Logistic Regression

Sourav Karmakar

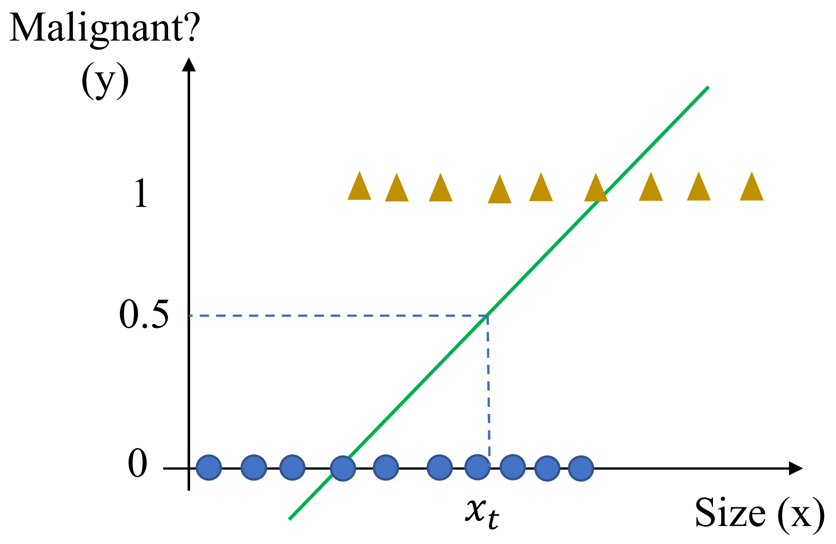
souravkarmakar29@gmail.com

Outline

Why Linear Regression is not suitable for Classification

Consider a classification problem where we are trying to **classify a Tumour to be Benign or Malignant** based on its Size.

We gather data on various Tumours and label them as 0 (benign) or 1 (Malignant). If we plot these data we see something like following.



Lets see what will happen if we put a Linear Regression line through our observed data.

- First problem that we observe is that, the linear regression produces continuous output which is more suitable when target variable is quantitative in nature. However, here the **target variable is discrete**.

- Say we now try to discretize the output by thresholding. That means for an unknown tumour of size 'x', let the corresponding value obtained from linear regression is 'y'. If $y \geq threshold$ then Malignant else Benign.

- This creates another problem of too many errors while classifying because determination of threshold is not trivial and intuitive in this case.

Why Linear Regression is not suitable for Classification

- In classification problem we are more interested to learn the conditional probability $\boldsymbol{P(y = 1|x)}$ *(i.e. Probability of target to be 1 given the value of the input variable x)*

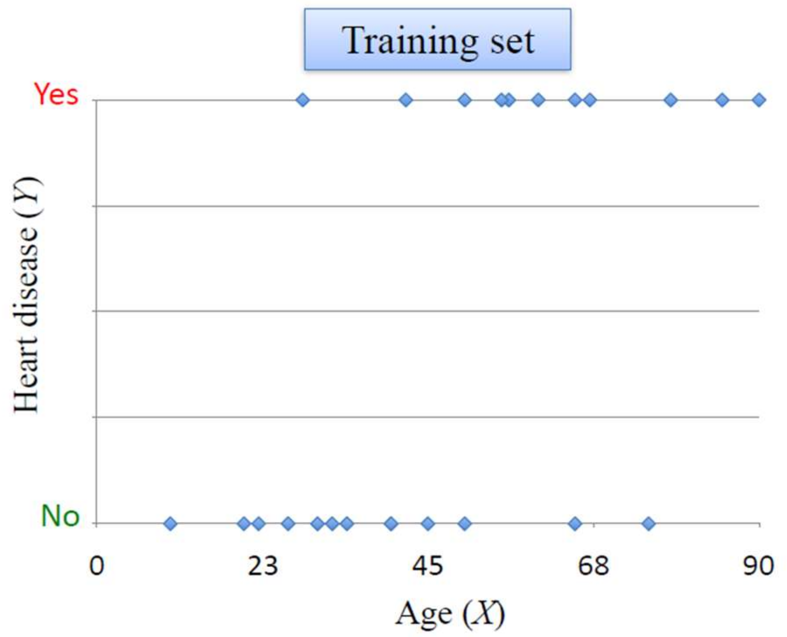
- Then we will choose a suitable threshold (usually 0.5). If the probability > threshold we will mark that observation as **1** else **0**.

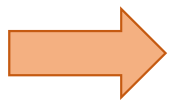
- Hence, our hypothesis function should produce legitimate probability value as output, i.e. provide output between **[0,1]**.

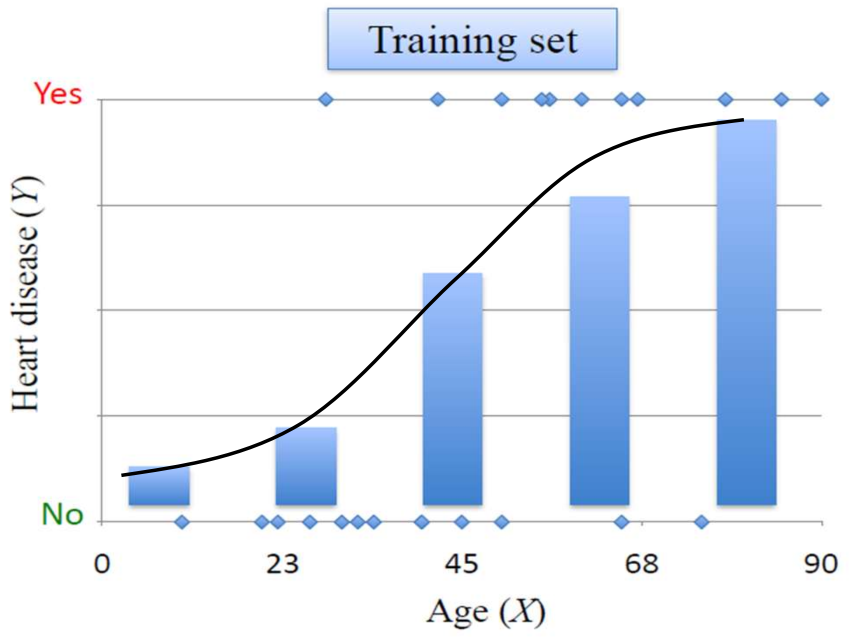
- However, linear regression produces real valued outputs, which theoretically can be within $(-\infty, \infty)$. Hence, Linear Regression is not suitable for modelling the hypothesis function for classification problem.

Example of estimating conditional probability from Data

Consider the following Scenario of Heart Disease vs. Age:







Task: Given a new person's age, predict if he/she has heart disease. Hence, mathematically speaking, our task is to estimate $P(Y = \text{Yes} \mid X)$

- First Calculate $P(Y = \text{Yes} \mid X)$ for different values of X
- Then fit a curve that estimates the probability $P(Y = \text{Yes} \mid X)$

Towards Logistic Regression for Binary Classification

- In real life scenario we usually have multiple predictor variable. Let's denote them by $x_1, x_2, \dots, x_k$

- If we linearly combine these predictors we get: $z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_k x_k$

- This is the formulation of linear regression. As we know z shall range within $(-\infty, \infty)$

- But, in Logistic Regression the output shall be a legitimate probability value p which shall range from $[0, 1]$

- So our task boils down to obtaining a function $f(p)$ which shall range between $(-\infty, \infty)$ and then estimate $f(p)$ using z

  or, finding a function $g(z)$ which will range between $[0, 1]$ and use that to estimate probability p.

- It is worth noting that, $g\big(f(p)\big) = g(z) = p$. Hence mathematically $g\ and\ f$ are inverse functions of one another.

Towards Logistic Regression for Binary Classification

- In many situations, instead of using explicit probability of an event we use **odds**. For example, odds of India winning a match against Australia is 3 to 4. The **odds of an event** is defined by

$$Odds\ of\ an\ Event = \frac{Probability\ of\ success\ of\ the\ event}{Probability\ of\ failure\ of\ the\ event}$$

- If probability of success of an event is p, then odds of the event is $odds = \frac{p}{1-p}$

- As $p \in [0, 1]$, $odds = \frac{p}{1-p} \in [0, \infty)$, $odds = 0\ when\ p = 0\ and\ odds \to \infty\ when\ p \to 1$

- Now if we take natural logarithm of odds, $\ln(odds) = \ln \frac{p}{1-p} \in (-\infty, \infty)$.
  However it is undefined when $p = 0$

- So, $f(p) = \ln \frac{p}{1-p}$ is one such function which ranges between $(-\infty, \infty)$ and can be estimated using $z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_k x_k$

Towards Logistic Regression for Binary Classification

- So we got a function $f(p) = \ln\frac{p}{1-p}$ which ranges between $(-\infty, \infty)$ and can be faithfully estimated using $z = \theta_0 + \theta_1\, x_1 + \theta_2\, x_2 + \cdots + \theta_k\, x_k$

$$z = \ln\frac{p}{1-p} \qquad \therefore \frac{p}{1\ -p} = e^z \quad (taking\ exponential\ on\ both\ side)$$

$$\therefore \frac{1\ -p}{p} = \frac{1}{e^z} = e^{-z}$$

$$\therefore \frac{1\ -p}{p} + 1 = 1 +\ e^{-z} \quad \Rightarrow \frac{1\ -p+p}{p} = 1 +\ e^{-z} \quad \Rightarrow \frac{1}{p} = 1 +\ e^{-z}$$

$$\therefore p = \frac{1}{1 + e^{-z}}$$

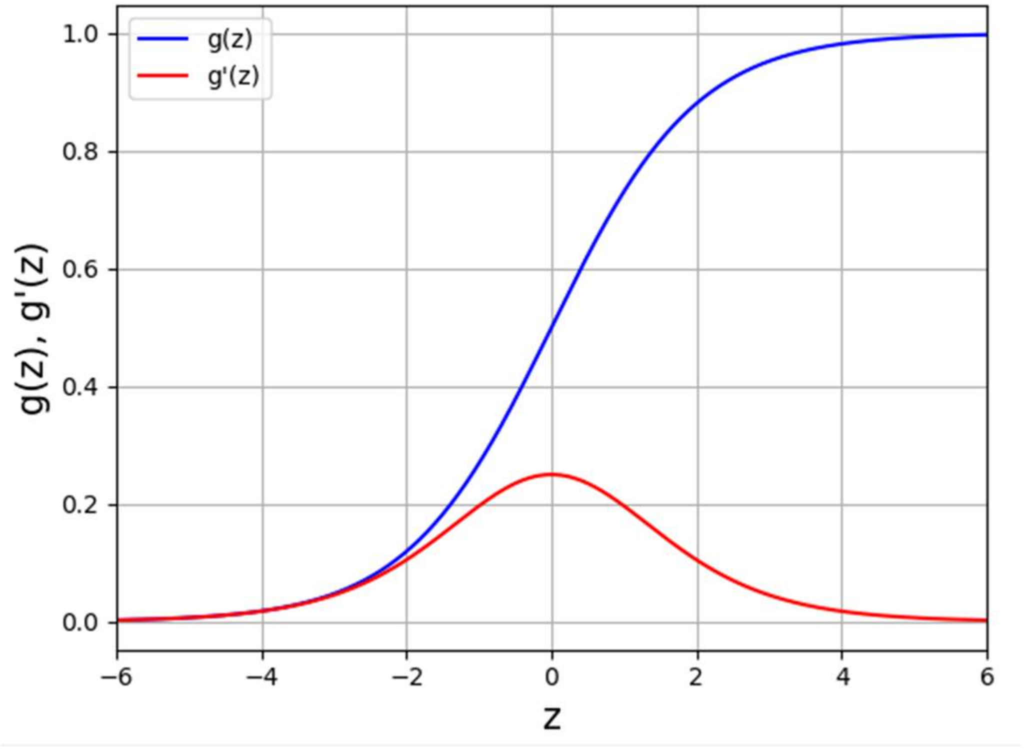
- So we obtained $\boxed{g(z) = \frac{1}{1+e^{-z}}}$ which provides output between $(0, 1)$

This is known as Logistic or Sigmoid function

The Logistic Function and its derivative

The Logistic Function, also known as the Sigmoid Function is defined by:

$$g(z) = logistic(z) = sigmoid(z) = \frac{1}{1+e^{-z}} = \frac{e^{z}}{1+e^{z}}\text{;} \ \text{Clearly, } 0 \leq g(z) \leq 1$$



If we differentiate $g(z)$ with respect to z, we get the following:

- $g(z) = \frac{1}{1+e^{-z}}$
- $g'(z) = -\left(\frac{1}{1+e^{-z}}\right)^2 \frac{d}{dz}\left(1+e^{-z}\right)$
- $g'(z) = \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1}{(1+e^{-z})}\frac{e^{-z}}{(1+e^{-z})} = \frac{1}{(1+e^{-z})}\frac{(1+e^{-z})-1}{(1+e^{-z})}$
- $g'(z) = \frac{1}{(1+e^{-z})}\left(1 - \frac{1}{(1+e^{-z})}\right)$
- $\boxed{g'(z) = g(z)\left(1 - g(z)\right)}$

We can also show the graph of $g'(z)$ along with $g(z)$.

Hypothesis of Logistic Regression

Let there is only one predictor variable/feature (x), then our **hypothesis function** for logistic regression is:

$$h_\theta(x) = g(\theta_0 + \theta_1 x); \quad where\ g(z) = \frac{1}{(1 + e^{-z})}$$

$$\therefore h_\theta(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}\ ; \ here\ \theta_0\ and\ \theta_1\ are\ the\ model\ parameters$$

Let there are more than one (say k no. of) predictor variables/ features $(\boldsymbol{x} = \{x_1, x_2, x_3, \dots, x_k\})$, then our **hypothesis function** for logistic regression is:

$$h_\theta(\boldsymbol{x}) = g\left(\theta_0 + \sum_{j=1}^{k} \theta_j x_j\right); \quad where\ g(z) = \frac{1}{(1 + e^{-z})}$$

$$\therefore h_\theta(\boldsymbol{x}) = \frac{1}{1 + e^{-\left(\theta_0 + \sum_{j=1}^{k} \theta_j x_j\right)}}\ ; \quad here\ \theta_j\ (j = 0,1,2, \dots, k)\ are\ the\ model\ parameters$$

Hypothesis of Logistic Regression

Interpretation of hypothesis function:

- $h_\theta(\boldsymbol{x}) = P\,(y = 1|\ \boldsymbol{x}\,; \theta) =$ Estimated Probability that "$y\ =\ 1$", on input $\boldsymbol{x}$, parameterized by θ. Note that for binary classification our predicted output $\hat{y} = h_\theta(\boldsymbol{x})$.
- Clearly, $1\ - h_\theta(\boldsymbol{x}) = P\,(y = 0|\boldsymbol{x}; \theta) =$ Estimated Probability that "$y\ = 0$", on input $\boldsymbol{x}$, parameterized by θ.
- Note that there are only two classes (one corresponds to y = 1 and other is y = 0). Hence, it is called **Binary Classification**.

We use the training data to estimate the model parameters $\boldsymbol{\theta} = \{\theta_0, \theta_1, \theta_2, \dots, \theta_k\}$

The Cost Function

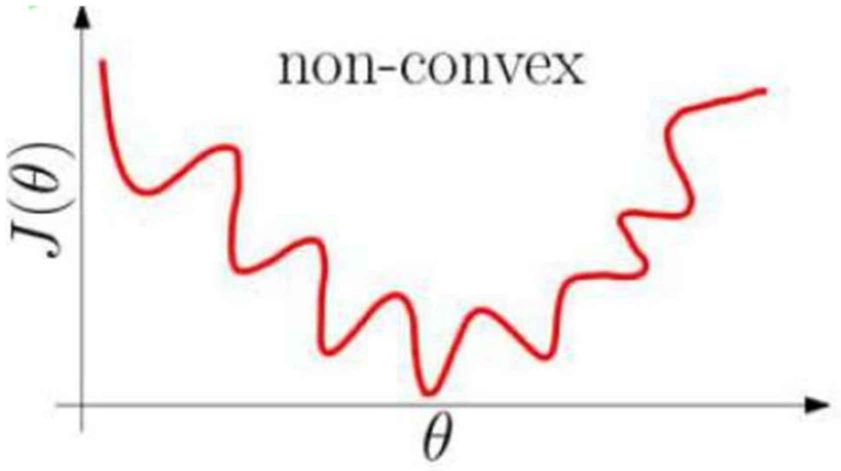
If we use the Mean Square Error (MSE) cost function, then for individual observation:

$$Cost(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Then the overall cost function:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Problem with MSE cost function in classification problem is that it has many **local minima**. i.e. MSE cost function becomes **non-convex** in classification problem.



If cost function has many local minima then optimizing the same using gradient descent is difficult. Because gradient descent being a convex optimization algorithm is susceptible to local minima.

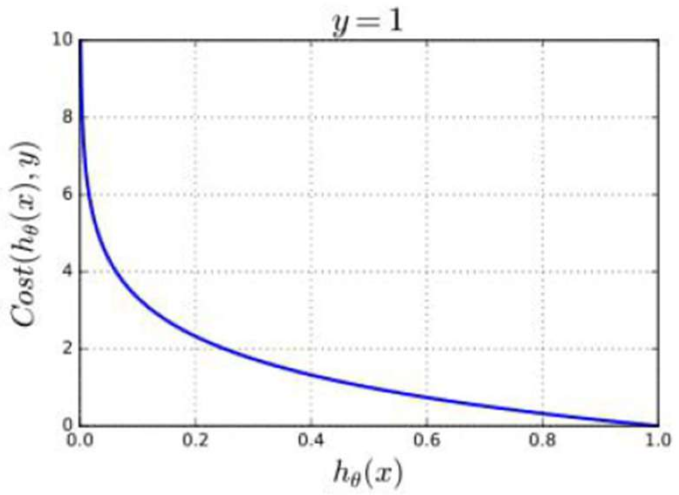
Hence we need different cost function for classification problem.

The Cost Function

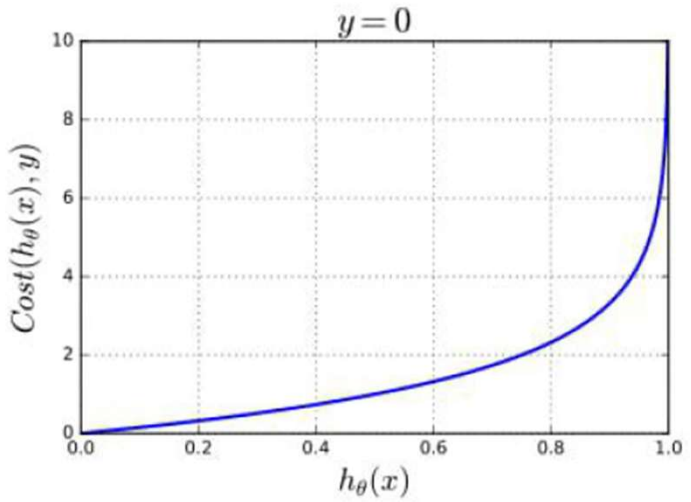
The cost function of for Binary Classification problem has following form:

$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)), & \text{if } y = 1 \\ -log(1 - h_\theta(x)), & \text{if} y = 0 \end{cases}$$

Interpretation of the cost function:



$h_\theta(x) = 1, Cost = 0$
$h_\theta(x) \to 0, Cost \to \infty$



$h_\theta(x) = 0, Cost = 0$
$h_\theta(x) \to 1, Cost \to \infty$

The Cost Function

We can express the cost function in Binary Classification as:

$$Cost(h_\theta(x), y) = -\left[\, y \log\left(h_\theta(\boldsymbol{x})\right) + (1-y) \log\left(1 - h_\theta(\boldsymbol{x})\right)\right]$$

Now the overall cost function is nothing but average of individual cost functions:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} Cost\left(h_\theta\left(\boldsymbol{x}^{(i)}\right), y^{(i)}\right)$$

Hence, in this case:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log\left(\underbrace{\boxed{h_\theta\left(\boldsymbol{x}^{(i)}\right)}}_{y_p^{(i)}}\right) + \left(1 - y^{(i)}\right) \log\left(1 - \underbrace{\boxed{h_\theta\left(\boldsymbol{x}^{(i)}\right)}}_{y_p^{(i)}}\right)\right]$$

$y_p^{(i)} \longleftarrow$ *The predicted output* $\longrightarrow y_p^{(i)}$

This is also called **“Binary Cross-entropy Loss”** or **“Log-loss”**

We optimize this cost function using gradient descent algorithm to find the values of the model parameters for which the cost function $J(\theta)$ is minimum.

Gradient Descent Optimization

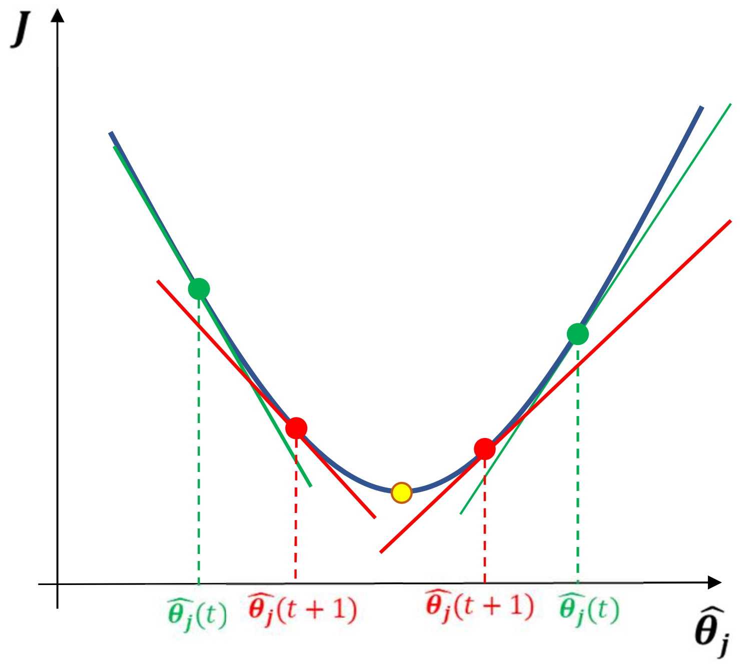
- **Algorithm:** Gradient Descent (an optimization algorithm used to find minimum of convex functions)
  - **Input:** Feature set $\left(X = \left[\left\{x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, \dots, x_k^{(i)}\right\}\right]_{i=1}^{m}\right)$ and corresponding target variables $\left(Y = \left[y^{(i)}\right]_{i=1}^{m}\right)$
  - **Output:** Model parameters $\boldsymbol{\Theta} = [\theta_0, \theta_1, \theta_2, \dots, \theta_k]$ for which the cost function $J(\theta)$ is minimum.
  - **Initialisation:** Initialise the model parameters $\boldsymbol{\Theta} = [\theta_0, \theta_1, \theta_2, \dots, \theta_k]$ with random numbers.
    Initialise iteration counter: $t \leftarrow 1$
  - **Repeat until Convergence** {

```
    → Compute Cost function: $J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(y_p^{(i)}\right) + \left(1 - y^{(i)}\right)\log\left(1 - y_p^{(i)}\right)\right]$
    → Compute the gradient of Cost function: $\nabla J(\theta) = \frac{\partial}{\partial\theta} J(\theta)$
    → Update the model parameters: $\theta(t+1) = \theta(t) - \alpha \nabla J(\theta)$
    → Increment the iteration counter: $t \leftarrow t + 1$
}
```

α is a small positive user defined parameter called **learning rate**

Intuition of Gradient Descent Algorithm

- The plot of cost function with respect to any of the model parameters (considering the other parameters fixed) is convex in nature as shown below.



- Now as per gradient descent update rule:

$$\theta(t+1) = \theta(t) - \alpha \frac{\partial}{\partial\theta} J(\theta) \Rightarrow \Delta\theta = -\alpha \frac{\partial}{\partial\theta} J(\theta)$$

- Now if the gradient $\frac{\partial J}{\partial\theta}$ is *positive*, then the update in the value of θ which is $\Delta\theta$ is *negative*. Indicating that, in the next update θ would be smaller than its previous value.

- On the other hand if the gradient $\frac{\partial J}{\partial\theta}$ is negative, then the update in the value of θ is positive. Indicating that, in the next update θ would be greater than its previous value.

- Ultimately it leads to the minimum of the convex curve after few iterations. Learing rate $\boldsymbol{\alpha}$ ensures smooth descent along the curve without much oscillations

Gradient Descent Optimization applied to Logistic Regression

We use **gradient descent** to optimize the cost function. In order to do so, we have to find $\frac{\partial J(\theta)}{\partial \theta_j}$ $for\ j = 0,1,2,\dots,k$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} \log\left(h_\theta\left(\boldsymbol{x}^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1 - h_\theta\left(\boldsymbol{x}^{(i)}\right)\right)\right], where\ h_\theta\left(\boldsymbol{x}^{(i)}\right) = \frac{1}{1+ e^{-\left(\theta_0+\sum_{j=1}^{k}\theta_j x_j^{(i)}\right)}}$$

$$Let, z^{(i)} = \theta_0 + \sum_{j=1}^{k}\theta_j x_j^{(i)}\ and\ h_\theta\left(x^{(i)}\right) = g\left(z^{(i)}\right) = \frac{1}{1+ e^{-z^{(i)}}}$$

$$then\ J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(g\left(z^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1 - g\left(z^{(i)}\right)\right)\right]$$

$$\therefore \frac{\partial J(\theta)}{\partial \theta_j} = -\frac{1}{m}\sum_{i=1}^{m}\boxed{\frac{\partial J(\theta)}{\partial g\left(z^{(i)}\right)}} \times \boxed{\frac{\partial g\left(z^{(i)}\right)}{\partial z^{(i)}}} \times \boxed{\frac{\partial z^{(i)}}{\partial \theta_j}} \Rightarrow \boxed{\frac{\partial z^{(i)}}{\partial \theta_j} = \begin{cases} 1, & for\ j=0 \\ x_j^{(i)}, & for\ j = 1,2,\dots,k \end{cases}}$$

$$\boxed{\begin{aligned}\frac{\partial J(\theta)}{\partial g\left(z^{(i)}\right)} &= \frac{y^{(i)}}{g\left(z^{(i)}\right)} - \frac{\left(1-y^{(i)}\right)}{\left(1-g\left(z^{(i)}\right)\right)} \\ = \frac{y^{(i)}\left(1-g\left(z^{(i)}\right)\right) - \left(1-y^{(i)}\right)g\left(z^{(i)}\right)}{g\left(z^{(i)}\right)\left(1-g\left(z^{i}\right)\right)} &= \frac{\left(y^{(i)} - g\left(z^{(i)}\right)\right)}{g\left(z^{(i)}\right)\left(1-g\left(z^{(i)}\right)\right)}\end{aligned}}$$

$$\boxed{\frac{\partial g\left(z^{(i)}\right)}{\partial z^{(i)}} = g\left(z^{(i)}\right)\left(1 - g\left(z^{(i)}\right)\right)}$$

*Proof of this is already provided in the slide where we discussed about **logistic function***

Gradient Descent Optimization applied to Logistic Regression

$$\frac{\partial J(\theta)}{\partial \theta_j} = -\frac{1}{m}\sum_{i=1}^{m}\left[\frac{\left(y^{(i)} - g\left(z^{(i)}\right)\right)}{\cancel{g\left(z^{(i)}\right)\left(1 - g\left(z^{i}\right)\right)}} \times \cancel{g\left(z^{(i)}\right)\left(1 - g\left(z^{(i)}\right)\right)} \times x_j^{(i)}\right] \quad ; \quad x_0^{(i)} = 1 \quad \forall i \in \{1,2,3, \ldots.., m\}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left(g\left(z^{(i)}\right) - y^{(i)}\right)x_j^{(i)} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$

This can be obtained by adding a feature column with all values **1** *to the left of the dataset table.*

Sl. No.	x_0	x_1	$\cdots$	x_k
1	1	$x_1^{(1)}$	$\cdots$	$x_k^{(1)}$
2	1	$x_1^{(2)}$	$\cdots$	$x_k^{(2)}$
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
m	1	$x_1^{(m)}$	$\cdots$	$x_k^{(m)}$

$$\therefore \frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)} \quad ; \quad \forall j \in \{0,1,2, \ldots.., k\}$$

Therefore the Gradient descent update rule is:

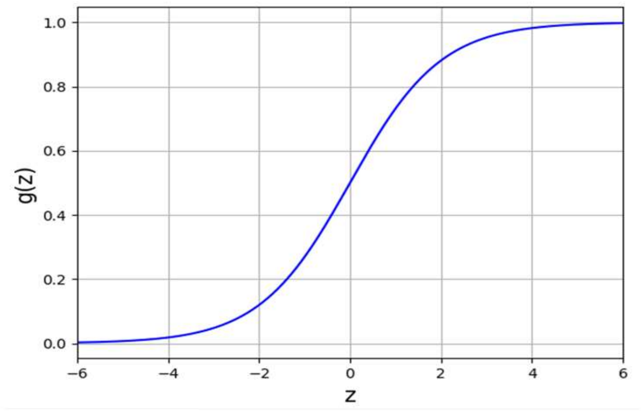
$$\boxed{\begin{aligned} \theta_j(t+1) &= \theta_j(t) - \alpha\frac{\partial J(\theta)}{\partial \theta_j} \\ &= \theta_j(t) - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)} \quad ; \quad \forall j \in \{0,1,2,\ldots,k\} \end{aligned}}$$

α is a user defined small positive value known as **“Learning Rate”**

Here, $\theta_j(t)$ implies the value of θ_j at t^{th} iteration during gradient descent update.

Decision Boundary of Logistic Regression

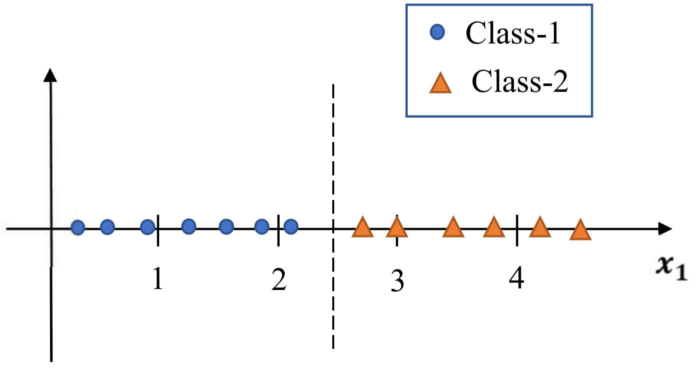
$$y = h_\theta(\boldsymbol{x}) = P\,(y = 1 \mid \boldsymbol{x}\,;\theta) = g\left(\theta_0 + \sum_{j=1}^{k} \theta_j x_j\right); \quad g(z) = \frac{1}{(1 + e^{-z})}$$

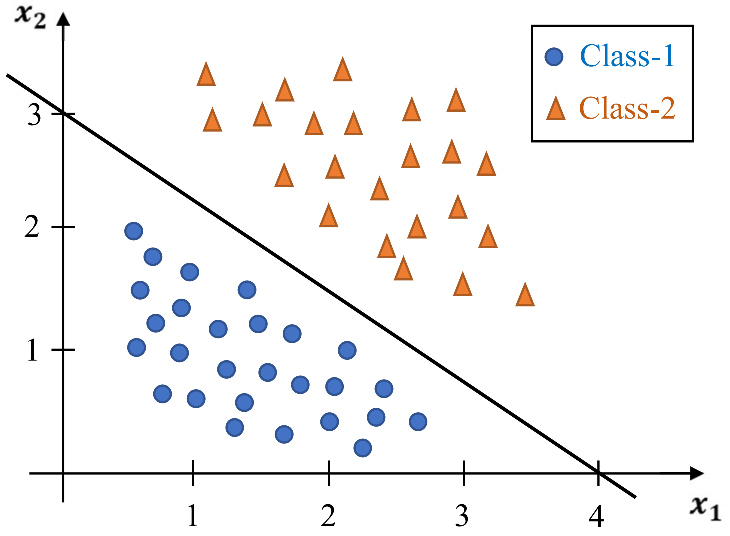


- This is a binary classification with y = 0 (for class-1) or 1 (for class-2)
- Let, $y = \begin{cases} 0, if\ h_\theta(\boldsymbol{x}) < \mathbf{0.5} \\ 1, if\ h_\theta(\boldsymbol{x}) \geq \mathbf{0.5} \end{cases}$; for some input feature vector $\boldsymbol{x}$
- Let, $y = \begin{cases} 0, if\ g(z) < \mathbf{0.5} \\ 1, if\ g(z) \geq \mathbf{0.5} \end{cases}$; where $z = \theta_0 + \sum_{j=1}^{k} \theta_j x_j$ and $g(z) = \frac{1}{(1+ e^{-z})}$
- Let, $y = \begin{cases} 0, if\ z < \mathbf{0} \\ 1, if\ z \geq \mathbf{0} \end{cases}$; where $z = \theta_0 + \sum_{j=1}^{k} \theta_j x_j$, Because $g(z) = \frac{1}{(1+e^{-z})} = 0.5 \ \ when\ z = 0$
- Hence, our decision boundary is $z = 0$; where $z = \theta_0 + \sum_{j=1}^{k} \theta_j x_j$. Decision boundary separates the two classes.
- $z = \theta_0 + \sum_{j=1}^{k} \theta_j x_j = 0 \ \Rightarrow \theta_o + \theta_1 x_1 + \theta_2 x_2 + \ \ldots + \theta_k x_k = 0$; This is the equation of **k-dimensional hyperplane.**

Decision Boundary of Logistic Regression

- Consider 1-D example. There is only one predictor variable x_1
- The form of the decision boundary is: $\theta_0 + \theta_1 x_1 = 0 \; or, x_1 = -\frac{\theta_0}{\theta_1}$
- Let $\theta_0 = -2.5$ and $\theta_1 = 1$, then $x_1 = 2.5$ is our decision boundary
- For an unknown sample x, if $x \geq 2.5$, then we say that x belongs to Class-2, else x belongs to Class-1





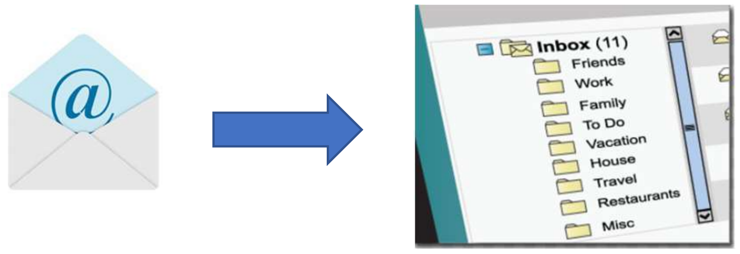
- For two dimension (i.e. there are only two features / predictors) the form the decision boundary is $\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$
- Consider the example beside. There are two classes (Class – 1 and Class – 2)
- Let $\theta_0 = -12 \,, \theta_1 = 3 \; and \; \theta_2 = 4$, then our decision boundary is: $-12 + 3x_1 + 4x_2 = 0$ or $3x_1 + 4x_2 = 12$, which separates the two classes.
- For an unknown sample $\boldsymbol{x} = [x_1, x_2]$, if $-12 + \; 3x_1 + 4x_2 \geq 0$, then $\boldsymbol{x}$ belongs to Class-2, else $\boldsymbol{x}$ belongs to Class-1

MULTICLASS CLASSIFICATION

There are plenty of examples of multiclass classification problems in real life.

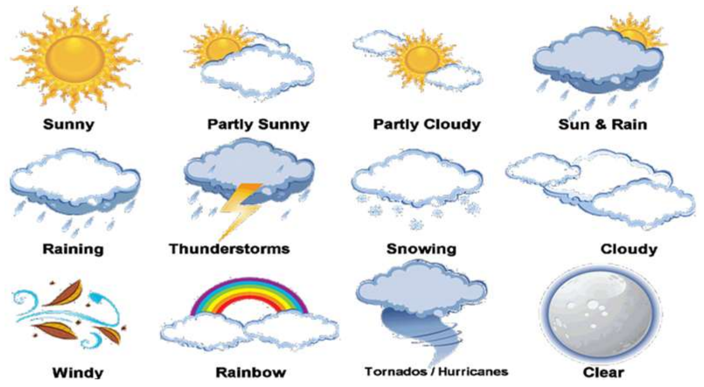
Email Foldering / Tagging: Work, Friend, Family, Hobby etc.

Based on previous foldering / tagging history a new email should be classified and placed into the relevant folder.



Identifying the weather condition: Sunny, Rainy, Overcast etc.

It is a Machine Learning Problem to predict the weather of the coming day based on the past records.



Identifying Genre of Music: Pop, Rock, Classical, Instrumental etc.

Medical Diagnosis: Not ill, Disease-1, Disease-2 etc.

Parts of Speech Tagging of a word: Noun, Pronoun, Verb, Adjective, Adverb, Preposition, Article

And Many More .....

IRIS-DATASET

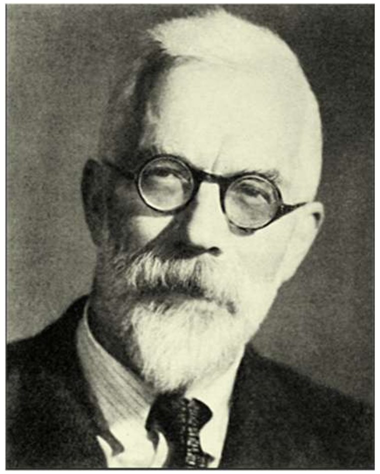
The ***Iris* flower data** is a multivariate data set introduced by the British statistician and biologist **Ronald A. Fisher** in his 1936 paper *The use of multiple measurements in taxonomic problems.*

The data set consists of 50 samples from each of three species of ***Iris***:
Iris setosa, Iris versicolor, Iris virginica

Iris setosa

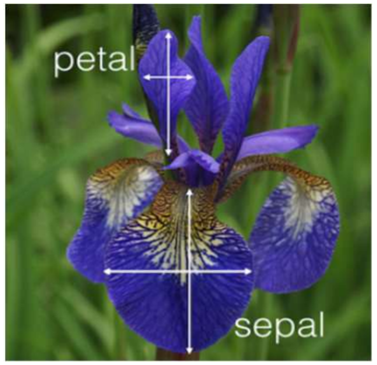
Iris versicolor

Iris virginica

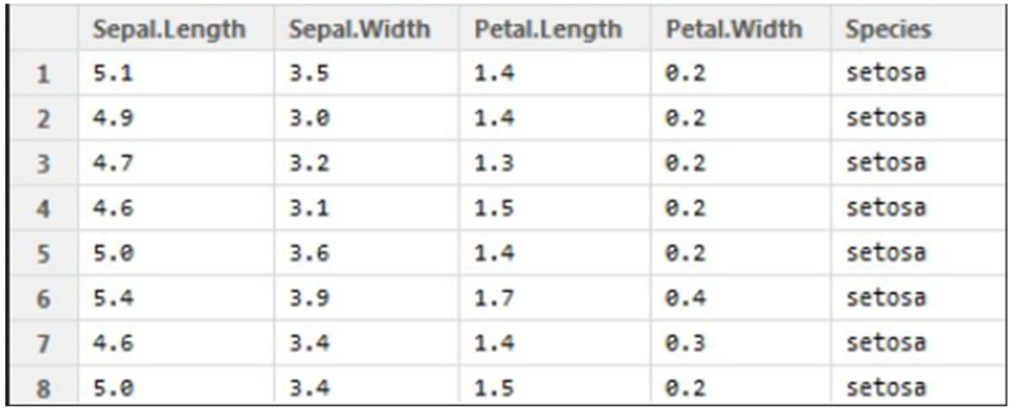

Sir Ronald Aylmer Fisher (FRS)
(17 February 1890 – 29 July 1962)

Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.

The data set contains 150 observations of iris flowers. There are four columns of measurements of the flowers in centimeters. The fifth column is the species of the flower observed. All observed flowers belong to one of three species.



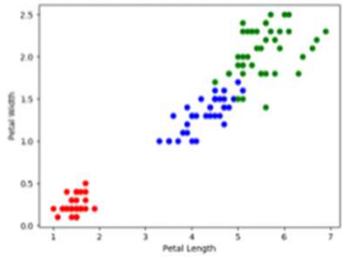
IRIS-DATASET



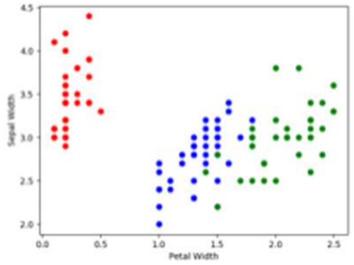
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

- This is the view of first 8 rows of Iris Dataset
- There are total 4 features: Sepal Length, Sepal Width, Petal Length and Petal Width.
- Here the target variable is “Species”. Species are usually coded in numbers. ‘0’ for setosa, ‘1’ for versicolor and ‘2’ for virginica.

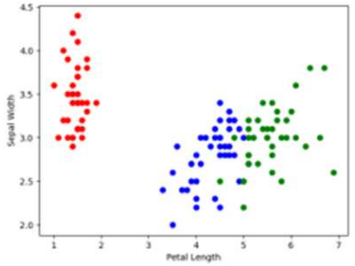
Following visualizations are obtained from data when considering two dimensions, that is, two features, at a time. Each graph or figure contains two different features, one along x-axis and another along y-axis.



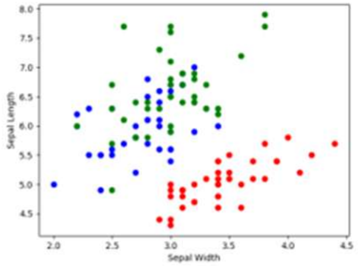
1. *Petal Width vs Petal Length*



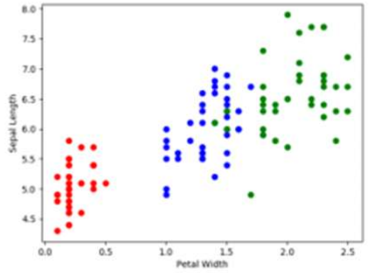
2. *Sepal Width vs Petal Width*



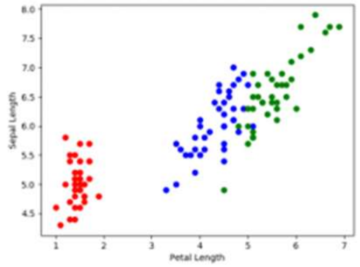
3. *Sepal Width vs Petal Length*



4. *Sepal Length vs Sepal Width*



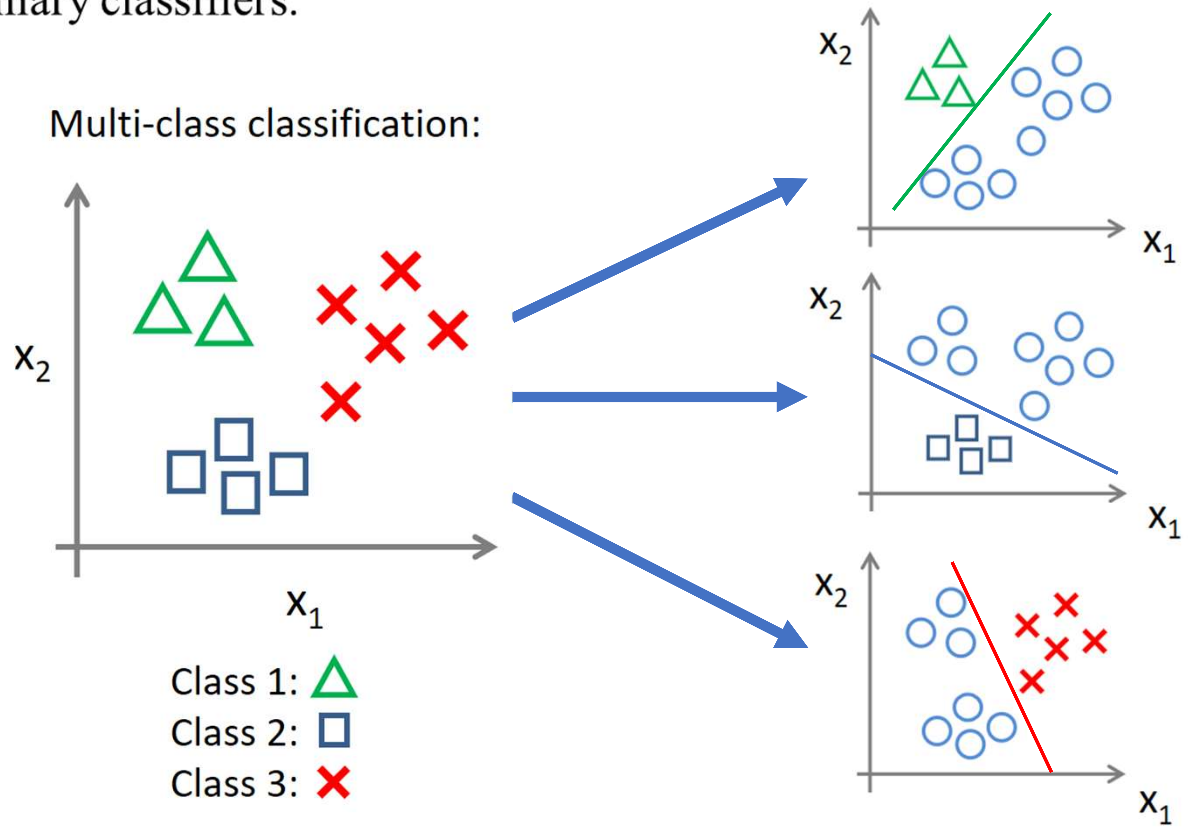
5. *Sepal Length vs Petal Width*



6. *Sepal Length vs Petal Length*

MULTICLASS CLASSIFICATION

There are various ways to deal with multi-class classification problem. We shall consider one model here known as **One-vs-All** model. The intuition behind this model is: for 'C' no. of classes ($C \geq 3$), we shall train C different binary classifiers.



$$h_\theta^{(1)}(\boldsymbol{x}) = P\,(y = 1 |\, \boldsymbol{x}, \theta)$$

$$h_\theta^{(2)}(\boldsymbol{x}) = P\,(y = 2 |\, \boldsymbol{x}, \theta)$$

$$h_\theta^{(3)}(\boldsymbol{x}) = P\,(y = 3 |\, \boldsymbol{x}, \theta)$$

$$\boxed{h_\theta^{(k)}(\boldsymbol{x}) = P\,(\,y = k \mid \boldsymbol{x}, \theta)\ for\ k = 1, 2, 3}$$

$$\boxed{\text{Predicted Class} = \max_{k}\left(h_\theta^{(k)}(\boldsymbol{x})\right)}$$

EVALUATION OF MULTI-CLASSIFIER

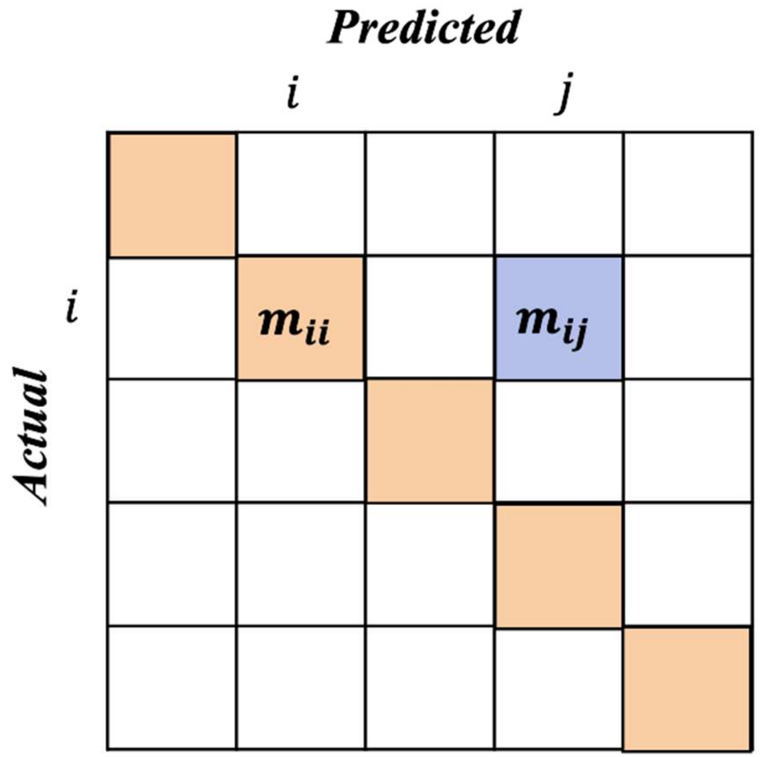
- **Confusion Matrix:**

  Confusion matrix (M_C) in a matrix of dimension $|\mathcal{C}| \times |\mathcal{C}|$. It records the performance of the classifier. Where $|\mathcal{C}|$ is number of classes in the dataset.

- The $(i, j)^{th}$ entry of confusion matrix (m_{ij}) denotes the number of test instances which originally belong to class - i but predicted as members of class - j.

- The diagonal entries of confusion matrix (m_{ii}) denotes the number of correct identification of test samples. Hence, total number of correct identifications = $\boldsymbol{trace}(M_C)$

- All the entries of confusion matrix must be positive integer or zero. $i.e. m_{ij} \in \mathbb{N} \cup \{0\}$



EVALUATION OF MULTI-CLASSIFIER

- **Accuracy:**

$$Accuracy = \frac{\sum_{i=1}^{|\mathcal{C}|} m_{ii}}{\sum_{i=1}^{|\mathcal{C}|}\sum_{j=1}^{|\mathcal{C}|} m_{ij}} = \frac{trace(M_c)}{sum(M_c)}$$

- **Recall of a class and Average Recall:**

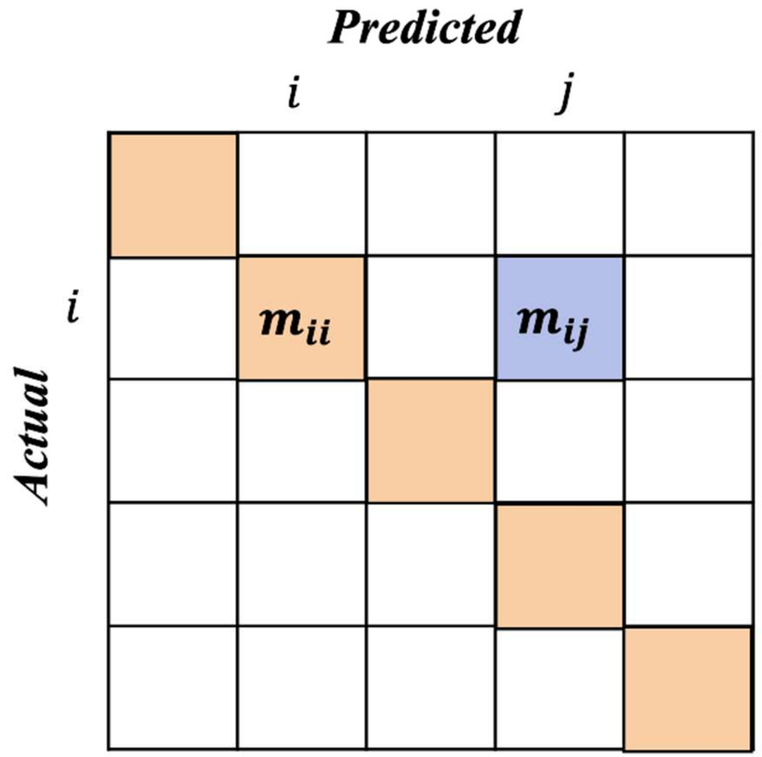
Recall of particular i^{th} class: $\beta_i = \frac{m_{ii}}{\sum_{j=1}^{|\mathcal{C}|} m_{ij}}$,

Hence, average recall of all classes: $\frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} \beta_c$

- **Precision of a class and Average Precision:**

Precision of particular i^{th} class: $\alpha_i = \frac{m_{ii}}{\sum_{i=1}^{|\mathcal{C}|} m_{ij}}$,

Hence, average precision of all classes: $\frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} \alpha_c$



Thank You