Logistic Regression for Binary Classification

Raghda Altaei

2024

1 Introduction

In this project, we aim to solve a binary classification problem using Logistic Regression. The goal is to predict user categories based on their ratings for various news items, leveraging training and test datasets stored in MATLAB .mat format. The code was written in PYTHON using Google Colab.

2 Problem Statement

The problem involves predicting a user's category (0 or 1) based on features derived from user ratings of news items. The dataset consists of a training set with 1600 samples and a test set with 400 samples. Each sample contains 1593 features (ratings) and 1 target label (user category).

3 Data Description

The datasets used in this project are:

- **Train Data:** Contains 1600 samples and 1594 features.
- **Test Data:** Contains 400 samples and 1594 features.

4 Methodology

The following steps outline the approach taken to solve the problem:

1. **Data Loading:** The datasets were loaded from .mat files using the `scipy` library.
2. **Feature and Label Extraction:** The last column of the datasets was used as the target variable (labels), while the remaining columns were treated as features.

3. **Hyperparameter Tuning:** We defined a range of regularization parameters (λ) and performed 10-fold cross-validation to find the optimal λ value.

4. **Model Training:** The Logistic Regression model was trained using the best λ value obtained from cross-validation.

5. **Model Evaluation:** The model was evaluated on both training and test datasets, and performance metrics including accuracy and ROC curve were computed.

5 Code Implementation

The following Python code was used to implement the solution:

```
# Import Libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, roc_curve, roc_auc_score
import matplotlib.pyplot as plt
from scipy.io import loadmat
from google.colab import drive

# Mount Google Drive
drive.mount('/content/drive', force_remount=True)

# Specify the .mat file paths
file_path1 = '/content/drive/MyDrive/Train_data.mat'
file_path2 = '/content/drive/MyDrive/Test_Data.mat'

# Load the .mat files
train_data = loadmat(file_path1)
test_data = loadmat(file_path2)

# Extract the data stored under the correct keys
train = train_data['train']
test = test_data['test']

# Prepare features and labels
X_train = train[:, :-1]  # Features for training data
y_train = train[:, -1]   # Labels for training data
X_test = test[:, :-1]    # Features for test data
y_test = test[:, -1]     # Labels for test data
```

```
# Define the range of  values for regularization
lambda_values = [10**i for i in range(-5, 3)]

# Perform 10-fold cross-validation to find the best  value
kf = KFold(n_splits=10, shuffle=True, random_state=42)

best_lambda = None
best_score = -np.inf

for lam in lambda_values:
    model = LogisticRegression(C=1/lam, penalty='l2', solver='liblinear', random_state=42)
    scores = cross_val_score(model, X_train, y_train, cv=kf, scoring='accuracy')
    mean_score = scores.mean()

    if mean_score > best_score:
        best_score = mean_score
        best_lambda = lam

# Train the Logistic Regression model with the best  value
final_model = LogisticRegression(C=1/best_lambda, penalty='l2', solver='liblinear', random_s
final_model.fit(X_train, y_train)

# Evaluate the model on the training and test sets
y_train_pred = final_model.predict(X_train)
y_test_pred = final_model.predict(X_test)

# Calculate accuracies
train_accuracy = accuracy_score(y_train, y_train_pred)
test_accuracy = accuracy_score(y_test, y_test_pred)

# Print results
print(f"Best  value: {best_lambda}")
print(f"Best cross-validated accuracy: {best_score}")
print(f"Training accuracy: {train_accuracy}")
print(f"Test accuracy: {test_accuracy}")

# Generate ROC curve
y_test_proba = final_model.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_test_proba)
roc_auc = roc_auc_score(y_test, y_test_proba)

# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='red', linestyle='--')
plt.xlabel('False Positive Rate')
```

```
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Logistic Regression')
plt.legend(loc='lower right')
plt.grid()
plt.show()
```

6 Results

The results obtained from the analysis are summarized as follows:

- **Best value:** 100
- **Best cross-validated accuracy:** 0.81
- **Training accuracy:** 0.95
- **Test accuracy:** 0.80

The ROC curve was generated to evaluate the performance of the model visually. The area under the curve (AUC) provides insight into the trade-off between the true positive rate and the false positive rate.

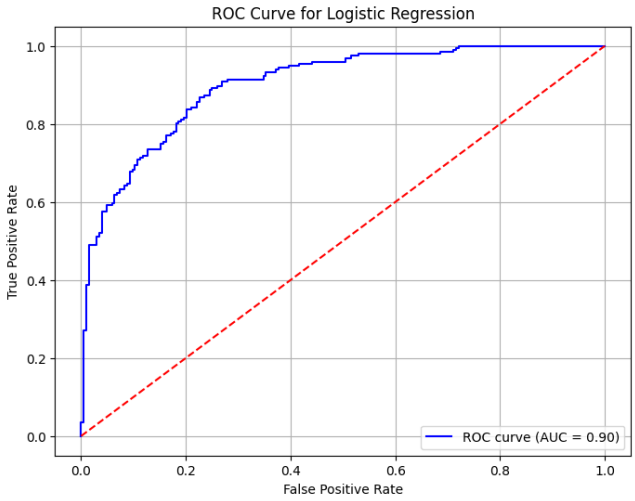

Figure 1: ROC Curve for Logistic Regression

7 Conclusion

The Logistic Regression model demonstrated good performance on the training set with a high accuracy of 95%. However, the test accuracy was slightly lower at

80%, indicating some level of overfitting. The optimal regularization parameter found through cross-validation was 100. The ROC curve further illustrated the model's ability to distinguish between the two classes.