Logistic Regression

The question is how should we model the relationship between

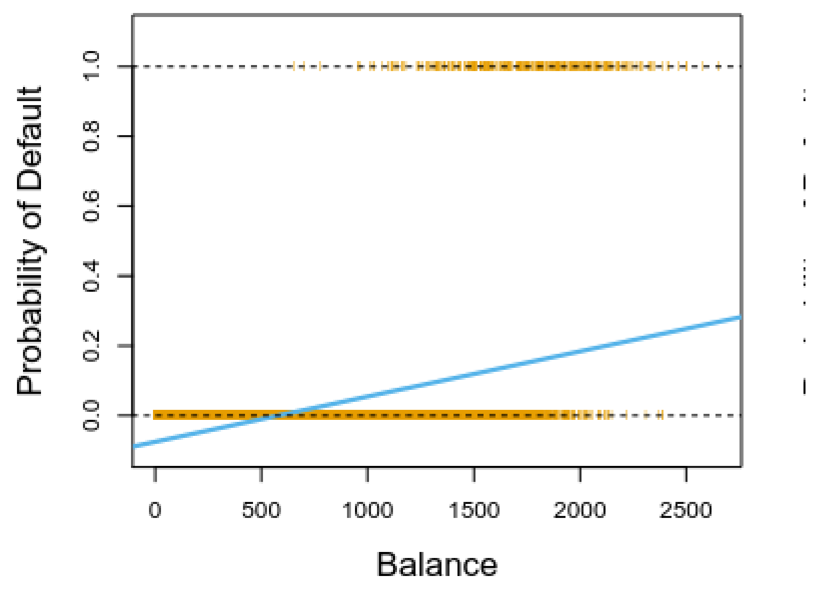
$$P(X) = \Pr(Y = 1|X)$$

- Relationship between the **Probability** of X and the **Classifying** Prediction for X
- Using 1 and 0 for the Response

Using Linear Regression model to represent these probabilities

$$p(X) = \beta_0 + \beta_1 X$$

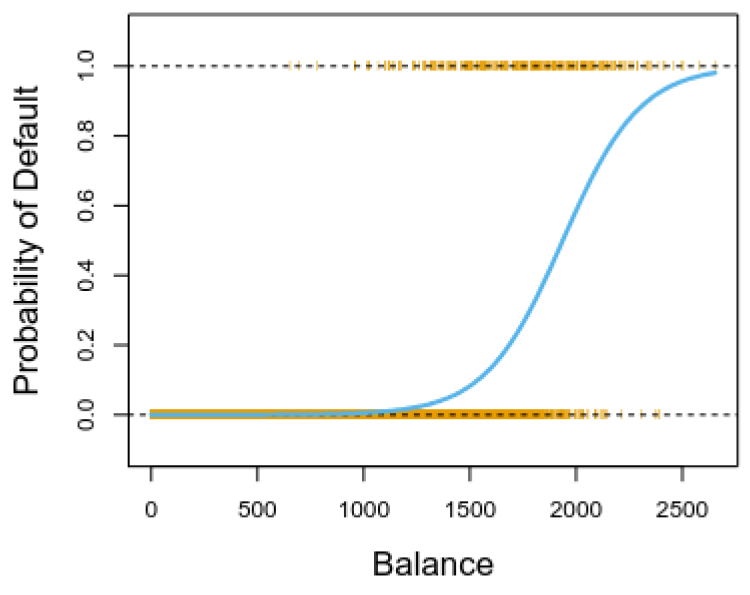
- If we fit the line to predict the **Probability**



- Notice that the `Balance` Lower than 500 our prediction for the probability is **negative**

To avoid this problem we model $p(X)$ to only fall between 1 and 0 for all values X **Logistic Function** which is a Sigmoid Function

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} = \frac{1}{e^{-(\beta_0 + \beta_1 X)} + 1}$$



- Any output of the **Logistic Function** Falls between 1 or 0

$$\frac{p(X)}{1 - p(X)} = \frac{\frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}}{1 - \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}} = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} \times \frac{1}{\frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}} = \frac{1}{1+e^{\beta_0+\beta_1 X}}} = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} \times 1 + e^{\beta_0+\beta_1 X}$$

$$\text{odds} = \frac{p(X)}{1 - p(X)} = e^{\beta_0 + \beta_1 X}$$

- the **quantity** $p(X)/[1 - p(X)]$ is called the $odds$ can can only take values between 0 to ∞
- Values close to 0 indicates low probability
- Values close to ∞ indicate higher probability

- if $p(X) = 0.5$ the $\text{odds} = 1$ equal chance
- if $p(X) = 0.8$ the $\text{odds} = 4$ 4x success chances

Probabilities vs Odds

- The odds are the **ratio** of something happening *divided by* something not happening
- The probability is the **ratio** of something happening *divided by* to everything could happen

$$\text{Probability} = \frac{\text{Number of favorable outcomes}}{\text{Total number of possible outcomes}}$$

$$\text{Odds} = \frac{\text{Probability of even occurs}}{\text{Probability event does not occur}} = \frac{p}{1-p}$$

- In the Logistic Regression setting the **odds** are just an alternative representation for the classification problem
- **Odds** are preferred cause they allow us to transform the the odds which are a correct and alternative representations to the probabilities of each class into a **Linear Combination of Feature**

Odds in logistic regression

Taking logarithm of both sides : log odds or logit

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

- the $\log$ of odds gives us a **Linear Combination of Features** which is easy to model and **interpret**

Why we use the odds?

- **Probabilities** lives in the interval $[0, 1]$
- Linear combinations like $\beta_0 + \beta_1 X$ lives on $(-\infty, +\infty)$
- The **odds** solves that by being in $(0, +\infty)$

Coefficients interpretability

In the **Linear Regression** β_1 gives the average change in Y associated with one unit increase in X
In the **Logistic Regression** β_1 does not correspond directly to the change in $p(X)$, if β_1 is positive increasing one unit in X will increase the **Probability** $p(X)$ but the increase depends on the current value of $p(X)$

$$\beta_1 > 0 \rightarrow \text{One unit increase X} \rightarrow \text{Increase } p(X)$$

$$\beta_1 < 0 \rightarrow \text{One unit increase X} \rightarrow \text{Decrease } p(X)$$

- The amount "Degree" of increase in the **Probability** $p(X)$ depends on the current value of X

Why the effect of the probability depends on current $p(X)$

- if $p(X) = 0.5$ the Sigmoid Function curve is steep, small changes in $X \rightarrow$ big changes in $p(X)$
- if $p(X) \approx 0.01 \text{ or } 0.09$ the **Sigmoid Function** curve is flat, changes in $X \rightarrow$ small changes in $p(X)$

Statistical Inference

The **Log odds** results in a linear equation which will allow us to preform the **Linear Regression** inference and tests :

- **Hypothesis Testing**
  - F-test
  - T-test
- Construct **Confidence Intervals**
- **MLE** for optimization

Multiple Logistic Regression

Generalizing the **Simple Logistic Regression** equation of the **log-odds**

$$\log\left(\frac{p(x)}{1-p(x)} = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p\right)$$

While $p(X)$ can be written as :

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}$$

Matrix form :

$$p(X) = \frac{e^{X\beta}}{1+e^{X\beta}}$$

- The **Maximization** of the coefficients is done via **MLE** Maximum Likelihood Estimator Derivation Logistic Regression
- The Response is binary 1 or 0
- With multiple predictors **Variables** to make the prediction
- it uses **Linear decision boundary** in the log-odds space

Same case as in multiple linear regression it give different results than the simple linear regression for the same predictor cause often the predictors will be **correlated**

Multinomial Logistic Regression

Also know as **softmax regression**

It's an extension to the **Binary Logistic Regression** where the response Y has more than two classes $K > 2$,

Softmax Coding

the **softmax regression** outputs a vector which we can interpret each elements of that vector as a the probability of the input of a class k

$$f(x;B) = \begin{pmatrix} P(y=1|X=x) \\ P(y=2|X=x) \\ \vdots \\ P(y=K|X=x) \end{pmatrix}$$

- The elements of this vectors are the probabilities of the data x being in that class

1. For input x , the score for class k is given by $x\beta_k$
2. Taking the exponential $e^{x\beta_k}$, so its always positive
3. Normalize it , $P(y=k|X=x) = \frac{e^{x\beta_k}}{\sum_{j=1} e^{x\beta_j}}$

$$f(x;\beta) = \frac{1}{\sum_{j=1}^{K} e^{\beta j^T x}} \begin{bmatrix} e^{\beta_1^T x} \\ e^{\beta_2^T x} \\ \vdots \\ e^{\beta_k^T x} \end{bmatrix} = \text{softmax}(Bx)$$

- The softmax regression doesn't require a baseline
  Note :

$$B_k = \begin{bmatrix} \dots \beta_1^T \dots \\ \dots \beta_2^T \dots \\ \vdots \\ \dots \beta_k^T \dots \end{bmatrix}$$

- B_k is a parameter matrix
- β_k is a vector of coefficients for each class k

Multinomial Coding

Unlike the **softmax regression** here we choose one of the classes K to be the baseline for the model so we estimate $K-1$ coefficient vectors $\beta_1 \dots \beta_{K-1}$

Baseline

- The decision of choosing a baseline wont effect the predictions so we pick the class K as a baseline
- Subtract the estimated coefficients of the baseline from all the other classes k estimated coefficients
- Including the baseline class K which will result in $X\beta = 0$ and when exponent it $e^0 = 1$

Which results in the baseline class being :

$$P(Y = K|x) = \frac{1}{1 + \sum_j^{K-1} e^{X\beta_j}}$$

and other classes :

$$P(Y = k|x) = \frac{e^{X\beta_k}}{1 + \sum_j^{K-1} e^{X\beta_j}}$$

Example

- Studying the **CRP** and it effect on infection types **(No infection ,Viral infection, Bacterial infection)**
  Here we have 3 **logistic regressions** : with $CRP = 25$
- $0 =$ (Viral ,Bacterial infections) and $1 =$ No infection, resulted in 0.009
- $0 =$ (No,Bacterial infections) and $1 =$ Viral infection, resulted in 0.360
- $0 =$(No,Viral infection) and $1 =$ Bacterial infection, resulted 0.001

These results don't sum up to one which means they we cannot be represented as **probabilities** to each class

Multinomial vs Softmax regression

They are internally the same concept and achieve the same predictions, the main problem with the **softmax regression** coding approach is that :

$$P(Y = K|X) = \frac{e^{X\beta_k}}{\sum_j^K e^{X\beta_j}}$$

- This representation **Overparameterize** and will results in infinite amount of solutions by adding a constant to the parameters β

$$P(Y = K|X) = \frac{e^{X(\beta_k + c)}}{\sum_j^K e^{X(\beta_j + c)}} = \frac{e^{X\beta_k} e^{Xc}}{\sum_j^K e^{(X\beta_j)} e^{Xc}}$$

- and the terms e^{Xc} will cancel out

As noted it allowed us to add any constant c and result in the same prediction, Here where picking a **Baseline** puts a constraint on this by :

- By setting the **baseline** class coefficient to zero
- It wont allow us to add constants cause it will violet this rule that $\beta_K = 0$

Why it matter ?

- **Numerical Stability** it will make the optimization process faster and easier for **gradient descent** since there wont be infinite solution to model equation (**Overparameterzation**)
- **Interpretability** Coefficients will make sense relative to the baseline which is the reference

Note : The softmax regression approach still works and avoid all of these problems when we set on the background one of the classes as a **Baseline**

Estimating The Regression Coefficients

In **Linear Regression** we use the least squares to estimate the coefficients, in the logistic regression the **Maximum Likelihood**.
Intuition :

- We seek estimates for $\hat{\beta}_0, \hat{\beta}_1$ such that we maximize the probability $\hat{p}(x_i)$
- More explanation in Maximum Likelihood Estimation

$$\mathcal{L}(\beta_0, \beta_1) = \prod_{i:y_{i=1}} p(x_i) \ \prod (1 - p(x_{i'}))$$

- This called the **Likelihood function**
- Our estimates $\hat{\beta}_0, \hat{\beta}_1$ are chosen to **maximize** this likelihood function and that **best to separates classes based on labels**
- **The least squares** is a special case of $\text{maximum Likelihood}$ Residual Sum of Squares
- Unlike Linear Regression the **Likelihood function** in logistic regression is nonlinear in the parameter β so there is no closed form solution for β Maximum Likelihood Estimator Derivation Logistic Regression

Loss Function

The loss function in the **Logistic Regression** is also called the **log loss** or **cross-entropy loss** , which is derived from **Log Likelihood Function** of the **Bernoulli distribution** Which models the **Binary Logistic Regression**
Likelihood Function :

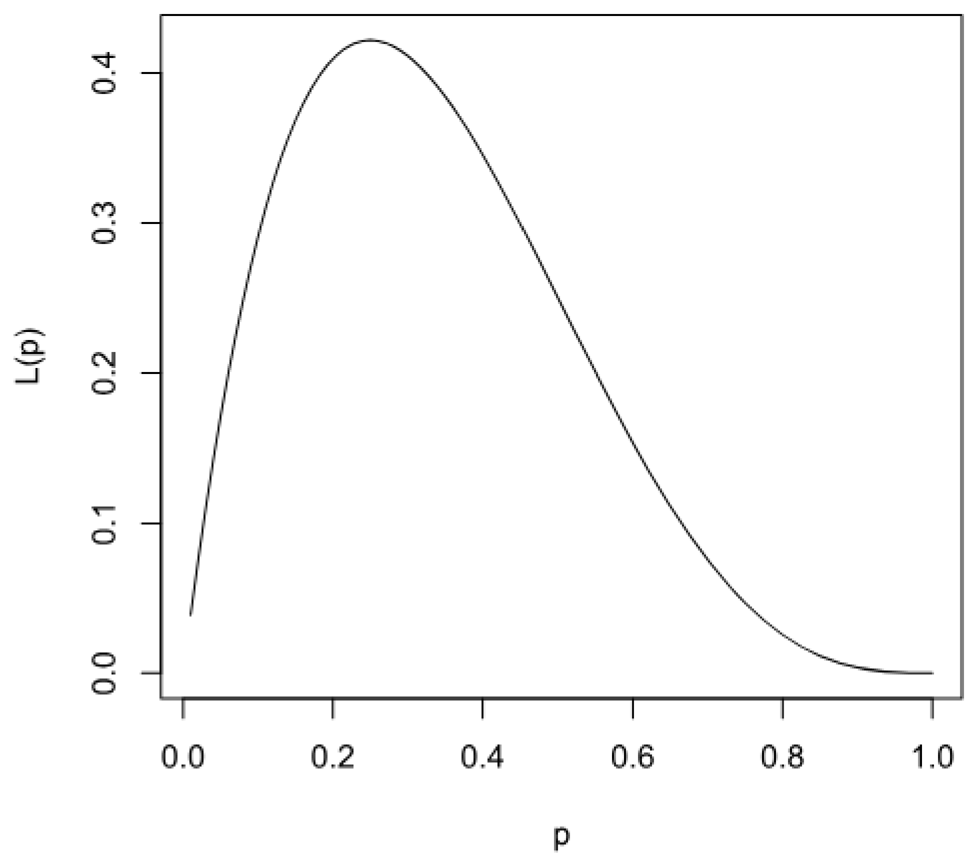
$$\mathcal{L}(\beta) = \prod_{i=1}^{n} P_i^{y_i}(1 - P_i)^{1-y_i}$$

- Where $P_i = \sigma(X\beta) = \frac{1}{1+e^{-X_i\beta}}$ which the **Logistic/Sigmoid function**

Log Likelihood Function

$$l(\beta) = \log \mathcal{L}(\beta) = y^T \log(P) + (1-y)^T \log(1-P)$$

the **Likelihood Function** is used to maximize the likelihood of the probability given the parameter β

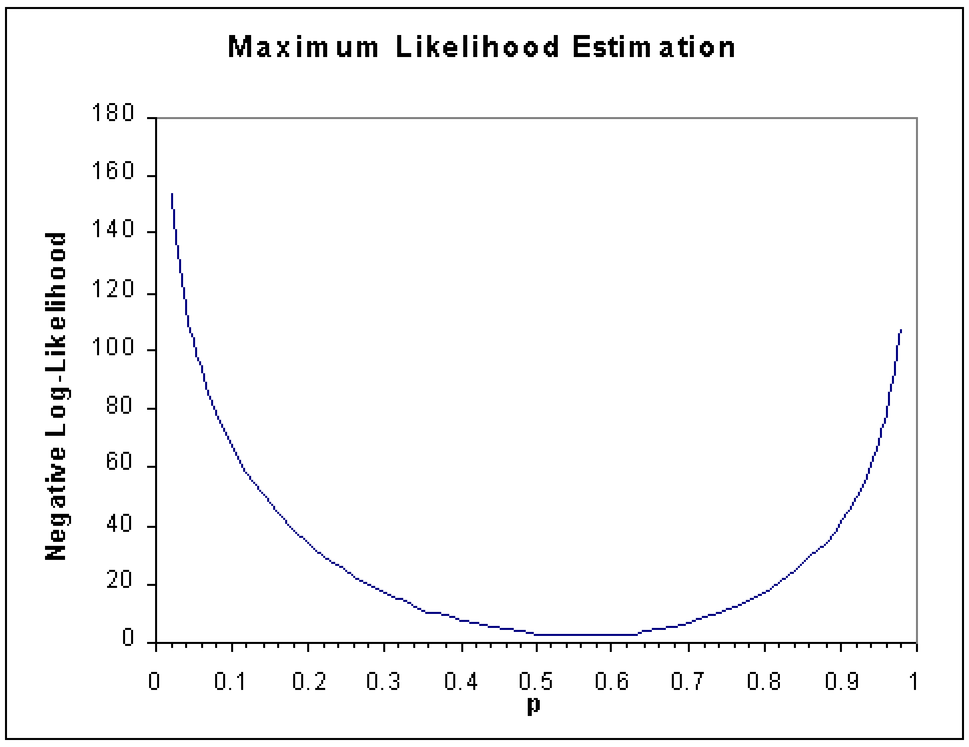


- This the graph of the **Likelihood Function** for the Bernoulli distribution data , and **MLE** select the best parameter estimate that explains the observed data (making it the most probable)

By taking the negative of the **Likelihood Function** which by nature maximize it will turn into a minimize which is useful to apply the loss function algorithms

$$l(\beta) = -[y^T \log(P) + (1-y)^T \log(1-P)]$$

- Its called **cross-entropy loss** $\rightarrow$ The cost function for the binary logistic regression



Why does minimizing the negative log probability do what we want?

Cause the probability of the correct answer is maximized and the probability of the incorrect answer is minimized, since the sum of the two is one so:

- Increase in the probability of the correct answer will come at the expense of the incorrect answer
  That's why its called cross-entropy loss

And to find the minimum of the negative log likelihood function (cross-entropy loss) we use :

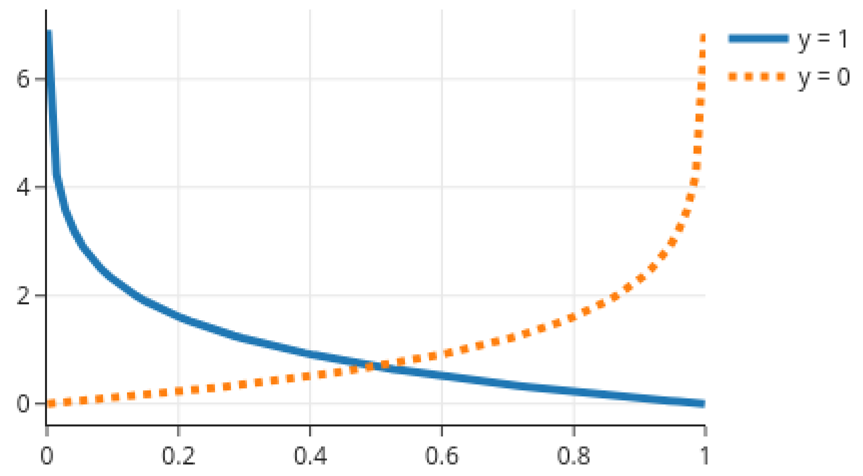
Gradient Descent

This follows the Gradient Descent algorithm, Where it finds the best fit parameters θ that minimize the error in the loss function which in this case is the **cross-entropy loss**

$$\hat{\theta} = \text{argmin}_{\theta} \frac{1}{n} \sum_{i=1}^{n} l(f(x^{(i)};\theta), y^{(i)})$$

- Where θ is the parameter β we wanna estimate
- $l(f(x^{(i)};\theta), y^{(i)})$ is the loss function where $f(x;\theta)$ is the estimated value

The **cross-entropy loss** is a **convex** function which means that at most it has one minimum, There is no local minima to get stuck on



The first important step in the Gradient Descent is to derive the gradient ∇ for the **cross-entropy loss**

$$\nabla_{\beta} J(\beta) = \nabla_{\beta} \frac{1}{n} (-[y^T \log(P) + (1-y)^T \log(1-P)])$$

With $P = \frac{1}{1+e^{-X\beta}} = \sigma(X\beta)$

- Calculating the $y_i = 1$ gradient

$$\nabla_{\beta} \log \sigma(X\beta) = \log(1) - \log(1 + e^{-X\beta})$$

$$\nabla_\beta \log \sigma(X\beta) = \nabla_\beta(-\log(1+e^{-X\beta}))$$

Calculating the gradient of $\nabla_\beta(1+e^{-X\beta})$

$$\frac{\partial}{\partial\beta}(-\log(1+e^{-X\beta})) = \frac{1}{1+e^{-X\beta}}\nabla_\beta(1+e^{-X\beta})$$

Plug it back will result in :

$$\nabla_\beta(1+e^{-X\beta}) = e^{-X\beta}\nabla_\beta(-X\beta) = -e^{-X\beta}X = -X^Te^{-X\beta}$$

Which gives :

$$\frac{\partial}{\partial\beta}(-\log(1+e^{-X\beta})) = -\frac{1}{1+e^{-X\beta}}e^{-X\beta}X = X^T\frac{1}{e^{X\beta}+1} = X^T[1-\sigma(X\beta)]$$

With : $\frac{e^{-X\beta}}{1+e^{-X\beta}} = \frac{1}{e^{X\beta}+1} = 1-\sigma(X\beta)$

So :

$$\nabla_\beta \log \sigma(X\beta) = X^T[1-\sigma(X\beta)]$$

Now lets Calculating the $y_i = 0$ gradient, its know that :

$$\nabla_\beta \log(1-\sigma(X\beta)) = \nabla_\beta \log \sigma(-X\beta)$$

$$\nabla_\beta \log \sigma(-X\beta) = \nabla_\beta \log\left(\frac{1}{1+e^{X\beta}}\right) = \nabla_\beta(\log(1) - \log(1+e^{X\beta}))$$

$$\nabla_\beta(-\log(1+e^{X\beta})) = \frac{\partial}{\partial\beta}(-\log(1+e^{X\beta})) = -\frac{1}{1+e^{X\beta}}\nabla_\beta(1+e^{X\beta})$$

Calculating the gradient of $\nabla_\beta(1+e^{X\beta})$

$$\nabla_\beta(1+e^{X\beta}) = \frac{\partial}{\partial\beta}(1+e^{X\beta}) = e^{X\beta}X = X^Te^{X\beta}$$

Plugging it back :

$$\frac{\partial}{\partial\beta}(-\log(1+e^{X\beta})) = -\frac{1}{1+e^{X\beta}}e^{X\beta}X$$

So :

$$\nabla_\beta \log(1-\sigma(X\beta)) = \nabla_\beta \log \sigma(-X\beta) = -X^T\frac{e^{X\beta}}{1+e^{X\beta}} = -X^T\frac{1}{e^{-X\beta}+1} = -X^T\sigma(X\beta)$$

Now we plug both $\nabla_\beta \log(1-\sigma(X\beta))$ and $\nabla_\beta \log \sigma(X\beta)$ in the gradient of the **cross-entropy loss** function :

$$\nabla_\beta = -\frac{1}{n}\sum_{i=1}^{n}[y^T[1-\sigma(x_i^t\beta)]x_i - (1-y)^T\sigma(x_i^t\beta)x_i] = -\frac{1}{n}X^T(y-\sigma(X\beta))$$

So the Gradient of the **cross-entropy loss** function is :

$$\nabla J(\beta) = -\frac{1}{n}X^T(y-\sigma(X\beta))$$

- Now we just apply the common steps of the gradient descent algorithm by plugging it in :

$$\beta^+ = \beta^- - \alpha\nabla_\beta J(\beta)$$

- To calculate the updated parameter β and check convergence
  Details in Gradient Descent

Newton-Raphson Method

Once the **Coefficients** are estimated (Using iterative numerical optimization methods),we can calculate the predicted probability for the observation X_i