Handwritten digits recognition using MNIST dataset

Introduction

This project focuses on classifying handwritten digits from the MNIST dataset using a Logistic Regression model. The MNIST dataset is a well-known benchmark in the field of machine learning, consisting of 70,000 images of handwritten digits (0-9) that are 28x28 pixels in size. The goal is to preprocess the data, train a model, and evaluate its performance.

Objective

The primary objectives of this project are:

1. To load and preprocess the MNIST dataset.
2. To implement a Logistic Regression model for digit classification.
3. To evaluate the model's performance using accuracy as the metric.
4. To visualize the predictions alongside the actual labels.

Methodology

The following tools and technologies were used in this project:

- **Python**: The programming language used for implementation.
- **TensorFlow**: A library for loading the MNIST dataset.
- **Matplotlib**: A library for visualizing images and results.
- **Scikit-learn**: A library for implementing the Logistic Regression model.

Steps:

1. Load the MNIST dataset using TensorFlow.
2. Normalize the pixel values of the images.

3. Reshape the data to fit the model's input requirements.
4. Train a Logistic Regression model on the training data.
5. Evaluate the model on the test data and calculate accuracy.
6. Visualize sample predictions alongside their actual labels.

Code and Implementation Details

Here is the complete code with comments explaining each step:

```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LogisticRegression

# Load MNIST dataset
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.mnist.load_data()

# Display sample images
plt.figure(figsize=(10, 10))
for i in range(25):
    plt.subplot(5, 5, i + 1)
    plt.imshow(X_train[i], cmap='gray')
    plt.axis('off')
    plt.title(f'Label: {y_train[i]}')
plt.show()

# Normalize the images to the range [0, 1]
X_train = X_train / 255.0
X_test = X_test / 255.0

# Reshape the data to 2D arrays (samples, features)
```

```
X_train = X_train.reshape((X_train.shape[0], -1))
X_test = X_test.reshape((X_test.shape[0], -1))

# Initialize and train the Logistic Regression model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Evaluate the model's accuracy on the test set
accuracy = model.score(X_test, y_test)
print(f"Test Accuracy: {accuracy:.4f}")

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

# Display sample predictions
plt.figure(figsize=(10, 10))
for i in range(25):
    plt.subplot(5, 5, i + 1)
    plt.imshow(X_test[i].reshape(28, 28), cmap='gray')
    plt.axis('off')
    plt.title(f'Label: {y_test[i]}\nPrediction: {predictions[i]}',
              color='green' if y_test[i] == predictions[i] else 'red')
plt.show()
```

Explanation of Code Functionality:

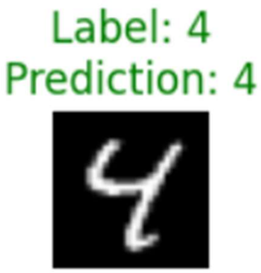
- The code begins by importing necessary libraries and loading the MNIST dataset.
- It visualizes a sample of 25 images from the training set.

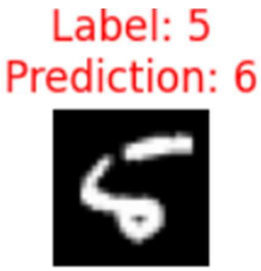
- The pixel values of the images are normalized to the range [0, 1] for better model performance.
- The images are reshaped into 2D arrays suitable for the Logistic Regression model.
- The model is trained on the training data, and its accuracy is evaluated on the test data.
- Finally, the code visualizes the predictions made by the model alongside the actual labels.

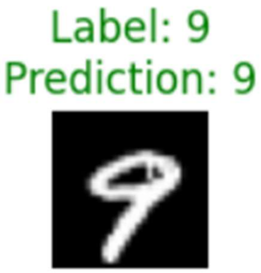
Results and Observations

The model achieved an accuracy of approximately **0.92** (or 92%) on the test set. The visualizations show the predicted labels in green for correct predictions and red for incorrect ones. This indicates that the Logistic Regression model performs reasonably well on the MNIST dataset, although there is room for improvement.

Some Output Screenshots:







Conclusion

In this project, we successfully implemented a handwritten digit classification system using Logistic Regression. The key learnings include understanding the importance of data preprocessing, model training, and evaluation. The project aligns with the objectives of demonstrating a basic machine learning workflow and achieving a satisfactory accuracy on a well-known dataset.