Contents

[Classfication 2](#_Toc518308483)

[Logistic function/Sigmoid function 2](#_Toc518308484)

[Decision Boundary 3](#_Toc518308485)

[Logistic Regression Assumptions 3](#_Toc518308486)

[Interpretation of Logistic Regression Equation 4](#_Toc518308487)

[Advantages and Disadvantages of Logistic Regression 5](#_Toc518308488)

[Maximum Likelihood Estimation (MLE ) 5](#_Toc518308489)

[Cost function 5](#_Toc518308490)

[The cross-entropy loss function 7](#_Toc518308491)

[Binary cross-entropy loss 8](#_Toc518308492)

[Calculating accuracy 8](#_Toc518308493)

[Logits 8](#_Toc518308494)

[The Softmax function 9](#_Toc518308495)

[Cross Entropy 9](#_Toc518308496)

[One-Hot-Encoding matrix. 9](#_Toc518308497)

[Multiclass Classification 9](#_Toc518308498)

[Interview Questions 9](#_Toc518308499)

[Why is logistic regression a linear classifier? Or Why Logistic regression is not Logistic classification 9](#_Toc518308500)

[Why cant we use Linear regression for classification model 12](#_Toc518308501)

[How would you create a logistic regression model? 12](#_Toc518308502)

[Do I need to add interaction of variables in logistic regression? 12](#_Toc518308503)

[How to Solve the Logistic Regression Equation for the Probability p. 12](#_Toc518308504)

# Logistic regression

It predicts the probability of occurrence of an event by fitting data to a logit function.

Logistic regression is based on **Maximum Likelihood (ML) 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.

**Fisher Scoring** is the most popular iterative method of estimating the regression parameters.

logit(p) = b0 + b1X1 + b2X2 + ------ + bk Xk

where logit(p) = loge(p / (1-p))

# Distribution

Binary logistic regression model assumes binomial distribution of the response with N (number of trials) and p(probability of success). Logistic regression is in the 'binomial family' of GLMs.

The dependent variable does not need to be normally distributed.

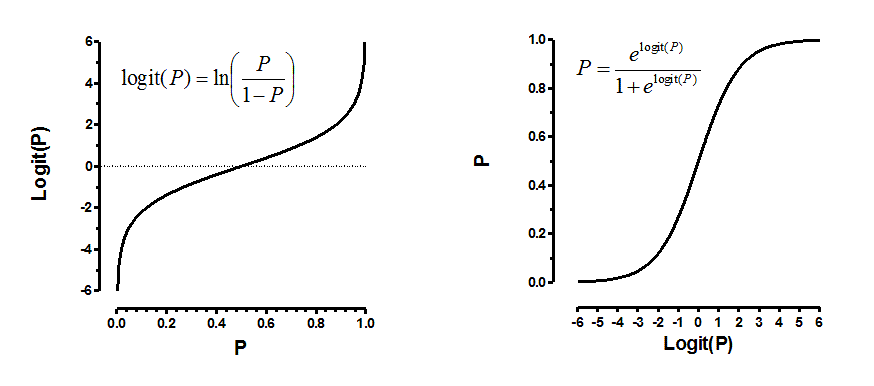
# Classfication

To attempt classification, one method is to use linear regression and map all predictions greater than 0.5 as a 1 and all less than 0.5 as a 0. However, this method doesn't work well because classification is not actually a linear function.

The classification problem is just like the regression problem, except that the values we now want to predict take on only a small number of discrete values. For now, we will focus on the **binary classification** **problem** in which y can take on only two values, 0 and 1. (Most of what we say here will also generalize to the multiple-class case.) For instance, if we are trying to build a spam classifier for email, then *x*(*i*) may be some features of a piece of email, and y may be 1 if it is a piece of spam mail, and 0 otherwise. Hence, y∈{0,1}. 0 is also called the negative class, and 1 the positive class, and they are sometimes also denoted by the symbols “-” and “+.” Given *x*(*i*), the corresponding *y*(*i*) is also called the label for the training example.

## Logistic function/Sigmoid function

### Logit Function:

Logistic regression is an estimate of a logit function. Here is how the logit function looks like:[](https://www.analyticsvidhya.com/wp-content/uploads/2015/10/logit.png)

**Why we use logistic function**

We could approach the classification problem ignoring the fact that y is discrete-valued, and use our old linear regression algorithm to try to predict y given x. However, it is easy to construct examples where this method performs very poorly. Intuitively, it also doesn’t make sense for *hθ*(*x*) (which is to take values larger than 1 or smaller than 0 when we know that y ∈ {0, 1}.

To fix this, let’s change the form for our hypotheses *hθ*(*x*) to satisfy 0≤*hθ*(*x*)≤1. This is accomplished by plugging *θTx* into the Logistic Function.

Our new form uses the "Sigmoid Function," also called the "Logistic Function":



hθ(X) = probability that y =1 given x parametrized by θ

*hθ*(*x*) will give us the **probability** that our output is 1. For example, *hθ*(*x*)=0.7 gives us a probability of 70% that our output is 1. Our probability that our prediction is 0 is just the complement of our probability that it is 1 (e.g. if probability that it is 1 is 70%, then the probability that it is 0 is 30%).

In logistic regression, the output is related to the predictors via the logistic sigmoid function.

Model: y^=Sigmoid(θTx)

## Decision Boundary

The **decision boundary** is the line that separates the area where y = 0 and where y = 1. It is created by our hypothesis function.

The input to the sigmoid is linear in the parameters, not the predictors. The decision boundary is the solution to the equation, θTx =0. As a result, if x contains non-linear terms (such as interactions or squared terms), the decision boundary will be nonlinear.

## Odds

Odds(class = 1) = Pr(class =1) / [1 - Pr(class =1)]

# Logistic Regression Assumptions

Equation

**log (p/(1-p)) = *β0 + β1x***

**Assumptions**

|  |
| --- |
| Independence of errors  Linearity between independent variable ( x variables) and log odds  Little or no multi collinearity  No influential observations (Outliers)  Large Sample Size - It requires at least 10 events per independent variable. |

1. Binary logistic regression requires the dependent variable to be binary.
2. Second, logistic regression requires the observations to be independent of each other.  In other words, the observations should not come from repeated measurements or matched data.
3. Third, logistic regression requires there to be little or no multicollinearity among the independent variables.  This means that the independent variables should not be too highly correlated with each other.
4. **Linearity** - Fourth, logistic regression assumes linearity of independent variables and log odds.  although this analysis does not require the dependent and independent variables to be related linearly, it requires that the independent variables are linearly related to the log odds.

**Odds = probability of occurring / probability of not occurring**

1. Finally, logistic regression typically requires a large sample size.  A general guideline is that you need at minimum of 10 cases with the least frequent outcome for each independent variable in your model. For example, if you have 5 independent variables and the expected probability of your least frequent outcome is .10, then you would need a minimum sample size of 500 (10\*5 / .10).

**Assumption which are not there**

First, logistic regression **does** **not** require a linear relationship between the dependent and independent variables.

Second, the error terms (residuals) do not need to be normally distributed.

Third, homoscedasticity is not required.

Finally, the dependent variable in logistic regression is not measured on an interval or ratio scale.

|  |
| --- |
| Equation  **log (p/(1-p)) = β0 + β1x** |

# Interpretation of Logistic Regression Equation

**Intercept** – log odds when all X’s are 0

**Slope:**

* difference in log odds for a 1 unit increase in X, controlling for other X’s
* log odds ratio associated with a 1 unit increase in X, controlling for other X’s

**Continuous variable**

An unit increase in years of experience increases the odds of getting a job by a multiplicative factor of 4.27, given the other variables in the model are held constant. In other words, the odds of getting a job are increased by 327% (4.27-1)\*100 for an unit increase in years of experience.

**For binary predictors**

The odds of a person having years of experience getting a job are 4.27 times greater than the odds of a person having no experience.

# Advantages and Disadvantages of Logistic Regression

* Pros
  + low variance
  + provides probabilities for outcomes
  + works well with diagonal (feature) decision boundaries
  + NOTE: logistic regression can also be used with kernel methods
* Cons
  + high bias

# Maximum Likelihood Estimation (MLE )

**Maximum-likelihood estimation** is a common learning algorithm used by a variety of machine learning algorithms, although it does make assumptions about the distribution of your data (more on this when we talk about preparing your data).

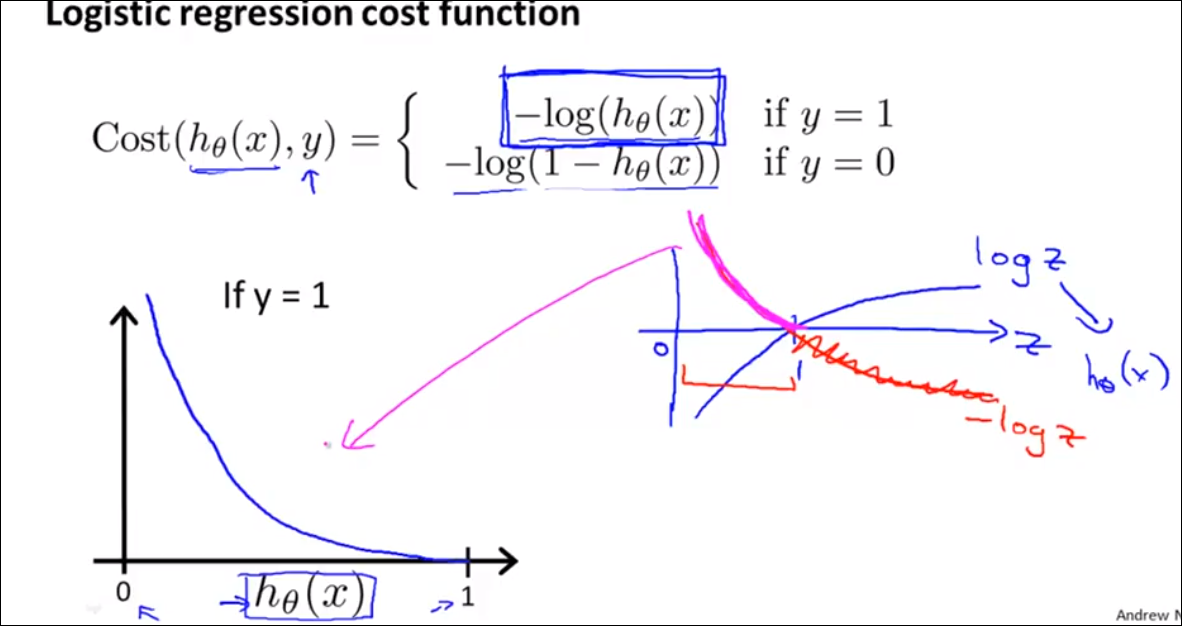
# [Cost function](http://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html#id22)

Unfortunately we can’t (or at least shouldn’t) use the same cost function [Mean Squared Error](http://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#mse) as we did for linear regression. Why? There is a great math explanation in chapter 3 of Michael Neilson’s deep learning book [[5]](http://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html#id11), but for now I’ll simply say it’s because our prediction function is non-linear (due to sigmoid transform). Squaring this prediction as we do in MSE results in a non-convex function with many local minimums.

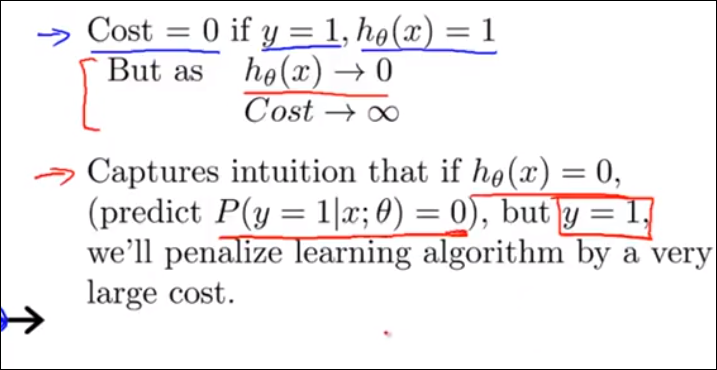
If our cost function has many local minimums, gradient descent may not find the optimal global minimum.

Instead of Mean Squared Error, we use a cost function called [Cross-Entropy](http://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#loss-cross-entropy), also known as Log Loss. Cross-entropy loss can be divided into two separate cost functions: one for y=1and one for y=0

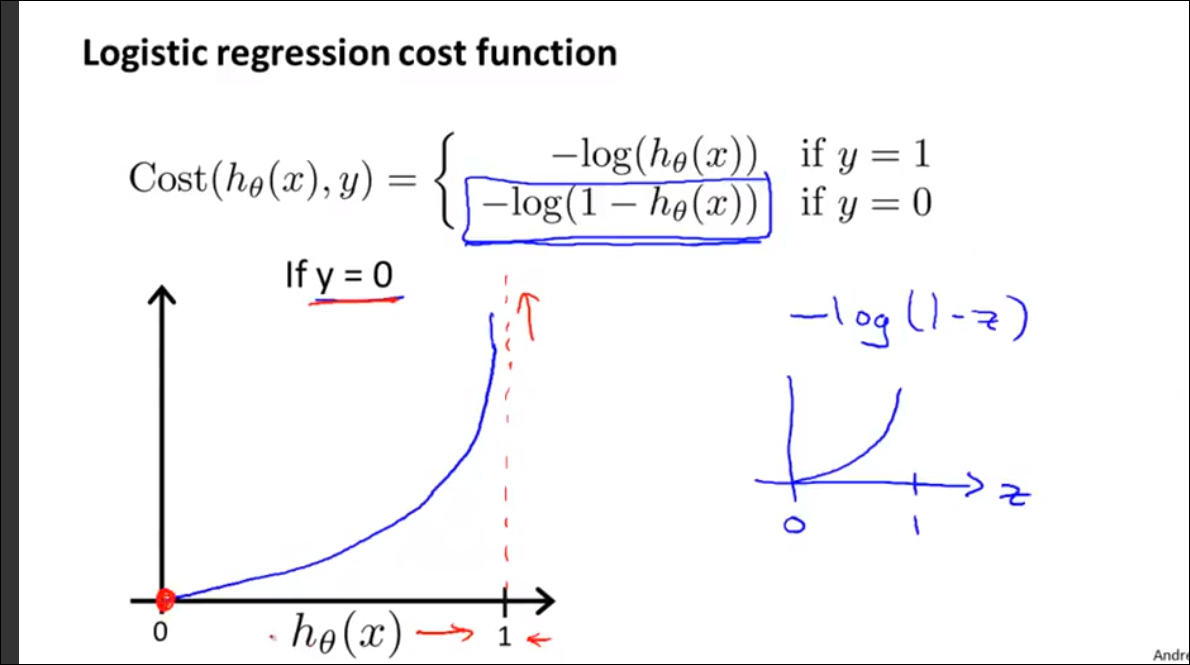
**Cost function when y=1**



**Intuition behind the cost function**



**Cost function when y=0**



## The cross-entropy loss function

Before we can start training, we’re going to need to define a loss function that makes

sense when our prediction is a probability distribution.

The relevant loss function here is called cross-entropy and it may be the most common

loss function you’ll find in all of deep learning. That’s because at the moment, classification

problems tend to be far more abundant than regression problems.

The basic idea is that we’re going to take a target Y that has been formatted as a one-hot

vector, meaning one value corresponding to the correct label is set to 1 and the others are

set to 0, e.g. [0, 1, 0, 0, 0, 0, 0, 0, 0, 0] .

The basic idea of cross-entropy loss is that we only care about how much probability the

prediction assigned to the correct label. In other words, for true label 2, we only care

about the component of yhat corresponding to 2. Cross-entropy attempts to maximize

the log-likelihood given to the correct labels.

The sigmoid function σ sometimes called a squashing function or a *logistic* function -

thus the name logistic regression - maps a real-valued input to the range 0 to 1.

Specifically, it has the functional form:

## Binary cross-entropy loss

Now that we’ve got a model that outputs probabilities, we need to choose a loss function.

When we wanted to predict *how much* we used squared error , as our measure

our model’s performance.

Since now we’re thinking about outputing probabilities, one natural objective is to say that

we should choose the weights that give the actual labels in the training data the highest

probability.

max θ P θ ((y 1 ,...,y n )| ***x*** 1 ,..., ***x*** n ) maxθPθ((y1,...,yn)|x1,...,xn)

Because each example is independent of the others, and each label depends only on the

features of the corresponding examples, we can rewirte the above as

max θ P θ (y 1 | ***x*** 1 )P θ (y 2 | ***x*** 2 )...P(y n | ***x*** n ) maxθPθ(y1|x1)Pθ(y2|x2)...P(yn|xn)

This function is a product over the examples, but in general, because we want to train by

stochastic gradient descent, it’s a lot easier to work with a loss function that breaks down

as a sum over the training examples.

max θ logP θ (y 1 | ***x*** 1 )+...+logP(y n | ***x*** n ) maxθlog⁡Pθ(y1|x1)+...+log⁡P(yn|xn)

Because we typically express our objective as a *loss* we can just flip the sign, giving us the

*negative log probability:*

Note that this loss function is commonly called *log loss* and is also commonly referred to as

*binary cross entropy* . It is a special case of negative log likelihood. And it is a special case of

cross-entropy, which can apply to the multi-class ( >2) setting.

# Calculating accuracy

While the negative log likelihood gives us a sense of how well the predicted probabilities

agree with the observed labels, it’s not the only way to assess the performance of our

classifiers. For example, at the end of the day, we’ll often want to apply a threshold to the

predicted probabilities in order to make hard predictions. For example, if we were building

a spam filter, we’ll need to either send the email to the spam folder or to the inbox. In

these cases, we might not care about negative log likelihood, but instead we want know

*how many errors* our classifier makes. Let’s code up a simple script that calculates the

accuracy of our classifier.

## Logits

The Logits also called as scores. These are just the outputs of the linear model. The Logits

will change with the changes in the calculated weights.

Softmax Function

The Softmax function is a probabilistic function which calculates the probabilities for the

given score. Using the softmax function return the high probability value for the high

scores and fewer probabilities for the remaining scores. This we can observe from the

image. For the logits 0.5, 1.5, 0.1 the calculated probabilities using the softmax function

are 0.2, 0.7, 0.1

For the Logit 1.5, we are getting the high probability value 0.7 and very less probability

value for the remaining Logits 0.5 and 0.1

## Cross Entropy

The cross entropy is the last stage of multinomial logistic regression. Uses the

cross-entropy function to find the similarity distance between the probabilities calculated

from the softmax function and the target one-hot-encoding matrix.

Before we learn more about Cross Entropy, let’s understand what it is mean by

## One-Hot-Encoding matrix.

# Multiclass Classification

Now we will approach the classification of data when we have more than two categories. Instead of y = {0,1} we will expand our definition so that y = {0,1...n}.

Since y = {0,1...n}, we divide our problem into n+1 (+1 because the index starts at 0) binary classification problems; in each one, we predict the probability that 'y' is a member of one of our classes.

# Test Overall Fit of the Model : -2 Log L , Score and Wald Chi-Square

These are Chi-Square tests. They test against the null hypothesis that at least one of the predictors' regression coefficient is not equal to zero in the model.

# Interview Questions

## [Why is logistic regression a linear classifier?](https://stats.stackexchange.com/questions/93569/why-is-logistic-regression-a-linear-classifier) Or Why Logistic regression is not Logistic classification

<https://www.quora.com/Why-is-logistic-regression-considered-a-linear-model>

Logistic regression predicts probabilities, and is therefore a regression algorithm

 Logistic regression is a regression model because it estimates the odds of class membership as a (transformation of a) multilinear function of the features.

Logistic regression is a generalized linear model, meaning it uses a link function to transform a range of probabilities into a range from negative infinity to positive infinity.

Remember that logistic regression does not produce a probability value; it produces an odds ratio, and an odds ratio has no upper limit. The interpretation of a logistic regression output would read something like this:

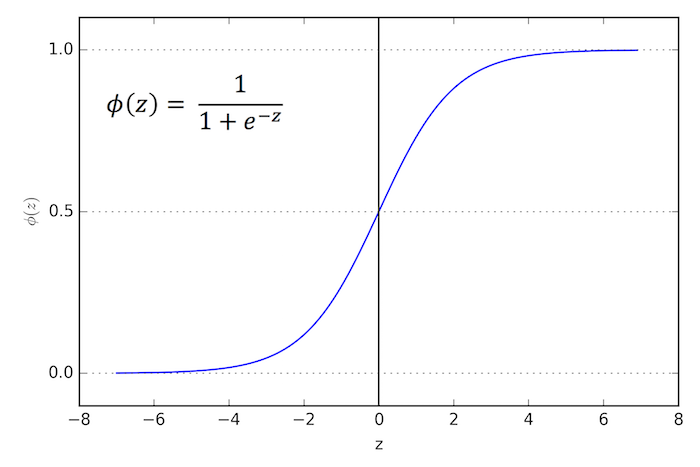
"People for whom independent variable I is TRUE have X.XX greater odds of dependent variable D being true than for those for whom I is FALSE"

The short answer is: Logistic regression is considered a generalized linear model **because the outcome always depends on the sum of the inputs and parameters. Or in other words, the output cannot depend on the product (or quotient, etc.) of its parameters!**

So, why is that? Let’s recapitulate the basics of logistic regression first, which hopefully makes things more clear. Logistic regression is an algorithm that learns a model for binary classification.

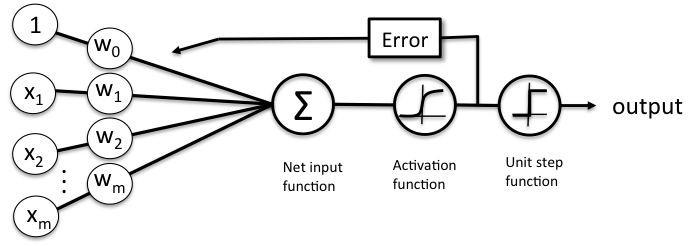
A nice side-effect is that it gives us the *probability* that a sample belongs to class 1 (or vice versa: class 0).

Our objective function is to minimize the so-called logistic function Φ (a certain kind of sigmoid function); it looks like this:

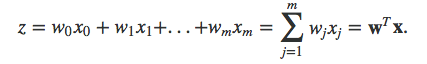


Now, if Φ(z) is larger than 0.5 (alternatively: if z is larger than 0), we classify an input as class 1 (and class 0, otherwise). Although logistic regression produces a linear decision surface (see the classification example in the figure below) this logistic (activation) function doesn't look very linear at all, right!?

So, let's dig a bit deeper and take a look at the equation we use to compute the net input *z*!



The net input function is simply the *dot product* of our input features and the respective model coefficients w:



Here, x\_0 refers to the weight of the bias unit which is always equal to 1 (a detail we don’t have to worry about here). I know, mathematical equations can be a bit "abstract" at times, so let's look at a concrete example. Let's assume we have a sample training point x of 4 features (e.g., *sepal length, sepal width, petal length, and petal width* in the *Iris dataset*):

x = [1, 2, 3, 4]

Now, let's assume our weight vector looks like this:

w = [0.5, 0.5, 0.5, 0.5]

Let's compute z now!

z = w^T x = 1\*0.5 + 2\*0.5 + 3\*0.5 + 4\*0.5 = 5

Not that it is important, but we have a 99.3% chance that this sample belongs to class 1: φ(z=148.41) = 1 / (1 + e-5) = 0.993

Anyways, the reason why logistic regression produces a linear decision boundary is the ***additivity***of the terms: Our outcome *z* depends on the additivity of the parameters, e.g., :

z = w\_1 \* x\_1 + w\_2 \* x\_2

There's no interaction between the parameter weights, nothing like w\_1\*x\_1 \* w\_2\* x\_2 or so, which would make our model ***non-linear!***

## Why cant we use Linear regression for classification model

First, linear regression model which is used to predict the continuous variable assumes the residuals to be normally distributed. However, in case of classification problem where the target variable is 0 and 1 this assumption is clearly violated.

Secondly, in classification problem where we are predicting the odds of target variable, linear regression will give us values which are above 1 and below 0.

* h(x) for linear regression interpolates, or extrapolates, the output and predicts the value for x we haven't seen. It's simply like plugging a new x and getting a raw number, and is more suitable for tasks like predicting, say car price based on *{car size, car age}* etc.
* h(x) for logistic regression tells you the **probability** that x belongs to the "positive" class. This is why it is called a regression algorithm - it estimates a continuous quantity, the probability. However, if you set a threshold on the probability, such as h(x)>0.5, you obtain a classifier, and in many cases this is what is done with the output from a logistic regression model. This is equivalent to putting a line on the plot: all points sitting above the classifier line belong to one class while the points below belong to the other class.

Also, the predictions made using linear regression can be highly inaccurate. This is when sigmoid function comes into the picture.

As we are predicting the probability of class 0 and 1 we would not want it to be less than 0 and more than 1. Sigmoid function helps in bounding the probability between 0 and 1. Also, since the derivative of a sigmoid function is easy to calculate we can further use that in the neural network.

## How would you create a logistic regression model?

[Answer](http://www.kdnuggets.com/faq/precision-recall.html). Recall describes what percentage of true positives are described as positive by the model. Precision describes what percent of positive predictions were correct. The ROC curve shows the relationship between model recall and specificity – specificity being a measure of the percent of true negatives being described as negative by the model.

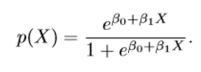
Recall, precision, and the ROC are measures used to identify how useful a given classification model is.

## Do I need to add interaction of variables in logistic regression?

If you have a significant interaction you  probably want to include it. If you include an interaction you almost certainly want to include the variables that go into that interaction.  That would mean model 2.

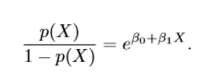
## How to Solve the Logistic Regression Equation for the Probability p.

In Logistic regression, the logistic function can be given as –



Here, P(X) is nothing but the probability of y =1 given the input vector X.

This can be further expressed as –



Here, the first term in above equation is termed as odds and it can take any value between 0 to ∞. So, to bound its value between 0 and 1, we take log on both the sides of above equation.

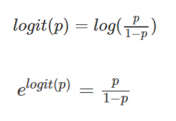
1

The left-hand side of above equation is called as log-odds or logit and the right-hand side of the equation is now similar to our linear regression. Thus, although non-linear in x, log odds of logistic is linear in X.

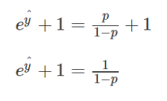
As we can see the logit function act as a link between logistic regression and linear regression and hence it is also called as a link function.

1

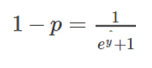
Our next step would be to calculate weights w which I will explain after we derive sigmoid function. Once the weights are estimated we take the inverse of logit function to find the probability p.



Adding 1 on both sides of above equation



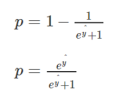
This can also be written as-



Subtracting 1 from both sides of equation-

1

Multiplying both sides by -1



Dividing both numerator and denominator by eyᶺ

1

The above equation represents a sigmoid function which is used in logistic regression and it is denoted by σ.

Hence the final logistic regression equation will look like this

yᶺ = σ (wT.x + b)

for simplicity let us denote wT.x + b = z

yᶺ = σ (z)

σ (z) = 1/1+ e-z

If the value of z is very large positive number then

σ (z) = 1/1+0

= 1

And if the value of z is very large negative number then

σ (z) = 1/1+ (big number)

and σ (z) will approximate to 0

So, we can see that the value is bound between 0 and 1.

Now to train our model we need to define the cost function for our logistic regression problem.

# Loss function of Logistic regression

The loss function is nothing but the error function which will tell us how good our output variable y^ is when the true label is y.

We generally use squared error method to find the loss function but in logistic regression, this method does not work well for gradient descent. Hence, we will define different loss function which plays a similar role like a squared error (for linear regression) and will give us optimization curve which will be convex in shape.

So, the loss function for logistic regression is given as-

**L(y^,y) = -(y log y^ + (1-y) log(1- y^))**

Here we want our loss function to be minimum just like the squared error function in Linear regression.

The key point here is Loss function is given for single training example. When we calculate the error for entire training set it is given by Cost function.

# Cost function

So, the cost function which is applied to entire training set is given as-

**J (w, b) = 1/m \* sum(L(y^,y) )***( for all the observation in training dataset.)*

**J (w, b) = -1/m \* sum ( y(i).log(y^(i)) + (1- y(i)).log(1- y^(i)))**

So, our aim is to find the parameter w and b that minimizes the overall cost function of the model.