University of Hertfordshire UH

School of Physics, Engineering and computer science, Hatfield, Hertfordshire

AL10 9AB

7PAM2021-0901-2024 – Machine Learning and Neural Networks

Title: “Understanding Logistic Regression with Regularization: A Practical Guide”

Github URL: https://github.com/RameshKelavath/Logistic-Regression

Name: Ramesh Kelavath

Student ID: 23029839

Supervisor: **Dr. Peter Scicluna**

1. Abstract

Logistic regression is a widely used supervised learning algorithm for binary classification tasks, where the goal is to predict probabilities for two possible outcomes. It utilizes the sigmoid function to convert linear predictions into probabilities, making it effective in fields such as healthcare and finance. However, logistic regression can overfit, especially in high-dimensional datasets or when features are highly correlated.

Regularization techniques, like L1 (Lasso) and L2 (Ridge), help mitigate overfitting. L1 regularization reduces some coefficients to zero, aiding in feature selection, while L2 regularization minimizes coefficient magnitudes, improving generalization and stability.

This tutorial explores the concepts and applications of logistic regression with regularization. It covers key topics, including the logistic cost function and regularization methods, with Python-based examples. By understanding L1 and L2 regularization, readers will learn how to prevent overfitting, improve model performance, and enhance predictive accuracy while maintaining model interpretability.

2. Introduction

Logistic regression is a supervised learning algorithm for binary classification, predicting the probability of two possible outcomes using the sigmoid function. This function maps linear predictions to a range between 0 and 1, making it ideal for probability estimation. Widely applied across diverse fields, logistic regression helps predict disease risks in healthcare, customer retention patterns in marketing, and credit risks in finance.

Despite its effectiveness, the algorithm faces significant challenges. In high-dimensional datasets, logistic regression can suffer from overfitting, learning noise instead of generalizable patterns. This results in poor performance on unseen data. Additionally, the model is sensitive to multicollinearity, where input features are highly correlated, leading to instability in coefficient estimates and difficulty in interpreting individual predictor importance.

Regularization techniques address these challenges by adding penalty terms to the logistic regression cost function. Two primary methods are L1 regularization (Lasso) and L2 regularization (Ridge). L1 regularization introduces an absolute penalty that shrinks some coefficients to zero, effectively selecting the most relevant features and creating a sparse model. Conversely, L2 regularization applies a squared penalty, reducing the magnitude of all coefficients and improving model stability, particularly in multicollinearity scenarios.

The tutorial aims to provide a comprehensive understanding of logistic regression and its regularization techniques. By exploring the theoretical foundations of L1 and L2 regularization and demonstrating their implementation in Python, it offers a practical

comparison of how these techniques prevent overfitting, enhance model interpretability, and improve predictive accuracy. Readers will gain insights into effectively applying logistic regression in real-world scenarios, understanding its nuanced approach to classification and predictive modeling.

3. Theoretical Background
Logistic Regression: Core Concepts

3.1. Sigmoid Function:

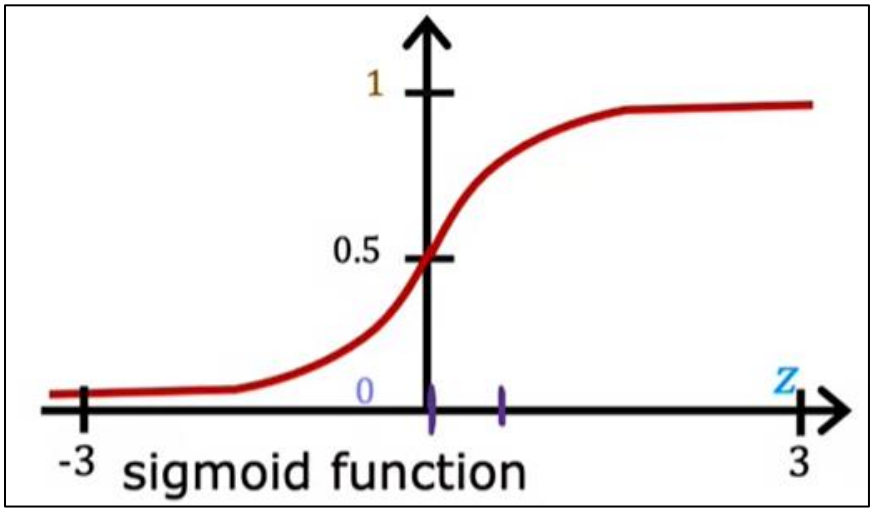
The sigmoid function is the core of logistic regression, transforming a linear combination of inputs into a probability value between 0 and 1. It is defined as:

$$g(z) = \frac{1}{1+e^{-z}}, \text{ where } z = w.x + b$$

Here, w represents the weights, x is the feature vector, and b is the bias term. This function maps input values to probabilities, making logistic regression particularly effective for binary classification tasks.

Interpretation:

- when z is very large and positive, g(z) approaches 1.
- when z is very large and negative, g(z) approaches 0.
- When z is zero, g(z) is 0.5



3.2. Cost Function:

To evaluate how well the model aligns with the data, logistic regression employs the Logistic Loss function or Log Loss or Binary Cross Entropy Loss.The function is expressed as:

$$J(w) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \log\left(f(x_i)\right) + (1 - y_i)\log\left(1 - f(x_i)\right)\right]$$

Here:

- y_i represents the actual label (either 0 or 1) for the i-th data point.
- $f(x_i)$ denotes the predicted probability for the i-th data point.
- m is the number of training samples.

This loss function penalizes the model heavily for incorrect predictions, particularly when the predicted probability significantly deviates from the actual label. By minimizing J(w), the model works to reduce the gap between the predicted probabilities and the true labels, improving its accuracy over time.

3.3. Gradient Descent for Logistic Regression:

Gradient Descent is an optimization algorithm used to determine the values of parameters w and b that minimize the cost function J(w). The update rules for the parameters are as follows:

$$w_j = w_j - \alpha\frac{\partial J(w)}{\partial w_j}, \quad b = b - \alpha\frac{\partial J(w)}{\partial b}$$

In this formula, α represents the learning rate, which controls the size of each step taken towards minimizing the cost function. For logistic regression, the gradients are calculated as:

$$\frac{\partial J(w)}{\partial w_j} = \frac{1}{m}\sum_{i=1}^{m}(f(x_i) - y_i)x_{ij}, \quad \frac{\partial J(w)}{\partial b} = \frac{1}{m}\sum_{i=1}^{m}(f(x_i) - y_i)$$

Through iterative updates of w and b, the gradient descent algorithm gradually decreases the cost function, eventually converging to its minimum.

3.4. The Problem of Overfitting

Overfitting happens when a model achieves high performance on the training data but fails to perform well on unseen data. In logistic regression, this issue often occurs when:

High Dimensionality:
Overfitting occurs when the number of features exceeds the observations, causing the model to memorize the training data rather than learning general patterns.

Multicollinearity:
Highly correlated features lead to unstable weight estimates, making predictions sensitive to small changes in data. This results in large weights, allowing the model to fit training data perfectly but perform poorly on unseen data.

3.5. Regularization: A Solution to Overfitting

Regularization is a method to reduce overfitting by incorporating a penalty term into the cost function. This helps to limit large weight values, encouraging the model to generalize more effectively to new data. By managing the size of the weights, regularization achieves a balance between the model's complexity and its ability to make accurate predictions.

L1 Regularization (Lasso)

- Adds an absolute penalty to the cost function:

$$J(w) = J(w) + \lambda \sum_{j=1}^{n} |w_j|$$

Promotes sparsity by shrinking some weights to zero, effectively eliminating irrelevant features. This makes L1 useful for feature selection.

L2 Regularization (Ridge)

- Adds a squared penalty to the cost function

$$J(w) = J(w) + \lambda \sum_{j=1}^{n} w_j^2$$

Reduces the magnitude of weights, ensuring stability without eliminating features. This makes L2 effective for handling multicollinearity.

Hyperparameter (λ)

The regularization strength is controlled by the hyperparameter λ:

- High λ: Strong regularization, leading to smaller weights and potentially underfitting.
- Low λ: Weak regularization, allowing larger weights and risking overfitting

4. Practical Implementation

In this practical implementation, we used the scikit-learn Breast Cancer dataset to apply logistic regression models for predicting whether a tumor is benign or malignant. Logistic regression is a suitable model for this task, as it is designed for binary classification problems and provides probabilities for class predictions. The dataset includes 30 continuous features, such as the mean radius, texture, and smoothness of cell nuclei, which are used to distinguish between benign and malignant tumors.

Dataset Overview

The Breast Cancer dataset contains 569 instances, each with 30 continuous features. The target variable y indicates whether a tumor is benign (0) or malignant (1). These features capture different physical characteristics of tumor cells, making the dataset well-suited for logistic regression, which handles continuous inputs and binary targets effectively. Logistic regression predicts the log-odds of a tumor being malignant using the provided features.

Preprocessing the Data

The initial step involves loading the dataset and dividing it into training and test sets, allocating 80% for training and 20% for testing. This ensures the model is assessed on unseen data to evaluate its ability to generalize. Feature scaling was performed using StandardScaler to normalize the data, ensuring all features are on the same scale, which is crucial for logistic regression due to its sensitivity to feature magnitudes.

```
In [4]: import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.datasets import load_breast_cancer
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import StandardScaler
        from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import classification_report, confusion_matrix,roc_curve, roc_auc_score

In [5]: # Step 1: Load dataset
        # Load the breast cancer dataset from sklearn
        X, y = load_breast_cancer(return_X_y=True)

In [6]: # Step 2: Split the data
        # Split the dataset into 80% training and 20% testing data

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

        # Step 3: Scale the features
        scaler = StandardScaler()
        X_train_scaled = scaler.fit_transform(X_train)
        X_test_scaled = scaler.transform(X_test)
```

Models Trained

Three versions of logistic regression were trained:

- **Logistic Regression without Regularization:** The first model was trained without regularization (penalty='none'), serving as a baseline. This model learns coefficients for all features without any penalty for large coefficients.

```
#Logistic Regression without Regularization

lr_no_reg = LogisticRegression(penalty='none', random_state=42)
lr_no_reg.fit(X_train_scaled, y_train)
y_pred_no_reg = lr_no_reg.predict(X_test_scaled)
y_prob_no_reg = lr_no_reg.predict_proba(X_test_scaled)


print(classification_report(y_test, y_pred_no_reg))
```

- **Logistic Regression with L1 Regularization (Lasso):** The second model incorporated L1 regularization (penalty='l1'), which encourages some coefficients to shrink to zero. This effectively acts as a form of feature selection, helping to pinpoint the most significant features for predicting the tumor type.

```
# Logistic Regression with L1 Regularization(Lasso)

lr_l1 = LogisticRegression(penalty='l1', solver='liblinear', random_state=42)
lr_l1.fit(X_train_scaled, y_train)
y_pred_l1 = lr_l1.predict(X_test_scaled)
y_prob_l1 = lr_l1.predict_proba(X_test_scaled)


print(classification_report(y_test, y_pred_l1))
```

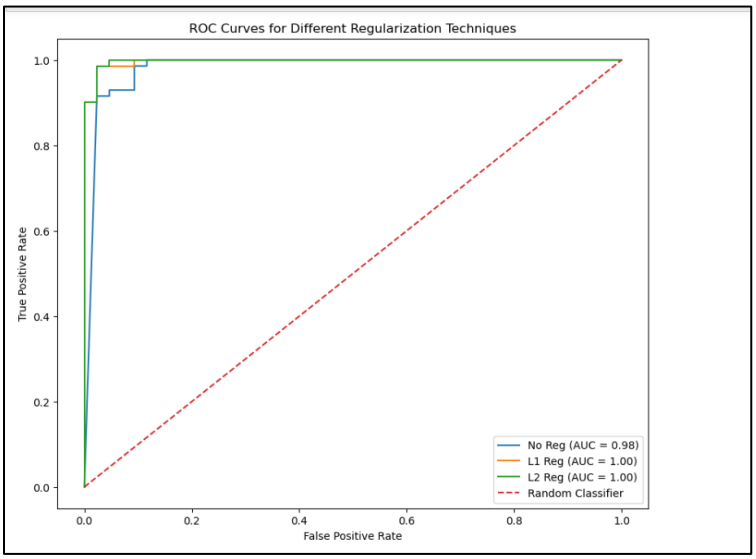
- **Logistic Regression with L2 Regularization (Ridge):** The third model used L2 regularization (penalty='l2'), which penalizes large coefficients but does not shrink them to zero. L2 regularization prevents overfitting by reducing the magnitude of coefficients, leading to a more generalized model.

```
# Logistic Regression with L2 Regularization(Ridge)

lr_l2 = LogisticRegression(penalty='l2', random_state=42)
lr_l2.fit(X_train_scaled, y_train)
y_pred_l2 = lr_l2.predict(X_test_scaled)
y_prob_l2 = lr_l2.predict_proba(X_test_scaled)


print(classification_report(y_test, y_pred_l2))
```

Model Evaluation

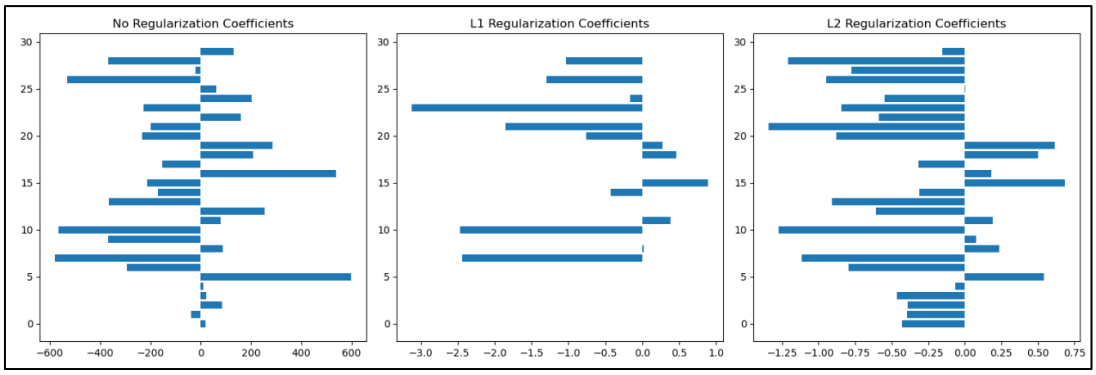
Once the models were trained, their performance was assessed using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. These measures offer a comprehensive understanding of the model's effectiveness in accurately classifying tumors.

- The classification report revealed that all models achieved comparable results, demonstrating high accuracy (over 90%) in differentiating between benign and malignant tumors.
- The ROC curve and AUC score were used to evaluate the models' ability to distinguish between classes. Models with higher AUC values are better at recognizing the difference between normal and cancerous tumors. The AUC score for the L1 and L2 regularized models was slightly higher than for the non-regularized model, indicating that regularization improved the model's performance slightly.



Feature Importance

We also compared the coefficients of the features in the three models. The models with regularization (L1 and L2) tended to shrink some feature coefficients more than the non-regularized model, especially in the case of L1 regularization, which resulted in many zero coefficients. This highlights the importance of regularization in identifying key features while eliminating less relevant ones, particularly in high-dimensional datasets.



5. Conclusion

In this project, I applied logistic regression to classify breast cancer tumors as either benign or malignant. I explored three different logistic regression models: one without regularization, one with L1 regularization (Lasso), and one with L2 regularization (Ridge). Regularization is a method designed to reduce overfitting by discouraging large coefficients in the model, thereby enhancing its ability to generalize to new data.

Key Findings

- The models performed well, with all achieving an accuracy greater than 90% and AUC-ROC scores above 0.97.
- L1 regularization proved highly effective for feature selection by reducing the coefficients of less significant features to zero. On the other hand, L2 regularization maintained all features while reducing their coefficient magnitudes, providing model stability.
- The models with regularization (L1 and L2) performed slightly better than the model without regularization, as indicated by their higher AUC-ROC scores.

Key Takeaways

- L1 regularization is useful for feature selection because it can eliminate irrelevant features, simplifying the model.
- L2 regularization helps improve model stability and generalization, especially when features are highly correlated or there is noise in the data.

Applications

- L1 regularization is often used in areas like bioinformatics (e.g., gene selection) or high-dimensional data problems.
- L2 regularization is more commonly applied in fields where model stability is crucial, such as financial modeling or predictive analytics in insurance.

Future Directions

- A natural next step would be to explore ElasticNet, a combination of L1 and L2 regularization, for better performance in cases where both feature selection and stability are needed.
- Additionally, extending logistic regression to handle multiclass classification problems would be a valuable future direction, especially for applications like image classification or text analysis.

References

James, G., Witten, D., Hastie, T. and Tibshirani, R., 2013. *An Introduction to Statistical Learning: with Applications in R*. New York: Springer.

Ng, A., 2024. *Machine Learning*. [online] Coursera. Available at: https://www.coursera.org/learn/machine-learning [Accessed 5 November 2024].

Pedregosa, F. et al., 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, pp. 2825-2830.

Hastie, T., Tibshirani, R. and Friedman, J., 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. New York: Springer.

Tibshirani, R., 1996. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1), pp. 267-288.

Friedman, J., Hastie, T. and Tibshirani, R., 2010. *Regularization Paths for Generalized Linear Models via Coordinate Descent*. *Journal of Statistical Software*, 33(1), pp. 1-22.

Scikit-learn developers, 2023. *Logistic Regression*. [online] Available at: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression [Accessed 1 December 2024].

Ridge, S., Van Der Heide, T. and Rijsdijk, L., 2019. *Regularization Methods for Machine Learning*. Journal of Data Science, 17(4), pp. 435-445.