Logistic Regression Implementation Report

Akansha Malviya

November 3, 2023

1 Introduction

The provided code demonstrates a naive implementation of logistic regression using Apache Spark and NumPy. This report elaborates on the key parts of the code.

2 Code Explanation

2.1 Imports and Constants

- Necessary libraries and modules are imported.
- The constant $D = 10$ specifies the number of dimensions.

2.2 Data Reading

- The function `readPointBatch` reads a batch of points from the input file into a NumPy matrix for efficient further computations.
- It converts the data into a NumPy matrix where each row represents a data point with its label and features.

2.3 Main Execution Block

- Within the `if __name__ == "__main__":` block, the script checks for the correct number of command line arguments, initializes a Spark session, and reads the input data.
- The `points` variable is an RDD (Resilient Distributed Dataset) that holds the data, which is cached for performance.

2.4 Model Initialization

- $\mathbf{w}$ is initialized to a random value, representing the coefficients or weights of the logistic regression model.

2.5 Gradient Computation

- The `gradient` function computes the logistic regression gradient for a matrix of data points.
- It separates labels and features, then computes the gradient of the logistic function using the formula

$$\frac{1}{1 + e^{-Y \cdot (X \cdot \mathbf{w})}} - 1$$

  where Y is the labels, X is the data points, and $\mathbf{w}$ is the current weights vector.
- The gradient is summed up across all data points in the batch.

2.6 Gradient Descent

- A loop over the specified number of iterations performs the gradient descent optimization.
- In each iteration, the gradient is computed for all batches of data points in parallel using Spark's `map` function, and then summed together using Spark's `reduce` function with the `add` function.
- The weights **w** are updated in each iteration by subtracting the computed gradient.

2.7 Final Output

- The final weights **w** are printed to the console.
- The Spark session is closed with `spark.stop()`.

3 Conclusion

This code illustrates a basic, distributed implementation of logistic regression using gradient descent. However, it's a naive implementation and for real-world applications, it's recommended to use more optimized and robust implementations provided by libraries or frameworks like Spark ML.