Single Neuron Learning:
Linear and Logistic Regression

Complete Mathematical Framework with Worked Examples

September 2, 2025

Contents

1 Introduction

This document provides a comprehensive mathematical treatment of single neuron learning for two fundamental machine learning tasks:

- **Linear Regression**: Predicting continuous output values
- **Logistic Regression**: Predicting binary classification probabilities

Both models use the same basic neuron structure but differ in their activation functions, loss functions, and interpretation of outputs.

2 Single Neuron Architecture

2.1 General Structure

A single neuron with n inputs has:

- **Inputs**: $x_1, x_2, \ldots, x_n$
- **Weights**: $w_1, w_2, \ldots, w_n$
- **Bias**: b
- **Pre-activation**: $z = \sum_{i=1}^{n} w_i x_i + b = \mathbf{w}^T\mathbf{x} + b$
- **Output**: $y = f(z)$ where f is the activation function

2.2 Example Setup

For our worked examples, we'll use a neuron with 3 inputs:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{1}$$

$$\mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} \tag{2}$$

Initial parameters:

$$\mathbf{w}_{\text{init}} = \begin{pmatrix} 0.5 \\ -0.3 \\ 0.8 \end{pmatrix} \tag{3}$$

$$b_{\text{init}} = 0.2 \tag{4}$$

3 Part I: Linear Regression

3.1 Model Definition

In linear regression, we use a linear activation function (identity function):

$$z = \mathbf{w}^T\mathbf{x} + b = \sum_{i=1}^{3} w_i x_i + b \tag{5}$$

$$y = f(z) = z \quad \text{(linear activation)} \tag{6}$$

The output y directly represents the predicted continuous value.

3.2 Loss Function

We use Mean Squared Error (MSE) loss:

$$L = \frac{1}{2}(y-t)^2 = \frac{1}{2}(z-t)^2 \tag{7}$$

where t is the target (true) value.

3.3 Gradient Computation

3.3.1 Loss Derivatives

$$\frac{\partial L}{\partial y} = \frac{\partial}{\partial y}\left[\frac{1}{2}(y-t)^2\right] = y - t \tag{8}$$

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial z} = (y-t) \cdot 1 = y - t = z - t \tag{9}$$

3.3.2 Parameter Gradients

For weights:

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial w_i} = (z-t) \cdot x_i \tag{10}$$

For bias:

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial b} = (z-t) \cdot 1 = z - t \tag{11}$$

3.4 Worked Example: Linear Regression

3.4.1 Training Data

Let's use the following training example:

$$\mathbf{x} = \begin{pmatrix} 2.0 \\ 1.5 \\ -0.5 \end{pmatrix} \tag{12}$$

$$t = 3.2 \quad \text{(target continuous value)} \tag{13}$$

3.4.2 Forward Pass

$$\begin{aligned} z &= w_1x_1 + w_2x_2 + w_3x_3 + b && (14)\\ &= 0.5(2.0) + (-0.3)(1.5) + 0.8(-0.5) + 0.2 && (15)\\ &= 1.0 - 0.45 - 0.4 + 0.2 && (16)\\ &= 0.35 && (17) \end{aligned}$$

Since we use linear activation:

$$y = z = 0.35 \tag{18}$$

3.4.3 Loss Calculation

$$\begin{aligned} L &= \frac{1}{2}(y-t)^2 = \frac{1}{2}(0.35 - 3.2)^2 && (19)\\ &= \frac{1}{2}(-2.85)^2 = \frac{1}{2}(8.1225) = 4.061 && (20) \end{aligned}$$

3.4.4 Backward Pass

Error term:

$$\frac{\partial L}{\partial z} = z - t = 0.35 - 3.2 = -2.85 \tag{21}$$

Weight gradients:

$$\frac{\partial L}{\partial w_1} = (-2.85) \times 2.0 = -5.70 \tag{22}$$

$$\frac{\partial L}{\partial w_2} = (-2.85) \times 1.5 = -4.275 \tag{23}$$

$$\frac{\partial L}{\partial w_3} = (-2.85) \times (-0.5) = 1.425 \tag{24}$$

Bias gradient:

$$\frac{\partial L}{\partial b} = -2.85 \tag{25}$$

3.4.5 Parameter Updates

Using learning rate $\alpha = 0.1$:

$$w_1^{\text{new}} = 0.5 - 0.1(-5.70) = 0.5 + 0.57 = 1.07 \tag{26}$$

$$w_2^{\text{new}} = -0.3 - 0.1(-4.275) = -0.3 + 0.4275 = 0.1275 \tag{27}$$

$$w_3^{\text{new}} = 0.8 - 0.1(1.425) = 0.8 - 0.1425 = 0.6575 \tag{28}$$

$$b^{\text{new}} = 0.2 - 0.1(-2.85) = 0.2 + 0.285 = 0.485 \tag{29}$$

4 Part II: Logistic Regression

4.1 Model Definition

In logistic regression, we use the sigmoid activation function:

$$z = \mathbf{w}^T\mathbf{x} + b = \sum_{i=1}^{3} w_i x_i + b \tag{30}$$

$$y = \sigma(z) = \frac{1}{1 + e^{-z}} \quad \text{(sigmoid activation)} \tag{31}$$

The output y represents the probability of the positive class (i.e., $P(class = 1)$).

4.2 Sigmoid Function Properties

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{32}$$

$$\sigma'(z) = \frac{d}{dz}\sigma(z) = \sigma(z)(1 - \sigma(z)) \tag{33}$$

Key properties:

- Range: $(0, 1)$ - perfect for probabilities
- $\sigma(0) = 0.5$
- $\lim_{z \to \infty} \sigma(z) = 1$
- $\lim_{z \to -\infty} \sigma(z) = 0$

4.3 Loss Function

We use Binary Cross-Entropy (BCE) loss:

$$L = -[t \log(y) + (1-t)\log(1-y)] \tag{34}$$

where $t \in \{0, 1\}$ is the true binary label.

This can be rewritten as:

$$L = \begin{cases} -\log(y) & \text{if } t = 1 \\ -\log(1-y) & \text{if } t = 0 \end{cases} \tag{35}$$

4.4 Gradient Computation

4.4.1 Loss Derivatives

$$\frac{\partial L}{\partial y} = -\frac{t}{y} + \frac{1-t}{1-y} = \frac{y-t}{y(1-y)} \tag{36}$$

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial z} \tag{37}$$

$$= \frac{y-t}{y(1-y)} \cdot y(1-y) \tag{38}$$

$$= y - t \tag{39}$$

Note the elegant simplification: $\frac{\partial L}{\partial z} = y - t$

4.4.2 Parameter Gradients

For weights:

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial w_i} = (y-t) \cdot x_i \tag{40}$$

For bias:

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial b} = (y-t) \cdot 1 = y - t \tag{41}$$

4.5 Worked Example: Logistic Regression

4.5.1 Training Data

Let's use the following training example:

$$\mathbf{x} = \begin{pmatrix} 1.2 \\ -0.8 \\ 0.6 \end{pmatrix} \tag{42}$$

$$t = 1 \quad \text{(positive class)} \tag{43}$$

Starting with the same initial parameters:

$$\mathbf{w} = \begin{pmatrix} 0.5 \\ -0.3 \\ 0.8 \end{pmatrix} \tag{44}$$

$$b = 0.2 \tag{45}$$

4.5.2 Forward Pass

$$\begin{aligned} z &= w_1x_1 + w_2x_2 + w_3x_3 + b && (46)\\ &= 0.5(1.2) + (-0.3)(-0.8) + 0.8(0.6) + 0.2 && (47)\\ &= 0.6 + 0.24 + 0.48 + 0.2 && (48)\\ &= 1.52 && (49) \end{aligned}$$

Sigmoid activation:

$$\begin{aligned} y = \sigma(1.52) &= \frac{1}{1 + e^{-1.52}} && (50)\\ &= \frac{1}{1 + 0.219} = \frac{1}{1.219} = 0.820 && (51) \end{aligned}$$

4.5.3 Loss Calculation

Since $t = 1$ (positive class):

$$\begin{aligned} L &= -\log(y) = -\log(0.820) && (52)\\ &= -(-0.198) = 0.198 && (53) \end{aligned}$$

4.5.4 Backward Pass

Error term:

$$\frac{\partial L}{\partial z} = y - t = 0.820 - 1 = -0.180 \quad (54)$$

Weight gradients:

$$\begin{aligned} \frac{\partial L}{\partial w_1} &= (-0.180) \times 1.2 = -0.216 && (55)\\ \frac{\partial L}{\partial w_2} &= (-0.180) \times (-0.8) = 0.144 && (56)\\ \frac{\partial L}{\partial w_3} &= (-0.180) \times 0.6 = -0.108 && (57) \end{aligned}$$

Bias gradient:

$$\frac{\partial L}{\partial b} = -0.180 \quad (58)$$

4.5.5 Parameter Updates

Using learning rate $\alpha = 0.1$:

$$\begin{aligned} w_1^{\text{new}} &= 0.5 - 0.1(-0.216) = 0.5 + 0.0216 = 0.5216 && (59)\\ w_2^{\text{new}} &= -0.3 - 0.1(0.144) = -0.3 - 0.0144 = -0.3144 && (60)\\ w_3^{\text{new}} &= 0.8 - 0.1(-0.108) = 0.8 + 0.0108 = 0.8108 && (61)\\ b^{\text{new}} &= 0.2 - 0.1(-0.180) = 0.2 + 0.018 = 0.218 && (62) \end{aligned}$$

5 Comparison: Linear vs Logistic Regression

5.1 Key Differences

5.2 Similarities

Both models share:

Aspect	**Linear Regression**	**Logistic Regression**
Activation Function	$f(z) = z$ (identity)	$f(z) = \frac{1}{1+e^{-z}}$ (sigmoid)
Output Range	$(-\infty, +\infty)$	$(0, 1)$
Output Interpretation	Continuous value	Probability
Loss Function	MSE: $\frac{1}{2}(y-t)^2$	BCE: $-[t \log y + (1-t)\log(1-y)]$
$\frac{\partial L}{\partial z}$	$z - t$	$y - t$
Use Case	Regression	Binary Classification

Table 1: Comparison of Linear and Logistic Regression

- Same neuron architecture with weights and bias
- Same gradient descent update rules
- Linear combination of inputs: $z = \mathbf{w}^T\mathbf{x} + b$
- Similar parameter gradient forms: $(error) \times (input)$

6 Multi-Example Training

6.1 Batch Gradient Descent

For a dataset with m examples, we average the gradients:

6.1.1 Linear Regression

$$\frac{\partial L}{\partial w_i} = \frac{1}{m}\sum_{j=1}^{m}(z^{(j)} - t^{(j)})x_i^{(j)} \tag{63}$$

$$\frac{\partial L}{\partial b} = \frac{1}{m}\sum_{j=1}^{m}(z^{(j)} - t^{(j)}) \tag{64}$$

6.1.2 Logistic Regression

$$\frac{\partial L}{\partial w_i} = \frac{1}{m}\sum_{j=1}^{m}(y^{(j)} - t^{(j)})x_i^{(j)} \tag{65}$$

$$\frac{\partial L}{\partial b} = \frac{1}{m}\sum_{j=1}^{m}(y^{(j)} - t^{(j)}) \tag{66}$$

6.2 Extended Example: Multiple Training Points

Let's train both models on multiple data points:

6.2.1 Linear Regression Dataset

$$\text{Example 1: } \mathbf{x}^{(1)} = (2.0, 1.5, -0.5)^T, \quad t^{(1)} = 3.2 \tag{67}$$

$$\text{Example 2: } \mathbf{x}^{(2)} = (1.0, -1.0, 2.0)^T, \quad t^{(2)} = 1.8 \tag{68}$$

$$\text{Example 3: } \mathbf{x}^{(3)} = (-0.5, 0.8, 1.2)^T, \quad t^{(3)} = 0.5 \tag{69}$$

6.2.2 Forward Pass for All Examples

Using initial parameters $\mathbf{w} = (0.5, -0.3, 0.8)^T, b = 0.2$:

Example 1:

$$z^{(1)} = 0.5(2.0) + (-0.3)(1.5) + 0.8(-0.5) + 0.2 = 0.35 \tag{70}$$
$$y^{(1)} = z^{(1)} = 0.35 \tag{71}$$

Example 2:

$$z^{(2)} = 0.5(1.0) + (-0.3)(-1.0) + 0.8(2.0) + 0.2 = 2.6 \tag{72}$$
$$y^{(2)} = z^{(2)} = 2.6 \tag{73}$$

Example 3:

$$z^{(3)} = 0.5(-0.5) + (-0.3)(0.8) + 0.8(1.2) + 0.2 = 0.915 \tag{74}$$
$$y^{(3)} = z^{(3)} = 0.915 \tag{75}$$

6.2.3 Batch Gradient Calculation

Error terms:

$$e^{(1)} = z^{(1)} - t^{(1)} = 0.35 - 3.2 = -2.85 \tag{76}$$
$$e^{(2)} = z^{(2)} - t^{(2)} = 2.6 - 1.8 = 0.8 \tag{77}$$
$$e^{(3)} = z^{(3)} - t^{(3)} = 0.915 - 0.5 = 0.415 \tag{78}$$

Weight gradients (averaged over 3 examples):

$$\frac{\partial L}{\partial w_1} = \frac{1}{3}[(-2.85)(2.0) + (0.8)(1.0) + (0.415)(-0.5)] \tag{79}$$
$$= \frac{1}{3}[-5.7 + 0.8 - 0.2075] = \frac{-5.1075}{3} = -1.702 \tag{80}$$

$$\frac{\partial L}{\partial w_2} = \frac{1}{3}[(-2.85)(1.5) + (0.8)(-1.0) + (0.415)(0.8)] \tag{81}$$
$$= \frac{1}{3}[-4.275 - 0.8 + 0.332] = \frac{-4.743}{3} = -1.581 \tag{82}$$

$$\frac{\partial L}{\partial w_3} = \frac{1}{3}[(-2.85)(-0.5) + (0.8)(2.0) + (0.415)(1.2)] \tag{83}$$
$$= \frac{1}{3}[1.425 + 1.6 + 0.498] = \frac{3.523}{3} = 1.174 \tag{84}$$

$$\frac{\partial L}{\partial b} = \frac{1}{3}[(-2.85) + (0.8) + (0.415)] \tag{85}$$
$$= \frac{-1.635}{3} = -0.545 \tag{86}$$

6.2.4 Parameter Updates

Using $\alpha = 0.1$:

$$w_1^{\text{new}} = 0.5 - 0.1(-1.702) = 0.5 + 0.1702 = 0.670 \tag{87}$$
$$w_2^{\text{new}} = -0.3 - 0.1(-1.581) = -0.3 + 0.1581 = -0.142 \tag{88}$$
$$w_3^{\text{new}} = 0.8 - 0.1(1.174) = 0.8 - 0.1174 = 0.683 \tag{89}$$
$$b^{\text{new}} = 0.2 - 0.1(-0.545) = 0.2 + 0.0545 = 0.255 \tag{90}$$

7 Implementation Considerations

7.1 Learning Rate Selection

- **Too high**: Oscillation, divergence, overshooting minimum
- **Too low**: Slow convergence, many iterations needed
- **Typical values**: 0.001 to 0.1
- **Adaptive methods**: Adam, RMSprop adjust learning rate automatically

7.2 Convergence Criteria

Stop training when:

- Loss change between iterations $< \epsilon$ (e.g., $\epsilon = 10^{-6}$)
- Maximum number of iterations reached
- Gradient magnitude $< \epsilon$

7.3 Numerical Stability

7.3.1 Logistic Regression

For very large $|z|$, direct computation of $\sigma(z)$ can cause:

- **Overflow**: When z is very positive
- **Underflow**: When z is very negative

Stable implementations use:

$$\log(\sigma(z)) = \begin{cases} -\log(1 + e^{-z}) & \text{if } z \geq 0 \\ z - \log(1 + e^{z}) & \text{if } z < 0 \end{cases} \tag{91}$$

8 Summary and Key Takeaways

8.1 Mathematical Framework

Both linear and logistic regression follow the same fundamental pattern:

1. **Forward pass**: Compute $z = \mathbf{w}^T\mathbf{x} + b$, then $y = f(z)$
2. **Loss computation**: Evaluate appropriate loss function
3. **Backward pass**: Compute $\frac{\partial L}{\partial z}$
4. **Gradients**: $\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial z} \cdot x_i$, $\frac{\partial L}{\partial b} = \frac{\partial L}{\partial z}$
5. **Updates**: $\theta \leftarrow \theta - \alpha \nabla_\theta L$

8.2 Key Insights

- The choice of activation function and loss function determines the model type
- Gradient computation follows the chain rule consistently
- Both models have closed-form gradient expressions
- The error term $\frac{\partial L}{\partial z}$ has elegant forms in both cases
- Parameter updates follow identical patterns regardless of model type

8.3 Practical Applications

- **Linear regression**: House prices, stock predictions, temperature forecasting
- **Logistic regression**: Email spam detection, medical diagnosis, marketing response

This mathematical foundation extends naturally to multi-layer networks where the same principles apply layer by layer through the backpropagation algorithm.