# Department of Computing

## CS 471: Machine Learning (3+1)

## Class: BSCS12-C

## Lab 4: Logistic Regression

## Date: 7-10-2024

## Time: 10:00 am – 1:00 pm

**Instructor: Dr. Muhammad Naseer Bajwa**

**Lab Engineer: Ms. Iram Tariq Bhatti**

### Course Learning Outcomes (CLOs)

| No. | Course Learning Outcomes | Graduate Attributes (CS) | BT Level | Teaching & Learning Methods | Assessment Methods |
|---|---|---|---|---|---|
| CLO-1 | Explain theoretical concepts behind various machine learning algorithms and techniques | GA-2 | C-2 | Active/Blended | Quizzes, Assignments, ESE, MSE |
| CLO-2 | Analyse a given problem and select the most suitable supervise / unsupervised machine learning algorithm for the task. | GA-3 | C-4 | Active/Blended | Quizzes, Assignments, ESE, MSE |
| CLO-3 | Implement various machine learning algorithms using modern software libraries for a range of applications. | GA-5 | P-3 | Cooperative Learning | Labs, Project |
| CLO-4 | Contribute effectively in a team to complete a given task | GA-6 | A-2 | Cooperative Learning | Labs, Project |

# Lab 4: Logistic Regression

**Learning Outcome**

CLO 1: Explain theoretical concepts behind various machine learning algorithms and techniques.

CLO 3: Implement various machine learning algorithms using modern software libraries for a range of applications.

**Tools / Software Requirement**

Google Colab, Python, Jupyter Notebook, Numpy, Pandas, Matplotlib, Scikit-learn, Seaborn

**Contents**



## 1) Objectives

With successful completion of this lab, you should be able to:

- Understand the fundamental concept of **Logistic Regression**.
- Know how to model a **binary classification** problem using logistic regression.
- Implement Logistic Regression using **Python** and **Scikit-learn**.
- Explore how **sigmoid function** works to predict probabilities.
- Learn how to **interpret results**, tune **hyperparameters**, and validate models.
- Gain information about **real-world applications** and **limitations** of logistic regression.

## 2) Core Ideas

- How logistic regression estimates probabilities? Why it uses the sigmoid function to map any input into a value between 0 and 1?

- How the threshold (often set at 0.5) is used to assign class labels?
- Introduction to Log Loss and why it's used in logistic regression instead of Mean Squared Error.
- How gradient descent helps in optimizing the weights of the model.?

### 3) Mathematics Behind Logistic Regression

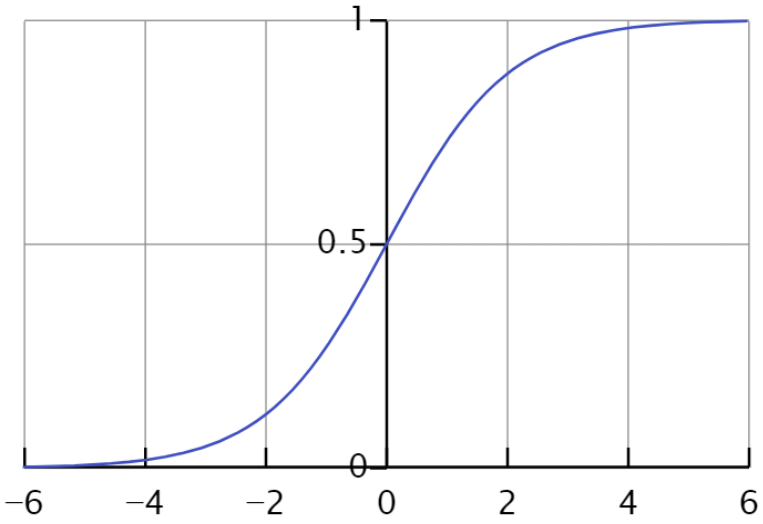
Logistic regression is a statistical method used for binary classification problems. It models the probability that a given input point belongs to a particular category (usually labeled as 1), versus another category (labeled as 0). Unlike linear regression, which predicts continuous values, logistic regression predicts discrete outcomes by transforming its output into a probability through the use of the **sigmoid function**.

#### The Logistic Function

At the core of logistic regression is the logistic function, often represented as the sigmoid function. The **sigmoid function** can be expressed mathematically as:

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

**Range** - Output of the sigmoid function ranges from 0 to 1, making it ideal for probability estimation. **S-shaped Curve** - Sigmoid curve has a characteristic S-shape (logistic curve), which smoothly transitions from 0 to 1.




#### Odds and Log-Odds

To understand how logistic regression works, we need to understand the concepts of odds and log-odds. Odds represent the ratio of the probability of an event occurring to the probability of it not occurring.

$$\text{Odds}(y=1|X) = \frac{P(y=1|X)}{P(y=0|X)} = \frac{P(y=1|X)}{1-P(y=1|X)}$$

Log-odds (or logit) is the natural logarithm of the odds. The log-odds transformation maps probabilities from (0, 1) to (-∞, +∞).

$$\text{Log-Odds}(y=1|X) = \log\left(\frac{P(y=1|X)}{1-P(y=1|X)}\right)$$

**Logistic Regression Model**

Logistic regression assumes that the log-odds of the outcome can be modeled as a linear combination of the input features:

$$\text{Log-Odds}(y=1|X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$$

Where:

$\beta_0$ is the intercept (constant term).

$\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients associated with each feature $X_1, X_2, \ldots, X_n$.

This equation can be rearranged to predict the probability of the target variable $y$

$$P(y=1|X) = \sigma(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n)$$

**Interpreting Coefficients**

Each coefficient $\beta_i$ in the logistic regression model represents the change in the log-odds of the outcome for a one-unit increase in the corresponding feature $X_i$, while holding all other features constant. A **positive coefficient** indicates that as the feature increases, the odds of the target variable being 1 increase. A **negative coefficient** indicates that as the feature increases, the odds of the target variable being 1 decrease.

| Coefficient ($\beta$) | Interpretation |
|---|---|
| $\beta > 0$ | Odds increase as $X$ increases |
| $\beta < 0$ | Odds decrease as $X$ increases |
| $\beta = 0$ | No effect on odds |

**Loss Function**

In logistic regression, we use the **Binary Cross-Entropy Loss (Log Loss)** as the cost function to evaluate model performance. The log loss quantifies the difference between the predicted probabilities and the actual binary outcomes.

$$L(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

Where:

$m$ is the number of training examples.

$y_i$ is the actual label (0 or 1).

$\hat{y}_i$ is the predicted probability of $y_i$ being 1.

### 4) Sigmoid Function: How It Works

The **sigmoid function** is a mathematical function that plays an important role in logistic regression and many other machine learning algorithms, especially in binary classification. Its ability to map any real-valued number into the range between 0 and 1 makes it particularly useful for predicting probabilities. Mathematically, it is defined as:

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

Where:

$\sigma(z)$ is the output of the sigmoid function.

$z$ is the input, which can be any real number.

$e$ is Euler's number, approximately equal to 2.71828.

**How the Sigmoid Function Works?**

**Shape of the Sigmoid Curve**

The sigmoid function produces an S-shaped curve, which has distinct characteristics. As $z$ approaches positive infinity, $\sigma(z)$ approaches 1. Conversely, as $z$ approaches negative infinity, $\sigma(z)$ approaches 0. The point at $z = 0$ is the inflection point where the function transitions from being less than 0.5 to greater than 0.5. At this point, the output of the sigmoid function is 0.5.

**Mapping Inputs to Probabilities**

The sigmoid function transforms any real-valued number (the linear combination of features in logistic regression) into a probability between 0 and 1. This transformation is important for binary classification.

- If $z < 0$: The output $\sigma(z) < 0.5$, indicating a higher likelihood of the negative class (0).
- If $z = 0$: The output $\sigma(0) = 0.5$, suggesting uncertainty between the classes.
- If $z > 0$: The output $\sigma(z) > 0.5$, indicating a higher likelihood of the positive class (1).

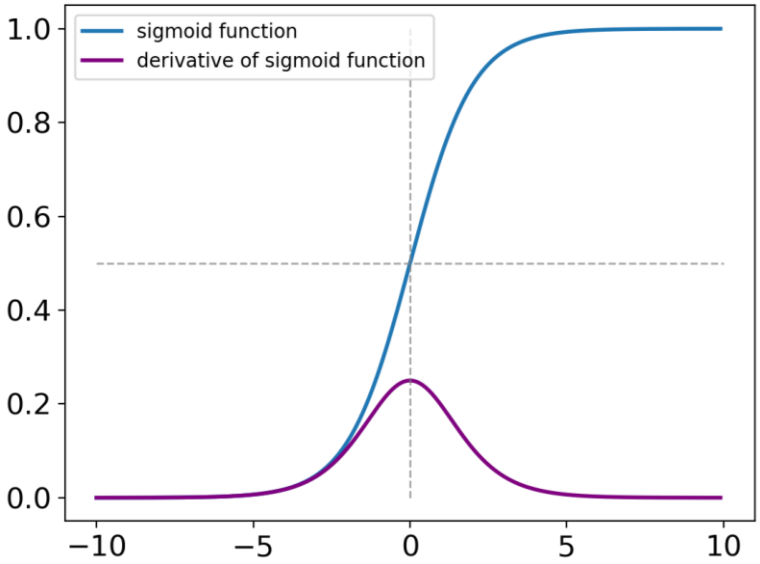
| Input ($z$) | Sigmoid Output ($\sigma(z)$) | Interpretation |
|---|---|---|
| -3 | 0.047 | High likelihood of class 0 |
| -1 | 0.268 | Likely class 0 |
| 0 | 0.500 | Uncertain between classes |
| 1 | 0.731 | Likely class 1 |
| 3 | 0.952 | High likelihood of class 1 |

**Derivative of the Sigmoid Function**

Understanding the derivative of the sigmoid function is essential for optimization algorithms, particularly during the training of models using gradient descent. The derivative of the sigmoid function can be expressed as:

$$\sigma'(z) = \sigma(z)(1 - \sigma(z))$$

This means the maximum slope of the sigmoid function occurs at $z = 0$ (where $\sigma(z) = 0.5$), making the learning process most efficient at this point. As $z$ moves away from 0, the slope diminishes, indicating that changes in $z$ lead to smaller changes in the output, slowing the learning process.



## 5) Model Training & Loss Function

Model training is the process of teaching a machine learning model to make predictions based on input data. In logistic regression, this involves adjusting the model's parameters (coefficients) to minimize the difference between the predicted and actual outcomes. The ultimate goal is to create a model that can generalize well to new, unseen data.

### Training Process

The training process consists of the following steps:

**Step 1: Initialization** - Start by initializing the model's parameters (weights) randomly or to zero.

**Step 2: Forward Pass** - For each input instance $X$, compute the predicted probability using the logistic function:

$$P(y = 1|X) = \sigma(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n)$$

**Step 3: Loss Calculation** - Calculate the loss (error) between the predicted probabilities and the actual binary outcomes using a loss function.

**Step 4: Backward Pass** - Adjust the model's parameters to minimize the loss using optimization techniques (like gradient descent).

**Step 5: Iteration** - Repeat steps 2 to 4 for a predefined number of iterations or until the loss converges.

### Loss Function: Heart of Model Training

The loss function quantifies how well the model's predictions match the actual outcomes. In logistic regression, we use the **Binary Cross-Entropy Loss (Log Loss)**, which is particularly suited for binary classification tasks.

#### Binary Cross-Entropy Loss

The Binary Cross-Entropy Loss can be mathematically expressed as:

$$L(\beta) = -\frac{1}{m} \sum_{i=1}^{m} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

Where:

$m$ is the number of training examples.

$y_i$ is the actual label (0 or 1).

$\hat{y}_i$ is the predicted probability of the instance being in class 1.

**Understanding the Loss Function**

Loss function penalizes incorrect predictions, where the penalty is higher when the model is more confident but wrong. For instance, if the true label is 1 but the predicted probability is very low, the loss will be significantly high. Loss value is always non-negative, with lower values indicating better model performance. A loss of 0 means perfect predictions.

| Actual Label | Predicted Probability | Loss Value |
|---|---|---|
| 0 | 0.1 | $-log(1-0.1)$ = 0.105 |
| 0 | 0.9 | $-log(1-0.9)$ = 2.302 |
| 1 | 0.1 | $-\log(0.1)$ = 2.302 |
| 1 | 0.9 | $-log(0.9)$ = 0.105 |

**Gradient Descent**

To minimize the loss function, we use gradient descent, an optimization algorithm that adjusts the model's parameters iteratively.

Compute the derivative of the loss function with respect to each parameter $\beta$:

$$\frac{\partial L}{\partial \beta_j} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)\,X_{ij}$$

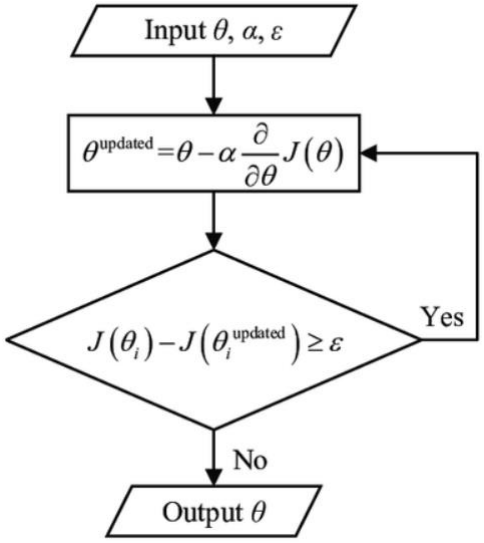
This tells us the direction and magnitude of the change needed for each parameter.

Update each parameter $\beta_j$ using the learning rate $\alpha$:

$$\beta_j \leftarrow \beta_j - \alpha\frac{\partial L}{\partial \beta_j}$$

Where: α is a small positive number (learning rate) that controls how much we adjust the weights during each iteration.

Repeat the process until the loss converges or a set number of iterations is reached.




## 6) Feature Scaling & Preprocessing

In machine learning, the quality of your data significantly influences the performance of your model. Feature scaling and preprocessing are really important steps that make sure that your model is trained effectively and can make accurate predictions. Proper scaling of features helps improve convergence during training and ensures that all features contribute equally to the model's learning process.

### Why Feature Scaling Matters

When features have different scales (e.g., age in years versus income in thousands), models may give undue weight to features with larger ranges. This can lead to:

**Slower Convergence** - Gradient descent may struggle to converge when the features are not on a similar scale, taking longer to find the optimal solution. **Poor Model Performance** - Models like logistic regression and K-nearest neighbors rely heavily on the distances between data points. If one feature dominates due to its scale, it can skew the results.

### Common Feature Scaling Techniques

**Min-Max Scaling** transforms the features to a fixed range, usually [0, 1].

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

Where:

$X'$ is the scaled feature.

$X_{\min}$ and $X_{\max}$ are the minimum and maximum values of the feature.

**Standardization (Z-score Normalization)** transforms the data to have a mean of 0 and a standard deviation of 1.

$$X' = \frac{X - \mu}{\sigma}$$

Where:

$\mu$ is the mean of the feature values.

$\sigma$ is the standard deviation.

**Other Preprocessing Steps**

**Handling Missing Values**

Missing values can skew results and reduce model accuracy. Common strategies include replacing missing values with the mean, median, or mode of the feature. You can also drop rows or columns with excessive missing data.

**Encoding Categorical Variables**

Categorical variables need to be converted into numerical form for the model to interpret them. You can either assign a unique integer to each category or perform One-Hot Encoding (creating binary columns for each category).

| Original Category | Label Encoded | One-Hot Encoded (A, B, C) |
|---|---|---|
| A | 0 | (1, 0, 0) |
| B | 1 | (0, 1, 0) |
| C | 2 | (0, 0, 1) |

## 7) Hands-On Implementation

Now, let us walk through the entire process of implementing logistic regression using Python. We'll cover data loading, preprocessing, feature scaling, model training, and evaluation using a sample dataset.

**7a) Setting Up the Environment**

Before we begin, make sure you have the following libraries installed. You can install them using pip:

***pip install pandas numpy scikit-learn matplotlib seaborn***

**7b) Loading the Dataset**

We are going to use the well-known Iris dataset, which contains data about iris flowers and their species. We will focus on predicting whether a flower is of species Iris-Virginica (1) or not (0) based on its features.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

# Load the iris dataset
url="https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
column_names = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
iris = pd.read_csv(url, header=None, names=column_names)

# Display the first few rows of the dataset
print(iris.head())
```

**7c) Data Exploration**

Let us visualize the data to understand the relationships between its features.

```
# Visualize the distribution of each feature
sns.pairplot(iris, hue='species')
plt.show()

# Check for missing values
print(iris.isnull().sum())
```

**7d) Data Preprocessing**

We will convert the species into a binary format where Iris-Virginica is labeled as 1, and all other species are labeled as 0.

```
# Binarize the target variable
iris['target'] = np.where(iris['species'] == 'Iris-virginica', 1, 0)

# Drop the original species column
iris = iris.drop(columns=['species'])

print(iris.head())
```

Now, we will split the dataset into training and testing sets.

```
# Split the data into features and target
X = iris.drop(columns=['target'])
y = iris['target']

# Split the dataset into training and testing sets (80-20 split)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print("Training set shape:", X_train.shape)
print("Testing set shape:", X_test.shape)
```

**7e) Feature Scaling**

Feature scaling is important for logistic regression. We will use ***StandardScaler*** from scikit-learn.

```
from sklearn.preprocessing import StandardScaler

# Initialize the scaler
scaler = StandardScaler()

# Fit the scaler to the training data and transform both training and testing data
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

print("Scaled training data:")
print(X_train_scaled[:5])
```

**7f) Model Training**

We will now implement logistic regression using scikit-learn.

```
from sklearn.linear_model import LogisticRegression

# Initialize the logistic regression model
model = LogisticRegression()

# Train the model
model.fit(X_train_scaled, y_train)
```

```
# Print the coefficients
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

**7g) Model Evaluation**

Now, let's evaluate the model's performance using accuracy and confusion matrix.

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Make predictions on the test set
y_pred = model.predict(X_test_scaled)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Display confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", conf_matrix)

# Display classification report
print("Classification Report:\n", classification_report(y_test, y_pred))
```

Let's visualize the confusion matrix for better understanding.

```
import seaborn as sns

# Visualize the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
        xticklabels=['Not Iris-Virginica', 'Iris-Virginica'],
        yticklabels=['Not Iris-Virginica', 'Iris-Virginica'])
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion Matrix')
plt.show()
```

### 8) Mini Challenge

Now that you have worked through a basic logistic regression model, it is time to challenge yourself. Using the **Banknote Authentication Dataset** from **UCI**, build a logistic regression model to classify whether a banknote is genuine or counterfeit. Your task is to:

- **Load the dataset, preprocess it, and train a logistic regression model.**
- **Evaluate your model using the confusion matrix and ROC-AUC.**
- **Explore the impact of regularization (L2 regularization) on your model performance.**

**Bonus:** Can you improve the model's performance by tuning its hyperparameters? Try using ***GridSearchCV*** to find the best parameters.

### 10) Deliverable

Please submit a single notebook on LMS with the above codes and mini challenge results before deadline.

### Additional Resources

- Scikit-learn Documentation
- Pandas Documentation
- Numpy Documentation
- Matplotlib Documentation
- Seaborn Documentation