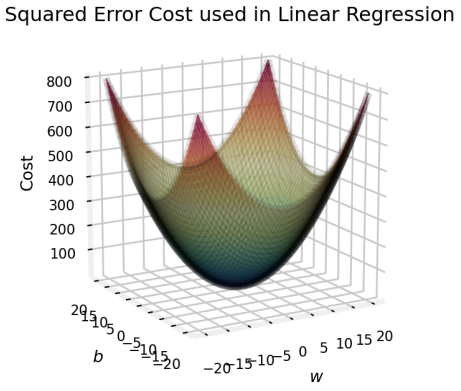
Cost function for logistic regression

Cost Function

Overview

The cost function for logistic regression differs from **linear regression** because the squared error cost function is non-convex, leading to potential local minima during optimization.

Logistic regression uses a convex cost function instead of squared error to avoid local minima and ensure convergence to a global minimum. It penalizes incorrect predictions to better match true labels.



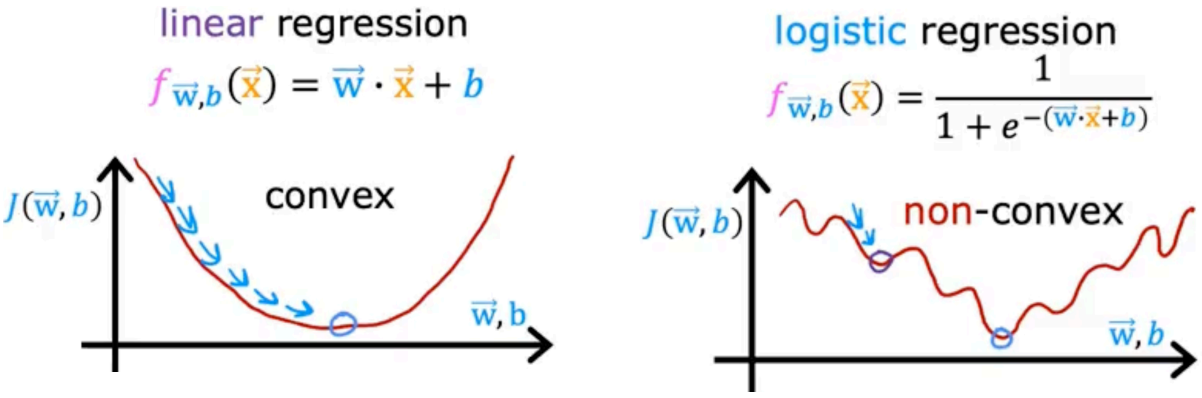
source: w3lab4

Comparison with Linear Regression

In linear regression, the cost function is based on squared error:

D

However, this squared error approach is unsuitable for logistic regression due to its non-convex nature when applied to the sigmoid function's output. Instead, a specialized loss function is used.



Defining the Loss Function

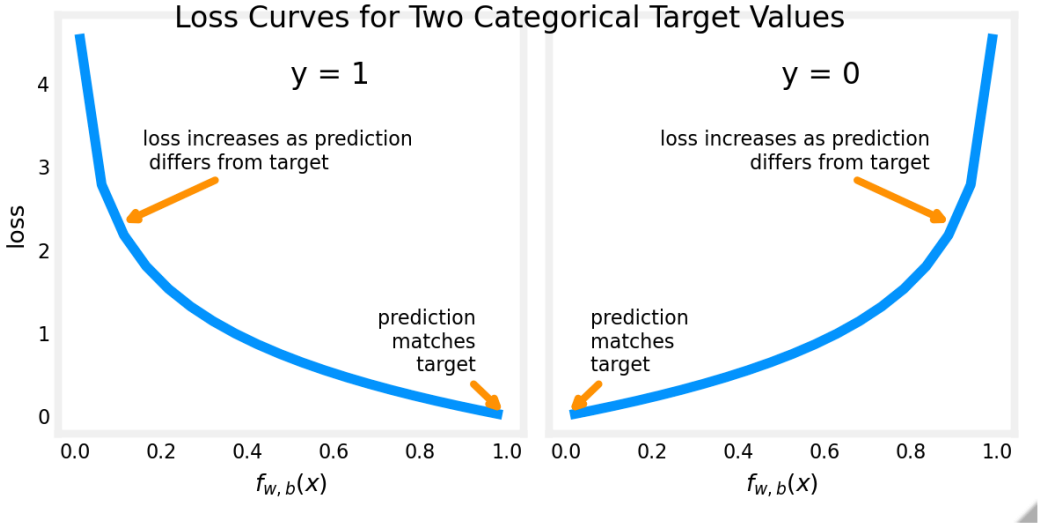
The loss function for logistic regression, $L\left(f_{\vec{w},b}(\vec{x}^i), y^i\right)$ or $loss\left(f_{\vec{w},b}(\vec{x}^i), y^i\right)$, depends on the true label y^i (0 or 1) and the predicted probability $f_{\vec{w},b}(\vec{x}^i)$. It is defined as:

Semi-simplified Loss Function

D

This formulation ensures that the loss increases significantly when predictions deviate from the true label.

For example, if $y^i = 1$ and $f_{\vec{w},b}(\vec{x}^i)$ approaches 0, the loss $(-\log(f_{\vec{w},b}(\vec{x}^i)))$ approaches ∞, strongly penalizing **incorrect predictions**.



- With $y^{(i)} = 0$ (real value), if $f_{\vec{w},b}(\vec{x}^i) \approx 1$ (predict value) then the loss: $-\log\left(1 - f_{\vec{w},b}(\vec{x}^i)\right)$ approaches ∞ (incorrect)

The loss function above can be rewritten to be easier to implement.

D

This is a rather formidable-looking equation. It is less daunting when you consider $y^{(i)}$ can have only two values, 0 and 1. One can then consider the equation in two pieces:

When $y^{(i)} = 0$, the left-hand term is eliminated:

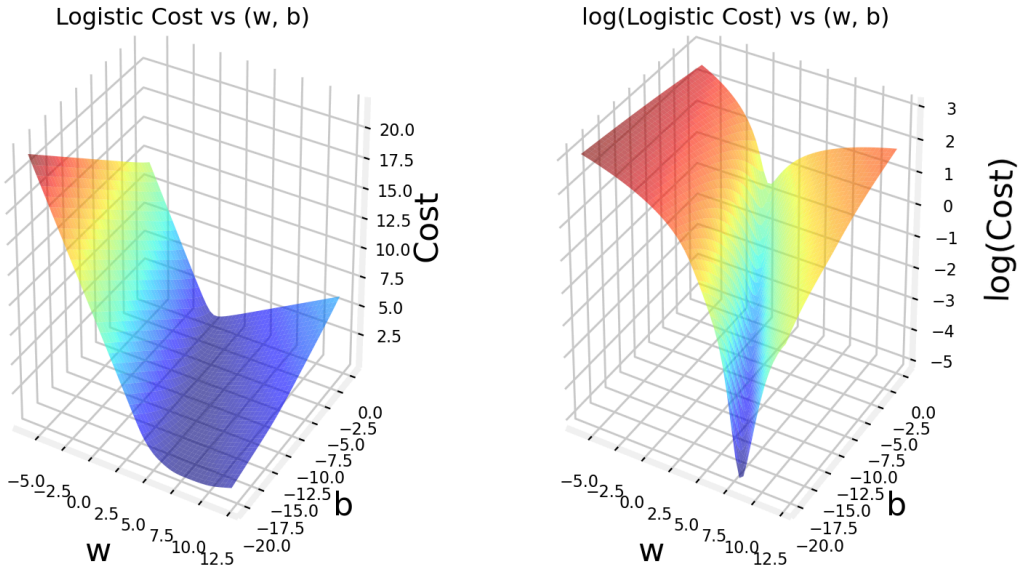
$$loss(f_{\mathbf{w},b}(\mathbf{x}^{(i)}), 0) = -(0)\log\left(f_{\mathbf{w},b}\left(\mathbf{x}^{(i)}\right)\right) - (1-0)\log\left(1 - f_{\mathbf{w},b}\left(\mathbf{x}^{(i)}\right)\right) \quad (1)$$

$$= -\log\left(1 - f_{\mathbf{w},b}\left(\mathbf{x}^{(i)}\right)\right) \quad (2)$$

And when $y^{(i)} = 1$, the right-hand term is eliminated:

$$loss(f_{\mathbf{w},b}(\mathbf{x}^{(i)}), 1) = -(1)\log\left(f_{\mathbf{w},b}\left(\mathbf{x}^{(i)}\right)\right) - (1-1)\log\left(1 - f_{\mathbf{w},b}\left(\mathbf{x}^{(i)}\right)\right) \quad (3)$$

$$= -\log\left(f_{\mathbf{w},b}\left(\mathbf{x}^{(i)}\right)\right) \quad (4)$$



This curve works well with gradient descent—it's smooth, with no plateaus, local minima, or jumps. Unlike the squared error's bowl shape, it steadily decreases even when the cost is small, as shown by both the cost and its log.

💡 **QUES:** Why is the squared error cost not used in logistic regression?

- The non-linear nature of the model results in a "wiggly", non-convex cost function with many potential local minima. 👍🏻
- The mean squared error is used for logistic regression.

⇒ If using the mean squared error for logistic regression, the cost function is "non-convex", so it's more difficult for gradient descent to find an optimal value for the parameters w and b.

Simplified Cost function for logistic regression

Simplified Loss Function

The loss function for logistic regression can be expressed as a single equation:

$$L\left(f_{\vec{w},b}(\vec{x}^i), y^{(i)}\right) = -y^{(i)} \log\left(f_{\vec{w},b}(\vec{x}^i)\right) - \left(1 - y^{(i)}\right) \log\left(1 - f_{\vec{w},b}(\vec{x}^i)\right)$$

Because y^i can only be 0 or 1, plugging in these values naturally reduces the equation to its semi-simplified form—eliminating the need to handle separate cases explicitly.

Cost Function Derivation

The cost function J is the average loss across the training set, calculated as:

$$J(\vec{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \left[L\left(f_{\vec{w},b}(\vec{x}^i), y^{(i)} \right) \right]$$

Replacing the simplified loss expression into the overall formula and reorganizing the terms leads to the final form of the **cost function**:

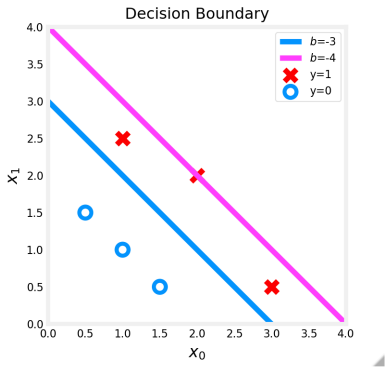
Final Cost Function

$$J(\vec{w}, b) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log\left(f_{\vec{w},b}(\vec{x}^i) \right) + \left(1 - y^{(i)} \right) \log\left(1 - f_{\vec{w},b}(\vec{x}^i) \right) \right]$$

Implications

This cost function, based on the maximum likelihood principle, is convex—ensuring a unique global minimum. This makes it well-suited for optimization with gradient descent. It drives the model toward accurate predictions by penalizing incorrect ones more heavily, thereby enhancing overall accuracy.

Examples



source: w3lab5

⇒ $b = -4$ is a worse model for the training data.

```
Cost for b = -3 :   0.36686678640551745
Cost for b = -4 :   0.5036808636748461
```

source: w3lab5

⇒ The cost function behaves as expected and the cost for $b = -4, w = [1, 1]$ is indeed higher than the cost for $b = -3, w = [1, 1]$.