Detailed Report on Logistic Regression: Titanic Survival Prediction

Atharv Bhatt

October 3, 2025

1 Introduction

Logistic regression is a widely used supervised machine learning algorithm for **binary classification problems**, where the output $y \in \{0, 1\}$.

In this project, we predict survival (`survived`) of passengers on the Titanic based on their **age, sex, and passenger class** (`pclass`). The implementation is from scratch in Python.

2 Dataset Description

The Titanic dataset contains passenger information:

- `age` – age of the passenger
- `sex` – gender (0: male, 1: female)
- `pclass` – passenger class (1, 2, 3)
- `survived` – survival status (0: died, 1: survived)

3 Data Preprocessing

We select relevant columns, handle missing values, and encode categorical features numerically:

```
df = df[["age", "sex", "pclass", "survived"]].dropna()
df["sex"] = df["sex"].map({"male": 0, "female": 1})
df["pclass"] = df["pclass"].astype(int)

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

4 Logistic Regression Model

The logistic regression model outputs a **probability**:

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}}, \quad z = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

where $\sigma(z)$ is the **sigmoid function**, mapping any real number to $[0, 1]$.

4.1 Cross-Entropy Loss

To measure the difference between predicted probabilities and true labels, we use **binary cross-entropy loss**:

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

This loss penalizes confident but wrong predictions heavily.

4.2 Gradient Descent Derivation

We compute gradients to update weights:

$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i) x_{ij}, \quad \frac{\partial \mathcal{L}}{\partial b} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)$$

Weights are updated iteratively:

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

where α is the learning rate.

5 Python Implementation

```
class LogisticRegression:
    def __init__(self, lr=0.01, epochs=10000):
        self.lr = lr
        self.epochs = epochs
        self.weights = None
        self.bias = None

    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))

    def fit(self, X, y):
        n_samples, n_features = X.shape
        self.weights = np.zeros(n_features)
```

```
        self.bias = 0

        for _ in range(self.epochs):
            linear_pred = np.dot(X, self.weights) + self.bias
            y_pred = self.sigmoid(linear_pred)

            dw = (1/n_samples) * np.dot(X.T, (y_pred - y))
            db = (1/n_samples) * np.sum(y_pred - y)

            self.weights -= self.lr * dw
            self.bias -= self.lr * db

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

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

6 Training the Model

We train the model using:

```
model = LogisticRegression(lr=0.01, epochs=10000)
model.fit(X_train, y_train)
accuracy = np.mean(model.predict(X_test) == y_test)
print("Test Accuracy:", accuracy)
```

7 Interpretation of Parameters

- Each w_j represents the contribution of the corresponding feature to the log-odds of survival.
- The bias b is the baseline log-odds when all features are zero.

8 Conclusion

This report demonstrates:

- Implementing logistic regression from scratch
- Mathematical derivation of cross-entropy loss
- Gradient computation and parameter updates
- Prediction of Titanic survival based on age, sex, and class

The logistic regression model captures nonlinear probability outputs using the sigmoid function, providing interpretable insights into survival likelihood.