Practice Exam: CS 583 Deep Learning

Solutions

Instructor: Justo E. Karell

Instructions

This practice exam consists of 8 problems. Answer each question, showing all steps. For handwritten work, please upload a clear photo of your work. Alternatively, you can type out your solutions in LaTeX or any document editor. The total points are 100, distributed as indicated.

Problem 1: Mathematical Modeling (10 points)

Problem: You are tasked with predicting house prices based on features such as size, number of rooms, and location. Formulate this problem as a mathematical model. Identify the goal, problem type, assumptions, and an appropriate model for this task. Be sure to clearly define your input variables and output target.

Solution

Objective:

We aim to predict house prices based on specific features such as size, number of rooms, and location. This involves creating a mathematical model that relates these features to the house price.

1. Identify the Goal:

- **Goal:** Develop a predictive model to estimate the price of a house given its features.

2. Define the Problem Type:

- **Problem Type:** This is a **supervised learning regression problem**. We have labeled data (house features and their corresponding prices), and we want to predict a continuous target variable (price).

3. Define Input Variables and Output Target:

Let's define our variables:

- **Input Variables (Features):**

- X_1 = Size of the house (in square feet)
- X_2 = Number of rooms
- X_3 = Location index (a numerical representation of location desirability)

- **Output Target:**

- Y = Price of the house (in dollars)

So, our input feature vector is:

$$\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}$$

4. Formulate the Mathematical Model:

We can model the relationship between the inputs and the output using a linear regression model:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$$

Where:

- β_0 is the intercept term.
- $\beta_1, \beta_2, \beta_3$ are the coefficients (weights) that represent the impact of each feature on the house price.
- ϵ is the error term (residual) accounting for the difference between the observed and predicted prices.

Alternatively, in vector form:

$$Y = \beta_0 + \boldsymbol{\beta}^\top \mathbf{X} + \epsilon$$

Where:

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}$$

5. State the Assumptions:

For the linear regression model to be valid, we make several assumptions:

1. **Linearity:** The relationship between the dependent variable and the independent variables is linear.
$$E[Y|\mathbf{X}] = \beta_0 + \boldsymbol{\beta}^\top \mathbf{X}$$
2. **Independence:** The observations are independent of each other.
3. **Homoscedasticity:** The variance of the error terms is constant across all levels of the independent variables.
$$\mathrm{Var}(\epsilon|\mathbf{X}) = \sigma^2$$
4. **Normality of Errors:** The error terms are normally distributed.
$$\epsilon \sim N(0, \sigma^2)$$
5. **No Multicollinearity:** The independent variables are not perfectly linearly related.

6. Choose an Appropriate Model:
Given the assumptions and the nature of the problem, a **multiple linear regression model** is appropriate. This model will allow us to estimate the coefficients β_i using methods like Ordinary Least Squares (OLS).
7. Model Estimation:
To estimate the coefficients, we minimize the sum of squared residuals:
$$\min_{\beta_0, \boldsymbol{\beta}} \sum_{i=1}^{n} \left(Y_i - \beta_0 - \boldsymbol{\beta}^\top \mathbf{X}_i \right)^2$$
Where:

- n is the number of observations (houses in our dataset).
- $\mathbf{X}_i$ is the feature vector for the i-th house.
- Y_i is the actual price of the i-th house.

8. Model Interpretation:

- **Intercept (β_0):** Represents the expected house price when all features are zero (may not have practical meaning depending on the context).
- **Coefficients (β_i):** Represent the expected change in house price for a one-unit increase in the corresponding feature, holding other features constant.
  - For example, β_1 indicates how much the house price is expected to increase per additional square foot.

Conclusion:
We have formulated the problem as a multiple linear regression model, defined our variables, stated our assumptions, and identified the goal and problem type. This mathematical model can now be used to predict house prices based on the given features.

Problem 2: Neural Network Output Calculation (15 points)

Problem: Consider a neural network with:

- Input vector: $\mathbf{x} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$
- Hidden layer 1 weights:
$$\mathbf{W}_1 = \begin{bmatrix} 0.5 & -0.3 \\ 0.2 & 0.8 \end{bmatrix}$$
- Hidden layer 1 bias:
$$\mathbf{b}_1 = \begin{bmatrix} 0.1 \\ -0.2 \end{bmatrix}$$
- Hidden layer 2 weights:
$$\mathbf{W}_2 = \begin{bmatrix} 0.4 & 0.1 \\ -0.5 & 0.3 \end{bmatrix}$$
- Hidden layer 2 bias:
$$\mathbf{b}_2 = \begin{bmatrix} 0.05 \\ 0.1 \end{bmatrix}$$

Both hidden layers use the sigmoid activation function:

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

Task:

1. Write out the formula used to compute the output of each layer, including the activation function.
2. Compute the output of the neural network after the second hidden layer.

Solution

1. Formulas for Computing Layer Outputs:
For each layer l, the computations are as follows:

- **Linear Transformation:**
$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}$$
Where:
  - $\mathbf{z}^{(l)}$ is the pre-activation output (weighted sum plus bias) of layer l.
  - $\mathbf{W}^{(l)}$ is the weight matrix for layer l.
  - $\mathbf{h}^{(l-1)}$ is the output from the previous layer.
    - For the first hidden layer, $\mathbf{h}^{(0)} = \mathbf{x}$ (the input vector).
  - $\mathbf{b}^{(l)}$ is the bias vector for layer l.
- **Activation Function (Sigmoid):**
$$\mathbf{h}^{(l)} = \sigma(\mathbf{z}^{(l)}) = \frac{1}{1 + e^{-\mathbf{z}^{(l)}}}$$
Where σ is applied element-wise to $\mathbf{z}^{(l)}$.

2. Compute the Output After the Second Hidden Layer:
Given Data:

- **Input Vector:**

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

- **Hidden Layer 1 Weights and Biases:**

$$\mathbf{W}_1 = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} = \begin{bmatrix} 0.5 & -0.3 \\ 0.2 & 0.8 \end{bmatrix}$$

$$\mathbf{b}_1 = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \end{bmatrix} = \begin{bmatrix} 0.1 \\ -0.2 \end{bmatrix}$$

- **Hidden Layer 2 Weights and Biases:**

$$\mathbf{W}_2 = \begin{bmatrix} w_{11}^{(2)} & w_{12}^{(2)} \\ w_{21}^{(2)} & w_{22}^{(2)} \end{bmatrix} = \begin{bmatrix} 0.4 & 0.1 \\ -0.5 & 0.3 \end{bmatrix}$$

$$\mathbf{b}_2 = \begin{bmatrix} b_1^{(2)} \\ b_2^{(2)} \end{bmatrix} = \begin{bmatrix} 0.05 \\ 0.1 \end{bmatrix}$$

Computations:
Step 1: Compute $\mathbf{z}^{(1)}$ (First Hidden Layer Pre-Activation):

$$\mathbf{z}^{(1)} = \mathbf{W}_1\mathbf{x} + \mathbf{b}_1$$

Calculate $\mathbf{W}_1\mathbf{x}$:

$$\mathbf{W}_1\mathbf{x} = \begin{bmatrix} 0.5 & -0.3 \\ 0.2 & 0.8 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} (0.5)(1) + (-0.3)(2) \\ (0.2)(1) + (0.8)(2) \end{bmatrix} = \begin{bmatrix} 0.5 - 0.6 \\ 0.2 + 1.6 \end{bmatrix} = \begin{bmatrix} -0.1 \\ 1.8 \end{bmatrix}$$

Add biases $\mathbf{b}_1$:

$$\mathbf{z}^{(1)} = \begin{bmatrix} -0.1 \\ 1.8 \end{bmatrix} + \begin{bmatrix} 0.1 \\ -0.2 \end{bmatrix} = \begin{bmatrix} -0.1 + 0.1 \\ 1.8 - 0.2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1.6 \end{bmatrix}$$

Step 2: Compute $\mathbf{h}^{(1)}$ (First Hidden Layer Activation):
Apply the sigmoid function element-wise:

$$\mathbf{h}^{(1)} = \sigma(\mathbf{z}^{(1)}) = \begin{bmatrix} \sigma(0) \\ \sigma(1.6) \end{bmatrix}$$

Compute each component:

- $\sigma(0) = \dfrac{1}{1 + e^0} = \dfrac{1}{1 + 1} = 0.5$
- $\sigma(1.6) = \dfrac{1}{1 + e^{-1.6}} \approx \dfrac{1}{1 + 0.2019} \approx 0.8320$

So:

$$\mathbf{h}^{(1)} = \begin{bmatrix} 0.5 \\ 0.8320 \end{bmatrix}$$

Step 3: Compute $\mathbf{z}^{(2)}$ (Second Hidden Layer Pre-Activation):

$$\mathbf{z}^{(2)} = \mathbf{W}_2\mathbf{h}^{(1)} + \mathbf{b}_2$$

Calculate $\mathbf{W}_2\mathbf{h}^{(1)}$:

$$\mathbf{W}_2\mathbf{h}^{(1)} = \begin{bmatrix} 0.4 & 0.1 \\ -0.5 & 0.3 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.8320 \end{bmatrix}$$

Compute each component:

- First element:
$$z_1^{(2)} = (0.4)(0.5) + (0.1)(0.8320) = 0.2 + 0.0832 = 0.2832$$
- Second element:
$$z_2^{(2)} = (-0.5)(0.5) + (0.3)(0.8320) = -0.25 + 0.2496 = -0.0004$$

Add biases $\mathbf{b}_2$:

$$\mathbf{z}^{(2)} = \begin{bmatrix} 0.2832 \\ -0.0004 \end{bmatrix} + \begin{bmatrix} 0.05 \\ 0.1 \end{bmatrix} = \begin{bmatrix} 0.3332 \\ 0.0996 \end{bmatrix}$$

Step 4: Compute $\mathbf{h}^{(2)}$ (Second Hidden Layer Activation):
Apply the sigmoid function element-wise:

$$\mathbf{h}^{(2)} = \sigma(\mathbf{z}^{(2)}) = \begin{bmatrix} \sigma(0.3332) \\ \sigma(0.0996) \end{bmatrix}$$

Compute each component:

- $\sigma(0.3332) = \dfrac{1}{1 + e^{-0.3332}} \approx \dfrac{1}{1 + 0.7162} \approx 0.5825$
- $\sigma(0.0996) = \dfrac{1}{1 + e^{-0.0996}} \approx \dfrac{1}{1 + 0.9053} \approx 0.5249$

So:

$$\mathbf{h}^{(2)} = \begin{bmatrix} 0.5825 \\ 0.5249 \end{bmatrix}$$

Final Output After the Second Hidden Layer:

$$\mathbf{h}^{(2)} = \begin{bmatrix} 0.5825 \\ 0.5249 \end{bmatrix}$$

Problem 3: Neural Network with Linear Output Layer (15 points)

Problem: Building upon the neural network in Problem 2, we now consider the output layer.

The neural network's final output is simply the output of the second hidden layer without any additional weights or biases (i.e., the identity function is used as the output layer).

Task:

1. Write out the mathematical formula for the neural network's output in terms of the parameters ($\mathbf{W}$ and $\mathbf{b}$), the activation function σ, and the layer outputs $h^{(l)}$.

2. Using your formula, compute the final output of the neural network.

Solution

1. Mathematical Formula for the Neural Network's Output:

Given that the output layer is the identity function (linear activation), the final output $\mathbf{y}$ is simply the output from the second hidden layer:

$$\mathbf{y} = \mathbf{h}^{(2)}$$

However, to express the output in terms of the parameters and activation functions, we can write:

- **First Hidden Layer:**
$$\mathbf{z}^{(1)} = \mathbf{W}_1\mathbf{x} + \mathbf{b}_1$$
$$\mathbf{h}^{(1)} = \sigma(\mathbf{z}^{(1)})$$

- **Second Hidden Layer:**
$$\mathbf{z}^{(2)} = \mathbf{W}_2\mathbf{h}^{(1)} + \mathbf{b}_2$$
$$\mathbf{h}^{(2)} = \sigma(\mathbf{z}^{(2)})$$

- **Output Layer:**
$$\mathbf{y} = \mathbf{h}^{(2)}$$

Alternatively, combining all steps:

$$\mathbf{y} = \sigma\left(\mathbf{W}_2\sigma\left(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1\right) + \mathbf{b}_2\right)$$

2. Compute the Final Output of the Neural Network:

From Problem 2, we have:

$$\mathbf{h}^{(2)} = \begin{bmatrix} 0.5825 \\ 0.5249 \end{bmatrix}$$

So the final output $\mathbf{y}$ is:

$$\mathbf{y} = \mathbf{h}^{(2)} = \begin{bmatrix} 0.5825 \\ 0.5249 \end{bmatrix}$$

Problem 4: Activation Function: Leaky ReLU (10 points)

Problem: Consider the Leaky ReLU activation function:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{if } x \leq 0 \end{cases}$$

Task:

1. Compute the output of the Leaky ReLU for the following inputs: $x = -3, -1, 0, 1, 3$.
2. Plot the Leaky ReLU function.

Solution

1. Compute the Output of the Leaky ReLU for Given Inputs:
Using the given function:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{if } x \leq 0 \end{cases}$$

Compute $f(x)$ for each input:

- For $x = -3$:
$$f(-3) = 0.01 \times (-3) = -0.03$$
- For $x = -1$:
$$f(-1) = 0.01 \times (-1) = -0.01$$
- For $x = 0$:
$$f(0) = 0.01 \times 0 = 0$$
- For $x = 1$:
$$f(1) = 1$$
- For $x = 3$:
$$f(3) = 3$$

Summary Table:

x	$f(x)$
-3	-0.03
-1	-0.01
0	0
1	1
3	3

2. Plot the Leaky ReLU Function:
any reasonable drawing of a horizontal line along with a change in the slope of that line works, as long as it correspond to your data.

Note: Since this is a text-based medium, please consider the following description for the plot:
- The Leaky ReLU function has two linear pieces: - For $x \leq 0$, the function is a straight line with a small positive slope (0.01), passing through the origin. - For $x > 0$, the function is a straight line with a slope of 1, also passing through the origin. - The function smoothly transitions at $x = 0$.

Problem 5: Softmax Function (10 points)

Problem: Given the following vector:

$$\mathbf{v} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$$

Task:

1. Compute the softmax of this vector:

$$\text{softmax}(v_i) = \frac{e^{v_i}}{\sum_{j=1}^{5} e^{v_j}}$$

2. Draw the resulting values on a distribution (e.g., a bar chart).

Solution

1. Compute the Softmax of the Vector v:

Given:

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$$

Compute e^{v_i} for each v_i:

- $e^{v_1} = e^1 \approx 2.7183$
- $e^{v_2} = e^2 \approx 7.3891$
- $e^{v_3} = e^3 \approx 20.0855$
- $e^{v_4} = e^4 \approx 54.5982$
- $e^{v_5} = e^5 \approx 148.4132$

Compute the Sum of Exponentials:

$$S = \sum_{j=1}^{5} e^{v_j} = 2.7183 + 7.3891 + 20.0855 + 54.5982 + 148.4132 \approx 233.2043$$

Compute Softmax Values:

- $\text{softmax}(v_1) = \dfrac{2.7183}{233.2043} \approx 0.0117$
- $\text{softmax}(v_2) = \dfrac{7.3891}{233.2043} \approx 0.0317$
- $\text{softmax}(v_3) = \dfrac{20.0855}{233.2043} \approx 0.0861$
- $\text{softmax}(v_4) = \dfrac{54.5982}{233.2043} \approx 0.2340$
- $\text{softmax}(v_5) = \dfrac{148.4132}{233.2043} \approx 0.6365$

Verify Sum to One:

$$\sum_{i=1}^{5} \text{softmax}(v_i) \approx 0.0117 + 0.0317 + 0.0861 + 0.2340 + 0.6365 = 1.0000$$

2. Draw the Resulting Values on a Distribution:
Plot the results we just did, making sure that no bar reaches or exceeds 1.0
Note: Since images cannot be displayed here, please consider the following description:
- The bar chart has five bars, each representing softmax(v_i) for $i = 1$ to 5. - The heights of the bars correspond to the computed softmax values:

- v_1: Approximately 1.17%
- v_2: Approximately 3.17%
- v_3: Approximately 8.61%
- v_4: Approximately 23.40%
- v_5: Approximately 63.65%

- The chart illustrates how higher input values correspond to higher probabilities after applying the softmax function.

Problem 6: Convexity of a Vector-Valued Loss Function (20 points)

Problem: Consider the following vector-valued loss function for a simple linear regression model:

$$L(\mathbf{W}) = \|\mathbf{y}_{\text{true}} - \mathbf{X}\mathbf{W}\|^2$$

where:

- $\mathbf{y}_{\text{true}}$ is the vector of true target values.
- $\mathbf{X}$ is the matrix of input features.
- $\mathbf{W}$ is the weight vector.

Task:

1. Prove that this loss function is convex with respect to the weight vector $\mathbf{W}$.
2. Determine the domain of this function.

Solution

1. Prove that $L(\mathbf{W})$ is Convex with Respect to $\mathbf{W}$:
Express the Loss Function in Terms of $\mathbf{W}$:

$$L(\mathbf{W}) = \|\mathbf{y}_{\text{true}} - \mathbf{X}\mathbf{W}\|^2 = (\mathbf{y}_{\text{true}} - \mathbf{X}\mathbf{W})^\top(\mathbf{y}_{\text{true}} - \mathbf{X}\mathbf{W})$$

Expand the Expression:

$$\begin{aligned} L(\mathbf{W}) &= (\mathbf{y}_{\text{true}})^\top \mathbf{y}_{\text{true}} - (\mathbf{y}_{\text{true}})^\top \mathbf{X}\mathbf{W} - (\mathbf{X}\mathbf{W})^\top \mathbf{y}_{\text{true}} + (\mathbf{X}\mathbf{W})^\top(\mathbf{X}\mathbf{W}) \\ &= (\mathbf{y}_{\text{true}})^\top \mathbf{y}_{\text{true}} - 2\mathbf{W}^\top\mathbf{X}^\top\mathbf{y}_{\text{true}} + \mathbf{W}^\top\mathbf{X}^\top\mathbf{X}\mathbf{W} \end{aligned}$$

Simplify:

$$L(\mathbf{W}) = \mathbf{W}^\top\mathbf{X}^\top\mathbf{X}\mathbf{W} - 2\mathbf{W}^\top\mathbf{X}^\top\mathbf{y}_{\text{true}} + (\mathbf{y}_{\text{true}})^\top\mathbf{y}_{\text{true}}$$

Analyzing Convexity:
- The term $\mathbf{W}^\top\mathbf{X}^\top\mathbf{X}\mathbf{W}$ is a quadratic form. - The matrix $\mathbf{X}^\top\mathbf{X}$ is symmetric and positive semidefinite because it's a Gram matrix. - The Hessian matrix of $L(\mathbf{W})$ with respect to $\mathbf{W}$ is:

$$\nabla^2 L(\mathbf{W}) = 2\mathbf{X}^\top\mathbf{X}$$

- Since $\mathbf{X}^\top\mathbf{X}$ is positive semidefinite, $\nabla^2 L(\mathbf{W})$ is positive semidefinite. - A function with a positive semidefinite Hessian is convex.
Conclusion:
$L(\mathbf{W})$ is convex with respect to $\mathbf{W}$.
2. Determine the Domain of the Function:
- $\mathbf{W} \in \mathbb{R}^d$, where d is the number of features. - There are no restrictions (e.g., divisions by zero, square roots of negative numbers). - All operations are well-defined for all $\mathbf{W} \in \mathbb{R}^d$.
Conclusion:
The domain of $L(\mathbf{W})$ is $\mathbb{R}^d$.

Problem 7: Convexity of a Loss Function (10 points)

Problem: Prove whether the following loss function is convex:

$$L(w) = \log(1 + e^w)$$

Task: Use the second derivative test to check for convexity.

Solution

1. Compute the First Derivative $L'(w)$:

$$L'(w) = \frac{d}{dw}\left[\log(1 + e^w)\right] = \frac{e^w}{1 + e^w}$$

2. Compute the Second Derivative $L''(w)$:

$$L''(w) = \frac{d}{dw}\left[\frac{e^w}{1 + e^w}\right] = \frac{e^w(1 + e^w) - e^w e^w}{(1 + e^w)^2} = \frac{e^w}{(1 + e^w)^2}$$

3. Analyze $L''(w)$:
- $e^w > 0$ for all $w \in \mathbb{R}$. - $(1 + e^w)^2 > 0$ for all $w \in \mathbb{R}$. - Therefore, $L''(w) > 0$ for all $w \in \mathbb{R}$.
Conclusion:
Since the second derivative $L''(w) > 0$ everywhere, the function $L(w)$ is **strictly convex** on $\mathbb{R}$.

Problem 8: Backpropagation (20 points)

Problem: Consider a simple neural network with:

- Input vector: $\mathbf{x} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$
- Hidden layer weights:
$$\mathbf{W}_1 = \begin{bmatrix} 0.3 & 0.5 \\ -0.6 & 0.2 \end{bmatrix}$$
- Hidden layer bias:
$$\mathbf{b}_1 = \begin{bmatrix} 0.1 \\ -0.1 \end{bmatrix}$$
- Output layer weights:
$$\mathbf{W}_2 = \begin{bmatrix} 0.4 & -0.3 \end{bmatrix}$$
- Output layer bias: $b_2 = 0.05$

The activation function for both the hidden and output layers is the sigmoid function:

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

The loss function is the Mean Squared Error (MSE):

$$L = \frac{1}{2}(y - y_{\text{true}})^2$$

where $y_{\text{true}} = 1$.

Task: Perform one iteration of backpropagation. Compute the forward pass, the gradients, and update the weights and biases for both the hidden and output layers.

Solution

Assumptions:

Assume a learning rate $\eta = 0.1$.

Forward Pass:

Step 1: Compute $\mathbf{z}^{(1)}$ (Hidden Layer Pre-Activation):

$$\mathbf{z}^{(1)} = \mathbf{W}_1\mathbf{x} + \mathbf{b}_1$$

Calculate $\mathbf{W}_1\mathbf{x}$:

$$\mathbf{W}_1\mathbf{x} = \begin{bmatrix} (0.3)(1) + (0.5)(2) \\ (-0.6)(1) + (0.2)(2) \end{bmatrix} = \begin{bmatrix} 0.3 + 1.0 \\ -0.6 + 0.4 \end{bmatrix} = \begin{bmatrix} 1.3 \\ -0.2 \end{bmatrix}$$

Add biases $\mathbf{b}_1$:

$$\mathbf{z}^{(1)} = \begin{bmatrix} 1.3 \\ -0.2 \end{bmatrix} + \begin{bmatrix} 0.1 \\ -0.1 \end{bmatrix} = \begin{bmatrix} 1.4 \\ -0.3 \end{bmatrix}$$

Step 2: Compute $\mathbf{a}^{(1)}$ (Hidden Layer Activation):

$$\mathbf{a}^{(1)} = \sigma(\mathbf{z}^{(1)})$$

Compute each component:

- $a_1^{(1)} = \sigma(1.4) = \dfrac{1}{1 + e^{-1.4}} \approx 0.8022$

- $a_2^{(1)} = \sigma(-0.3) = \dfrac{1}{1+e^{0.3}} \approx 0.4256$

So:

$$\mathbf{a}^{(1)} = \begin{bmatrix} 0.8022 \\ 0.4256 \end{bmatrix}$$

Step 3: Compute $z^{(2)}$ (Output Layer Pre-Activation):

$$z^{(2)} = \mathbf{W}_2\mathbf{a}^{(1)} + b_2 = (0.4)(0.8022) + (-0.3)(0.4256) + 0.05$$

Compute:

$$z^{(2)} = 0.3209 - 0.1277 + 0.05 = 0.2432$$

Step 4: Compute $a^{(2)}$ (Output Layer Activation):

$$a^{(2)} = \sigma(z^{(2)}) = \frac{1}{1+e^{-0.2432}} \approx 0.5605$$

Step 5: Compute Loss L:

$$L = \frac{1}{2}(a^{(2)} - y_{\text{true}})^2 = \frac{1}{2}(0.5605 - 1)^2 = \frac{1}{2}(-0.4395)^2 = 0.0966$$

Backward Pass:
Step 1: Compute Output Layer Error $\delta^{(2)}$:

$$\delta^{(2)} = \frac{\partial L}{\partial z^{(2)}} = \frac{\partial L}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial z^{(2)}}$$

Compute $\dfrac{\partial L}{\partial a^{(2)}}$:

$$\frac{\partial L}{\partial a^{(2)}} = a^{(2)} - y_{\text{true}} = 0.5605 - 1 = -0.4395$$

Compute $\dfrac{\partial a^{(2)}}{\partial z^{(2)}}$:

$$\frac{\partial a^{(2)}}{\partial z^{(2)}} = a^{(2)}(1 - a^{(2)}) = 0.5605 \times 0.4395 = 0.2463$$

Compute $\delta^{(2)}$:

$$\delta^{(2)} = (-0.4395) \times 0.2463 = -0.1083$$

Step 2: Compute Gradients for Output Layer Weights and Bias:

- **Weights $\mathbf{W}_2$:**

$$\frac{\partial L}{\partial \mathbf{W}_2} = \delta^{(2)}\mathbf{a}^{(1)\top} = -0.1083 \begin{bmatrix} 0.8022 & 0.4256 \end{bmatrix}$$

  So:

$$\frac{\partial L}{\partial \mathbf{W}_2} = \begin{bmatrix} -0.0869 & -0.0461 \end{bmatrix}$$

- **Bias b_2:**

$$\frac{\partial L}{\partial b_2} = \delta^{(2)} = -0.1083$$

Step 3: Compute Hidden Layer Error $\boldsymbol{\delta}^{(1)}$:
Compute $\sigma'(\mathbf{z}^{(1)})$:

$$\sigma'(z_i^{(1)}) = a_i^{(1)}(1 - a_i^{(1)})$$

Compute:

- $\sigma'(z_1^{(1)}) = 0.8022(1 - 0.8022) = 0.1580$
- $\sigma'(z_2^{(1)}) = 0.4256(1 - 0.4256) = 0.2444$

Compute $\mathbf{W}_2^\top \delta^{(2)}$:

$$\mathbf{W}_2^\top \delta^{(2)} = \begin{bmatrix} 0.4 \\ -0.3 \end{bmatrix} (-0.1083) = \begin{bmatrix} -0.0433 \\ 0.0325 \end{bmatrix}$$

Compute $\boldsymbol{\delta}^{(1)}$:

$$\boldsymbol{\delta}^{(1)} = \left(\mathbf{W}_2^\top \delta^{(2)}\right) \circ \sigma'(\mathbf{z}^{(1)}) = \begin{bmatrix} -0.0433 \times 0.1580 \\ 0.0325 \times 0.2444 \end{bmatrix} = \begin{bmatrix} -0.0068 \\ 0.0079 \end{bmatrix}$$

Step 4: Compute Gradients for Hidden Layer Weights and Biases:

- **Weights $\mathbf{W}_1$:**
$$\frac{\partial L}{\partial \mathbf{W}_1} = \boldsymbol{\delta}^{(1)} \mathbf{x}^\top = \begin{bmatrix} -0.0068 \\ 0.0079 \end{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} = \begin{bmatrix} -0.0068 & -0.0136 \\ 0.0079 & 0.0158 \end{bmatrix}$$

- **Biases $\mathbf{b}_1$:**
$$\frac{\partial L}{\partial \mathbf{b}_1} = \boldsymbol{\delta}^{(1)} = \begin{bmatrix} -0.0068 \\ 0.0079 \end{bmatrix}$$

Update Weights and Biases:
Step 1: Update Output Layer Weights and Bias:

- **Weights $\mathbf{W}_2$:**

$$\mathbf{W}_2^{\text{new}} = \mathbf{W}_2 - \eta \frac{\partial L}{\partial \mathbf{W}_2} = \begin{bmatrix} 0.4 & -0.3 \end{bmatrix} - 0.1 \times \begin{bmatrix} -0.0869 & -0.0461 \end{bmatrix} = \begin{bmatrix} 0.4087 & -0.2954 \end{bmatrix}$$

- **Bias b_2:**
$$b_2^{\text{new}} = b_2 - \eta \frac{\partial L}{\partial b_2} = 0.05 - 0.1 \times (-0.1083) = 0.0608$$

Step 2: Update Hidden Layer Weights and Biases:

- **Weights $\mathbf{W}_1$:**

$$\mathbf{W}_1^{\text{new}} = \mathbf{W}_1 - \eta \frac{\partial L}{\partial \mathbf{W}_1} = \begin{bmatrix} 0.3 & 0.5 \\ -0.6 & 0.2 \end{bmatrix} - 0.1 \times \begin{bmatrix} -0.0068 & -0.0136 \\ 0.0079 & 0.0158 \end{bmatrix} = \begin{bmatrix} 0.3007 & 0.5014 \\ -0.6008 & 0.1984 \end{bmatrix}$$

- **Biases $\mathbf{b}_1$:**
$$\mathbf{b}_1^{\text{new}} = \mathbf{b}_1 - \eta \frac{\partial L}{\partial \mathbf{b}_1} = \begin{bmatrix} 0.1 \\ -0.1 \end{bmatrix} - 0.1 \times \begin{bmatrix} -0.0068 \\ 0.0079 \end{bmatrix} = \begin{bmatrix} 0.1007 \\ -0.1008 \end{bmatrix}$$