Introduction to Logistic Regression

A Beginner Friendly Tutorial

Koushik Khan

koushik.khan.ds@gmail.com linkedin.com/in/koushikkhan/ koushikkhan.github.io

March 22, 2025

Contents

Motivation

The Binary Classification Problem

- Imagine situations where we would like to know,
  - the *eligibility* of getting a bank loan given the value of credit score (x_{credit_score}) and monthly income (x_{income}).
  - identifying a tumor as *benign* or *malignant* given its size (x_{tumor_size}).
  - classifying an email as *promotional* given the no. of occurrences for some keywords like {'win', 'gift', 'discount'} ($x_{n\_win}, x_{n\_gift}, x_{n\_discount}$).
  - finding a monetary transaction as *fraudulent* given the time of occurrence (x_{time_stamp}) and amount (x_{amount}).
- These problems occur frequently in real life & can be dealt with machine learning.
- All such problems come under the umbrella of what is known as *Classification*.
- In each scenario, only one of the two possible outcomes can occur, hence these are specifically known as *Binary Classification* problems.

Understanding The Dataset

- Any dataset containing *numerical* or *categorical* features can be used for classification.
- However, the target variable must be *categorical* in nature.
- Specifically, for binary classification, a target variable (Y) must take *any one of the two* distinct values like {'benign', 'malignant'}.
- To use with ML algorithms, values of the target variable are *encoded* into numeric representations e.g. `{"benign": 0, "malignant": 1}`, a.k.a 'class-0' & 'class-1' respectively.
- In literature, typically 'class-0' is tagged as *failure* and 'class-1' is tagged as *success*.

Machine Learning for Classification

How Does A Machine Perform Classification?

- During the inference, the goal is to have the ML model *predict the class label* for a given set of feature values.
- Specifically, a binary classification model estimates two probabilities $\hat{p}_0$ & $\hat{p}_1$ for 'class-0' and 'class-1' respectively where $\hat{p}_0 + \hat{p}_1 = 1$.
- The predicted label depends on $max\{\hat{p}_0, \hat{p}_1\}$ i.e. it's the one which is most probable based on the given features.
- In logistic regression, $\hat{p}_1$ (i.e. success probability) is compared with a *predefined threshold* p^* to predict the class label like below:

$$\begin{aligned} \text{predicted class} &= 1;\ \hat{p}_1 \geq p^* \\ &= 0;\ \text{otherwise} \end{aligned} \tag{1}$$

- To keep the notation simple and consistent, we will denote the *success probability as* p, and *failure probability as* $(1 - p)$ instead of p_1 and p_0 respectively.

Why NOT Linear Regression?

- Can't we really use linear regression to address classification? The answer is *NO!*
- Let's try to understand why:
  - To estimate p using linear regression, we would need:

$$\hat{p} = \hat{\alpha} + \hat{\beta} x_{tumor\_size} \tag{2}$$

  - Eqn. (2) *doesn't seem* to be feasible as the R.H.S, in principle, belongs to $\mathbb{R}\,(-\infty, +\infty)$ & the L.H.S belongs to $(0, 1)$.
- Can we convert $(\hat{\alpha} + \hat{\beta} x_{tumor\_size})$ to something belonging to $(0, 1)$? That may work as an estimate of a probability! The answer is *YES!*
- We need a *converter* (a function), say, $g(.)$ that will connect $p \in (0, 1)$ to $(\hat{\alpha} + \hat{\beta} x_{tumor\_size}) \in \mathbb{R}$.
- Fortunately, such functions do exist and they are often referred to as *link functions* in this context.

Logistic Regression

The Bernoulli Distribution

- Following the definition, a *link function* connects the *liner predictor* like $(\alpha + \beta x_{tumor\_size})$ to the *expected value* of the target variable.
- The *binary* target variable naturally suits the *Bernoulli Distribution* (Bernoulli's Trial) for explanation.

Bernoulli's Trial

A random experiment that results in one of the two possible outcomes, often called, a *success* and a *failure*, with a *constant probability* of success, say, p.

Examples

- tossing a fair coin - the coin shows either the 'HEAD' or the 'TAIL'
- performing COVID test - result will be either '+ve' or '-ve'
- detecting a tumor as 'benign' or 'malignant' etc.

PMF And Expectation

- Bernoulli's trial can be expressed mathematically with a random variable, say, Y as:

$$f(y|p) = P(Y = y) = p^y \times (1-p)^{1-y};\ y \in \{0,1\} \tag{3}$$

where eqn. (3) is known as *Probability Mass Function (PMF)*.

- The PMF maps the values (y) taken by Y to probabilities e.g.

$$\text{when } y = 1 \Rightarrow P(Y = 1) = p$$
$$\text{when } y = 0 \Rightarrow P(Y = 0) = (1-p)$$

- The expected value of Y is calculated as:

$$\begin{aligned} E(Y) &= 0 \times P(Y=0) + 1 \times P(Y=1) \\ &= P(Y=1) \\ &= p \\ &= \text{Probability of `success' or identifying a malignant case} \end{aligned} \tag{4}$$

The Logit Link Function

- The logit link function is defined as:

$$logit\,(E(Y)) = \log\left(\frac{E(Y)}{1 - E(Y)}\right) \tag{5}$$

- And by the definition of a link function, it connects the linear predictor i.e. $(\alpha + \beta x_{tumor\_size})$ to $E(Y)$ i.e. p as:

$$\log\left(\frac{p}{1-p}\right) = \alpha + \beta x_{tumor\_size} \tag{6}$$

  The eqn. (6) is formally called the *Logistic Regression* equation.

- For a linear regression, the link function is the *identity function* i.e. $g(x) = x$.

The Inverse of Logit - Sigmoid Function

- In logistic regression, we try to model $P(Y = 1)$ as:

$$\log\left(\frac{p}{1-p}\right) = \alpha + \beta x_{tumor\_size}$$

  where the L.H.S is also known as *log-odds*.

- Alternatively we can write:

$$p = \frac{e^{\alpha+\beta x_{tumor\_size}}}{1 + e^{\alpha+\beta x_{tumor\_size}}} \text{ (verify)} \tag{7}$$

- The R.H.S of eqn.(7) is referred to as *sigmoid* function, denoted by $\sigma(.)$ which is the *inverse* of the logit function.

Nature Of The Sigmoid

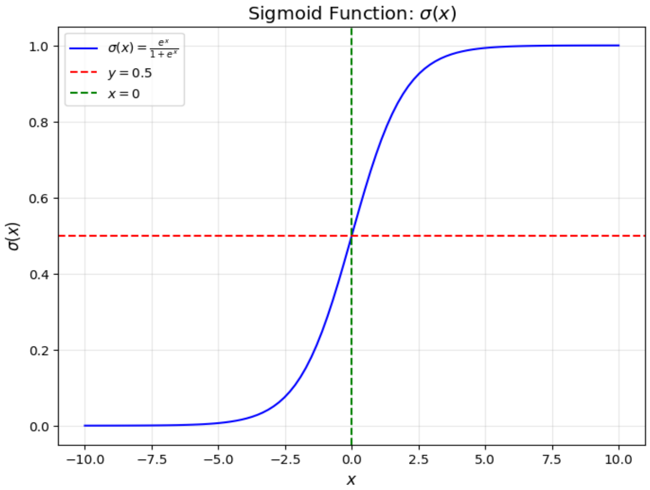

Figure: The sigmoid function

Math Behind The Sigmoid

- The sigmoid function:

$$\sigma(z) = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}};\ z \in \mathbb{R} \tag{8}$$

  maps any $z \in \mathbb{R}$ to a number belonging to $(0, 1)$.

- It has a very nice & important property too:

$$\frac{d\sigma}{dz} = \sigma'(z) = \sigma(z) \times (1 - \sigma(z)) \ \text{(verify)} \tag{9}$$

- It looks like an *elongated* 'S', that is where it gets its name from.

Linear And Non-linear Classification

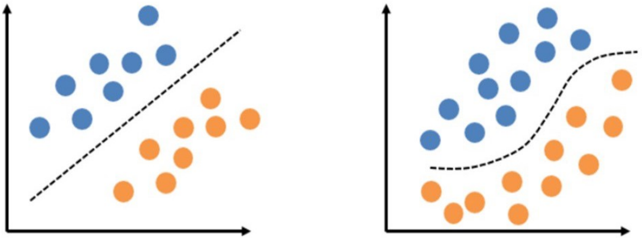

Figure: On the left, the classes are linearly separable as the boundary is a straight line, however they are not on the right

Logistic Regression Is A Linear Classifier

- The logistic regression equation (6) is actually a *straight line* (of the form $y = mx + c$).
- **Recall** the prediction rule:

$$\text{predicted class} = 1;\ \hat{p} \geq p^* \Rightarrow \hat{\alpha} + \hat{\beta} x_{tumor\_size} \geq \log\left(\frac{p^*}{1 - p^*}\right)$$
$$= 0;\ \text{otherwise}$$

- A simple logistic regression (the one we discussed) predicts the class label by identifying the regions on either side of a *straight line* (or hyperplane in general), hence it's a *linear classifier*.
- Logistic regression works well for *linearly separable* classes.

Estimation of Parameters

The Likelihood Function

- While defining PMF, it is assumed that the success probability (p) is *known*.
- However, in reality we *don't know* the p - consequently, PMF is *not suitable* for further use.
- The dataset involves N patients (say): $\{(x_{i,tumor\_size}, y_i)\}_{i=1}^{N},\ y_i \in \{0, 1\}$.
- Imagine the i^{th} patient, Y_i, has a probability p_i of developing a malignant case. Here $Y_i \sim Bernoulli(p_i)\ \forall i = 1, 2, \ldots, N$.
- Interestingly, the expression for $P(Y_i = y_i)$ is same as the PMF, eqn. (3),

$$\ell_i(p_i|y_i) = P(Y_i = y_i) = p_i^{y_i} \times (1 - p_i)^{1-y_i};\ y_i \in \{0, 1\}\ \forall i = 1, 2, \ldots, N \tag{10}$$

  The eqn. (10) is known as the likelihood for Y_i taking a value y_i.
- In Likelihood we know the dataset (*as we're observing it*), but the p_i is *unknown* to us.

The Joint Likelihood Function

- What's the likelihood for *observing the entire dataset*? Well, the *joint likelihood* gives that answer.
- It's computed as below:

$$L = P(Y_1 = y_1 \cap \ldots \cap Y_N = y_N) = \Pi_{i=1}^{N} p_i^{y_i} \times (1 - p_i)^{1-y_i} \tag{11}$$

  Eqn. (11) is called the *joint likelihood (L)*.
- It is much easier to work with joint likelihood after a *log-transformation*, also called the *log-likelihood (LL)*,

$$LL = \log(L) = \sum_{i=1}^{N} \left\{ y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right\} \tag{12}$$

- Joint likelihood measures the probability of observing the underlying dataset i.e. having $\{Y_1 = y_1, \ldots, Y_N = y_N\}$ for some unknown set of probabilities $\{p_1, \ldots, p_N\}$.

Maximum Likelihood Estimation

- Imagine that some process might have produced the observed dataset $\{(x_{i,tumor\_size}, y_i)\}_{i=1}^{N}$
- We are NOT sure what values of p_i's the process would have considered to produce the dataset.
- We can imagine *several potential candidates* for each p_i (say, all belong to a set $\mathbb{P}$) that might have been used to produce the dataset.
- In principle, the best candidate for each p_i would be the one that eventually makes the joint likelihood (L) or log likelihood (LL) (both are functions of p_i's) *maximum*.
- Mathematically, we would perform,

$$\underset{p_1,\ldots,p_N \in \mathbb{P}}{\arg\max} \sum_{i=1}^{N} \left\{ y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right\} \tag{13}$$

to find $\hat{p}_1, \ldots, \hat{p}_N$.

Going Deeper

- Finding $\hat{p}_1, \ldots, \hat{p}_N$ using eqn. (13) is basically an *optimization problem*.
- **Recall** $p_i = \frac{e^{\alpha+\beta x_{i,tumor\_size}}}{1+e^{\alpha+\beta x_{i,tumor\_size}}} = \sigma(\alpha + \beta x_{i,tumor\_size})$
- The eqn. (13) can be simplified as:

$$\underset{p_1,\ldots,p_N \in \mathbb{P}}{\arg\max} \left[\sum_{i=1}^{N} \left\{ y_i \log\left(\sigma(\alpha + \beta x_{i,tumor\_size})\right) + \right. \right. \\ \left. \left. (1 - y_i) \log\left(1 - \sigma(\alpha + \beta x_{i,tumor\_size})\right) \right\} \right] \tag{14}$$

- Here each p_i is a *function* of the parameters α & β and the known data $x_{i,tumor\_size}$.
- Finding $\hat{p}_1, \ldots, \hat{p}_N$ is *equivalent* to finding α and β with the help of eqn. (14).

Using Gradient Descent

- Though this is a *maximization* problem, the ML community prefers *minimizing* the *negative log-likelihood (NLL)* using gradient descent.
- In practice, a *scaled version* of NLL ($\frac{1}{N} NLL$) is used which is known as *Binary Cross Entropy (BCE)* loss function.
- Now the problem boils down to:

$$\underset{p_1,\ldots,p_N \in \mathbb{P}}{\arg\min} \frac{1}{N} NLL = \underset{p_1,\ldots,p_N \in \mathbb{P}}{\arg\min} -\frac{1}{N} \sum_{i=1}^{N} \left\{ y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right\} \tag{15}$$

or equivalently,

$$\begin{aligned} \underset{\alpha,\beta}{\arg\min} -\frac{1}{N} \sum_{i=1}^{N} \Big\{ & y_i \log\left(\sigma(\alpha + \beta x_{i,tumor\_size})\right) \\ & + (1 - y_i) \log(1 - \sigma(\alpha + \beta x_{i,tumor\_size})) \Big\} \end{aligned} \tag{16}$$

Computing Derivatives

- Gradient Descent computes *derivatives* of BCE w.r.t α and β.
- Here is how it works:
  Let us first consider $z_i = \alpha + \beta x_{i,tumor\_size}$
  we would like to compute:

$$\begin{aligned}\frac{\partial BCE}{\partial \alpha} &= -\frac{1}{N}\frac{\partial}{\partial \alpha}\sum_{i=1}^{N}\left\{y_i \log\left(\sigma(z_i)\right) + (1-y_i)\log(1-\sigma(z_i))\right\} \\ &= -\frac{1}{N}\sum_{i=1}^{N}\left\{y_i \frac{\partial}{\partial \alpha}\log\left(\sigma(z_i)\right) + (1-y_i)\frac{\partial}{\partial \alpha}\log\left(1-\sigma(z_i)\right)\right\}\end{aligned} \tag{17}$$

and similarly,

$$\frac{\partial BCE}{\partial \beta} = -\frac{1}{N}\sum_{i=1}^{N}\left\{y_i \frac{\partial}{\partial \beta}\log\left(\sigma(z_i)\right) + (1-y_i)\frac{\partial}{\partial \beta}\log\left(1-\sigma(z_i)\right)\right\} \tag{18}$$

The Magic Of Using The Sigmoid

- We will compute the derivatives one by one.
- Here is how we proceed for α:

$$
\begin{aligned}
\frac{\partial \log(\sigma(z_i))}{\partial \alpha} &= \overbrace{\frac{\partial \log(\sigma(z_i))}{\partial \sigma(z_i)} \times \frac{\partial \sigma(z_i)}{\partial z_i} \times \frac{\partial z_i}{\partial \alpha}}^{\text{chain rule of differentiation}} \\
&= \frac{1}{\sigma(z_i)} \times \sigma(z_i) \times (1 - \sigma(z_i)) \times 1 \\
&= (1 - \sigma(\alpha + \beta x_{i,tumor\_size}))
\end{aligned}
\tag{19}
$$

$$
\begin{aligned}
\frac{\partial \log(1 - \sigma(z_i))}{\partial \alpha} &= \frac{\partial \log(1 - \sigma(z_i))}{\partial (1 - \sigma(z_i))} \times \frac{\partial (1 - \sigma(z_i))}{\partial \sigma(z_i)} \times \frac{\partial \sigma(z_i)}{\partial z_i} \times \frac{\partial z_i}{\partial \alpha} \\
&= \frac{1}{(1 - \sigma(z_i))} \times (-1) \times \sigma(z_i) \times (1 - \sigma(z_i)) \times 1 \\
&= -\sigma(z_i) = -\sigma(\alpha + \beta x_{i,tumor\_size})
\end{aligned}
\tag{20}
$$

Continuing...

- And similarly for β:

$$\begin{aligned}\frac{\partial \log(\sigma(z_i))}{\partial \beta} &= \overbrace{\frac{\partial \log(\sigma(z_i))}{\partial \sigma(z_i)} \times \frac{\partial \sigma(z_i)}{\partial z_i} \times \frac{\partial z_i}{\partial \beta}}^{\text{chain rule of differentiation}} \\ &= \frac{1}{\sigma(z_i)} \times \sigma(z_i) \times (1-\sigma(z_i)) \times x_{i,tumor\_size} \\ &= (1-\sigma(\alpha+\beta x_{i,tumor\_size})) \times x_{i,tumor\_size}\end{aligned} \tag{21}$$

$$\begin{aligned}\frac{\partial \log(1-\sigma(z_i))}{\partial \beta} &= \frac{\partial \log(1-\sigma(z_i))}{\partial(1-\sigma(z_i))} \times \frac{\partial(1-\sigma(z_i))}{\partial \sigma(z_i)} \times \frac{\partial \sigma(z_i)}{\partial z_i} \times \frac{\partial z_i}{\partial \beta} \\ &= \frac{1}{(1-\sigma(z_i))} \times (-1) \times \sigma(z_i) \times (1-\sigma(z_i)) \times x_{i,tumor\_size} \\ &= -\sigma(\alpha+\beta x_{i,tumor\_size}) \times x_{i,tumor\_size}\end{aligned} \tag{22}$$

The Iterative Rule Of Gradient Descent

- We can now compute $\frac{\partial BCE}{\partial \alpha}$ and $\frac{\partial BCE}{\partial \beta}$.
- And finally make use of the Gradient Decent *update rule*:

$$\overbrace{\begin{pmatrix} \hat{\alpha}^{(t+1)} \\ \\ \hat{\beta}^{(t+1)} \end{pmatrix}}^{\text{estimates at } (t+1)^{th} \text{ step}} = \overbrace{\begin{pmatrix} \hat{\alpha}^{(t)} \\ \\ \hat{\beta}^{(t)} \end{pmatrix}}^{\text{estimates at } t^{th} \text{ step}} - \eta \cdot \overbrace{\begin{pmatrix} \left.\frac{\partial BCE}{\partial \alpha}\right|_{\hat{\alpha}^{(t)}} \\ \\ \left.\frac{\partial BCE}{\partial \beta}\right|_{\hat{\beta}^{(t)}} \end{pmatrix}}^{\text{gradient adjustments at } t^{th} \text{ step}} \tag{23}$$

Here η is the *learning rate*.

Evaluating Model Performance

Confusion Matrix And Related Metrics

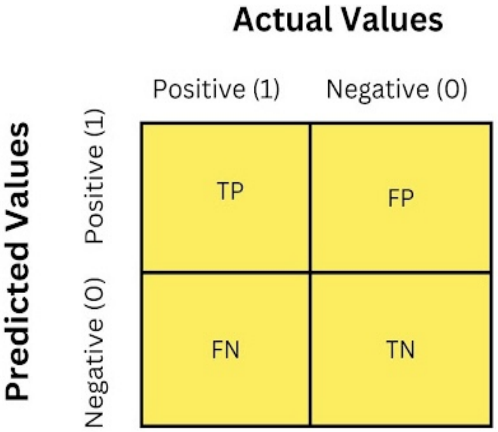

Figure: Confusion Matrix

- All cases $= TP + TN + FP + FN$
- Correctly classified cases $= TP + TN$
- Misclassified cases $= FP + FN$
- Precision $= \frac{TP}{TP+FP}$
- Recall (Sensitivity) $= \frac{TP}{TP+FN}$
- Specificity $= \frac{TN}{TN+FP}$
- Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$
- F1-score $= \frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$
- All the above metrics except the first one *depends on the threshold p^**.

Understanding The Precision

- Precision measures the *probability of predicting a true positive case* by a fitted model.
- According to the formula: $\frac{TP}{TP+FP}$, the *lesser* the number of false positive cases, the *higher* will be the precision.
- Precision is important where is making *false positive mistakes* is *risky*.
- In a email spam detection system, it's crucial that a *non-spam* email is *not getting tagged* as a spam email, otherwise an user may miss an important email. - *expecting a high precision*
- In medical diagnosis (e.g. cancer detection) high precision gives *confidence* to the doctors to start treatment without further tests. - *expecting a high precision*
- Precision is also important when the dataset is highly imbalanced (e.g. credit fraud detection, where getting a fraudulent transaction is rare). If the precision is low, even if the accuracy is very high, the model would probably raise many *false alarms*, which is misleading. - *expecting a high precision*

Understanding The Recall

- Recall (a.k.a **Sensitivity**) measures the *probability of detecting a true positive case* when it's *actually positive*.
- According to the formula: $\frac{TP}{TP+FN}$, the *lesser* the number of false negative cases, the *higher* will be the recall.
- Recall is important where making *false negative mistakes* is *risky*.
- In case of cancer detection diagnosis, it's very important that a cancer is *getting detected* in the body if it is *actually there*, otherwise it will be a *life risk*. - *expecting a high recall*
- While using medical kit for detecting COVID, it's important that a person is *NOT tagged* as 'COVID -VE' when he is *actually* 'COVID +VE'. Having such cases will *infect* many other people. - *expecting a high recall*
- Detecting as many threats as possible is important for an airport security system. Having higher values of recall will make sure almost all positive cases are identified. - *expecting a high recall*

Understanding The F1 Score

- The idea of using F1-score is to keep *a balance* between precision and recall.
- According to the formula: $\frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$, F1-score calculates *harmonic mean* of precision and recall.
- Harmonic mean *penalizes* the extreme values of both precision and recall.
- For example, in case of credit fraud detection, actual fraudulent cases (positive) are very rare and this may make the model biased towards *legitimate cases* with a *very high accuracy*, however it *may not* make any sense.
- Here the model should actually:
  - have *high precision* i.e. lower chance of raising false alarm by identifying a legitimate case as fraudulent
  - have *high recall* i.e. lower chance of missing fraudulent transaction

  The F1-score keeps a balance between these and gives a much more *realistic evaluation* of how well a model is performing in detecting the *minority class (fraud)*.

The Receiver Operating Characteristic Curve (ROC)

- It's a *graphical tool* depending on two metrics derived from the *confusion matrix*:
  - True Positive Rate = P(Predicted Positive | Actually Positive) = $\frac{TP}{TP+FN}$
  - False Positive Rate = P(Predicted Positive | Actually Negative) = $\frac{FP}{FP+TN}$
- By *varying* p^* within a *permissible range* a set of $\{(FPR_k, TPR_k)\}_{k=1}^{K}$ are obtained, and are plotted to form what is known as *ROC Curve*.
- The FPR varies *along the X-axis* and TPR varies *along the Y-axis*.
- Both TPR and FPR vary within $[0, 1]$ making the *total area* of the plotting canvas to be 1.
- The *diagonal line*, connecting the coordinates $(0, 0)$ and $(1, 1)$ indicates $FPR = TPR$, which is how a *random model* would behave.
- It's always good to have a model which produces *TPR values on the higher side* and *FPR values on the lower side*.

Area Under The [ROC] Curve (AUC)

- It's one of the robust measures to *compare* different models or model configurations.
- The diagonal line (— — —) divides the plotting canvas in to *two halves* having an *area of* 0.5 each. This line indicates a random classifier which is *equally good and bad*.
- Any model better than the random one will *cover an area* > 0.5.
- The *higher* the AUC the model achieves, the *better* its performance.

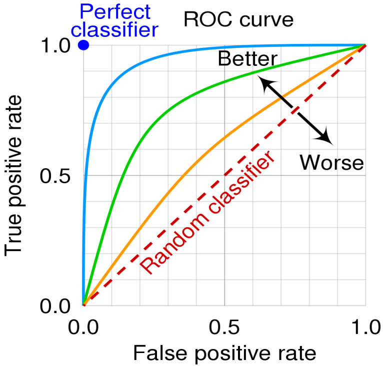

Figure: Comparing models with ROC curves and AUC values [1]

[1] Image Source: Wikipedia

Interpretations Of The Parameters

- α: When x_{tumor_size} *is zero*, it's the *value of the log-odds*. It's often called the *baseline log-odds*.
- β: It's the *change in log-odds* for an *unit change in* x_{tumor_size}.
- Interpretations are just like the *linear regression* as the alternate form of logistic regression (eqn. (6)) is exactly a linear regression w.r.t the log-odds.
- log-odds belongs to $(-\infty, +\infty)$ (verify)
- Even though logistic regression is used for *classification*, it actually *estimates a probability*, which is *continuous* within $(0, 1)$ - it's a *bridge* between *continuous modeling* and *discrete outcomes*, classification is just a *practical application* of this model.

References

- Generalized Linear Models by by P. McCullagh, J. A. Nelder
- Categorical Data Analysis by Alan Agresti
- Machine Learning Crash Course by Google

Thank You