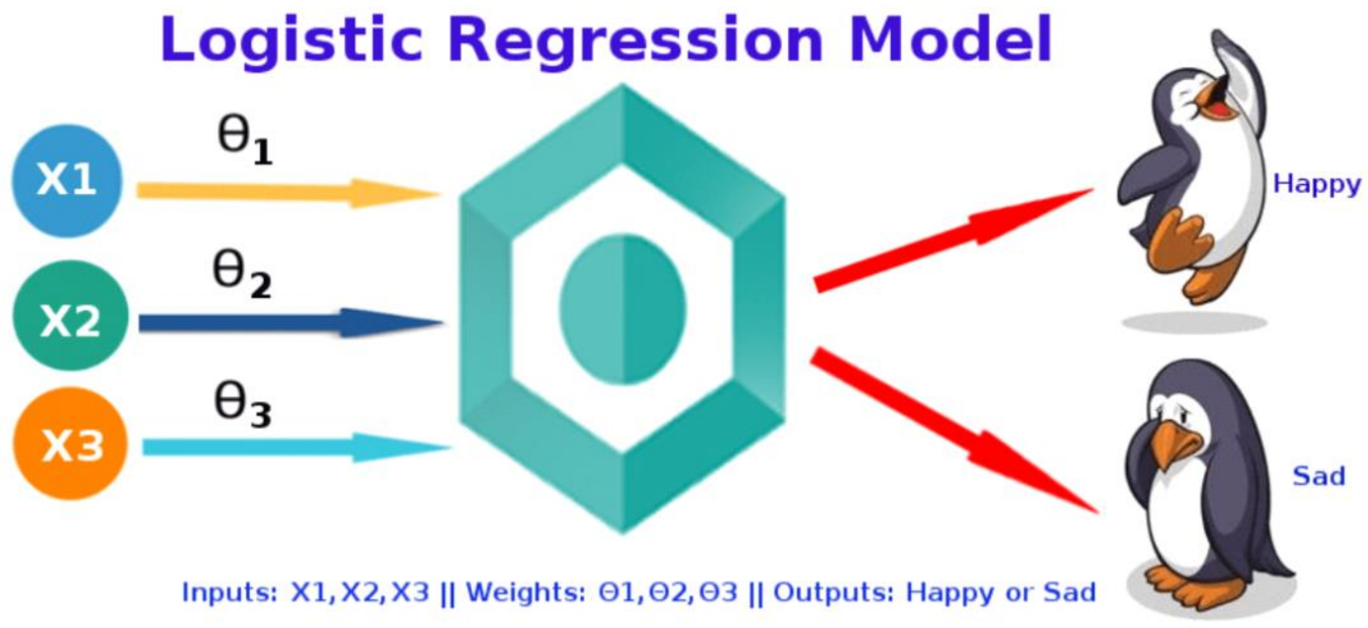
Logistic Regression

Sunday, September 2, 2018 2:47 AM


Logistic Regression Model
X1
X2
X3
θ_1
θ_2
θ_3
Happy
Sad
Inputs: X1,X2,X3 || Weights: Θ1,Θ2,Θ3 || Outputs: Happy or Sad

Logistic Regression was used in the biological sciences in early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable(target) is categorical.

For example,

- To predict whether an email is spam (1) or (0)
- Whether the tumor is malignant (1) or not (0)

Consider a scenario where we need to classify whether an email is spam or not. If we use linear regression for this problem, there is a need for setting up a threshold based on which classification can be done. Say if the actual class is malignant, predicted continuous value 0.4 and the threshold value is 0.5, the data point will be classified as not malignant which can lead to serious consequence in real time.

From this example, it can be inferred that linear regression is not suitable for classification problem. Linear regression is unbounded, and this brings logistic regression into picture. Their value strictly ranges from 0 to 1.

Simple Logistic Regression

Code @ Jupiter Note Book (Please refer that code here)

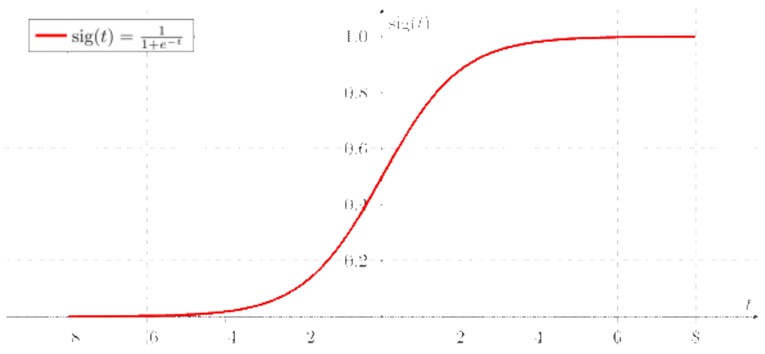
Model

Output = 0 or 1

Hypothesis => Z = WX + B

hΘ(x) = sigmoid (Z)

Sigmoid Function



If ‘Z’ goes to infinity, Y(predicted) will become 1 and if ‘Z’ goes to negative infinity, Y(predicted) will become 0.

Analysis of the hypothesis

The output from the hypothesis is the estimated probability. This is used to infer how confident can predicted value be actual value when given an input X. Consider the below example,

X = [x0 x1] = [1 IP-Address]

Based on the x1 value, let’s say we obtained the estimated probability to be 0.8. This tells that there is 80% chance that an email will be spam.

Mathematically this can be written as,

$h_\Theta(x) = P\ (Y=1|X;\ \text{theta})$

Probability that Y=1 given X which is parameterized by ‘theta’.

$P\ (Y=1|X;\ \text{theta}) + P\ (Y=0|X;\ \text{theta}) = 1$

$P\ (Y=0|X;\ \text{theta}) = 1 - P\ (Y=1|X;\ \text{theta})$

This justifies the name ‘logistic regression’. Data is fit into linear regression model, which then be acted upon by a logistic function predicting the target categorical dependent variable.

Types of Logistic Regression

1. Binary Logistic Regression

The categorical response has only two 2 possible outcomes. Example: Spam or Not

2. Multinomial Logistic Regression

Three or more categories without ordering. Example: Predicting which food is preferred more (Veg, Non-Veg, Vegan)

3. Ordinal Logistic Regression

Three or more categories with ordering. Example: Movie rating from 1 to 5

Decision Boundary

To predict which class a data belongs, a threshold can be set. Based upon this threshold, the obtained estimated probability is classified into classes.

Say, if predicted_value ≥ 0.5, then classify email as spam else as not spam.

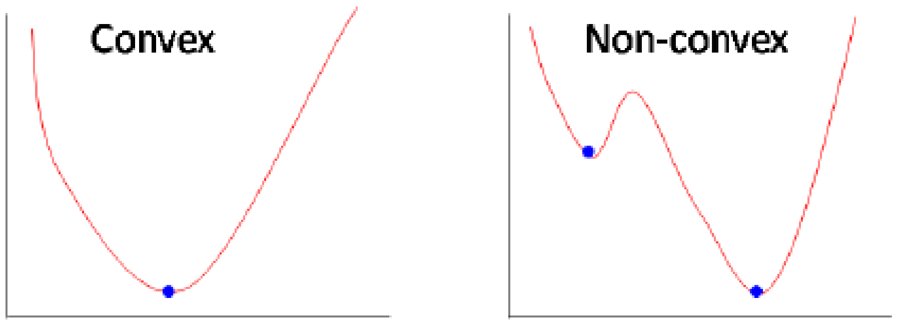
Decision boundary can be linear or non-linear. Polynomial order can be increased to get complex decision boundary.

Cost Function

$$\text{Cost}(h_\Theta(x), Y(\text{actual})) = -\log(h_\Theta(x)) \text{ if } y=1$$
$$-\log(1 - h_\Theta(x)) \text{ if } y=0$$

Why cost function which has been used for linear cannot be used for logistic?

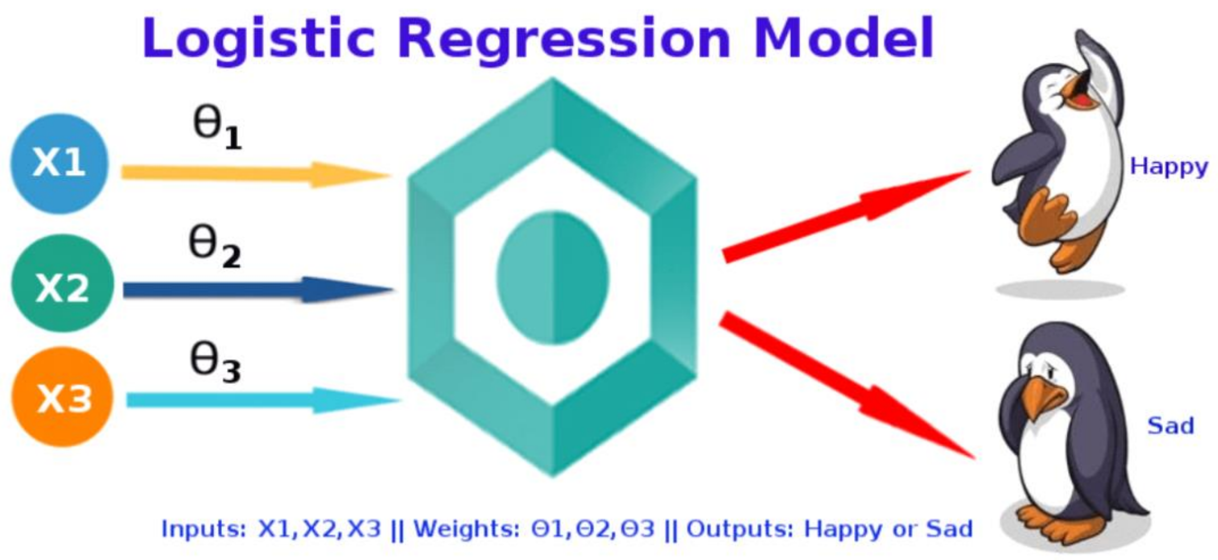
Linear regression uses mean squared error as its cost function. If this is used for logistic regression, then it will be a non-convex function of parameters (theta). Gradient descent will converge into global minimum only if the function is convex.



This implementation is for binary logistic regression. For data with more than 2 classes, softmax regression has to be used.

HOW THE LOGISTIC REGRESSION MODEL WORKS

Sunday, September 2, 2018 4:14 AM


Logistic Regression Model
X1
X2
X3
θ_1
θ_2
θ_3
Happy
Sad
Inputs: X1,X2,X3 || Weights: ϴ1,ϴ2,ϴ3 || Outputs: Happy or Sad

How the Logistic Regression Model Works in Machine Learning

In this article, we are going to learn how the logistic regression model works in machine learning. The logistic regression model is one member of the supervised classification algorithm family. The building block concepts of logistic regression can be helpful in **deep learning** while building the neural networks.

Logistic regression classifier is more like a **linear classifier** which uses the calculated logits (score) to predict the target class. If you are not familiar with the concepts of the logits, don't frighten. We are going to learn each and every block of logistic regression by the end of this post.

Before we begin, let's check out the table of contents.

Table of Contents

What is logistic regression?

Below is the most accurate and well-defined definition of logistic regression from Wikipedia.

> *"Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function"* (Wikipedia)

Let's understand the above logistic regression model definition word by word. What logistic regression model will do is, It uses a black box function to understand the relation between the categorical **dependent variable**and the **independent variables**. This black box function is popularly known as the **Softmax funciton**.

Dependent and Independent Variables

The dependent and the independent variables are the same which we were discussed in the building simple linear regression model. Just to give you a glance. The dependent variable is the target class variable we are going to predict. However, the independent variables are the features or attributes we are going to use to predict the target class.

Suppose the shop owner would like to predict the customer who entered into the shop will buy the **Macbook or Not**. To predict whether the customer will buy the MacBook or not. The shop owner will observe the customer features like.

- **Gender:**
  - Probabilities wise male will high chances of purchasing a MacBook than females.
- **Age:**
  - Kids won't purchase MacBook.

The shop owner will use the above, similar kind of features to predict the **likelihood occurrence** of the event (Will buy the Macbook or not.)

In the mathematical side, the logistic regression model will pass the likelihood occurrences through the **logistic function** to predict the corresponding target class. This popular logistic function is the Softmax function. We are going to learn about the softmax function in the coming sections of this post.

Before that. Let's quickly see few examples to understand the sentence **likelihood occurrence of an event**.

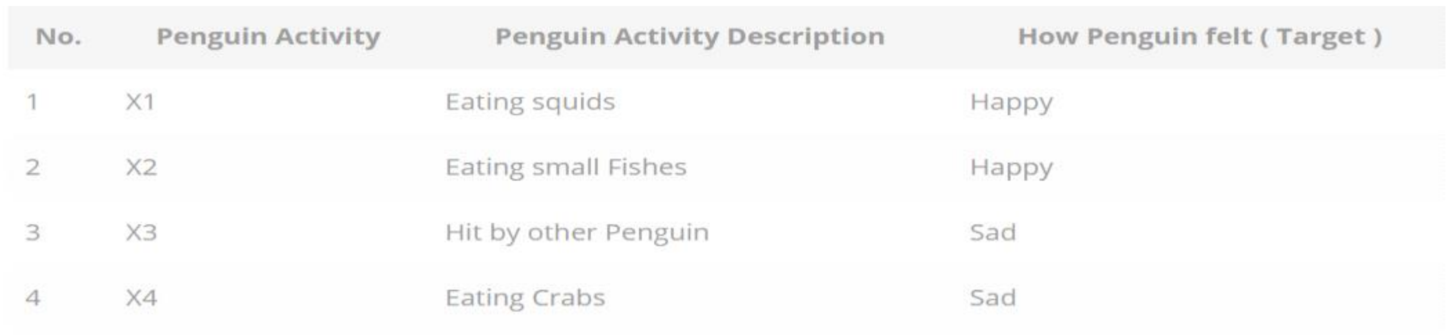
Examples of likelihood occurrence of an event

- How likely a customer will buy iPod having iPhone in his/her pocket.
- How likely Argentina team will win when Lionel Andrés Messi in rest.
- What is the probability to get into best university by scoring decent marks in mathematics, physics?
- What is the probability to get a kiss from your girlfriend when you gifted her favorite dress on behalf of your birthday?

Hope the above examples gives you the better idea about the sentence **predict the likelihood occurrence of an event.**
Before drive into the underline mathematical concept of logistic regression. let's brush up the logistic regression understanding level with an example.

Logistic Regression Model Example

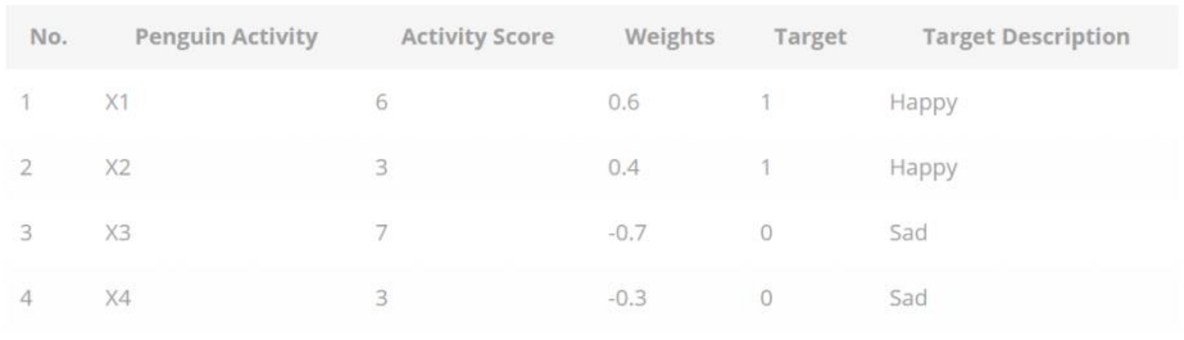
Suppose **HPenguin** wants to know, how likely it will be happy based on its daily activities. If the penguin wants to build a logistic regression model to predict it happiness based on its daily activities. The penguin needs both the happy and sad activities. In machine learning terminology these activities are known as the Input parameters (**features**).
So let's create a table which contains penguin activities and the result of that activity like **happy** or **sad**.



No.	Penguin Activity	Penguin Activity Description	How Penguin felt (Target)
1	X1	Eating squids	Happy
2	X2	Eating small Fishes	Happy
3	X3	Hit by other Penguin	Sad
4	X4	Eating Crabs	Sad

Penguin is going to use the above activities (features) to train the logistic regression model. Later the trained logistic regression model will predict how the penguin is feeling for the new penguin activities.
As it's not possible to use the above categorical data table to build the logistic regression. The above activities data table needs to convert into activities score, weights, and the corresponding target.

Penguin Activity Data table



No.	Penguin Activity	Activity Score	Weights	Target	Target Description
1	X1	6	0.6	1	Happy
2	X2	3	0.4	1	Happy
3	X3	7	-0.7	0	Sad
4	X4	3	-0.3	0	Sad

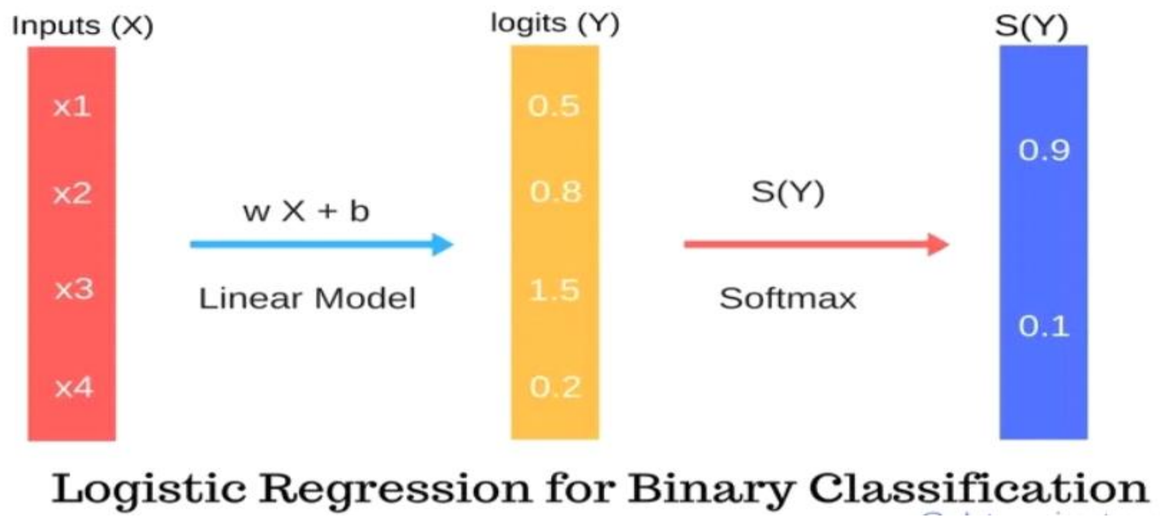
The updated dataset looks like this. Before we drive further let's understand more about the above data table.

- **Penguin Activity:**
  - The activities penguin do daily like eating small fishes, eating crabs .. etc
- **Activity Score:**
  - The activity score is more like the numerical equivalent to the penguin activity. For eating squids activity, the corresponding activity score is 6 and likewise, for other activities the scores are 3, 7, 3.
- **Weights:**
  - The Weights more like the weightages corresponding to the particular target.
  - Suppose for the activity X1 we have the weight as 0.6.
  - It means to say if the penguin performs the activity X1 the model is 60% confident to say the penguin will be happy.
  - If you observe the weights for the target class **happy** are positive, and the weights for the target class **sad** are negative.
  - This is because the problem we are addressing a binary classification. Will talk about the binary classification in the next sections of this post.
- **Target:**
  - The target is just the binary values. The value 1 represents the target **Happy**, and the value 0 represents the target **Sad**.

Now we know the activity score for each activity and the corresponding weights. To predict how the penguin will feel given the activity we just need to **multiply the activity score and the corresponding weight** to get the score. The calculated score is also known as the **logits**

The logit (Score) will pass into the softmax function to get the probability for **each target class**. In our case, if we pass the logit through the softmax function will get the probability for the target **happy class** and for the target **sad class**. Later we can consider the target class with **high probability** as the predicted target class for the given activity.
In fact, we can predict whether the Penguin is feeling happy or sad with the calculated **logits (Score) in this case**. As we were given the positive

weights for the target class happy and the negative weights for the target class sad. We can say If the Logit is **greater than 0** the target class is happy and if the logit is **less than 0** the target class is sad.



Binary classification with Logistic Regression model

In the Penguin example, we pre-assigned the **activity scores and the weights** for the logistic regression model. This won't be the simple while modeling the logistic regression model for real word problems. The weights will be calculated over the training data set. Using the calculated the weights the Logits will be computed. Till here the model is similar to the linear regression model.

Note: The Logits in the image were just for example, and **not the calculated** logits from the penguin example

The calculated Logits (score) for the linear regression model will pass through the softmax function. The softmax function will return the probabilities for each target class. The high probability target class will be the predicted target class.

The target classes In the Penguin example, are having two target classes (Happy and Sad). If we are using the logistic regression model for predicting the binary targets like yes or no, 1 or 0. which is known as the Binary classification.

Till now we talk about the **softmax function** as a black box which takes the calculated scores and returns the probabilities. Now let's learn how the softmax function calculates the probabilities. Next, we are going to implement the simple softmax function to calculate the probabilities for given Logits (Scores)

What is Softmax function?

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{k} e^{z_k}} \quad for\ j = 1, \ldots, k$$

Softmax function is the popular function to calculate the **probabilities** of the events. The other mathematical advantages of using the softmax function are the output range. Softmax function output values are always in the **range of (0, 1)**. The sum of the output values will always **equal to the 1**. The Softmax is also known as the **normalized exponential** function.
The above is the softmax formula. Which takes each value (Logits) and find the probability. The numerator the e-power values of the Logit and the denominator calculates the sum of the e-power values of all the Logits.
Softmax function used in:

- Naive Bayes Classifier
- Multinomial Logistic Classifier
- Deep Learning (While building Neural networks)

Before we implementing the softmax function, Let's study the special cases of the Softmax function inputs.

The special cases of softmax function input

The two special cases we need to consider about the Softmax function output, If we do the below modifications to the Softmax function inputs.

- Multiplying the Softmax function inputs (Multiplying the Logits with any value)
- Dividing the Softmax function inputs (Dividing the Logits with any value)

Multiplying the Softmax function inputs:
If we multiply the Softmax function inputs, the inputs values will become large. So the logistic regression will be more confident (High Probability

value) about the predicted target class.

Dividing the Softmax function inputs:
If we divide the Softmax function inputs, the inputs values will become small. So the Logistic regression model will be not confident (Less Probability value) of the predicted target class.
Enough of the theoretical concept of the Softmax function. Let's do the fun part (Coding).

Implementing the softmax function in Python

Let's implement a softmax function which takes the logits in list or array and returns the softmax function outputs in a list.

Softmax function is the popular function to calculate the **probabilities** of the events. The other mathematical advantages of using the softmax function are the output range. Softmax function output values are always in the **range of (0, 1)**. The sum of the output values will always **equal to the 1**. The Softmax is also known as the **normalized exponential** function.
The above is the softmax formula. Which takes each value (Logits) and find the probability. The numerator the e-power values of the Logit and the denominator calculates the sum of the e-power values of all the Logits.
Softmax function used in:

- Naive Bayes Classifier
- Multinomial Logistic Classifier
- Deep Learning (While building Neural networks)

Before we implementing the softmax function, Let's study the special cases of the Softmax function inputs.

The special cases of softmax function input

The two special cases we need to consider about the Softmax function output, If we do the below modifications to the Softmax function inputs.

- Multiplying the Softmax function inputs (Multiplying the Logits with any value)
- Dividing the Softmax function inputs (Dividing the Logits with any value)

Multiplying the Softmax function inputs:
If we multiply the Softmax function inputs, the inputs values will become large. So the logistic regression will be more confident (High Probability value) about the predicted target class.
Dividing the Softmax function inputs:
If we divide the Softmax function inputs, the inputs values will become small. So the Logistic regression model will be not confident (Less Probability value) of the predicted target class.
Enough of the theoretical concept of the Softmax function. Let's do the fun part (Coding).
Implementing the softmax function in Python
Let's implement a softmax function which takes the logits in list or array and returns the softmax function outputs in a list.

```
# Required Python Packages
import numpy as np
def softmax(scores):
    """
    Calculate the softmax for the given scores
    :param scores:
    :return:
    """
    return np.exp(scores) / np.sum(np.exp(scores), axis=0)

scores = [8, 5, 2]

if __name__ == "__main__":
    logits = [8, 5, 2]
    print "Softmax :: ", softmax(logits)
```

To implement the softmax function we just replicated the Softmax formula.

- The input to the softmax function is the logits in a list or array.
- The numerator computes the exponential value of each Logit in the array.
- The denominator calculates the sum of all exponential values.
- Finally, we return the ratio of the numerator and the denominator values.

Script Output:

Softmax function output

Python

```
Softmax ::  [ 0.95033021  0.04731416  0.00235563]
```

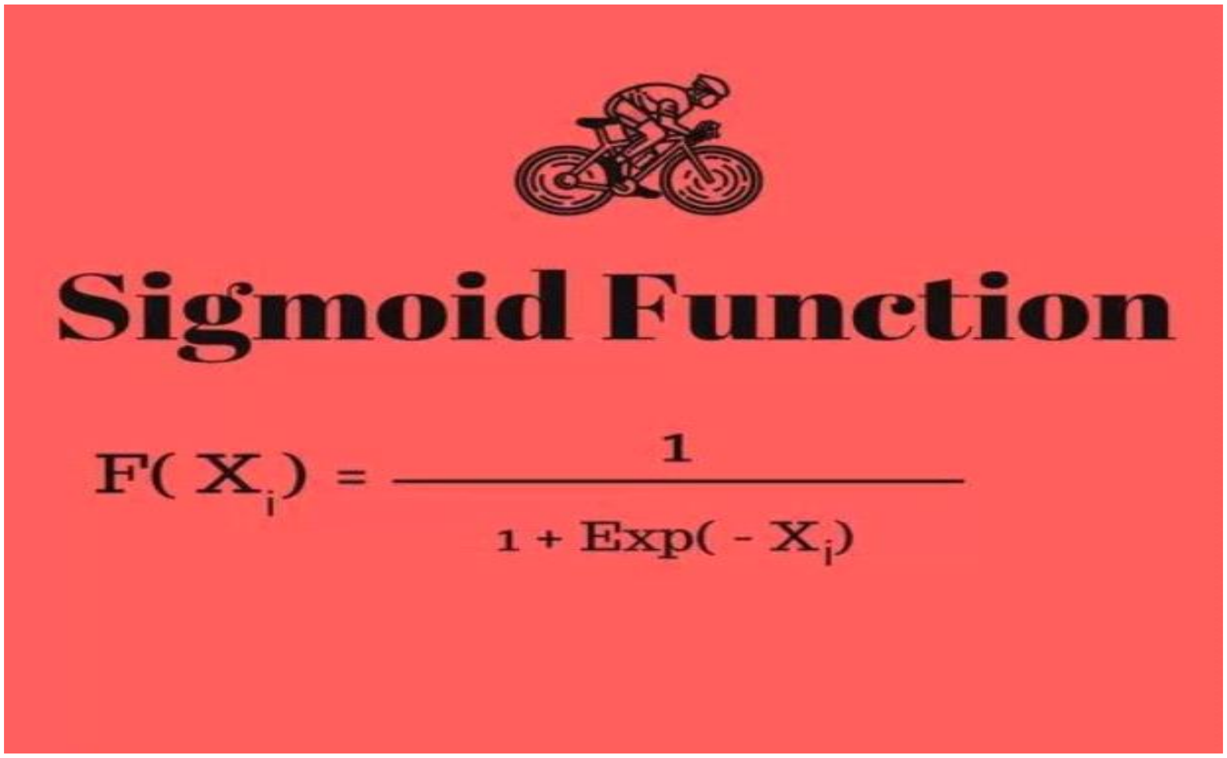
DIFFERENCE BETWEEN SOFTMAX FUNCTION AND SIGMOID FUNCTION

Sunday, September 2, 2018 4:44 AM

Softmax Function Vs Sigmoid Function

While learning the logistic regression concepts, the primary confusion will be on the functions used for calculating the probabilities. As the calculated probabilities are used to predict the target class in logistic regression model. The two principal functions we frequently hear are Softmax and Sigmoid function.
Even though both the functions are same at the **functional level.** (Helping to predict the target class) many noticeable mathematical differences are playing the vital role in using the functions in deep learning and other fields of areas.
So In this article, we were going to learn more about the fundamental differences between these two function and the usages.
Before we begin, let's quickly look at the table of contents.
Table of Contents:

What is Sigmoid Function?



$$F(X_i) = \frac{1}{1 + Exp(-X_i)}$$

In mathematical definition way of saying the sigmoid function take any range real number and returns the output value which falls in the range of **0 to 1.** Based on the convention we can expect the output value in the range of **-1 to 1.**

The sigmoid function produces the curve which will be in the Shape **"S."** These curves used in the statistics too. With the cumulative distribution function (The output will range from 0 to 1)

Properties of Sigmoid Function

- The sigmoid function returns a real-valued output.
- The first derivative of the sigmoid function will be non-negative or non-positive.
  - **Non-Negative:** If a number is greater than or equal to zero.
  - **Non-Positive:** If a number is less than or equal to Zero.

Sigmoid Function Usage

- The Sigmoid function used for **binary classification** in logistic regression model.
- While creating artificial neurons sigmoid function used as the **activation function**.
- In statistics, the **sigmoid function graphs** are common as a cumulative distribution function.

Implementing Sigmoid Function In Python

Now let's implement the sigmoid function in Python

```
# Required Python Package
import numpy as np
def sigmoid(inputs):
    """
    Calculate the sigmoid for the give inputs (array)
    :param inputs:
    :return:
    """
    sigmoid_scores = [1 / float(1 + np.exp(- x)) for x in inputs]
    return sigmoid_scores

sigmoid_inputs = [2, 3, 5, 6]
print "Sigmoid Function Output :: {}".format(sigmoid(sigmoid_inputs))
```

The above is the implementation of the sigmoid function.

- The function will take a list of values as an input parameter.
- For each element/value in the list will consider as an input for the sigmoid function and will calculate the output value.
- The code **1 / float(1 + np.exp(- x))** is the fucuntion is used for calcualting the sigmoid scores.
- Next, we take a list sigmiod_inputs having the values 2,3,5,6 as an input the function we implemented to get the sigmoid scores.

Script Output

Python

```
Sigmoid Function Output :: [0.8807970779778823, 0.9525741268224334, 0.9933071490757153, 0.9975273768433653]
```

Creating Sigmoid Function Graph

Now let's use the above function to create the graph to understand the nature of the sigmoid function.

- We are going to pass a list which contains numbers in the range 0 to 21.
- Will compute the sigmoid scores for the list we passed.
- Then we will use the outputs values to visualize the graph.

```python
# Required Python Packages
import numpy as np
import matplotlib.pyplot as plt


def sigmoid(inputs):
    """
    Calculate the sigmoid for the give inputs (array)
    :param inputs:
    :return:
    """
    sigmoid_scores = [1 / float(1 + np.exp(- x)) for x in inputs]
    return sigmoid_scores


def line_graph(x, y, x_title, y_title):
    """
    Draw line graph with x and y values
    :param x:
    :param y:
    :param x_title:
    :param y_title:
    :return:
    """
    plt.plot(x, y)
    plt.xlabel(x_title)
    plt.ylabel(y_title)
    plt.show()


graph_x = range(0, 21)
graph_y = sigmoid(graph_x)

print "Graph X readings: {}".format(graph_x)
print "Graph Y readings: {}".format(graph_y)

line_graph(graph_x, graph_y, "Inputs", "Sigmoid Scores")
```

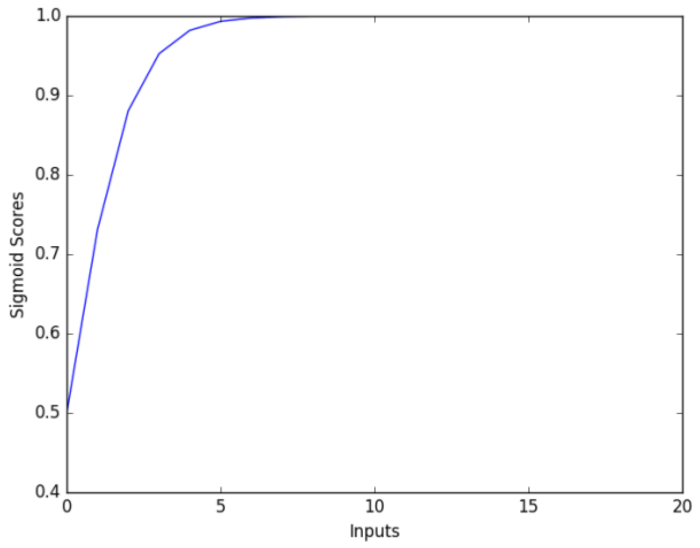
- Creating a **graph_x** list which contains the numbers in the range of 0 to 21.
- Next, in the **graph_y** list, we are storing the calculated **sigmoid scores** for the given graph_x inputs.
- Calling the **line_graph** function, which takes the x, y, and titles of the graph to create the line graph.

Script Output

```
Graph X readings: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

Graph Y readings: [0.5, 0.7310585786300049, 0.8807970779778823, 0.9525741268224334, 0.9820137900379085, 0.9933071490757153,
0.9975273768433653, 0.9990889488055994, 0.9996646498695336, 0.9998766054240137, 0.9999546021312976, 0.999983298578152,
0.9999938558253978, 0.999997739675702, 0.9999991684719722, 0.999999694097773, 0.9999998874648379, 0.9999999586006244,
0.9999999847700205, 0.9999999943972036, 0.9999999979388463]
```

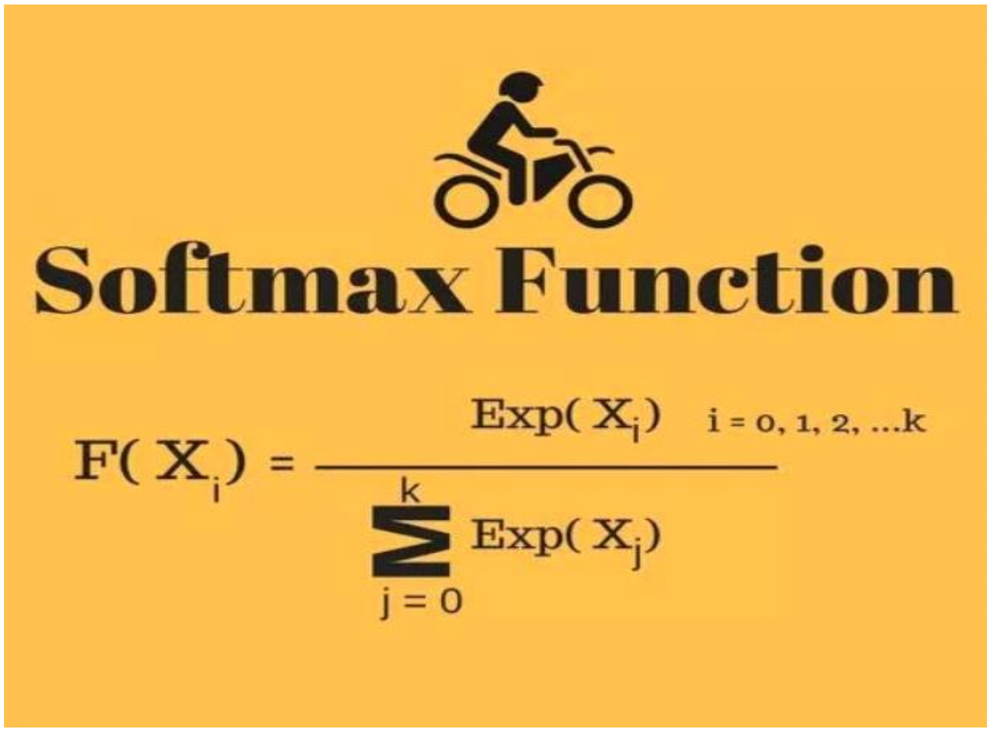
Graph

On successfully running the above code the below image will appear on your screen. If the above code failed in your system. Check the machine learning packages setup.



From the above graph, we can observe that with the increase in the input value the sigmoid score increase **till 1**. The values which are touching at the top of the graph are the values in the range of **0.9 to 0.99**

What is Softmax Function?



$$F(X_i) = \frac{\text{Exp}(X_i)}{\sum_{j=0}^{k} \text{Exp}(X_j)} \quad i = 0, 1, 2, \ldots k$$

Softmax function calculates the probabilities distribution of the event over ‘n’ different events. In general way of saying, this function will calculate the probabilities of each target class over all possible target classes. Later the calculated probabilities will be helpful for determining the target class for the given inputs.
The main advantage of using Softmax is the output probabilities range. The range will **0 to 1**, and the sum of all the probabilities will be **equal to one**. If the softmax function used for multi-classification model it returns the probabilities of each class and the target class will have the high probability.
The formula computes the **exponential (e-power)** of the given input value and the **sum of exponential values**of all the values in the inputs. Then the ratio of the exponential of the input value and the sum of exponential values is the output of the softmax function.

Properties of Softmax Function

Below are the few properties of softmax function.

- The calculated probabilities will be in the range of 0 to 1.
- The sum of all the probabilities is equals to 1.

Softmax Function Usage

- Used in multiple classification logistic regression model.
- In building neural networks softmax functions used in different layer level.

```
# Required Python Package
import numpy as np


def softmax(inputs):
    """
    Calculate the softmax for the give inputs (array)
    :param inputs:
    :return:
    """
    return np.exp(inputs) / float(sum(np.exp(inputs)))


softmax_inputs = [2, 3, 5, 6]
print "Softmax Function Output :: {}".format(softmax(softmax_inputs))
```

Script Output

Softmax Functin Output

```
Softmax Function Output :: [ 0.01275478  0.03467109  0.25618664  0.69638749]
```

If we observe the function output for the input value 6 we are getting the high probabilities. This is what we can expect from the softmax function. Later in classification task, we can use the high probability value for predicting the target class for the given input features.

Creating Softmax Function Graph

Now let's use the implemented Softmax function to create the graph to understand the behavior of this function.

- We are going to create a list which contains values in the range of 0 to 21.
- Next, we are going to pass this list to calculate the scores from the implemented function.
- To create the graph we are going to use the list and the estimated scores.

Code:

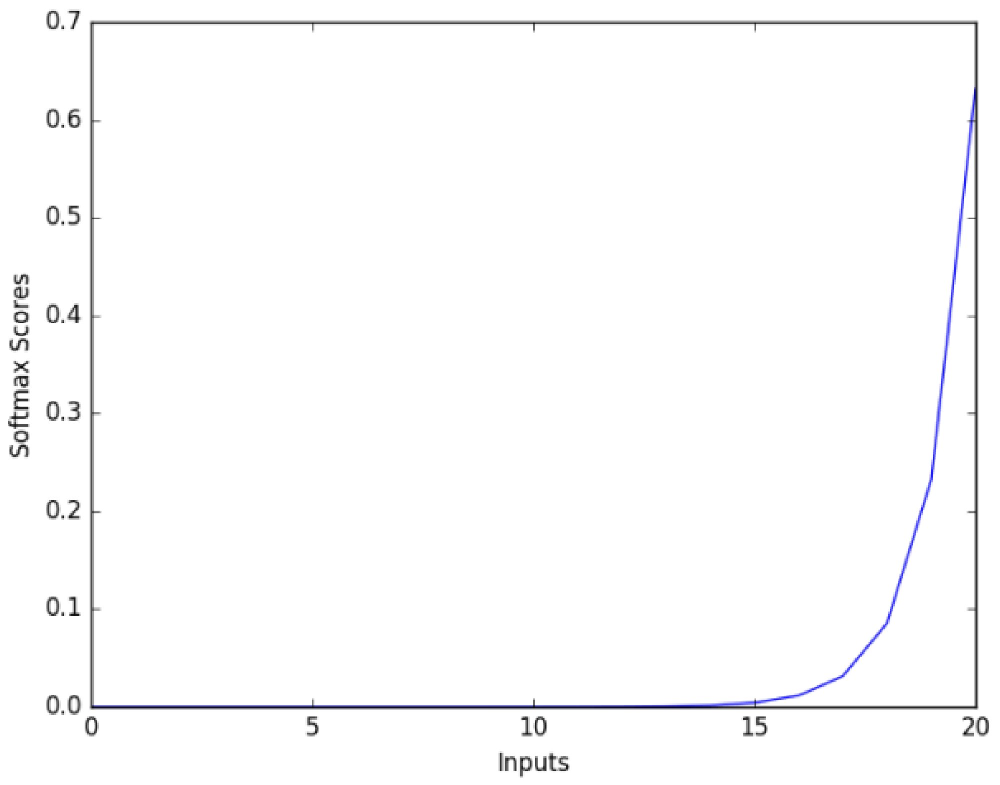
```
# Required Python Packages
import numpy as np
import matplotlib.pyplot as plt


def softmax(inputs):
    """
    Calculate the softmax for the give inputs (array)
    :param inputs:
    :return:
    """
    return np.exp(inputs) / float(sum(np.exp(inputs)))


def line_graph(x, y, x_title, y_title):
    """
    Draw line graph with x and y values
    :param x:
    :param y:
    :param x_title:
    :param y_title:
```

```
    :return:
    """
    plt.plot(x, y)
    plt.xlabel(x_title)
    plt.ylabel(y_title)
    plt.show()


graph_x = range(0, 21)
graph_y = softmax(graph_x)

print "Graph X readings: {}".format(graph_x)
print "Graph Y readings: {}".format(graph_y)

line_graph(graph_x, graph_y, "Inputs", "Softmax Scores")
```

Script Output

```
Graph X readings: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
Graph Y readings: [ 1.30289758e-09 3.54164282e-09 9.62718331e-09 2.61693975e-08 7.11357976e-08 1.93367146e-07 5.25626399e-07
1.42880069e-06 3.88388295e-06 1.05574884e-05 2.86982290e-05 7.80098744e-05 2.12052824e-04 5.76419338e-04 1.56687021e-03 4.25919483e-03
1.15776919e-02 3.14714295e-02 8.55482149e-02 2.32544158e-01 6.32120559e-01]
```

Graph



The figure shows the fundamental property of softmax function. The **high value will have the high probability**.

Difference Between Sigmoid Function and Softmax Function

The below are the tabular differences between Sigmoid and Softmax function.

	Softmax Function	**Sigmoid Function**
1	Used for multi-classification in logistic regression	Used for binary classification in logistic regression model.

	model.	
2	The probabilities sum will be 1	The probabilities sum need not be 1.
3	Used in the different layers of neural networks.	Used as activation function while building neural networks.
4	The high value will have the higher probability than other values.	The high value will have the high probability but not the higher probability.

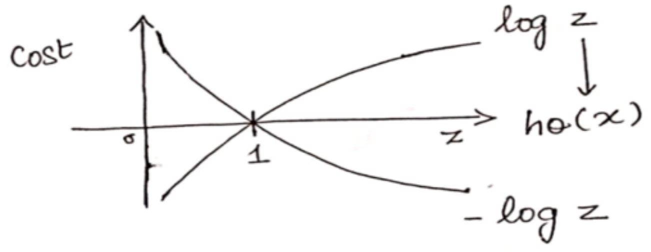
Conclusion

you learn in details about two functions which determine the logistic regression model. Just for a glance.

- **Softmax:** Used for the multi-classification task.
- **Sigmoid:** Used for the binary classification task.

Cost function explanation

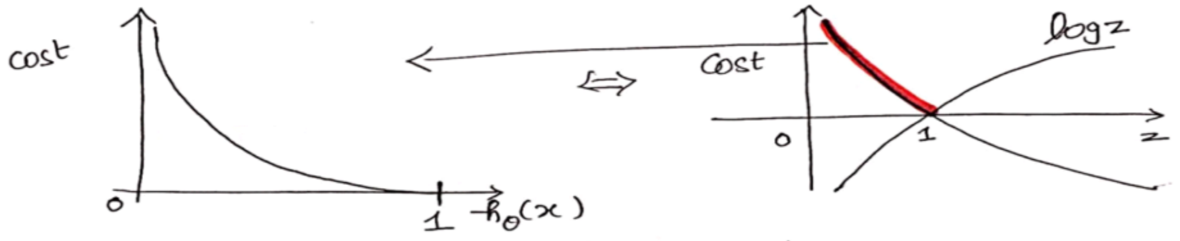
Sunday, September 2, 2018 3:04 AM



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

If y = 1,

Cost $(h_\theta(x), y) = -\log(h_\theta(x))$

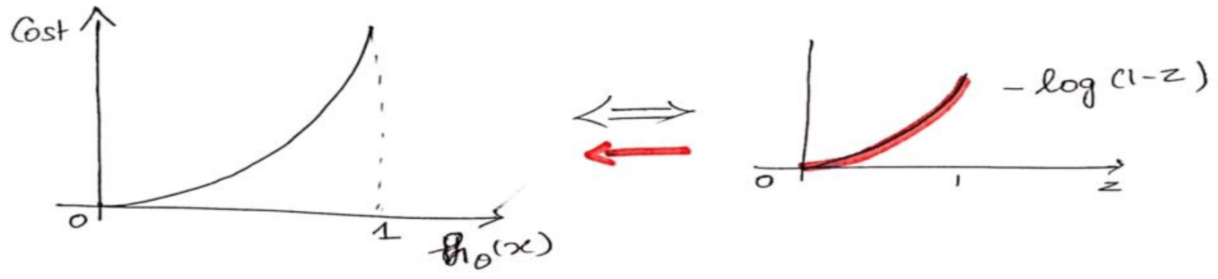


If, Cost = 0 ⇒ y = 1 ⇒ $h_\theta(x) = 1$

Cost = infinity for $h_\theta(x) = 0$

If $h_\theta(x) = 0$, it is similar to predicting $P(y = 1 | x; \theta) = 0$

Figure 1: Cost Function part 1



If cost = 0 ⇒ $h_\theta(x) = 0$ ⇒ y = 0

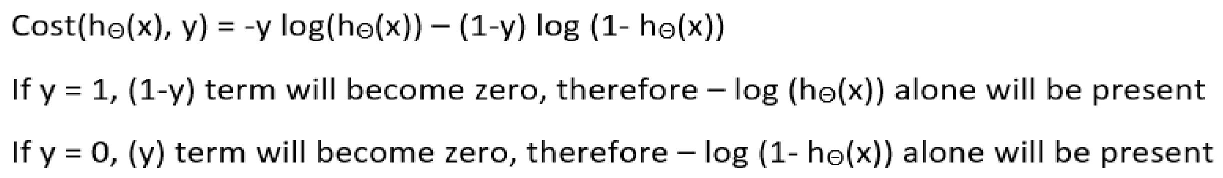
cost = ∞ ⇒ $h_\theta(x) = 1$

If $h_\theta(x) = 1$, it is similar to predicting

$P(y = 0 | x; \theta) = 0$

Figure 1: Cost Function part 2

Simplified cost function

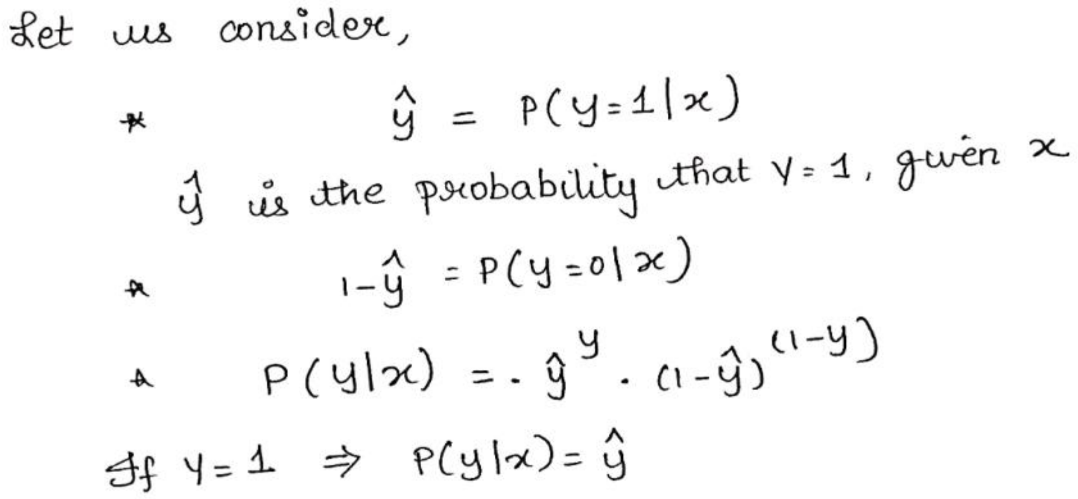


Cost($h_\Theta(x)$, y) = -y log($h_\Theta(x)$) – (1-y) log (1- $h_\Theta(x)$)

If y = 1, (1-y) term will become zero, therefore – log ($h_\Theta(x)$) alone will be present

If y = 0, (y) term will become zero, therefore – log (1- $h_\Theta(x)$) alone will be present

Why this cost function?



Let us consider,

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

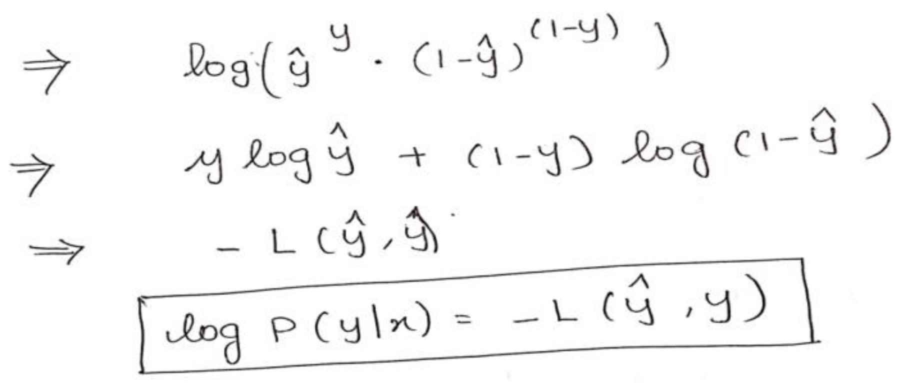
  $\hat{y}$ is the probability that $y=1$, given x

* $1-\hat{y} = P(y=0|x)$

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

If $y=1 \Rightarrow P(y|x)=\hat{y}$

Figure 3: Maximum Likelihood Explanation part-1



$\Rightarrow \log(\hat{y}^{y} \cdot (1-\hat{y})^{(1-y)})$

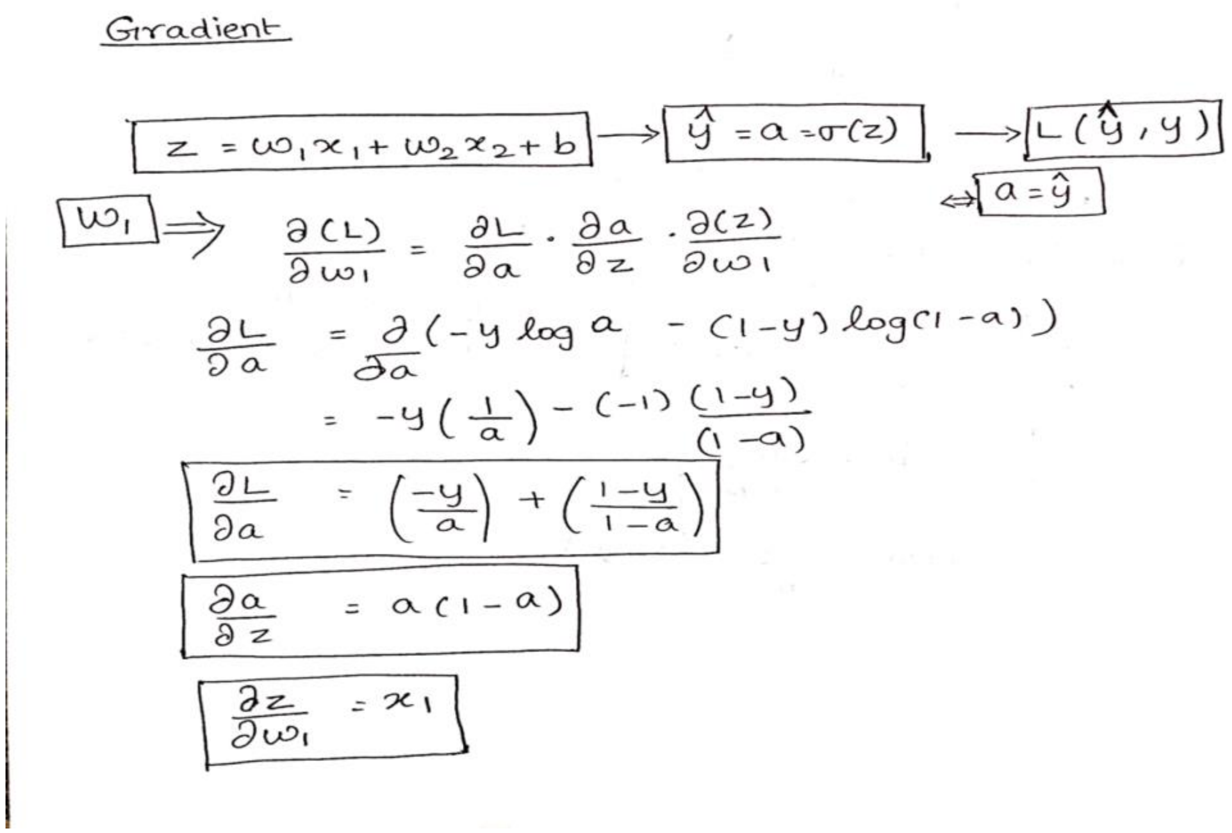
$\Rightarrow y\log\hat{y} + (1-y)\log(1-\hat{y})$

$\Rightarrow -L(\hat{y},\hat{y})$

$$\boxed{\log P(y|x) = -L(\hat{y},y)}$$

Figure 2: Maximum Likelihood Explanation part-2

This negative function is because when we train, we need to maximize the probability by minimizing loss function. Decreasing the cost will increase the maximum likelihood assuming that samples are drawn from an identically independent distribution.



<u>Gradient</u>

$$\begin{tikzcd} \boxed{z = w_1x_1 + w_2x_2 + b} \arrow[r] & \boxed{\hat{y} = a = \sigma(z)} \arrow[r] & \boxed{L(\hat{y}, y)} \end{tikzcd}$$

$\Leftrightarrow \boxed{a = \hat{y}.}$

$\boxed{w_1} \Longrightarrow \quad \frac{\partial(L)}{\partial w_1} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial(z)}{\partial w_1}$

$$\frac{\partial L}{\partial a} = \frac{\partial}{\partial a}(-y \log a - (1-y)\log(1-a))$$

$$= -y\left(\frac{1}{a}\right) - (-1)\frac{(1-y)}{(1-a)}$$

$$\boxed{\frac{\partial L}{\partial a} = \left(\frac{-y}{a}\right) + \left(\frac{1-y}{1-a}\right)}$$

$$\boxed{\frac{\partial a}{\partial z} = a(1-a)}$$

$$\boxed{\frac{\partial z}{\partial w_1} = x_1}$$

Figure 5: Gradient Descent Algorithm part 1

$$\frac{\partial L}{\partial w_1} = \left(\left(\frac{-y}{a} + \frac{(1-y)}{1-a} \right) \cdot (a)(1-a) \right) \cdot x_1$$

$$= (a-y) \cdot x_1$$

Update for w_1,

$$\frac{\partial L}{\partial w_1} = (a-y) \cdot x_1$$

Here, $(a-y) = \frac{\partial L}{\partial z}$

$$\boxed{w_1 = w_1 - \alpha \frac{\partial L}{\partial w_1}}$$

Similarly, for all parameters

$$\boxed{w_i = w_i - \alpha \frac{\partial L}{\partial w_i}}$$

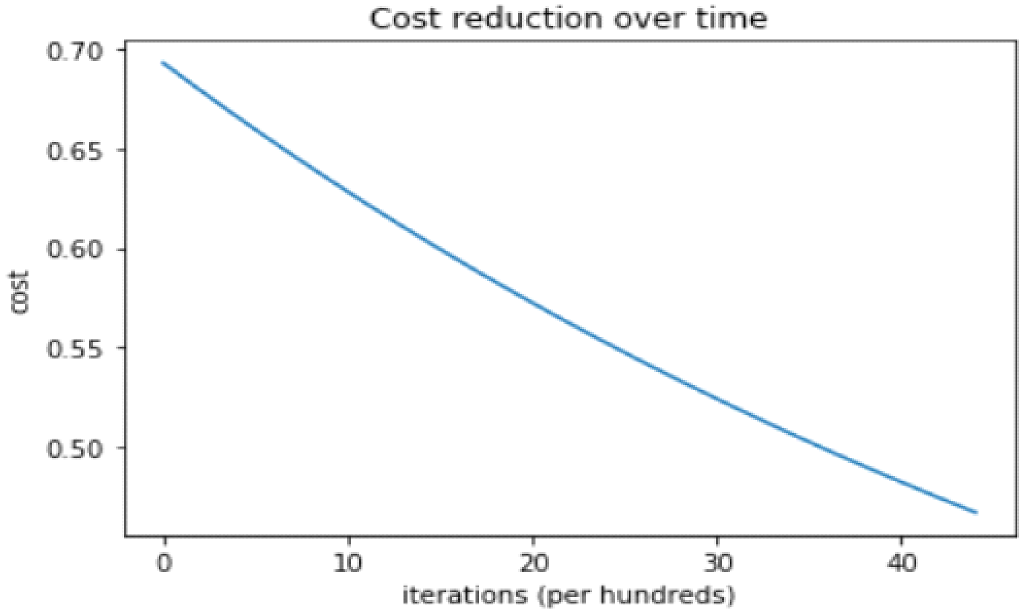
$i = 1, 2, \dots, m$
m = no. of parameters

$$\boxed{b = b - \alpha \frac{\partial L}{\partial b}}$$

Where, $\frac{\partial L}{\partial b} = (a-y)$

Figure 6 Gradient Descent part 2

Cost vs Number_of_Iterations



Train and test accuracy of the system is 100 %

DIFFERENCE BETWEEN LINEAR REGRESSION AND LOGISTIC REGRESSION

Sunday, June 3, 2018 10:35 AM

The purpose of this is to help you understand the difference between linear regression and logistic regression. These regression techniques are two most popular statistical techniques that are generally used practically in various domains. Since these techniques are taught in universities, their usage level is very high in predictive modeling world. In this article, we have listed down 13 differences between these two algorithms.

Difference between Linear and Logistic Regression

1. Variable Type : Linear regression requires the dependent variable to be continuous i.e. numeric values (no categories or groups).

While Binary logistic regression requires the dependent variable to be binary - two categories only (0/1). Multinominal or ordinary logistic regression can have dependent variable with more than two categories.

2. Algorithm : Linear regression is based on ***least square estimation*** which says regression coefficients should be chosen in such a way that it minimizes the sum of the squared distances of each observed response to its fitted value.

While logistic regression is based on ***Maximum Likelihood Estimation*** which says coefficients should be chosen in such a way that it maximizes the **Probability of Y given X** (likelihood). With ML, the computer uses different "iterations" in which it tries different solutions until it gets the maximum likelihood estimates.

3. Equation :

Multiple Regression Equation :

$$Y = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + \dots\dots\dots + b_kX_k$$

Linear Regression Equation

Y is target or dependent variable, b0 is intercept. x1,x2,x3...xk are predictors or independent variables. b1,b2,b3....bk is coefficients of respective predictors.

Logistic Regression Equation :

P(y=1) = e(b0 + b1x1 + b2x2 +-----bkxk) / (1+e(b0+b1x1+ b2x2+------bkxk))

Which further simplifies to :

P(y=1) = 1 / (1+ exp **-**(b0+b1x1+ b2x2+------bkxk))

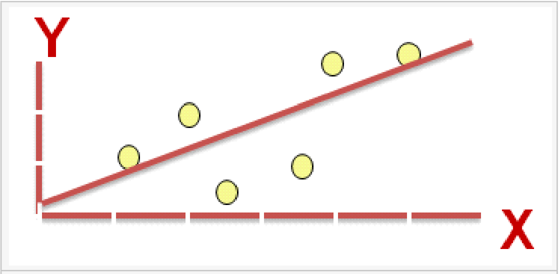
$$p = \frac{1}{1 + e^{-(b_0 + b_1X_1 + b_2X_2 + --- + b_kX_k)}}$$

Logistic Regression Equation

The above function is called logistic or sigmoid function.

4. Curve :

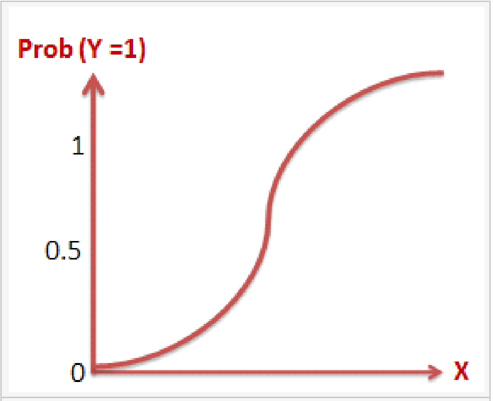
Linear Regression : Straight line



Straight Line : Linear Regression

Linear regression aims at finding the best-fitting straight line which is also called a regression line. In the above figure, the red diagonal line is the best-fitting straight line and consists of the predicted score on Y for each possible value of X. The distance between the points to the regression line represent the errors.

Logistic Regression : S Curve



Logistic S-shaped curve

Changing the coefficient leads to change in both the direction and the steepness of the logistic function. It means positive slopes result in an S-shaped curve and negative slopes result in a Z-shaped curve.

5. Linear Relationship : Linear regression needs a linear relationship between the dependent and independent variables. While logistic regression does not need a linear relationship between the dependent and independent variables.

6. Normality of Residual : Linear regression requires error term should be normally distributed. While logistic regression does not require error term should be normally distributed.

7. Homoscedasticity : Linear regression assumes that residuals are approximately equal for all predicted dependent variable values. While Logistic regression does not need residuals to be equal for each level of the predicted dependent variable values.

8. Sample Size : Linear regression requires 5 cases per independent variable in the analysis.While logistic regression needs at least 10 events per independent variable.

9. Purpose : Linear regression is used to estimate the dependent variable incase of a change in independent variables**.** For example, relationship between number of hours studied and your grades.
Whereas logistic regression is used to calculate the probability of an event. For example, an event can be whether customer will attrite or not in next 6 months.

10. Interpretation : Betas or Coefficients of linear regression is interpreted like below -

> *Keeping all other independent variables constant, how much the dependent variable is expected to increase/decrease with an unit increase in the independent variable.*

In logistic regression, we interpret odd ratios -

> *The effect of a one unit of change in X in the predicted odds ratio with the other variables in the model held constant.*

11. Distribution :

Linear regression assumes normal or gaussian distribution of dependent variable. Whereas, **Logistic regression** assumes binomial distribution of dependent variable. **Note :** Gaussian is the same as the normal distribution. See the implementation in R below -