Logistic Regression

Logistic Regression

Can we use the linear regression model for binary classification?

- Class 0 – labeled as 0
- Class 1 – labeled as 1
- We would like to learn f : **x** → {0,1}

Furthermore, we would like to estimate of the probability P(y=1|x), as the output:

$$\mathbf{x} \longrightarrow \boxed{} \longrightarrow P(y = 1 \mid \mathbf{X} = \mathbf{x}) = \sigma(\mathbf{w}^{\mathsf{T}}\mathbf{x})$$

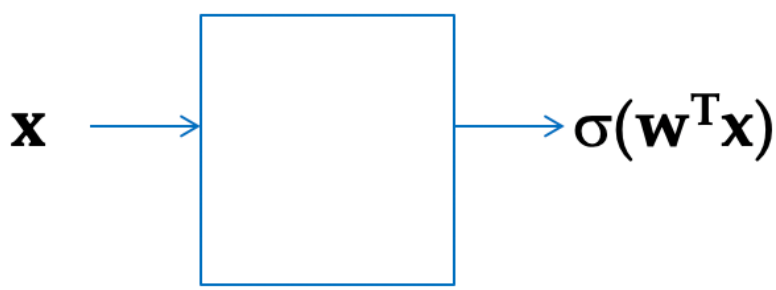
- The output of linear regression can be any real number. ☹
  - Use the sigmoid function to obtain output between 0 and 1.

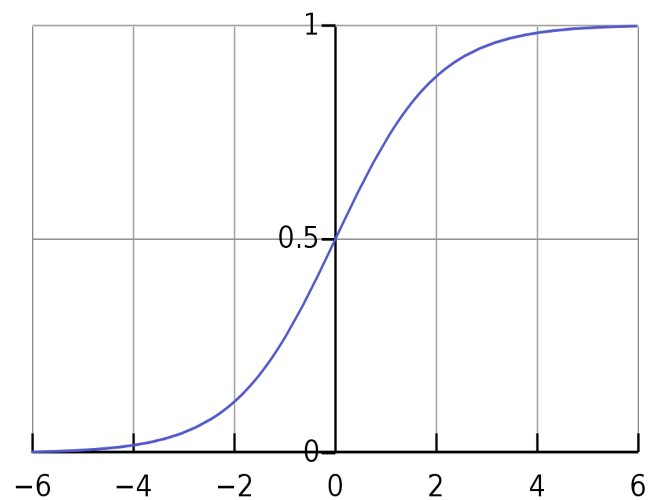
$$\sigma(z) = \frac{1}{1+e^{-z}}$$

where:

- $z = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$ (just like in linear regression)

$$f(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$$





$$\mathbf{x} \longrightarrow \boxed{} \longrightarrow P(y = 1 \mid \mathbf{X} = \mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$$

- We want :
  - Large positive net input for + samples (P will be close to 1)
  - Large negative net input for – samples (P will be close to 0).

- *For* **x**, we have

  $P(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$

  $P(y = 0 \mid \mathbf{x}) = 1 - \sigma(\mathbf{w}^T\mathbf{x})$

- A suitable loss function in logistic regression is called the Log-Loss, or binary cross-entropy.

$$L_{\text{BCE}} = -\sum_{i=1}^{N} \left[y_i \log \sigma(z_i) + (1 - y_i) \log(1 - \sigma(z_i)) \right]$$

where

- N is the number of samples, indexed by i,
- y_i is the true class for the index i,
- z_i is the net input for observation i.

Notice what happens when y_i=1 or y_i=0

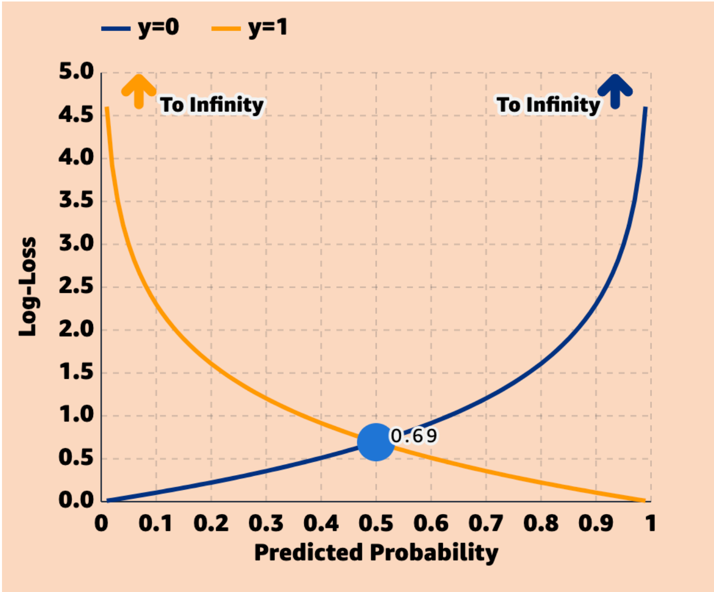
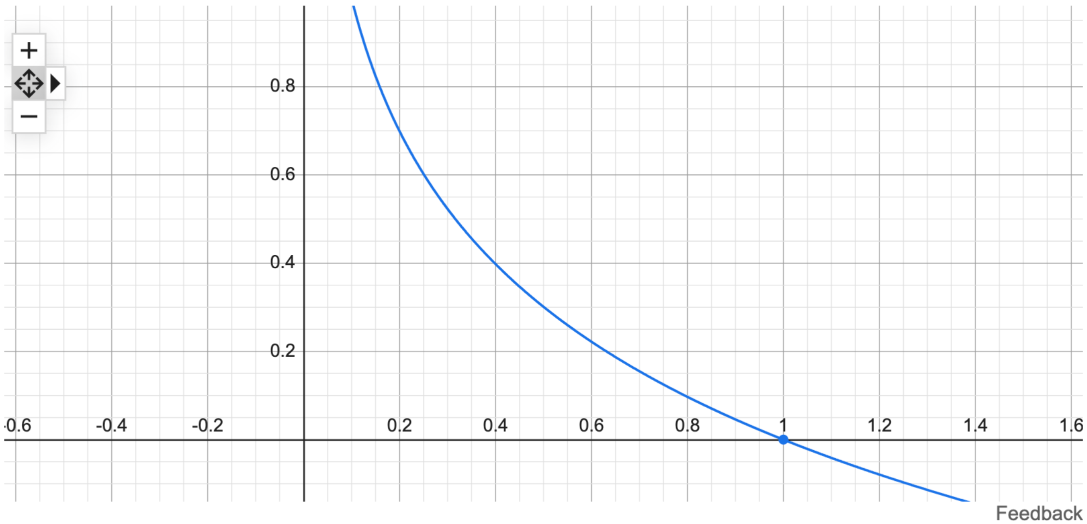


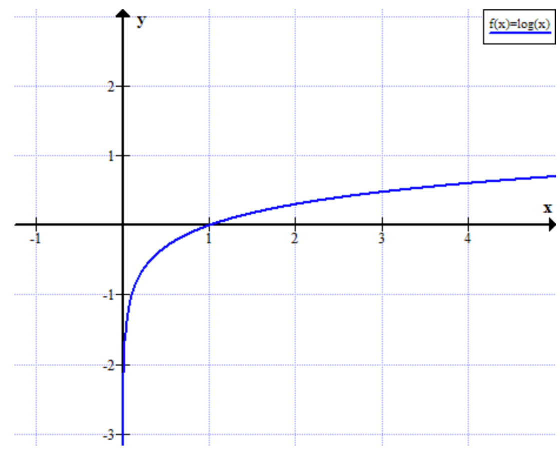
Figure from https://mlu-explain.github.io/logistic-regression/ with some good interactive tools

Why Binary Cross-entropy?

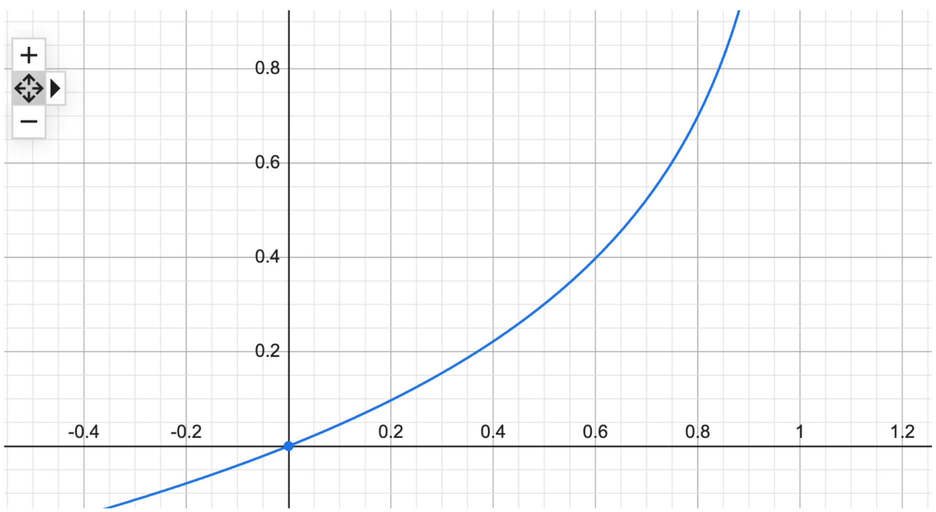
Graph for $-\log(p)$

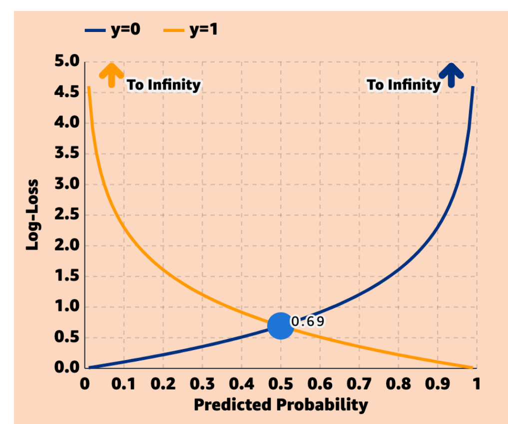


$y = f(x) = \log_{10}(x)$



Graph for $-\log(1-p)$





Optimization

- We can use Gradient Descent to find the optimal weights! ☺

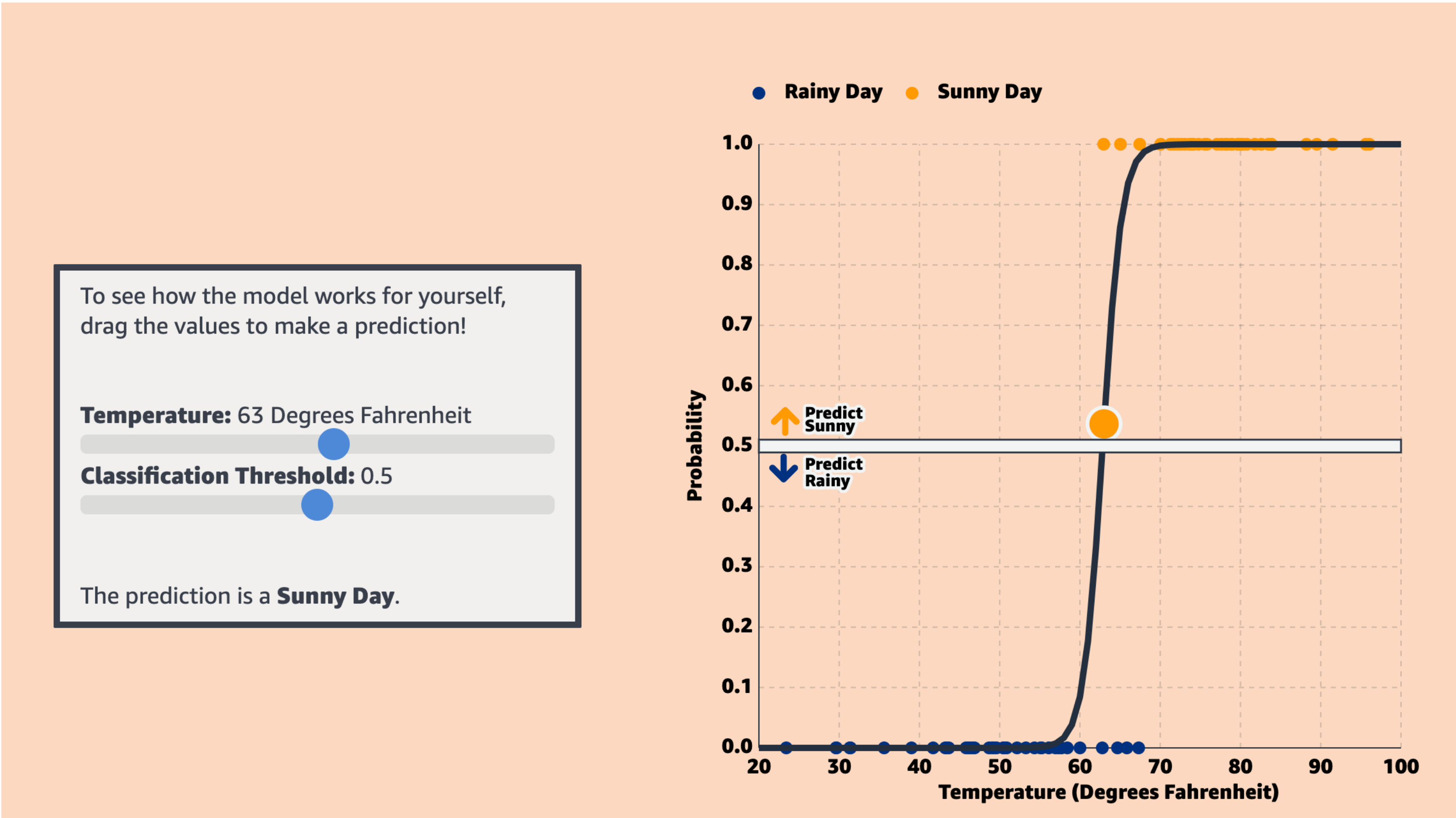


Figure from https://mlu-explain.github.io/logistic-regression/

- Learning progress: See how the solution evolves and understand the corresponding decision boundaries at https://mlu-explain.github.io/logistic-regression/

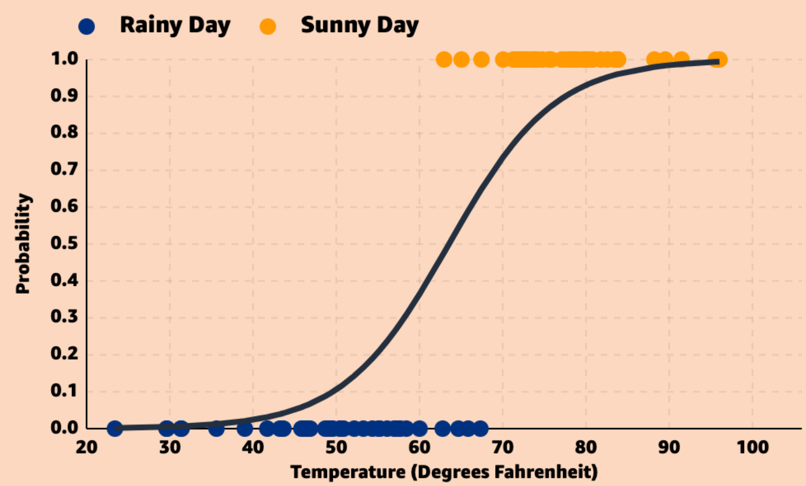
Let's see how gradient descent works for our logistic regression model. We'll use the algorithm to identify which values for bias $(\hat{\beta}_0)$ and weight $(\hat{\beta}_1)$ we should select. Click the buttons to run 1, 5, 10, or 25 steps of gradient descent, and see the curve update live. The error for each iteration is shown in the bottom error chart.

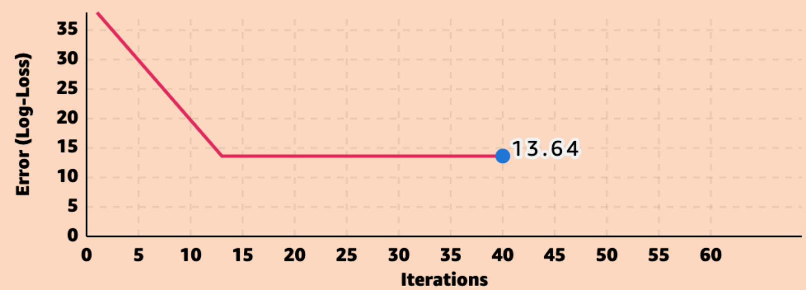
1 Step | 5 Steps | 10 Steps | 25 Steps

Weight: 0.1574

Bias: -10.0007

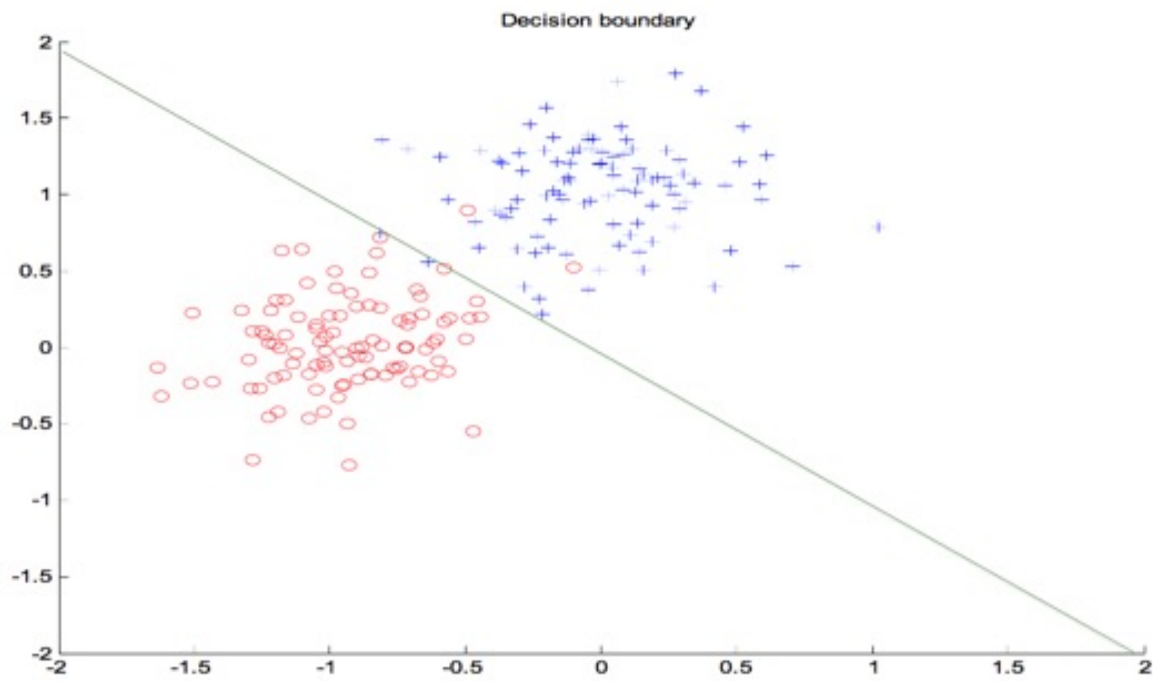
Our Model: $P(y=1|x) = \frac{1}{1+e^{-(-10.0007+0.1574x)}}$





Classification with Logistic Regression

- If f(x) = P(y=1 | x) > 0.5 then choose Class 1
  Else, choose Class 0
- Result is a linear boundary
  - Note that all points x on the boundary have output = 0.5, corresponding to net input of 0, which is a linear function of x.



- Summary:
  - Linear Regression and Logistic Regression can both be solved by Gradient descent, with appropriate loss functions.
    - MSE loss in linear regression and Binary Cross-Entropy for Logistic regression

Multi-class classification:

- If we have a multi-class classification problem, we can use multinomial logistic regression, or softmax logistic regression.
- In that case, we assign an input **x** into the class for which $P(y_i \mid \mathbf{x})$ is largest.
- Resulting decision boundaries are piece-wise linear