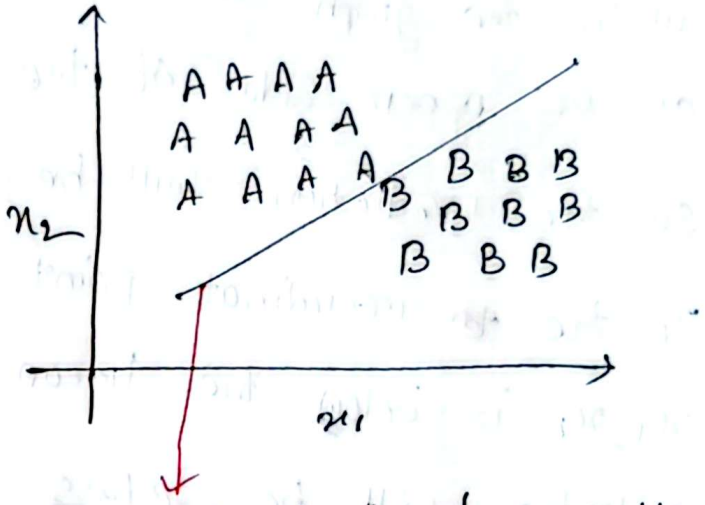
Logistic Regression

Unlike linear regression, Logistic regression is used to predict categorical outcomes. So, Logistic Regression is use for **classification problems.**

Let is say we have a dataset with two features x_1 and x_2 and with class A, B →

x_1	x_2	class
–	–	A
–	–	B
–	–	A
⋮	⋮	⋮
–	–	A

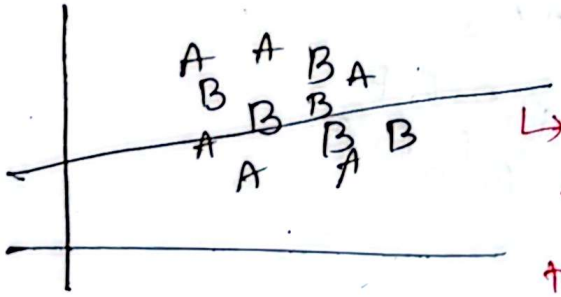
Plot →



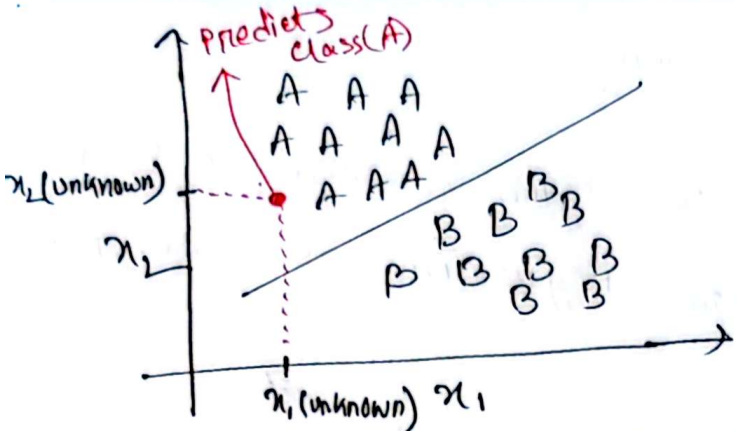
This data is linearly separable by this line.

We can apply logistic regression, if the data is linearly seperable by a line (in two dimention) or hyperplane (in higher dimention)

If data is non linear like →



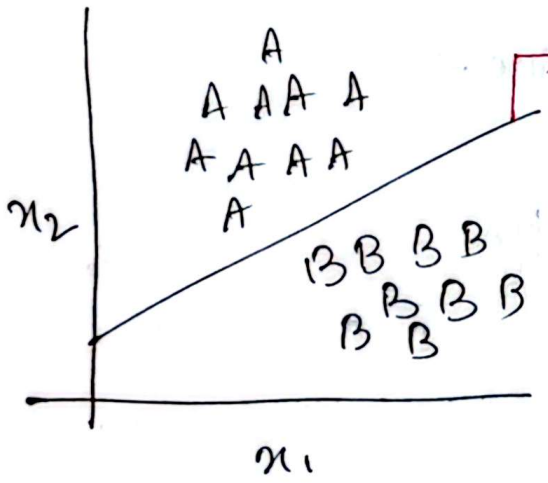
↳ we can linearly seperate the class A, B with this line.



To predict for a new value of x_1 and x_2 we can ~~just~~ easily use this model. For two values x_1(unknown) and x_2(unknown) In the graph we can see we get the coordinate at the upper side of the ~~sep~~ linear line (our model). So the prediction will be **class A**.

If the coordinate point for a new unknown x_1, x_2 is below the linear line ~~will be~~ the Prediction will be **class B**

Here this linear line is our logistic regression model, which we need to find. Basically, we need to find the equation of this linear line.



The equation will be,

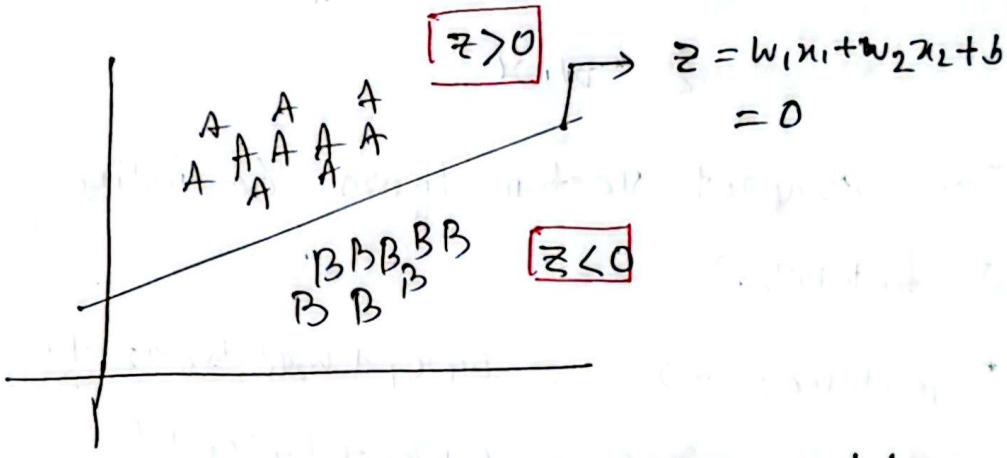
$$z = x_1 w_1 + x_2 w_2 + b$$

for two feature x_1 and x_2

Just like linear regression we need to find the weights here → w_1, w_2, and b.

For higher dimention like for 3 features, the equation will be $z = x_1 w_1 + x_2 w_2 + x_3 w_3 + b$.

Now ~~for~~ over this line/equation $z = w_1 x_1 + w_2 x_2 + b$, we will get $z = 0$.



So, the upper region of the line is $z > 0$ and lower region of the line is $z < 0$.

So, if $z > 0$, the predicted class will be A

if $z < 0$, the predicted class will be B.

so, using this equation we can easily find prediction for any new ~~value~~ unknown value of x_1 and x_2.

if $w_1 x_1(\text{unknown}) + w_2 x_2(\text{unknown}) + b \geq 0 \rightarrow$ class A

if $w_1 x_1(\text{unknown}) + w_2 x_2(\text{unknown}) + b < 0 \rightarrow$ class B.

We can use vector notations for this linear equation.

The **input vector** → $\langle x_1, x_2 \rangle = x$

and **weight vector** → $\langle w_1, w_2 \rangle = w$

So, we can write this $w_1 x_1 + w_2 x_2$ part as vector dot product →

$$w_1 x_1 + w_2 x_2 = w \cdot x$$

So, we can rewrite the equation as,

$$z = w \cdot x.$$

This compact vector form generalizes the number of features →

3 features → $z = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$

2 features → $z = w_1 x_1 + w_2 x_2 + b$

1 feature → $z = w_1 x_1 + b$

∴ for n feature $z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$

$$= \left(\sum_{i=1}^{n} w_i x_i \right) + b$$

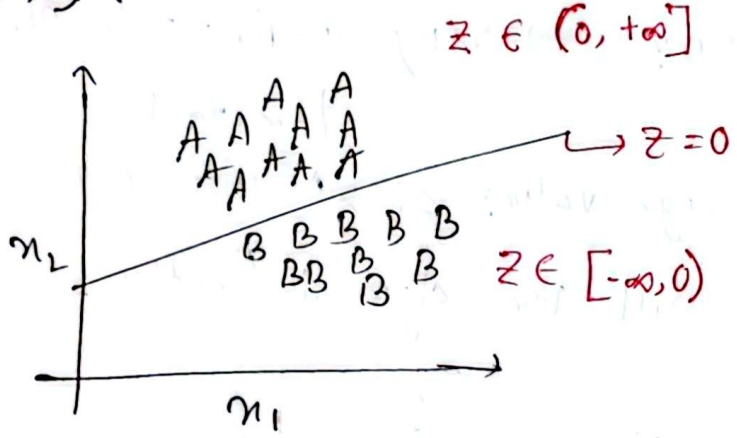
$$z = w \cdot X + b$$

Where $w = \langle w_1, w_2, w_3 \cdots w_n \rangle$

$$x = \langle x_1, x_2, \cdots x_n \rangle$$

Why do we need sigmoid function!

Now this linear equation z can take any value from $[-\infty, +\infty]$. We have already seen $z=0$ over then line. Above the line is $z>0$, and the range is $[0, +\infty]$. Below the line z can take $[-\infty, 0]$.



predict class = $\begin{cases} A/1, & \text{if } z \geq 0 \\ B/0, & \text{if } z \leq 0 \end{cases}$

But instead of taking large value we want to take the probabilistic values. For example, $P(y=A \mid <x_1, x_2>)$ $= P(y=A \mid x)$ = Probability of class A for a given value of feature x_1 and x_2.

It is Because logistic regression is probabilistic not just binary. the probability gives the confidence of a prediction to being in a class.

$P(y=A \mid x) = 0.993 \rightarrow$ very confident in class A

$P(y=A \mid x) = 0.55 \rightarrow$ slighty confident in class A

$P(y=A|x) = 0.007 \rightarrow$ very less confident in class A.
or very confident in class B.

↳ in contrast,

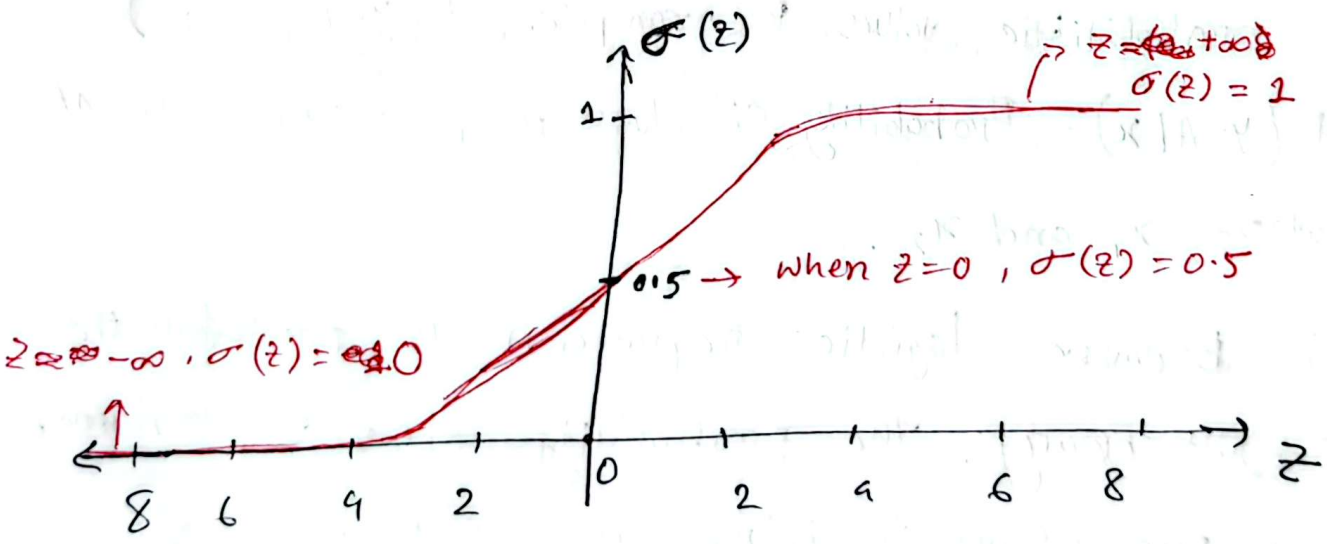
$$P(y=B) = 1 - P(y=A)$$
$$= 1 - 0.007$$
$$= 0.993 \rightarrow \text{very confident in class B.}$$

$[-\infty, +\infty]$

To map the large values of z into probabilistic value $[0, 1]$, we will use sigmoid function.

Sigmoid function $\sigma(z) = \dfrac{1}{1+e^{-z}}$ where, $z = W.x + b$.

Let us plot $\sigma(z)$ vs $z \rightarrow$



So, sigmoid function brings the $[-\infty, +\infty]$ values into the range of $[0, 1]$

Till now we have predicted class y,

$$\text{Prection } \hat{y} = \begin{cases} A, & \text{if } z \geq 0 \rightarrow \text{to } +\infty \\ B, & \text{if } z < 0 \rightarrow \text{to } -\infty \end{cases}$$

No, we will use the sigmoid function →

$$\text{Prediction } \hat{y} = \begin{cases} A, & \text{if } \sigma(z) \geq 0.5 \rightarrow \text{to } 1 \\ B, & \text{if } \sigma(z) < 0.5 \rightarrow \text{to } 0 \end{cases}$$

here $z = w.x + b$

↓

the previous linear line use as separator.

so, we can calculate, the probability of class A,

$P(y=A|x)$ with the sigmoid function.

$$\therefore \hat{y} = P(y=A|x) = \sigma(z) = \sigma(wx+b)$$

previously we have seen, probability of class B,

$$P(y=B|x) = 1 - P(y=A|x)$$

$$= 1 - \sigma(z)$$

$$= 1 - \sigma(w.x+b)$$

Let us use class 1 and class 0, instead of class A and class B.

$$\therefore \boxed{P(y=1|x) = \sigma(z)} = \sigma(w.x+b)$$

$$= \frac{1}{1+e^{-(w.x+b)}}$$

and $\boxed{P(y=0|x) = 1-\sigma(z)}$

$$= 1 - \sigma(w.x+b)$$

$$= 1 - \frac{1}{1+e^{-(w.x+b)}}$$

$$= \frac{1+e^{-(w.x+b)} - 1}{1+e^{-(w.x+b)}}$$

$$= \frac{e^{-(w.x+b)}}{1+e^{-(w.x+b)}}$$

$$\Rightarrow 1-\sigma(z) = \boxed{\frac{e^{-z}}{1+e^{-z}}} \quad \text{——————(1)}$$

$$= \frac{e^{-z} \times e^{z}}{(1+e^{-z}) \times e^{z}}$$ [multiplying nominator and denominator by e^z]

$$= \frac{1}{e^{z}+1} \Rightarrow [\because e^{-z} = \frac{1}{e^z}]$$

$$= \frac{\frac{1}{e^z} \times e^z}{e^z + \frac{1}{e^z} \times e^z} \quad [e^{-z} = \frac{1}{e^z}]$$

$$= \frac{1}{e^z+1}$$

$$= \frac{1}{1+e^{z}}$$

$$= \frac{1}{1+e^{-(-z)}} \quad [-(-z) = +z]$$

$$= \sigma(-z) \quad \left[\sigma(z) = \frac{1}{1+e^{-z}}\right]$$

$$\therefore \boxed{1 - \sigma(z) = \sigma(-z)}$$

Derivative of $\sigma(z)$:

$$\frac{d}{dz}\sigma(z) = \frac{d}{dz}\left(\frac{1}{1+e^{-z}}\right)$$

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

$$= -1 \times \left(1+e^{-z}\right)^{-2} \times \frac{d}{dz}\left(1+e^{-z}\right) \quad \text{[Chain rule]}$$

$$= -\left(1+e^{-z}\right)^{-2} \times \left(-e^{-z}\right)$$

$$= e^{-z} \times \left(1+e^{-z}\right)^{-2}$$

$$= \frac{e^{-z}}{\left(1+e^{-z}\right)^{2}}$$

$$= \frac{1}{\left(1+e^{-z}\right)} \times \frac{e^{-z}}{\left(1+e^{-z}\right)}$$

$$= \sigma(z) \times \left(1 - \sigma(z)\right) \rightarrow \text{From (1)}$$

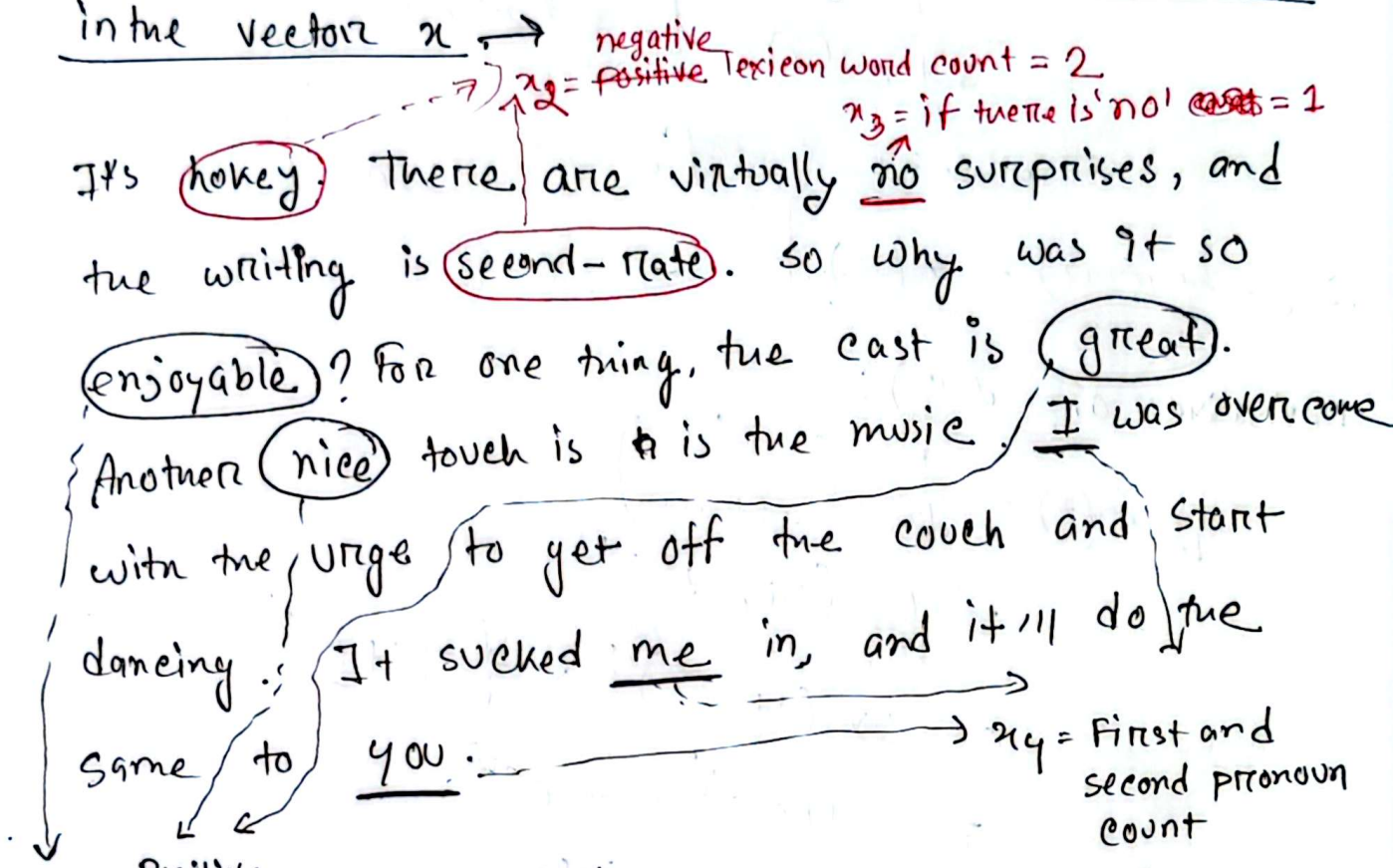
$\therefore$ derivative of sigmoid function

$$\boxed{\frac{d}{dz}\sigma(z) = \sigma(z)\left(1-\sigma(z)\right)} \quad \text{---(4)}$$

value.

Example:

(*) A sample text document showing extracted features in the vector x →



Find out if this paragraph/text carries positive emotion or negative emotion:

The weight vector,

$$W = \langle w_1, w_2, w_3, w_4, w_5, w_6 \rangle$$

$$= \langle 2.5, -5.0, -1.2, 0.5, 2.0, 0.7 \rangle$$

and bias $b = 0.1$.

Solution:

Features:	Definition	value
x_1	count(positive lexicon words $\in$ doc)	3
x_2	count(negative lexicon words $\in$ doc)	2
x_3	$\begin{cases} x_3 = 1 & \text{if "no" } \in \text{doc} \\ x_3 = 0 & \text{if otherwise} \end{cases}$	1 → there is "No" in the doc
x_4	count (1st and 2nd pronoun	3
x_5	$\begin{cases} x_5 = 1 & \text{if "!" } \in \text{Doc} \\ x_5 = 0 & \text{if otherwise} \end{cases}$	0 → there is no "!" sign
x_6	ln(word cont of doc)	ln(66) = 4.19 → there are 66 words in the doc.

So, input/feature vector

$$x = \langle x_1, x_2, x_3, x_4, x_5, x_6 \rangle$$
$$= \langle 3, 2, 1, 3, 0, 4.19 \rangle$$

Probability of text/paragraph/doc containing positive emotion, (Class 1 = positive emotion class)

$$P(+|x) = P(y=1|x) = \sigma(z)$$
$$= \sigma(w \cdot x + b)$$

$= \sigma(\langle 2.5, -5.0, -1.2, 0.5, 2.0, 0.7\rangle \cdot \langle 3, 2, 1, 3, 0, 4.19\rangle + 0.1)$

(w · dot multiplication · x; b)

$= \sigma((2.5 \times 3 - 5.0 \times 2 - 1.2 \times 1 + 0.5 \times 3 + 2.0 \times 0 + 0.7 \times 4.19) + 0.1)$

$= \sigma(0.833)$

$= \frac{1}{1+e^{-0.833}}$ $\quad \left[\sigma(z) = \frac{1}{1+e^{-z}}\right]$

$= 0.70$

∴ probability of text containing negative emotion

▷ class 0 = negative emotion class

$P(-|x) = P(y=0|x) = 1 - \sigma(w \cdot x + b)$

$= 1 - 0.70$

$= 0.30$.

∴ the text contains positive emotion.

In this example, the model / logistic regression classifier is given ^. So, we did not have to find the value of w and b. and w and.

Processing multiple examples in Matrix form:

Let ~~us~~, ~~we~~ assume,

we have a logistic regression model,

$$z = w_1 x_1 + w_2 x_2 + b \quad \text{and} \quad y = \sigma(z) = \frac{1}{1+e^{-z}}$$

where weight vector $w = \langle w_1, w_2 \rangle = \begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$

and bias $b = -1$.

as we have two features x_1 and x_2 if we have one test ~~case~~ case/example it will have both x_1 and x_2 value.

Let's assume test dataset →

	x_1	x_2
$x^{(1)}$	2	1
$x^{(2)}$	3	4
$x^{(3)}$	5	2

$x_1^{(3)}$ → 3rd test example's feature 1 value

$x_2^{(3)}$ → 3rd test example's feature 2 value.

∴ our input training matrix $x = \begin{bmatrix} 2 & 1 \\ 3 & 4 \\ 5 & 2 \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} \\ x_1^{(2)} & x_2^{(2)} \\ x_1^{(3)} & x_2^{(3)} \end{bmatrix}$

We know,

prediction $\hat{y} = \sigma(z)$

$= \sigma(w.x + b)$

weight vector, bias, input matrix

Let us calculate,

$$x.w = \begin{bmatrix} 2 & 1 \\ 3 & 4 \\ 5 & 2 \end{bmatrix}_{3\times2} * \begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix}_{2\times1}$$

$$= \begin{bmatrix} 2\times0.4 + 1\times0.6 \\ 3\times0.4 + 4\times0.6 \\ 5\times0.4 + 2\times0.6 \end{bmatrix} = \begin{bmatrix} 1.4 \\ 3.6 \\ 3.2 \end{bmatrix}_{3\times1}$$

$$\therefore z = x.w + b = \begin{bmatrix} 1.4 \\ 3.6 \\ 3.2 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0.4 \\ 2.6 \\ 2.2 \end{bmatrix}$$

bias vector

$$\therefore \hat{y} = \sigma(z) = \sigma\left(\begin{bmatrix} 0.4 \\ 2.6 \\ 2.2 \end{bmatrix}\right) = \begin{bmatrix} \sigma(0.4) \\ \sigma(2.6) \\ \sigma(2.2) \end{bmatrix}$$

$$= \begin{bmatrix} 0.5987 \\ 0.9309 \\ 0.9002 \end{bmatrix}$$

$\sigma(z) = \frac{1}{1+e^{-z}}$

So, in general →

if there are f features,

$x_1, x_2 \ldots\ldots x_f$

...

m test examples →

1st test example ← $x^{(1)}, x^{(2)}, x^{(3)} \ldots\ldots x^{(m)}$

input matrix $X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \ldots\ldots & x_f^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \ldots\ldots & x_f^{(2)} \\ \vdots & & & \\ x_1^{(m)} & x_2^{(m)} & \ldots\ldots & x_f^{(m)} \end{bmatrix}$

As there are f features, then will be f number of weights, as we know,

$$z = x_1 w_1 + x_2 w_2 \ldots\ldots x_f w_f + b$$

$$\therefore \quad W = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_f \end{bmatrix}$$

We need to predict class for m examples, so

$P(y^{(1)} = 1 \mid x^{(1)})$ = Probability of class 1 for training example 1

$$= \sigma(W \cdot x^{(1)} + b)$$

$$P(y^{(2)} = 1 \mid x^{(2)}) = \sigma(w \cdot x^{(2)} + b)$$

$$\vdots$$

$$P(y^{(m)} = 1 \mid x^{(m)}) = \sigma(w \cdot x^{(m)} + b)$$

So, prediction y will be a vector as well.

vector $\hat{y} = \sigma(x \cdot w + b)$

$$= \sigma\left(\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_f^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_f^{(2)} \\ \vdots & & & \\ x_1^{(m)} & x_2^{(m)} & \cdots & x_f^{(m)} \end{bmatrix}_{m \times f} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_f \end{bmatrix}_{f \times 1} + \begin{bmatrix} b \\ b \\ \vdots \\ b \end{bmatrix}_{m \times 1} \right)$$

$$\Rightarrow y = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y^{(m)} \end{bmatrix}_{m \times 1}$$

Finding the Logistic regression model!

To find the Logistic regression model,

$$\hat{y} = \sigma(z) = \sigma(w.x+b)$$

$$= \sigma(w_1x_1 + w_2x_2 \ldots\ldots w_nx_n + b)$$

We need to find the value of weight vector $\langle w_1, w_2 \ldots\ldots w_n \rangle$ and bias b. Then we can easily predict $\hat{y}$ vector.

In linear regression we have used Gradient Descent to optimize the weights ~~for~~. ~~we used~~ Here we will also use gradient descent to optimize the weight and bias. ~~using gradient descent.~~

We will start with a random value of weight w and b bias and iteratively update this parameters.

We know the parameter updates are done as below →

$$w_{1_{new}} = w_{1\,old} - \eta \frac{d(\text{Loss function})}{dw_1}$$

$$b_{new} = b_{old} - \eta \frac{d}{db}(\text{Loss function})$$

η learning rate.

So, we need to derive a loss function for logistic regression.

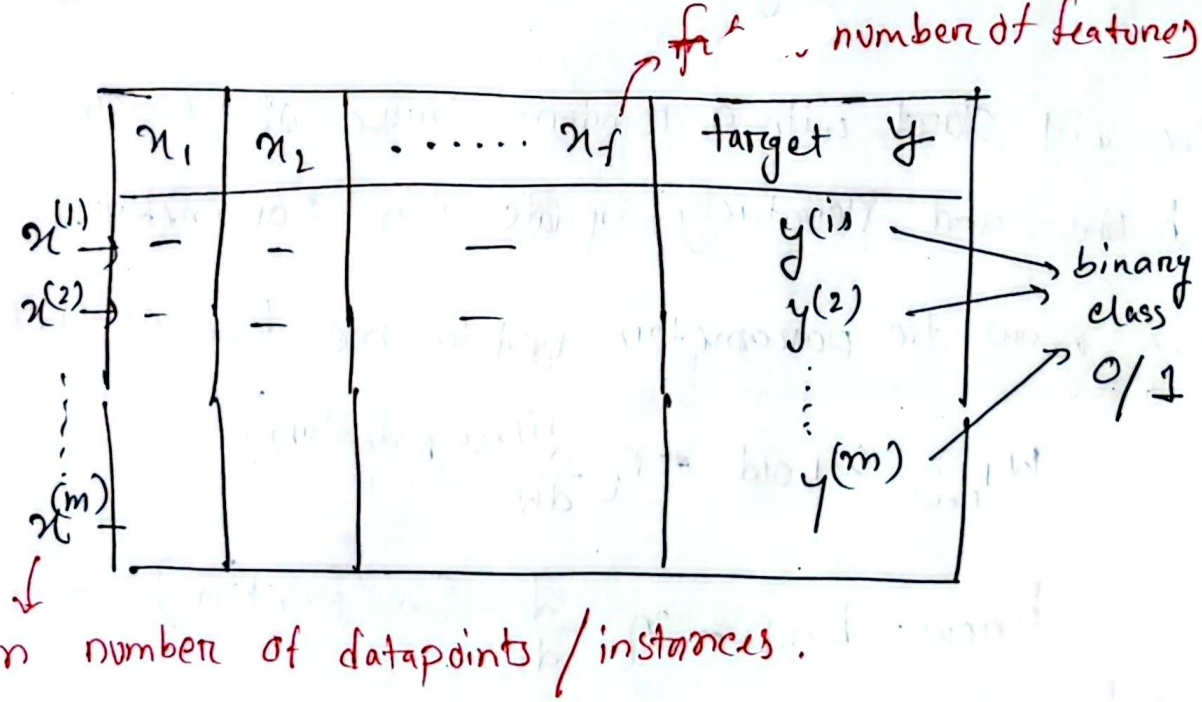
Here ^ we use a loss function Called in Logistic Regression

Cross Entropy

Deriving the Loss Function (BCE):

The loss function will be used to update the parameter weights and bias. And we need to do this while training our model with dataset.

Lets say we have a training dataset with f features and m data points like below →



For each data point in the dataset,

Probability of being in a class for feature vector x, $P(y = class | x)$

↓ actual $y^{(i)}$ from the dataset

for $y = 1$ class,

Prediction $P(y=1|x) = \sigma(z) = \hat{y}$ — (2)

and for $y=0$ class,

Prediction $P(y=0|x) = 1 - \sigma(z)$

$= 1 - \hat{y}$ — (3)

we can write a general equation for this two probabilities

Actual y from dataset

$$P(y|x) = (\hat{y})^{y} (1 - \hat{y})^{1-y}$$

Predicted y / $\sigma(z)$

here if $y = 1$, [if any data point has class labelled as 1]

then prediction $P(y=1|x) = (\hat{y})^{1} (1-\hat{y})^{1-1}$

$= \hat{y}$ [we get equation (2)]

and if $y=0$, [if any data point has class Labelled as 0]

then prediction $P(y=0|x) = (\hat{y})^0 (1-\hat{y})^{1-0}$

$= 1 \times 1 - \hat{y}$

$= 1 - \hat{y}$ → [we get equation (3)]

so, this equation, $P(y|x) = (\hat{y})^y (1-\hat{y})^{1-y}$ generalizes probability for both class.

As we are finding probability of $y=1$ and $y=0$ class, we will not get exactly 1 and 0 as prediction probability from $P(y=1|x)$ and $P(y=0|x)$.

→ if $y=1$ class gets correctly classified as 1, the prediction probability $P(y=1|x) \in [0.5, 1]$

→ if $y=0$ class gets correctly classified as 0, the prediction probability $P(y=0|x) \in [0.5, 1]$

But we want to maximize these probabilities near to 1. So,

→ if any datapoint belongs to class 1, we want $P(y=1|x) = 1$ [perfectly 1]

→ if any data point belongs to class 0, we want

$P(y=0|x) = 1$ [perfectly 1]

So, we want to maximize $P(y=1|x)$ and $P(y=0|x)$.

In general terms, we want to maximize,

$$P(y|x) = (\hat{y})^{y} (1-\hat{y})^{y-y}$$

In the dataset, there are m datapoints, we are training with m datapoints and trying to optimize w weights and b bias. So, for m datapoints the total prediction probability / Likelihood will be the joint probability of observing all training inputs.

So, total probability / likelihood for m datapoints,

$$P(y^{(1)}, y^{(2)} \ldots y^{(m)} | x^{(1)}, x^{(2)}, \ldots x^{(m)}).$$

↳ Just like before we want to maximize this likelihood as well.

If we assume all m datapoints are independent, We can write.

$$P(y^{(1)}, y^{(2)} \ldots y^{(m)} | x^{(1)}, x^{(2)} \ldots x^{(m)})$$

$$= P(y^{(1)}|x^{(1)}) \times P(y^{(2)}|x^{(2)}) \times \ldots P(y^{(m)}|x^{(m)})$$

$$= \prod_{i=1}^{m} P(y^{(i)} | x^{(i)})$$

$$= \prod_{i=1}^{m} (\hat{y}^{(i)})^{y^{(i)}} (1-\hat{y}^{(i)})^{1-y^{(i)}}$$

So, we want to maximize this product to get the accurate w an b.

This product hold both $P(y=1|x) = (\hat{y}^{(i)})$ and $P(y=0|x) = 1-\hat{y}^{(i)}$. It is because some data points are labelled as 1 and some as 0. We want to maximize both class probabilities to 1.

However it is very difficult to calculate this multiplication, when m is very large. to simplify we take "log" [while calculation use 'ln'] of this product, which gives us summation series in-place of product. We find **log-likelihood** after taking log.

$$\log \text{likelihood} = \sum_{i=1}^{m} \log \left[P(y^{(i)} | x^{(i)}) \right]$$

$$= \sum_{i=1}^{m} \log \left[(\hat{y}^{(i)})^{y^{(i)}} (1-\hat{y}^{(i)})^{1-y^{(i)}} \right]$$

$$= \sum_{i=1}^{m} \log (\hat{y}^{(i)})^{y^{(i)}} + \log (1-\hat{y}^{(i)})^{1-y^{(i)}}$$

$$= \sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log (1-\hat{y}^{(i)})$$

Now want to maximize this log-likelihood.

Now if we take negative of this log-likelihood, we will want to minimize it instead of maximizing. This minimizing function is our Cross Entropy Loss function.

minus makes it minimization function.

$$\text{Loss function (CE)} = -\sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log (1-\hat{y}^{(i)})$$

When class $y^{(i)} = 1$, we want to minimize

$$-\sum_{i=1}^{m} 1 \log \hat{y}^{(i)} + (1-1) \log (1-\hat{y}^{(i)})$$

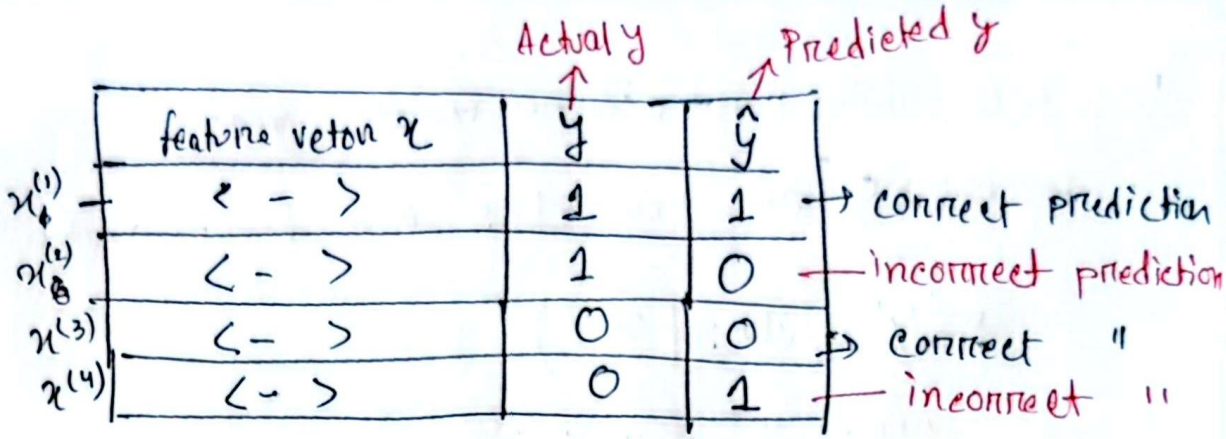
$$= -\sum_{i=1}^{m} \log \hat{y}^{(i)}$$

When data point's belongs to $y^{(i)} = 0$ class, we want to minimize,

$$-\sum_{i=1}^{m} 0 \times \log \hat{y}^{(i)} + (1-0) \log (1-\hat{y}^{(i)})$$

$$= -\sum_{i=1}^{m} \log (1-\hat{y}^{(i)})$$

Now let's check if this Loss function give us error while predicting correctly and incorrectly.



Let us assume, we have 4 data points in this dataset. and we have run our logistic regression model over this. and found some predicted $\hat{y}$. Let us check if our loss function CE gives us Error correctly.

For datapoint $x^{(1)} \rightarrow y^{(1)} = 1$ and $\hat{y}^{(1)} = 1$,

$$\text{Loss (CE)} = -y^{(i)} \log \hat{y}^{(i)} - (1 - y^{(i)}) \log (1 - \hat{y}^{(i)})$$

$$= -1 \log 1 - (1-1) \log (1-1)$$

$$= -1 \log 1 = 0$$ [correct prection Loss=0]

For datapoint $x^{(2)} \rightarrow y^{(2)} = 1$ and $\hat{y}^{(2)} = 0$

$$\text{Loss (CE)} = -y^{(2)} \log \hat{y}^{(2)} - (1 - y^{(2)}) \log (1 - \hat{y}^{(2)})$$

$$= -1 \log 0 - (1-1) \log (1-0)$$

$$= -\log 0 = \text{undefind}$$ [in a rangeits -infinity]

[Incorrect prediction Loss=∞]

For datapoint $x^{(3)} \rightarrow y^{(3)} = 0, \hat{y}^{(3)} = 0$

$$\text{Loss(CE)} = -y^{(3)} \log \hat{y}^{(3)} - (1-y^{(3)}) \log(1-\hat{y}^{(3)})$$

$$= -0 \log 0 - (1-0) \log(1-0)$$

$$= -\log 1 = 0$$

[correct prediction Loss = 0]

For datapoint $x^{(4)} \rightarrow y^{(4)} = 0, \hat{y}^{(4)} = 1$

$$\text{Loss (CE)} = -y^{(4)} \log \hat{y}^{(4)} - (1-y^{(4)}) \log (1-\hat{y}^{(4)})$$

$$= -0 \log 1 - (1-0) \log (1-1)$$

$$= -\log 0$$

= undefind [in a range it is -Infinity]

[incorrect prediction Loss = infinity]

So, we can use this loss function for the gradient descent process to update the weights and bias parameters.

For updating, we need derivative of loss function as below,

$$\frac{\delta \text{ Loss function}}{\delta w_i} \rightarrow \text{every weight } w_i$$

$$\frac{\delta \text{ Loss function}}{\delta b} \rightarrow \text{bias.}$$

Derivative of Loss function:

Loss function $L_{CE} = -\sum_{i=1}^{n} \cdot y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log (1-\hat{y}^{(i)})$

Let us consider it for one datapoint then,

$$L_{CE} = -y \log \hat{y} - (1-y) \log (1-\hat{y}).$$

Let us assume, $z = w_1 x_1 + w_2 x_2 + b$ [For a problem]

derivative interms of $w_1 \rightarrow$

$$\frac{\delta L_{CE}}{\delta w_1} = \frac{\delta L_{CE}}{\delta \hat{y}} \times \frac{\delta \hat{y}}{\delta z} \times \frac{\delta z}{\delta w_1}$$

↳ We do this because we don't have w_1 in L_{CE}.

So, we ~~derivate it~~ differenciate it with respect to $\hat{y}$. So, we need to differenciate $\hat{y}$ now.

$$\hat{y} = \sigma(z) = \frac{1}{1+e^{-z}}$$

So, we can differenciate it with respect to z. taen finally z has w_1 in it so, we differenciate it with respect to w_1. This is the chain rule.

Let us do the differenciation part by part.

$$\frac{\delta L_{CE}}{\delta \hat{y}} = \frac{\delta}{\delta \hat{y}}\left[-y \log \hat{y} + -(1-y)\log(1-\hat{y})\right]$$

$$= -y \times \frac{1}{\hat{y}} - \frac{1-y}{1-\hat{y}} \times \frac{\delta}{\delta \hat{y}} \times (1-\hat{y})$$

$$= -\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}} \times (-1)$$

$$= -\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}}$$

$$= \frac{-y + \cancel{y\hat{y}} + \hat{y} - \cancel{y\hat{y}}}{\hat{y}(1-\hat{y})}$$

$$= \frac{(\hat{y} - y)}{\hat{y}(1-\hat{y})} \quad \longrightarrow (5)$$

$$\frac{\delta \hat{y}}{dz} = \frac{\delta}{dz}\sigma(z) = \sigma(z)\,(1-\sigma(z)) \quad [\text{from } (4)]$$

$$\therefore \frac{\delta \hat{y}}{dz} = \hat{y}(1-\hat{y}) \quad [\hat{y} = \sigma(z)]$$

$\longrightarrow (6)$

$$\frac{\delta z}{\delta w_1} = \frac{d}{\delta w_1}(w_1 x_1 + w_2 x_2 + b)$$

$$= x_1$$

$$\therefore \frac{\delta L_{CE}}{\delta w_1} = \frac{\delta L_{CE}}{\delta \hat{y}} \times \frac{\delta \hat{y}}{dz} \times \frac{\cancel{dz}}{\cancel{\delta w_1}}$$

$$= \frac{\hat{y} - y}{\cancel{\hat{y}(1-\hat{y})}} \times \cancel{\hat{y}(1-\hat{y})} \times x_1$$

$$\therefore \frac{\delta L_{CE}}{\delta w_1} = (\hat{y} - y) x_1$$

Similarly,

$$\frac{\delta L_{CE}}{\delta w_2} = \boxed{\frac{\delta L_{CE}}{\delta \hat{y}} \times \frac{\delta \hat{y}}{\delta z}} \times \frac{\delta z}{\delta w_2}$$

(5) (6)

→ need to find this

we already know this part

$$\therefore \frac{\delta z}{\delta w_2} = \frac{\delta}{\delta w_2}(w_1 x_1 + w_2 x_2 + b)$$

$$= x_2$$

$$\therefore \frac{\delta L_{CE}}{\delta w_2} = \frac{\delta L_{CE}}{\delta \hat{y}} \times \frac{\delta \hat{y}}{\delta z} \times \frac{\delta z}{w_2}$$

$$= \frac{\hat{y} - y}{\hat{y}(1-\hat{y})} \times \hat{y}(1-\hat{y}) \times x_2$$

$$= (\hat{y} - y) x_2$$

→ we can see the derivative with respect to w_2 is same ~~except~~ as w_1 except the x term.

We can generalize derivative of L_{CE} with respect to weight w_i as →

$$\frac{\delta L_{CE}}{\delta w_i} = (\hat{y} - y) x_i$$

Now, derivative with respect to b:

$$\frac{\delta L_{EE}}{\delta b} = \boxed{\frac{\delta L_{EE}}{\delta \hat{y}} \times \frac{\delta \hat{y}}{\delta z}} \times \frac{\delta z}{\delta b}$$

↳ this part is,

$$\frac{\hat{y} - y}{\hat{y}(1-\hat{y})} \times \hat{y}(1-\hat{y})$$

$$= \boxed{\hat{y} - y}$$

$$\therefore \frac{\delta z}{\delta b} = \frac{\delta}{\delta b}(w_1 x_1 + w_2 x_2 + b) = 1.$$

$$\therefore \frac{\delta L_{EE}}{\delta b} = \frac{\delta L_{EE}}{\delta \hat{y}} \times \frac{\delta \hat{y}}{\delta z} \times \frac{\delta z}{\delta b}$$

$$= (\hat{y} - y) \times 1$$

$$= \hat{y} - y$$

Example-1:

x_1	x_2	y
3	2	1

Let's assume initial weight and bias ~~in~~ are are set to zero. learning rate is 0.1. update the weights and bias [for one cycle] using gradient descent in one epoch.

Solution:

for two feature x_1, x_2

$$z = w_1 x_1 + w_2 x_2 + b \quad \text{and} \quad \hat{y} = \sigma(z) = \frac{1}{1+e^{-z}}$$

So, we have to update two weights and b. Given,

$$\theta^0 = \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

we know, updates will be done as follows,

$$w_i(\text{new}) = w_i(\text{old}) - \eta \frac{\delta L_{CE}}{\delta w_i}$$

$$b(\text{new}) = b(\text{old}) - \eta \frac{\delta L_{CE}}{\delta b}$$

[Here old is θ^0 new will be θ']

∴ Gradient of Loss function with respect to w and b,

$$\nabla_{w,b} L = \begin{bmatrix} \frac{\delta L_{CE}}{\delta w_1} \\ \frac{\delta L_{CE}}{dw_2} \\ \frac{\delta L_{CE}}{\delta b} \end{bmatrix} = \begin{bmatrix} (\hat{y}-y)x_1 \\ (\hat{y}-y)x_2 \\ (\hat{y}-y) \end{bmatrix}$$

$\hat{y} = \sigma(z)$

$= \sigma(w_1 x_1 + w_2 x_2 + b)$

$$= \begin{bmatrix} [\sigma(w_1 x_1 + w_2 x_2 + b) - y] x_1 \\ [\sigma(w_1 x_1 + w_2 x_2 + b) - y] x_2 \\ \sigma(w_1 x_1 + w_2 x_2 + b) - y \end{bmatrix}$$

$$= \begin{bmatrix} [\sigma(0) - 1] \times 3 \\ [\sigma(0) - 1] \times 2 \\ \sigma(0) - 1 \end{bmatrix} \quad \left[z = w_1 x_1 + w_2 x_2 + b \right.$$
$$= 0 \times 3 + 0 \times 2 + 0$$
$$= 0 \quad]$$

$$= \begin{bmatrix} -0.5 \times 3 \\ -0.5 \times 2 \\ -0.5 \end{bmatrix} \qquad \sigma(0) = \frac{1}{1+e^{-0}} = 0.5$$

$$= \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix}$$

Now we have the gradients, we can compute the new updated value of the weights and bias

$$\theta' = \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} - \eta \times \begin{bmatrix} \frac{\delta L_{CE}}{\delta w_1} \\ \frac{\delta L_{CE}}{\delta w_2} \\ \frac{\delta L_{CE}}{\delta b} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} - 0.1 \times \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 0.15 \\ 0.1 \\ 0.05 \end{bmatrix}$$

we can also calculate / update the weights individually instead of doing it in vector / matrix form.

Example-2

Find the linear classifier for an OR gate using Logistic Regression.

The linear equation for a two input OR gate is $z = W.X = w_1x_1 + w_2x_2 + b$. Let us assume,

$w_1 = 0.5$, $w_2 = 0.75$ and $b = -1.25$. $\eta = 0.2$

initial $z = 0.5x_1 + 0.75x_2 - 1.25$

Do two cycle of gradient descent and update w_1, w_2 and b.

Solution:

We know the truth table of OR gate →

	x_1	x_2	y
① ←	0	0	0
② ←	0	1	1
③ ←	1	0	1
④ ←	1	1	1

features

actual target.

1st cycle →

Datapoint ① $x_1 = 0$, $x_2 = 0$ and $y = 0$.

$\therefore z = w_1x_1 + w_2x_2 + b$

$= 0.5 \times 0 + 0.75 \times 0 - 1.25$

$= -1.25$.

and $\hat{y} = \sigma(z) = \sigma(-1.25) = 0.22 < 0.5$

↓

Class is 0

So, our prediction/ hypothesis $h_w(x) = 0$

∴ Actual class $y = 0$ and predicted class $h_w = 0$.

So we will not update w_1, w_2, and b.

Datapoint - ② → $x_1 = 0$, $x_2 = 1$ and $y = 1$

$$z = 0.5x_1 + 0.75x_2 - 1.25$$

$$= 0.5 \times 0 + 0.75 \times 1 - 1.25 = -0.5$$

and $\hat{y} = \sigma(z) = \sigma(-0.5) = 0.38 \rightarrow h_w = 0$ as $\hat{y} < 0.5$

but actual class $y = 1$. So we need to update w_1 w_2 and b.

$$\therefore w_1 = w_1(\text{old}) - \eta \frac{\delta L_{CE}}{\delta w_1} = w_1(\text{old}) - \eta (\hat{y} - y) x_1$$

$$= 0.5 - 0.2 \times (0.38 - 1) \times 0$$

$$= 0.5$$

$$\therefore w_2 = w_2(\text{old}) - \eta (\hat{y} - y) x_2 = 0.75 - 0.2 \times (0.38 - 1) \times 0$$

$$= 0.87$$

$$\therefore b = b(\text{old}) - \eta \times \frac{\delta L_{CE}}{\delta b} = b(\text{old}) - \eta \times (\hat{y} - y)$$

$$= -1.25 - 0.2 \times (0.38 - 1)$$

$$= -1.13.$$

$w_1(\text{new})$ $w_2(\text{new})$ $b(\text{new})$

So, new $z = 0.5x_1 + 0.87x_2 - 1.13$

Datapoint → (3) $x_1 = 1, x_2 = 0, y = 1$

$$z = 0.5 \times 1 + 0.87 x_2 - 1.13 = -0.63$$

$$\hat{y} = \sigma(z) = \sigma(-0.63) = 0.35$$

$$\hookrightarrow h_w(x) = 0 \quad \text{as } \hat{y} < 0.5$$

Actual class $y = 1$ and predicted class $h_w(x) = 0$

So, we need to update weights and bias.

$$w_1(new) = w_1(old) - \eta \times (\hat{y} - y) x_1 = 0.5 - 0.2(0.35 - 1) \times 1$$
$$= 0.63$$

$$w_2(new) = w_2(old) - \eta \times (\hat{y} - y) x_2 = 0.87 - 0.2(0.35 - 1) \times 0$$
$$= 0.87$$

$$b(new) = b(old) - \eta \times (\hat{y} - y) = -1.13 - 0.2(0.35 - 1) = -1.0$$

$\therefore$ So, $z = 0.63 x_1 + 0.87 x_2 - 1.0$

Datapoint (4) → $x_1 = 1$, $x_2 = 1$ and $y = 1$

$$z = 0.63 \times 1 + 0.87 \times 1 - 1.0 = 0.5$$

$$\hat{y} = \sigma(z) = \sigma(0.5) = 0.62 > 0.5$$

$\therefore h_w(x) = 1$.

Actual class $y = 1$ and predicted class $h_w(x) = 1$ are same.

, no need to update the weights and bias.

After first cycle,

$w_1 = 0.63$

$w_2 = 0.87$

$b = -1.0$

And $z = 0.63x_1 + 0.87x_2 - 1.0$

Similarly we can do the ~~same~~ second cycle for datapoint ① → ④ Again.

We will keep doing the cycles until all samples are being correctly classified by the logistic regression model.

Also, this process is called Stochastic Gradient Descent where we check misclassification for every sample and update the parameters whenever needed.