Logistic Regression on Titanic Dataset: From Scratch

Your Name

October 3, 2025

1 Introduction

Logistic regression is a fundamental classification algorithm used to model the probability of a binary outcome. In this report, we implement logistic regression from scratch to predict survival on the Titanic dataset. The model uses features such as `age`, `sex`, and `pclass` to estimate survival probabilities.

2 Data Preprocessing

We begin by importing the dataset and selecting relevant features:

- `age` - continuous variable representing passenger age.
- `sex` - categorical variable (converted to numeric: male = 0, female = 1).
- `pclass` - categorical variable for passenger class (1, 2, 3).
- `survived` - target variable (0 = did not survive, 1 = survived).

Missing values are dropped to ensure clean data for model training. The features are converted to NumPy arrays for efficient computation:

```
X = df[["age","sex","pclass"]].values
y = df["survived"].astype(int).values
```

The dataset is split into training and test sets:

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

3 Model Implementation

We implement logistic regression from scratch with the following key components:

3.1 Sigmoid Function

The sigmoid activation function maps a linear combination of features to a probability between 0 and 1:

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

3.2 Linear Combination of Features

For each sample, the linear predictor is calculated as:

$$z = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

where w_1, w_2, w_3 are the model weights and b is the bias.

3.3 Parameter Update using Gradient Descent

Weights and bias are updated using the following gradients:

$$\frac{\partial L}{\partial w} = \frac{1}{n} X^T (\hat{y} - y), \quad \frac{\partial L}{\partial b} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)$$

Update rules:

$$w := w - \alpha \frac{\partial L}{\partial w}, \quad b := b - \alpha \frac{\partial L}{\partial b}$$

Here, α is the learning rate and n is the number of samples.

4 Training the Model

The model is trained for a fixed number of epochs, adjusting weights and bias iteratively. The prediction probabilities are computed as:

```
def predict_proba(self, X):
    return self.sigmoid(np.dot(X, self.weights) + self.
       bias)
```

Binary predictions are obtained using a threshold of 0.5:

```
def predict(self, X, threshold=0.5):
    return (self.predict_proba(X) >= threshold).astype(
       int)
```

5 Evaluation

The model is evaluated on the test set using accuracy:

```
accuracy = np.mean(model.predict(X_test) == y_test)
print("Test Accuracy:", accuracy)
```

6 Visualization

To illustrate the model's predictions, survival probabilities are plotted against `age` for different combinations of `sex` and `pclass`.

```
plt.scatter(df["age"], df["survived"], ...)
plt.plot(x_vals, y_probs, ...)
plt.axhline(0.5, ...)
plt.xlabel("Age")
plt.ylabel("Survival Probability")
plt.title("Logistic Regression: Survival vs Age")
plt.show()
```

The resulting curves demonstrate the characteristic S-shaped probability curve of logistic regression.

7 Conclusion

This project demonstrates the full pipeline of implementing logistic regression from scratch:

- Data preprocessing and feature encoding.
- Sigmoid-based probabilistic modeling.
- Gradient descent for optimizing weights and bias.
- Visualization of predictions showing survival probability trends.

The model achieves reasonable accuracy on the Titanic dataset and illustrates the fundamental concepts behind logistic regression.