Faculty of Computing and Information Technology

University of the Punjab,

Lahore

Artificial Intelligence Lab 12

Instructor: Qamar U Zaman

Logistic Regression

Objective

- Implement Logistic Regression to classify binary data using multiple features.
- Calculate predictions manually using the sigmoid function.
- Adjust weights using gradient descent.
- Evaluate the model using metrics like accuracy and cross-entropy loss.
- Visualize the decision boundary.

Key Concepts

1. **Logistic Regression**
   Logistic Regression is used to model the probability of a binary outcome based on one or more predictor variables.
2. **Sigmoid Function**
   The sigmoid function maps any real-valued number to a value between 0 and 1.
3. **Binary Cross-Entropy Loss**
   Measures the performance of a classification model by comparing predicted probabilities with actual labels.
4. **Gradient Descent**
   Used to minimize the loss function by adjusting weights iteratively.

Dataset

Feature 1 (X1)	Feature 2 (X2)	Target (Y)
0.1	1.1	0
1.2	0.9	0
1.5	1.6	1
2.0	1.8	1
2.5	2.1	1
0.5	1.5	0
1.8	2.3	1
0.2	0.7	0
1.9	1.4	1
0.8	0.6	0

Tasks for Students

1. **Data Preprocessing**
   - Normalize the dataset using standardization
   - Plot the data points on a 2D graph (X1 vs. X2) with different colors for the two classes.
2. **Train and Evaluate the Model**
   - Implement the logistic regression model using the templates provided below.
   - Calculate accuracy and cross-entropy loss on the dataset.
3. **Decision Boundary Visualization (Optional)**
   - Visualize the decision boundary generated by the model after training.

Code Template

```
def sigmoid(z):
    """
    Compute the sigmoid of z.
    """
    pass  # Implement sigmoid function here


def cross_entropy_loss(y_true, y_pred):
    """
    Compute binary cross-entropy loss.
    """
    pass  # Implement loss calculation here


def gradient_descent(X, y, weights, learning_rate, iterations):
    """
    Perform gradient descent to optimize weights.

    """
    pass  # Implement gradient descent here


def predict(X, weights):
    """
    Predict using sigmoid function.
    """
    pass  # Implement prediction logic here


def logistic_regression(X, y, learning_rate=0.01, iterations=1000):
    """
    Fit logistic regression model.
    """
    pass  # Implement the logistic regression fitting process here


def evaluate(y_true, y_pred):
    """
    pass  # Implement accuracy evaluation here
```

Instructions

1. **Data Preprocessing**
    - Standardize the input features X to improve the performance of gradient descent.
    - Visualize the data points before training to get a sense of the distribution of the classes.
2. **Model Training**
    - Use the **logistic_regression** function to fit the model on the dataset.
    - Adjust the learning rate and the number of iterations to observe the effect on model performance.
3. **Model Evaluation**
    - After training, evaluate the model by calculating accuracy and loss.
    - Experiment with different learning rates and iterations to optimize the performance.
4. **Visualization**
    - Plot the decision boundary to visualize how the model is separating the classes.

Tasks

1. **Experiment with Learning Rates:** Try different learning rates (e.g., 0.01, 0.1) and see how it affects the model's convergence and performance.
2. **Decision Boundary Visualization:** Visualize the decision boundary generated by the trained logistic regression model to understand how it separates the two classes.
3. **Model Tuning:** Experiment with the number of iterations to improve its accuracy.