Lab: Implementing Logistic Regression

Logistic regression is a classification algorithm used to model the probability of a binary outcome. Given a set of input features, the algorithm outputs a probability value between 0 and 1, which represents the likelihood of the positive class.

The hypothesis for logistic regression is represented by the **sigmoid function**, which takes any real-valued input and maps it to a value between 0 and 1.

The sigmoid function is given by:

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

where $z = X \cdot w$ (the dot product of the input feature vector X and the weight vector w). The sigmoid function's output is interpreted as the probability that a given input belongs to the positive class.

Lab Instructions

1. **Generate a Dataset:**
   - We need a dataset for binary classification, where each data point belongs to one of two classes (e.g., Class 0 or Class 1). Instead of collecting real-world data, we can use Scikit-learn's `make_classification` function to quickly create a synthetic dataset that is easy to work with and ideal for understanding logistic regression. Please read this page to learn how to use the `make_classification` function effectively for generating synthetic datasets. [1].
   - Split the dataset into training and test sets to separate data for training and evaluating the model.
2. **Visualize the Data:**

   Plot the data points to see the distribution of the two classes.
3. **Define the Sigmoid Function:**

   Implement the sigmoid function as defined in Equation (1).

[1] `https://scikit-learn.org/1.5/modules/generated/sklearn.datasets.make_classification.html`

4. **Define the Cost Function and Gradient Descent:**

   - Define a cost function to measure the error in predictions.
   - Implement gradient descent to iteratively optimize the weights.

   The cost function for logistic regression is:

$$J(w) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(h_w(x^{(i)})) + (1-y^{(i)})\log(1-h_w(x^{(i)}))\right] \quad (2)$$

   where:

   - m is the number of training samples.
   - $y^{(i)}$ is the true label for the i-th sample.
   - $h_w(x^{(i)}) = \sigma(w^T x^{(i)})$, which is the predicted probability.

   Gradient descent is used to minimize the cost function by updating the weights as follows:

$$w := w - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_w(x^{(i)}) - y^{(i)}\right)x^{(i)} \quad (3)$$

   where α is the learning rate.

   - Add an intercept term to the dataset to improve model fit.
   - Initialize the weights to zero.

5. **Run Gradient Descent:**

   Use gradient descent along with the cost function to optimize the weights.

6. **Plot the Cost Over Iterations:**

   Track and visualize the cost over iterations to observe convergence.

7. **Define a Prediction Function:**

   Implement a function that uses the sigmoid output to classify new samples.

8. **Evaluate the Model:**

   Use the test set to evaluate the model's accuracy and assess its performance.