Logistic Regression: From Theory to Decision Boundaries

Gregory Glickert

February 5, 2026

Introduction

This document explores logistic regression, a fundamental classification algorithm in machine learning:

- **From Linear to Logistic Regression**: We explain why linear regression fails for classification and motivate the need for a different approach.
- **The Sigmoid Function**: We introduce the sigmoid (logistic) function and explain how it transforms outputs into probabilities.
- **The Decision Boundary**: We derive the decision boundary that separates classes in feature space.
- **Binary Cross-Entropy Loss**: We explain why we use log-loss instead of mean squared error.
- **Gradient Descent for Logistic Regression**: We derive the update rules for optimizing the model.
- **Worked Example**: We train a logistic regression classifier on a simple 2D dataset.

From Linear to Logistic Regression

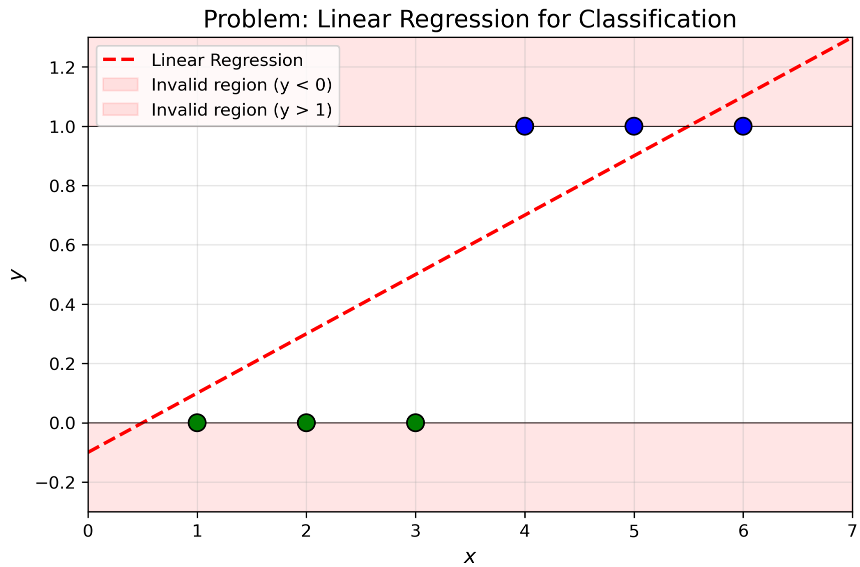
In **linear regression**, we predict a continuous output:

$$\hat{y} = w \cdot x + b.$$

However, in **classification**, we want to predict a discrete label (e.g., 0 or 1). If we use linear regression directly, the output could be any value from $-\infty$ to $+\infty$. This is problematic because:

- Probabilities must lie in $[0, 1]$.

- A prediction of $\hat{y} = 2.5$ or $\hat{y} = -0.3$ has no clear interpretation.
- Mean squared error with a linear model creates a non-convex optimization landscape.



The figure above illustrates the problem: a linear fit can produce values outside the valid range $[0, 1]$.

The Solution: Logistic Regression

Instead of directly predicting y, we pass the linear combination $z = w \cdot x + b$ through a **sigmoid function** that squashes the output to $(0, 1)$.

The Sigmoid Function

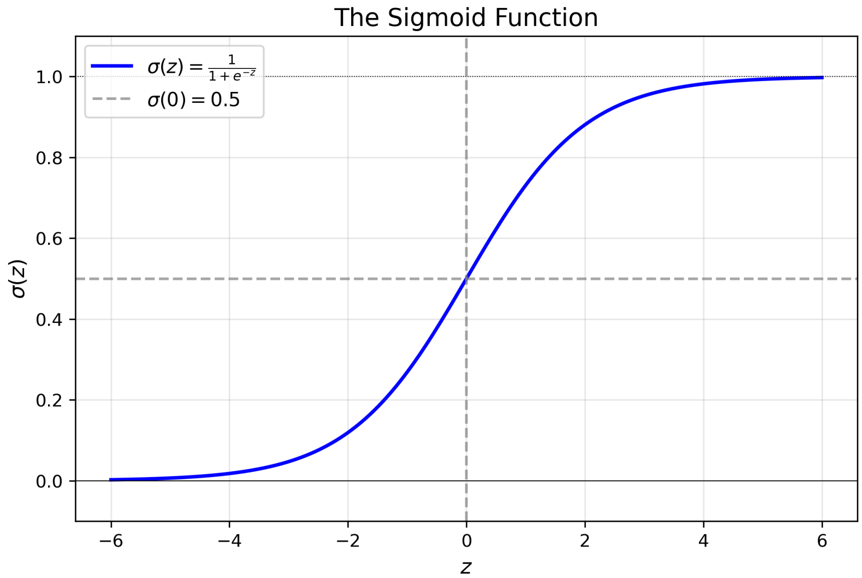
The **sigmoid function** (also called the logistic function) is defined as:

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

Key Properties

1. **Range**: The output is always in $(0, 1)$, making it interpretable as a probability.
2. **Center**: When $z = 0$, we have $\sigma(0) = \frac{1}{1+1} = 0.5$.
3. **Asymptotic behavior**:
   - As $z \to +\infty$, $\sigma(z) \to 1$.
   - As $z \to -\infty$, $\sigma(z) \to 0$.

4. **Smooth and differentiable**: This is essential for gradient-based optimization.



The Logistic Regression Model

Combining the linear function with the sigmoid, the logistic regression model is:

$$\hat{y} = \sigma(w \cdot x + b) = \frac{1}{1 + e^{-(w \cdot x + b)}}.$$

The output $\hat{y}$ is interpreted as $P(y = 1 \mid x)$, the probability that the input x belongs to class 1.

The Decision Boundary

To make a final classification, we apply a **threshold** (typically 0.5):

- If $\hat{y} \geq 0.5$, predict class 1.
- If $\hat{y} < 0.5$, predict class 0.

Deriving the Boundary

The threshold $\hat{y} = 0.5$ occurs when $\sigma(z) = 0.5$, which happens when $z = 0$.

For a two-dimensional input $x = (x_1, x_2)$ with weights $w = (w_1, w_2)$ and bias b:

$$z = w_1 x_1 + w_2 x_2 + b = 0.$$

Solving for x_2:

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2}.$$

This is a **linear equation** of the form $x_2 = m x_1 + c$, where:

- Slope: $m = -\frac{w_1}{w_2}$
- Intercept: $c = -\frac{b}{w_2}$

The decision boundary is therefore a straight line in 2D (or a hyperplane in higher dimensions).

Binary Cross-Entropy Loss

Why Not Mean Squared Error?

For linear regression, we minimize the mean squared error (MSE). However, MSE is problematic for logistic regression:

1. **Non-convexity**: When combined with the sigmoid, MSE creates a cost surface with many local minima.
2. **Weak gradients**: When the prediction is far from the target, the sigmoid saturates and gradients become very small.

The Log-Loss Function

Instead, we use **Binary Cross-Entropy** (also called log-loss):

$$J(w, b) = -\frac{1}{m} \sum_{i=1}^{m} \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right].$$

Intuition

- When $y_i = 1$: The cost is $-\log(\hat{y}_i)$. If $\hat{y}_i \approx 1$, cost ≈ 0. If $\hat{y}_i \approx 0$, cost $\to \infty$.
- When $y_i = 0$: The cost is $-\log(1 - \hat{y}_i)$. If $\hat{y}_i \approx 0$, cost ≈ 0. If $\hat{y}_i \approx 1$, cost $\to \infty$.

This function heavily penalizes "confident and wrong" predictions, which is exactly what we want.

Convexity

A key advantage of binary cross-entropy is that it is **convex** when used with the sigmoid function, guaranteeing a unique global minimum.

Gradient Descent for Logistic Regression

To minimize the cost function, we use gradient descent.

Computing the Gradients

The gradients of $J(w, b)$ with respect to the parameters are:

$$\frac{\partial J}{\partial w} = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i) x_i,$$

$$\frac{\partial J}{\partial b} = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i).$$

Note: These have the same form as linear regression gradients, but $\hat{y}_i = \sigma(w \cdot x_i + b)$ instead of a linear prediction.

Update Rules

At each iteration, we update the parameters:

$$w := w - \alpha \cdot \frac{\partial J}{\partial w} = w - \frac{\alpha}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i) x_i,$$

$$b := b - \alpha \cdot \frac{\partial J}{\partial b} = b - \frac{\alpha}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i),$$

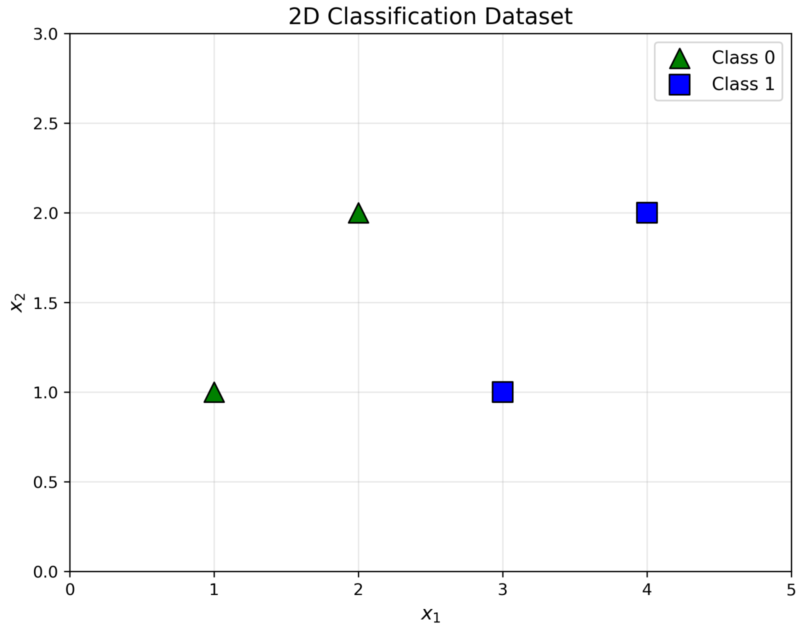
where α is the learning rate.

Worked Example: 2D Classification

We now apply logistic regression to a simple 2D dataset with 4 data points.

1. The Dataset

Point	x_1	x_2	y (class)
1	1	1	0
2	2	2	0
3	3	1	1
4	4	2	1



2. Model Setup

Our model is:

$$\hat{y} = \sigma(w_1 x_1 + w_2 x_2 + b).$$

We initialize:

- Weights: $w_1 = 0$, $w_2 = 0$
- Bias: $b = 0$
- Learning rate: $\alpha = 0.5$

3. Initial State (Iteration 0)

With $w = (0, 0)$ and $b = 0$, for any input x:

$$z = 0 \cdot x_1 + 0 \cdot x_2 + 0 = 0 \quad \Rightarrow \quad \hat{y} = \sigma(0) = 0.5.$$

All predictions are 0.5, meaning the model has no discriminative power yet.
The initial cost is:

$$J = -\frac{1}{4} \sum_{i=1}^{4} [y_i \log(0.5) + (1 - y_i) \log(0.5)] = -\log(0.5) = \ln(2) \approx 0.693.$$

4. First Gradient Descent Step

Compute the gradients. Since $\hat{y}_i = 0.5$ for all points and the true labels are $y = (0, 0, 1, 1)$:

$$\hat{y}_i - y_i = (0.5 - 0, 0.5 - 0, 0.5 - 1, 0.5 - 1) = (0.5, 0.5, -0.5, -0.5).$$

Gradient for w_1:

$$\frac{\partial J}{\partial w_1} = \frac{1}{4}\sum_{i=1}^{4}(\hat{y}_i - y_i)x_{i,1} = \frac{1}{4}[0.5(1) + 0.5(2) + (-0.5)(3) + (-0.5)(4)]$$

$$= \frac{1}{4}[0.5 + 1 - 1.5 - 2] = \frac{-2}{4} = -0.5.$$

Gradient for w_2:

$$\frac{\partial J}{\partial w_2} = \frac{1}{4}\sum_{i=1}^{4}(\hat{y}_i - y_i)x_{i,2} = \frac{1}{4}[0.5(1) + 0.5(2) + (-0.5)(1) + (-0.5)(2)]$$

$$= \frac{1}{4}[0.5 + 1 - 0.5 - 1] = 0.$$

Gradient for b:

$$\frac{\partial J}{\partial b} = \frac{1}{4}[0.5 + 0.5 - 0.5 - 0.5] = 0.$$

Update parameters:

$$w_1 := 0 - 0.5 \cdot (-0.5) = 0.25,$$

$$w_2 := 0 - 0.5 \cdot 0 = 0,$$

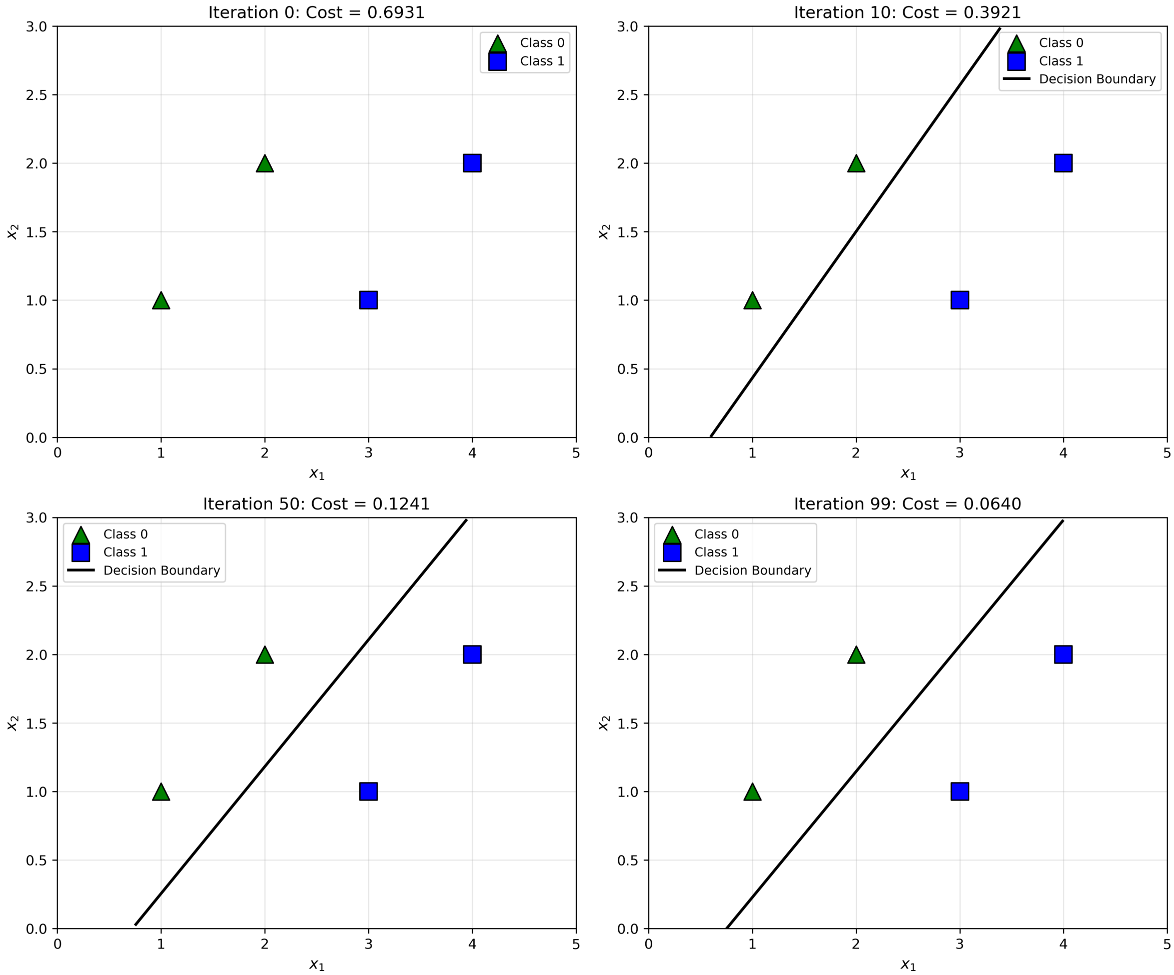
$$b := 0 - 0.5 \cdot 0 = 0.$$

After iteration 1: $w = (0.25, 0)$, $b = 0$, Cost ≈ 0.625.

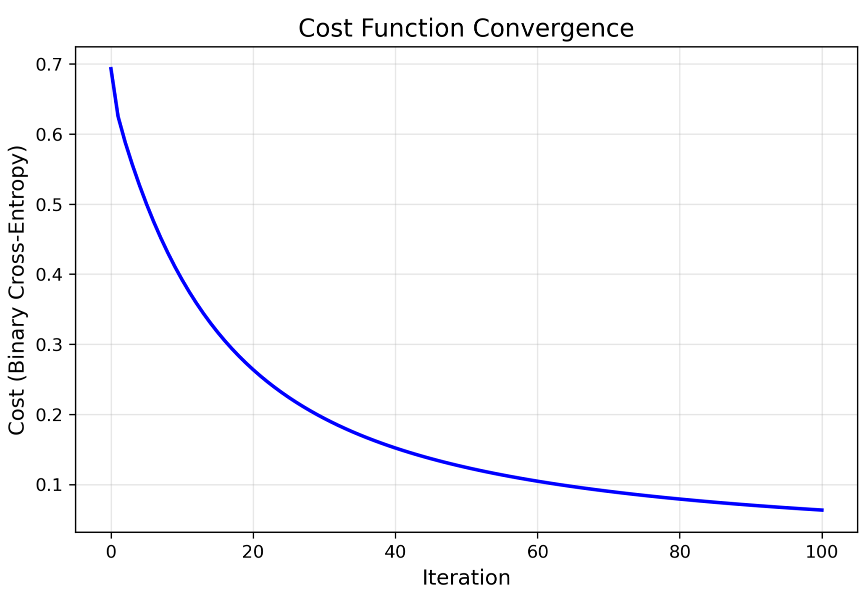
5. Gradient Descent Progress

Continuing the iterations:

Iteration	w_1	w_2	b	Cost
0	0.0000	0.0000	0.0000	0.6931
1	0.2500	0.0000	0.0000	0.6250
2	0.2789	-0.1185	-0.0744	0.5882
3	0.3571	-0.2037	-0.1280	0.5565
10	0.7863	-0.7354	-0.4687	0.3921
50	2.0726	-2.2368	-1.5028	0.1241
100	2.7805	-3.0275	-2.0951	0.0633



6. Cost Convergence



The cost decreases monotonically, demonstrating the convexity of the binary cross-entropy loss.

7. Final Decision Boundary

After 100 iterations, the learned parameters are:

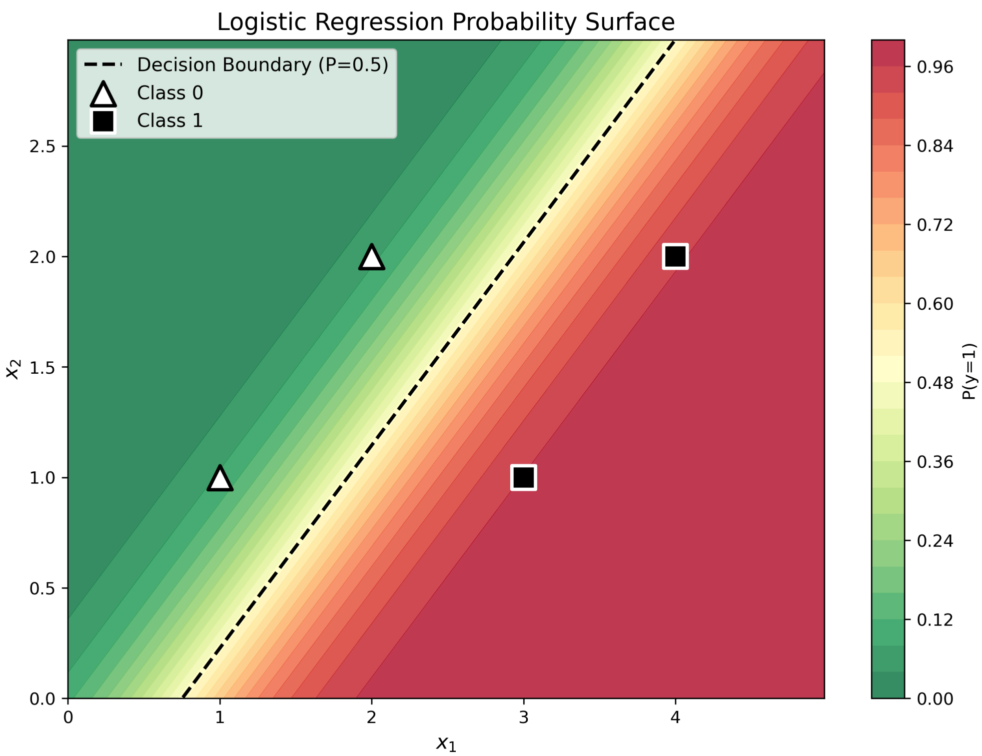
$$w_1 \approx 2.78, \quad w_2 \approx -3.03, \quad b \approx -2.10.$$

The decision boundary is where $w_1 x_1 + w_2 x_2 + b = 0$:

$$2.78x_1 - 3.03x_2 - 2.10 = 0.$$

Solving for x_2:

$$x_2 = \frac{2.78}{3.03}x_1 - \frac{2.10}{3.03} \approx 0.92x_1 - 0.69.$$



The probability surface shows smooth transitions from low probability (green, class 0) to high probability (red, class 1), with the decision boundary at $P = 0.5$.

Conclusion

We have covered the key concepts of logistic regression:

1. **The Problem**: Linear regression outputs unbounded values; classification needs probabilities.

2. **The Sigmoid Function**: Transforms $z \in (-\infty, \infty)$ to $(0, 1)$.
3. **The Decision Boundary**: A linear surface where $w \cdot x + b = 0$.
4. **Binary Cross-Entropy**: A convex loss function that penalizes incorrect confident predictions.
5. **Gradient Descent**: Iteratively minimizes the loss by updating w and b.

Connection to Neural Networks

Logistic regression can be viewed as a **single-layer neural network** with one neuron and a sigmoid activation. By stacking multiple layers of such units with non-linear activations, we obtain deep neural networks capable of learning complex, non-linear decision boundaries.