \left( \sum_i log(\phi(q_i x_i^T \beta)) \right) \\
&= \sum_i \frac{\varphi(q_i x_i^T \beta) q_i x_{ij}}{\phi(q_i x_i^T \beta)} \\
\frac{\partial \mathcal{L}}{\partial \beta_j} &= \sum_i \lambda_i x_{ij} \qquad (1)
\end{aligned}$$

Thus,

$$\nabla\mathcal{L} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \beta_1} \\ \frac{\partial \mathcal{L}}{\partial \beta_2} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial \beta_p} \end{bmatrix} = \begin{bmatrix} \lambda_1 x_{11} + \lambda_2 x_{21} + \lambda_3 x_{31} + \dots \\ \lambda_1 x_{12} + \lambda_2 x_{22} + \lambda_3 x_{32} + \dots \\ \vdots \\ \lambda_1 x_{1p} + \lambda_2 x_{2p} + \lambda_3 x_{3p} + \dots \end{bmatrix} = X^T \Lambda$$

where each row in X represents a data point, and $\Lambda$ is the column vector where the $i^{th}$ element is $\lambda_i$.
We can now calculate the hessian as well. To do so, we first calculate the partial derivative of (1) wrt $\beta_k$.

$$\begin{aligned}\frac{\partial^2 \mathcal{L}}{\partial \beta_j \partial \beta_k} &= \sum_i \left( \frac{\phi(q_i x_i^T \beta)\varphi(q_i x_i^T \beta)(-q_i x_i^T \beta) - \varphi^2(q_i x_i^T \beta)}{\phi^2(q_i x_i^T \beta)} \right) (q_i x_{ik}) q_i x_{ij} \\ &= -\sum_i (\lambda_i x_i^T \beta + \lambda_i^2) x_{ij} x_{ik}\end{aligned}$$

We define a $(m \times m)$ matrix $M$, where $m$ is the total number of data points such that $M$ is a diagonal matrix and $M_{ii} = \lambda_i x_i^T \beta + \lambda_i^2$ for all $i \in \{1, 2, \ldots, m\}$. Then, the hessian matrix can be written as

$$\nabla^2 \mathcal{L} = -X^T M X$$

# 4 Implementation in Python

- The algorithm was implemented as a Python class with all the necessary methods.
- In each iteration $i$ of the algorithm, we perform the following,

  $$\beta^{(i)} = \beta^{(i-1)} - (\nabla^2 \mathcal{L})^{-1} (\nabla \mathcal{L})$$

  with $\beta^{(0)} = \mathbf{0}$.
  For calculation of the inverse of the hessian, the **linalg** module of **numpy** library was used, which has optimized algorithms for calculating the inverse.
- **stats** module of **scipy** library was used to implement the $\phi(.)$ and $\varphi(.)$ functions.
- During the calculation of the hessian, instead of calculating the matrix $M$, the broadcasting feature of numpy library was used which makes the calculation a bit faster.
- $\beta$ : $[0.93223979, -1.53195285, -0.01229736, -0.23082798, -0.13581631, 0.00963642]$
- The probabilities obtained are:
  Probability of survival of Jack: 0.21964665512742015
  Probability of survival of Rose: 0.9999999302065994

# 5 References


[1] Amemiya, T. (1985). Qualitative Response Models, Advanced Econometrics (1st ed., pp. 273-274). Cambridge, Massachusetts: Harvard University Press.

[2] Newton-Raphson method for finding the root of a non-linear equation