LOGISTIC REGRESSION

Dr. Brian Mc Ginley

LOGISTIC REGRESSION

- To start on classification, do we really need a different algorithm? Let's see if we can "tweak" what we know about regression.
- Note:
  - Logistic Regression is a confusing name because we are not trying to predict a continuous value.
  - We do predict a continuous values (between 0 and 1) but we then threshold it, to take on discrete (non-continuous) values.
  - Logistic Regression, in the end, is actually a classification algorithm.

classification - discrete output

LOGISTIC REGRESSION

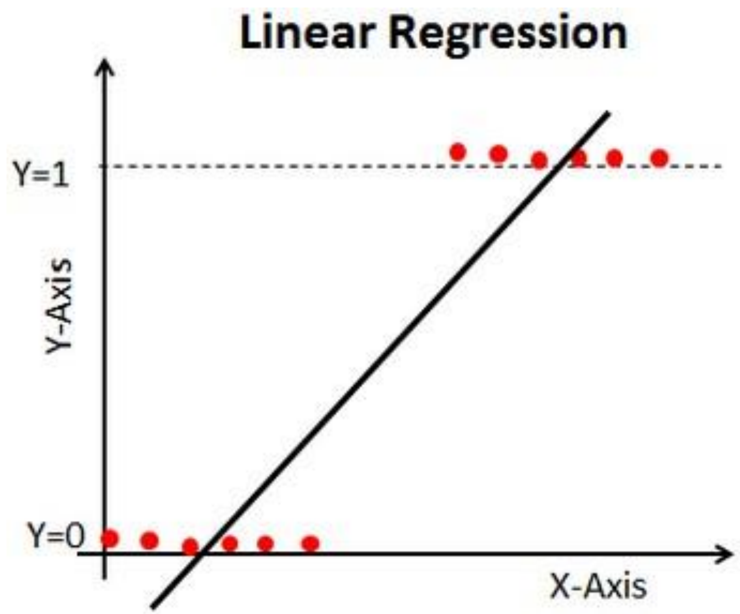
- Despite its name, Logistic Regression is a classification ML model. Something is either in category 0 or 1 and the *predicted* value is between 0 and 1. The predicted value is a *probability (p).*
- The response variable y is either 0 or 1. With our standard formula for regression,

$$\hat{y} = w_0 + w_1 x_1$$

The estimation of $\hat{y}$ has a good chance of being <0 or >1. We want an estimation for p between 0 and 1.

LOGISTIC REGRESSION

- Additionally, we get a plot (only one feature) like:

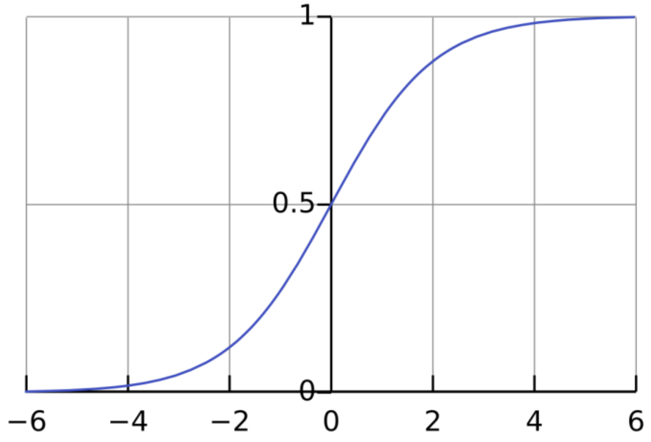


- The picture clearly is not going to be very accurate. We would much rather a picture like:

MODEL FOR LOGISTIC REGRESSION

- The **Sigmoid function**, also called the **logistic** function, takes any real-valued number and maps it to a value between 0 and 1.
  - If the curve tends towards $+\infty$, y goes towards 1 and if it tends towards $-\infty$, y goes towards 0

$$y = f(x) = \frac{1}{1 + e^{-x}}$$



- **Note**: Sigmoid function is regularly been used for Neural Networks, Deep Learning.
- We have the function that gives us the shape we want. How do we relate this to our

$$\hat{y} = w_0 + w_1 x \,?$$

MODEL FOR LOGISTIC REGRESSION

- Recall, the sigmoid function maps a value ranging from $-\infty$ to $+\infty$ into a range of 0 to 1.
- We can map our $\hat{y} = w_0 + w_1 x$ (which also ranges from $-\infty$ to $+\infty$) using the sigmoid function into a range of 0 to 1.

$$\hat{y} = w_0 + w_1 x$$

$$y = f(x) = \frac{1}{1 + e^{-x}}$$

$$y\ (or\ p) = \frac{1}{1 + e^{-(w_0 + w_1 x)}}$$

- Or the general for can be written as:

$$y\ (or\ p) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + \cdots + w_n x_n)}}$$

MODEL FOR LOGISTIC REGRESSION

- Sometimes the following equation:

$$y\ (or\ p) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + \cdots + w_n x_n)}}$$

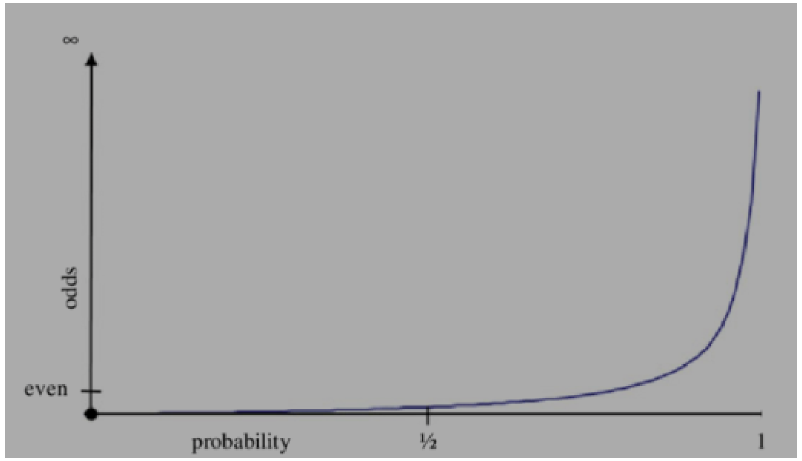
will also be written as (we'll come back to this) (& where *ln* is natural log):

$$ln\left(\frac{p}{1-p}\right) = w_0 + w_1 x_1 + \cdots + w_n x_n$$

- This is known as the **logistic regression function**. Often the weights/coefficients ($w_0 \ldots$) will be written in the literature as ($\beta_0, \ldots, \beta_n$)

MODEL FOR LOGISTIC REGRESSION

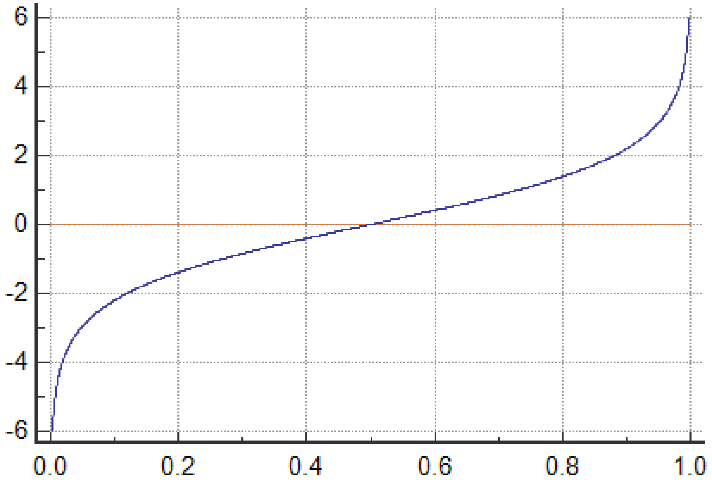
- Note: in the formula: $ln\left(\frac{p}{1-p}\right) = w_0 + w_1x_1 + \cdots + w_nx_n$
  - The term (p/1-p) is known colloquially as the odds
    - E.g. say if a team has a 50% chance of winning or a p=0.5 $\left(\frac{p}{1-p}\right) = \left(\frac{0.5}{0.5}\right) = 1:1$
    - E.g. say if a team has an 80% chance of winning or a p=0.8 $\left(\frac{p}{1-p}\right) = \left(\frac{0.8}{0.2}\right) = 4:1$
    - E.g. say if a team has an 5% chance of winning or a p=0.05 $\left(\frac{p}{1-p}\right) = \left(\frac{0.05}{0.95}\right) = 0.053:1$
- A problem with the odds function is that it’s asymmetrical.
  - If the probability of winning is between 0 and 0.5, the odds will be between 0 and 1
  - If the probability of winning is between 0.5 and 1, the odds will be between 1 and ∞
    - Problem!



MODEL FOR LOGISTIC REGRESSION

- To solve this asymmetrical problem with odds, the log of the odds function is often used (as it is a symmetrical function)

$$ln\left(\frac{p}{1-p}\right) = logit(p)$$



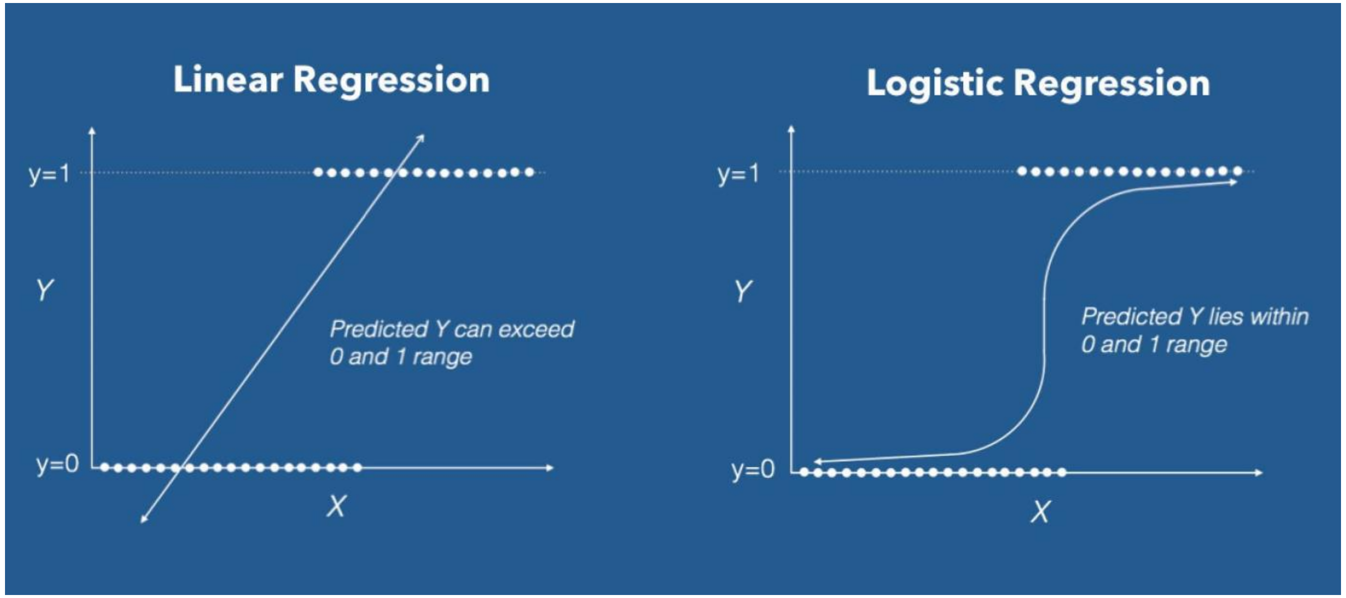
- This logit function forms the basis for logistic regression

$$ln\left(\frac{p}{1-p}\right) = w_0 + w_1 x_1 + \cdots + w_n x_n$$

- It just so happens that the inverse of the logit function is the sigmoid function.

LOGISTIC REGRESSION

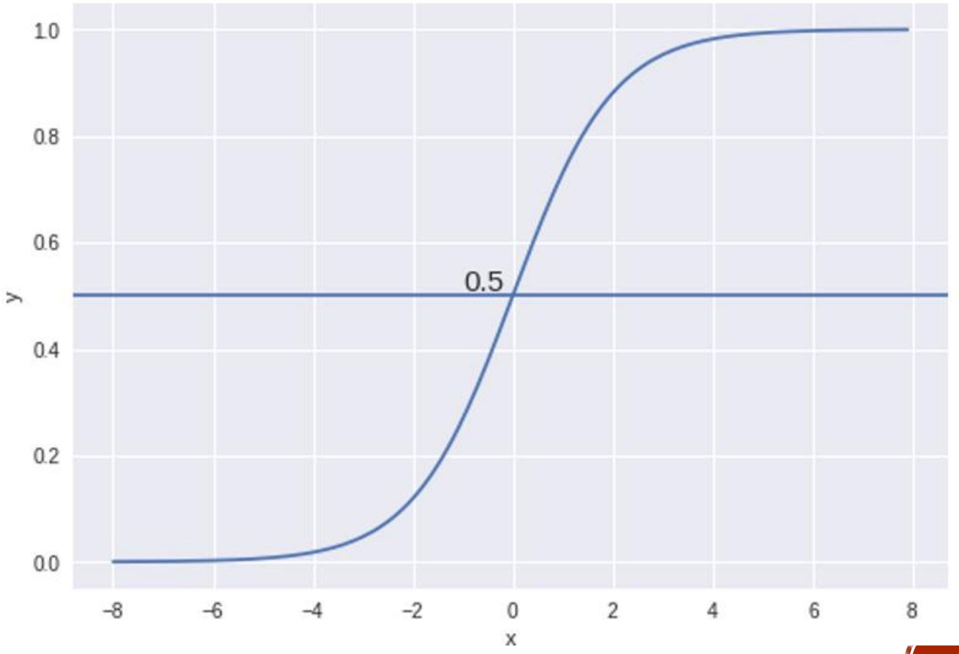
- Despite its name, Logistic Regression is a classification ML model.



- Something is either in category 0 or 1 and the *predicted* value is between 0 and 1. The predicted value is a *probability*

DECISION BOUNDARY

- We expect our classifier to give us a set of outputs or classes, when we pass the inputs through a prediction function and to return a probability score between 0 and 1.
- Example:
  - We have 2 classes, cats and dogs (Class 1: dog (1), Class 2 – cats (0)). We decide a threshold value above which we classify values into Class 1, and if the value goes below the threshold, we classify it into Class 2.
  - Having chosen the threshold as 0.5: if the prediction function returned a value of 0.7, then we would classify this observation as Class 1 (DOG). If our prediction returned a value of 0.2 then we would classify the observation as Class 2 (CAT).
  - We can choose a different boundary to adjust for FP vs FN.



THRESHOLD

- The end result of our ML system (for a binary classifier) is either the number 0 or 1.
- ML systems will often only output the class it thinks is most likely with this threshold, i.e. if one side of decision boundary always output 0, if the other output 1
- A lot of ML libraries that deal with Logistic Regression you can retrieve the "probability" for its result, however, some may just output 0 or 1.

LOGISTIC REGRESSION

- Clearly this is a binary classification system, so what happens if we have multiple categories?
- For now, we'll just take the scikitlearn method as "magic" in how it deals with multiple categories.
- A version of Logistic Regression is very often applied on the last layer of a neural network so understanding it will help when we get to that later.

LOGISTIC REGRESSION

- Advantages:
  - Interpretable
  - Small number params
  - Computationally efficient to train (estimate weights)
- Disadvantages:
  - Performance not necessarily as good as best classifiers (always depends on the problem) - sometimes it's the correct model
    - Random forests/boosted trees, SVM, etc. would outperform Logistic Regression over a large set of real-world problems
- Applications:
  - Biostats and social sciences
  - Foundation for neural networks/generalised linear models