```python
import numpy as np
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Load dataset
data = load_breast_cancer()
X = data.data     # Features
y = data.target.reshape(-1, 1)  # Labels (0 = malignant, 1 = benign)

# Normalize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

[3]:
```python
# Sigmoid and its derivative
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def sigmoid_derivative(x):
    return x * (1 - x)
```

[5]:
```python
# Set random seed for reproducibility
np.random.seed(42)

# Layer sizes
input_size = X_train.shape[1]  # 30 features
hidden_size = 10
output_size = 1  # Binary output

# Weights and biases
W1 = np.random.rand(input_size, hidden_size)
b1 = np.zeros((1, hidden_size))
W2 = np.random.rand(hidden_size, output_size)
b2 = np.zeros((1, output_size))

# Training parameters
epochs = 5000
learning_rate = 0.1

# Training loop
for epoch in range(epochs):
    # Forward pass
    z1 = np.dot(X_train, W1) + b1
    a1 = sigmoid(z1)
    z2 = np.dot(a1, W2) + b2
    output = sigmoid(z2)
```

```
    # Compute loss (binary cross-entropy approximation)
    error = y_train - output
    d_output = error * sigmoid_derivative(output)

    error_hidden = d_output.dot(W2.T)
    d_hidden = error_hidden * sigmoid_derivative(a1)

    # Update weights and biases
    W2 += a1.T.dot(d_output) * learning_rate
    b2 += np.sum(d_output, axis=0, keepdims=True) * learning_rate
    W1 += X_train.T.dot(d_hidden) * learning_rate
    b1 += np.sum(d_hidden, axis=0, keepdims=True) * learning_rate

    # Print loss every 500 epochs
    if epoch % 500 == 0:
        loss = np.mean(np.square(error))
        print(f"Epoch {epoch} - Loss: {loss:.4f}")
```

```
Epoch 0 - Loss: 0.4594
Epoch 500 - Loss: 0.0028
Epoch 1000 - Loss: 0.0024
Epoch 1500 - Loss: 0.0023
Epoch 2000 - Loss: 0.0023
Epoch 2500 - Loss: 0.0022
Epoch 3000 - Loss: 0.0022
Epoch 3500 - Loss: 0.0016
Epoch 4000 - Loss: 0.0002
Epoch 4500 - Loss: 0.0001
```

[6]:
```
# Test the model
z1 = np.dot(X_test, W1) + b1
a1 = sigmoid(z1)
z2 = np.dot(a1, W2) + b2
predictions = sigmoid(z2)
predicted_classes = (predictions > 0.5).astype(int)

# Accuracy
accuracy = np.mean(predicted_classes == y_test)
print(f"\nTest Accuracy: {accuracy * 100:.2f}%")
```

```
Test Accuracy: 95.61%
```