Neural Network Implementation Report

Contents

Problem Definition and Motivation

The task at hand is to implement a neural network from scratch using Python, NumPy, and Pandas. The primary motivation is to gain a deeper understanding of the fundamental concepts behind neural networks, including feedforward and backpropagation. Furthermore, the goal is to apply the implemented neural network on two different datasets, namely the IRIS dataset and the MNIST dataset, to solve classification problems.

Dataset Description

IRIS Dataset

The IRIS dataset consists of 150 samples of iris flowers, each belonging to one of three species: Setosa, Versicolor, or Virginica. The features include sepal length, sepal width, petal length, and petal width.

MNIST Dataset

The MNIST dataset is a collection of 28x28 pixel grayscale images of handwritten digits (0 through 9). It is a widely used benchmark dataset for image classification tasks.

Approach and Methodology

Neural Network Architecture

The implemented neural network consists of an input layer, a hidden layer, and an output layer. The activation function used is the sigmoid function. The weights are initialized with random values.

Data Preprocessing

For the IRIS dataset, the class labels are converted to numerical values (0, 1, 2), and one-hot encoding is applied to the target variable. The data is then split into training and testing sets.

For the MNIST dataset, the data is fetched using the fetch_openml function from scikit-learn. The images are normalized to the range [0, 1], and the data is split into training and testing sets.

Training

The neural network is trained using the backpropagation algorithm. The training process involves forward passes and backward passes for a specified number of iterations. The hyperparameters include the learning rate and the number of iterations.

Model Evaluation

The trained model is evaluated on the test sets of both datasets. For the IRIS dataset, the accuracy and confusion matrix are computed. For the MNIST dataset, the accuracy and confusion matrix are also calculated, considering the conversion from one-hot encoded predictions to class indices.

Implementation Details

The code is implemented in Python and relies on the NumPy and Pandas libraries for numerical operations and data manipulation. The scikit-learn library is used for loading the MNIST dataset and for splitting the data into training and testing sets.

Conclusion

The implemented neural network demonstrates its capability to handle classification tasks on both the IRIS and MNIST datasets. The model's performance is evaluated using accuracy and confusion matrices. The implementation process provides insights into the inner workings of a neural network, and the results on the datasets highlight its ability to generalize to different types of data.

Lessons Learned

- Understanding the importance of data preprocessing for different types of datasets.
- Gaining hands-on experience in implementing key components of a neural network.
- Appreciating the impact of hyperparameters, such as learning rate and the number of iterations, on model performance.

Tips for Improvement

- Experiment with different architectures, activation functions, and optimization techniques to enhance model performance.
- Explore regularization techniques to prevent overfitting.
- Consider using more advanced libraries or frameworks for large-scale neural network projects.