Classification: Logistic Regression

Logistic Regression

The **logistic regression**, is a type of supervised learning algorithm used for **binary classification tasks**. Logistic regression is called that even though it is a classification algorithm because it is **based on a regression model that predicts the probability of an event occurring**.

The algorithm works by **modeling the relationship between the input features and the probability of a binary outcome using a logistic function**, also known as a **sigmoid function**. The sigmoid function takes any input and **outputs a value between 0 and 1**, **representing the probability of the positive outcome**.

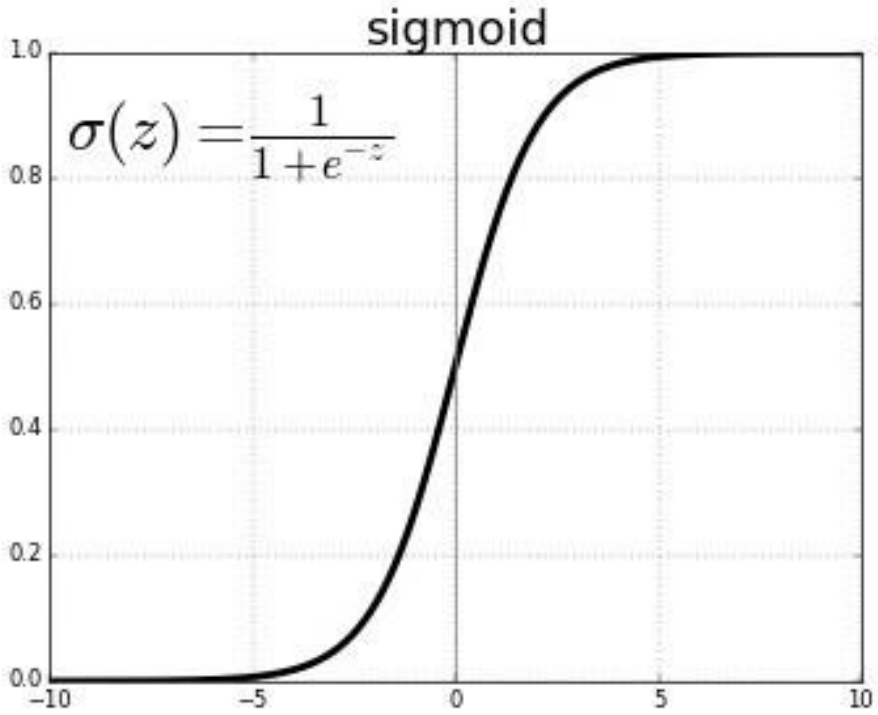
This output is obtained by applying a logistic or sigmoid function to the output of a linear regression model. **The logistic function transforms the output of the linear regression model into a probability value that is bounded between 0 and 1**, making it suitable for classification tasks. Therefore, while logistic regression is technically a classification algorithm, it uses a regression model to predict the probability of an event occurring, which explains why it is called logistic regression.

Logistic Function

A Sigmoid function is a mathematical function which has a characteristic S-shaped curve. There are numbers of common sigmoid functions, such as the **logistic function**, the **hyperbolic tangent**, and the **arctangent**. **Logistic function** is a type of sigmoid function that squishes values between 0 and 1.

$$\lim_{z \to -\infty} logistic(z) = 0 \qquad \lim_{z \to \infty} logistic(z) = 1$$

One important characteristic of the logistic function is that it has two horizontal asymptotes at 0 and 1. If z goes to infinity, logistic function will become 1 and if z goes to negative infinity, the function will become 0.



Logistic Regression – Mathematical Formula

For binary classification, we have two target classes we want to predict. Let's refer to them as positive (y=1) and negative (y=0) classes. When we combine linear regression and logistic function, we get the logistic regression equation:

$$\hat{p}(y=1) = logistic(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n)$$

where:

$\hat{p}(y=1)$: predicted probability of belonging to the positive class

θ_0 : intercept (aka bias term)

$\theta_1,\ \theta_2,\ ...,\ \theta_n$: coefficients (aka weights)

$x_1,\ x_2,\ ...,\ x_n$: features

$$\hat{p}(y=1) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n)}}$$

Logistic regression predicts the probability of a record belonging to the positive class given features.

Logistic Regression – Mathematical Formula

Finding the probability of belonging to the negative class:

$$\hat{p}(y = 0) = 1 - \hat{p}(y = 1)$$

Once we have probability values, it's easy to convert them to a predicted class. Using a threshold of 0.5 (i.e. probability of 50%), we can convert the probability values to classes:

$$\hat{y} = \begin{cases} 0 & \text{if } \hat{p}(y = 1) < 0.5 \\ 1 & \text{if } \hat{p}(y = 1) \geq 0.5 \end{cases} \longrightarrow \hat{y} = \begin{cases} 0 & \text{if } \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n < 0 \\ 1 & \text{if } \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n \geq 0 \end{cases}$$

where:
$\hat{y}$: predicted class

We can say that the predicted class is 0 for negative values of z, otherwise, the predicted class is 1

Logistic Regression – Example

Here's a simple example on a small toy dataset with a binary target and two features:

```
from sklearn.linear_model import LogisticRegression
# Create sample data
df = pd.DataFrame({'x1': [1, 2, 2, 3, 2, 4, 3, 4, 3, 4],
                   'x2': [2, 3, 4, 2, 1, 3, 5, 2, 3, 6],
                   'y': [0, 0, 0, 0, 0, 1, 1, 1, 1, 1]})
# Train a logistic regression
target = 'y'
features = ['x1', 'x2']
model = LogisticRegression()
model.fit(df[features], df[target])
# Predict
df['z'] = model.intercept_[0] + df['x1']*model.coef_[0][0] + df['x2']*model.coef_[0][1]
df['p_hat'] = model.predict_proba(df[features])[:,1]
df['y_hat'] = model.predict(df[features])
df
```

	x1	x2	y	z	p_hat	y_hat
0	1	2	0	-2.978560	0.048404	0
1	2	3	0	-1.114241	0.247081	0
2	2	4	0	-0.557769	0.364064	0
3	3	2	0	-0.362868	0.410266	0
4	2	1	0	-2.227187	0.097336	0
5	4	3	1	1.501451	0.817791	1
6	3	5	1	1.306550	0.786935	1
7	4	2	1	0.944979	0.720104	1
8	3	3	1	0.193605	0.548251	1
9	4	6	1	3.170869	0.959723	1

We can see that for negative values of z, probability values are under 0.5 and for positive values of z, the probability values are above 0.5.

Decision Boundary

In logistic regression, the decision boundary is a boundary that separates the positive class from the negative class in the feature space. The decision boundary is determined by the parameters of the logistic regression model, which are learned during the training process.

Decision boundary for logistic regression is given by:

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n = 0$$

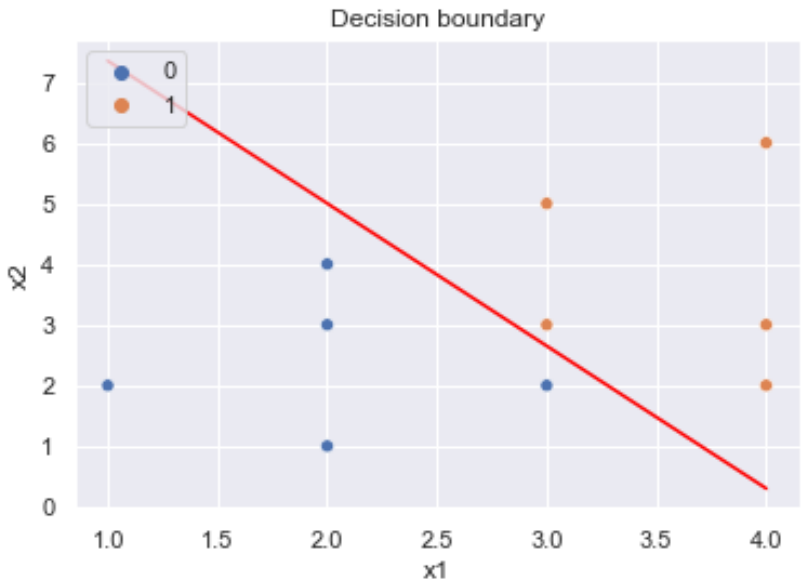
In our example, since we only have two features, the equation becomes:

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$$

Let's plot x_1 on the horizontal axis and x_2 on the vertical axis along with the decision boundary. Rearrange the above equation such that x_2 is expressed in terms of x_1

$$\theta_2 x_2 = -(\theta_0 + \theta_1 x_1)$$

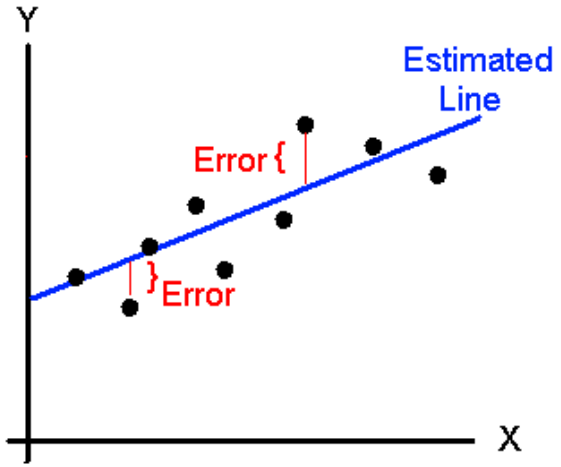
$$x_2 = -\frac{\theta_0 + \theta_1 x_1}{\theta_2}$$



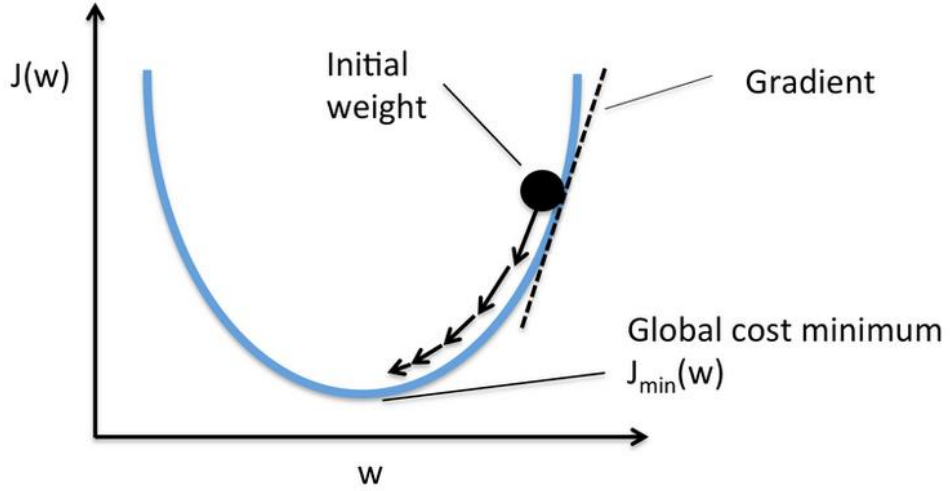
Cost Function

In ML, cost functions (you may also see this referred to as *loss* or *error)* are used to estimate how badly models are performing. ***A cost function is a measure of how wrong the model is in terms of its ability to estimate the relationship between X and y.*** This is typically expressed as a difference or distance between the predicted value and the actual value.

In **linear regression**, we can use MSE as its cost function



$$J = \frac{\sum_{i=1}^{n} (\hat{Y}_i - Y_i)^2}{n}$$



Gradient Descent is an algorithm that is used to optimize the cost function or the error of the model. It is used to find the minimum value of error possible in your model. This can be done by the help of 'gradient'

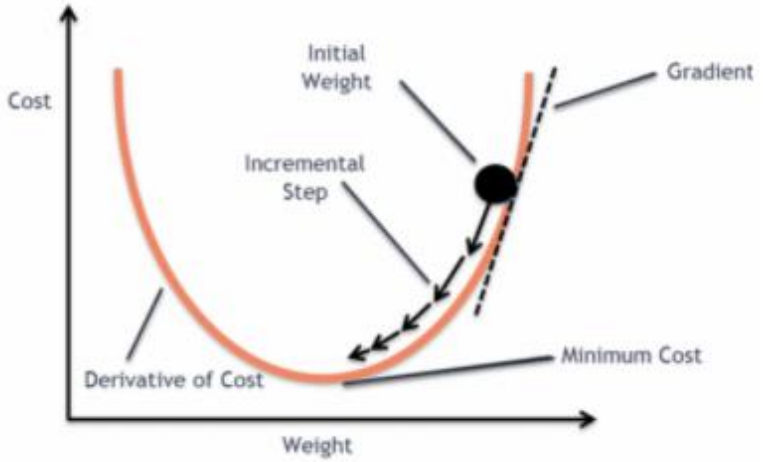
Cost Function

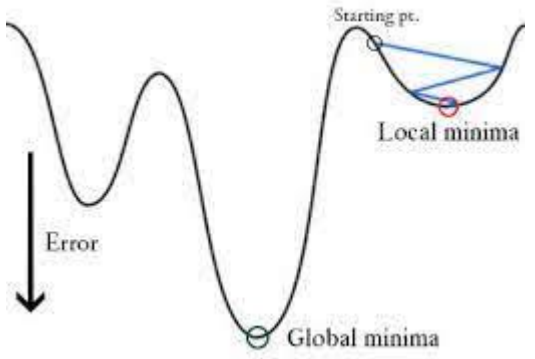
Gradient descent will converge into global minimum only if the function is **convex**.

$$J = \frac{\sum_{i=1}^{n} (\hat{Y}_i - Y_i)^2}{n}$$

In Logistic Regression Ŷi is a nonlinear function ($\hat{Y}=1/1+ e^{-z}$), if we put this in the above cost function it will give a non-convex function. There will exist many local optima on which optimization algorithm might prematurely converge before finding the true minimum.

Cost function in Linear Regression:





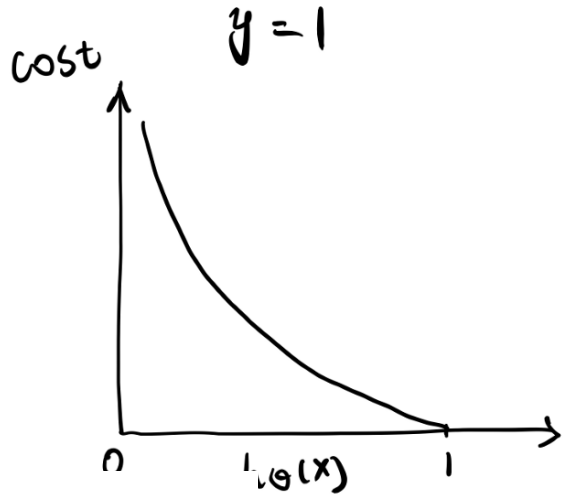
Cost Function

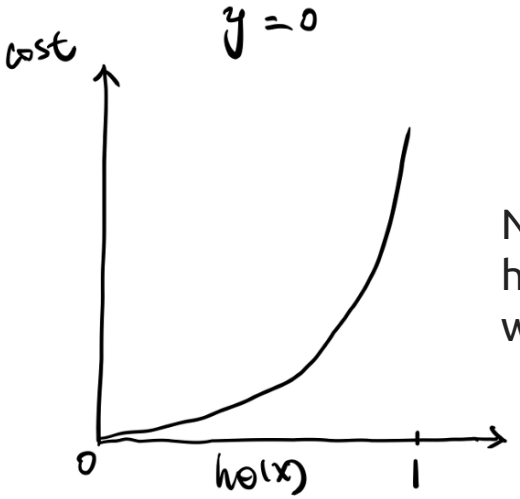
Using the Maximum Likelihood Estimator from statistics, we can obtain the cost function which produces a convex space friendly for optimization. This function is known as the binary cross-entropy loss or Log loss, which measures the difference between the predicted output of the logistic regression model and the actual output. It is given by the formula:

$$\text{Log loss} = \frac{1}{N} \sum_{i=1}^{N} -\left(y_i * \log(p_i) + (1-y_i) * \log(1-p_i) \right)$$

Here, pi is the probability of class 1, and (1-pi) is the probability of class 0. You will notice that the first half of the equation is 0 when y=0 (when observation belongs to class 0) whereas second half of the equation is 0 when y=1 (when observation belongs to class 1).

Now we know how to measure the error for a given set of parameters, the next question is how to find the best parameters that minimizes the error. This is where optimization algorithm such as gradient descent comes into play





Note:
$h_\theta(x)$ is the same as p_i we have calculated

Advantages of Logistic Regression

- It makes no assumptions about distributions of classes in feature space.
- It is very fast at classifying unknown records.
- Good accuracy for many simple data sets and it performs well when the dataset is linearly separable
- Simplicity and transparency. Logistic Regression is just a bit more involved than Linear Regression, which is one of the simplest predictive algorithms out there. It is also transparent, meaning we can see through the process and understand what is going on at each step, contrasted to the more complex ones (e.g. SVM, Deep Neural Nets) that are much harder to track
- Giving probabilistic output. Some other algorithms (e.g. Decision Tree) only produce the most seemingly matched label for each data sample, meanwhile, Logistic Regression gives a decimal number ranging from 0 to 1, which can be interpreted as the probability of the sample to be in the Positive Class. With that, we know how confident the prediction is, leading to a wider usage and deeper analysis.

Disadvantages of Logistic Regression

- If the number of observations is lesser than the number of features, Logistic Regression should not be used, otherwise, it may lead to overfitting.
- The major limitation of Logistic Regression is the assumption of linearity between the dependent variable and the independent variables.
- Non-linear problems can’t be solved with logistic regression because it has a linear decision surface. Linearly separable data is rarely found in real-world scenarios.
- Logistic Regression requires average or no multicollinearity between independent variables.
- It is tough to obtain complex relationships using logistic regression. More powerful and compact algorithms such as Neural Networks can easily outperform this algorithm.

Thanks!

This is the end of Classification: Logistic Regression,

see you in the next topic.