AI and Machine Learning, Homework8

Author: Ying Yiwen
Number: 12210159

Contents

1 Introduction

1.1 Task

The task of handwritten digit recognition has long been a benchmark problem in machine learning, providing valuable insights into pattern recognition and neural network performance. This report explores the implementation of a Multilayer Perceptron (MLP) model using Python and the NumPy library to classify handwritten digits from the Optical Recognition of Handwritten Digits (Optical Digits) dataset. The dataset consists of 8x8 grayscale images of handwritten digits, with each image represented as a 64-dimensional feature vector, and the corresponding label ranging from 0 to 9. In this project, we constructed an MLP model to classify these digits, trained it using the provided training dataset, and evaluated its performance on the test dataset. The primary goal is to assess the model's ability to accurately classify the digits and to analyze the factors influencing its performance, including the model architecture and training process.

1.2 Dataset

The Optical Digits dataset is a classical machine learning dataset containing 5620 handwritten digit samples for a 0-9 classification task. Each sample is captured by an optical character recognition device and is represented as an 8×8 grayscale image, where each pixel point has a grayscale value from 0 to 16. The data has been preprocessed into 64-dimensional feature vectors, which are suitable for training and evaluating classification models, and are an important benchmark dataset for studying pattern recognition and machine learning algorithms.

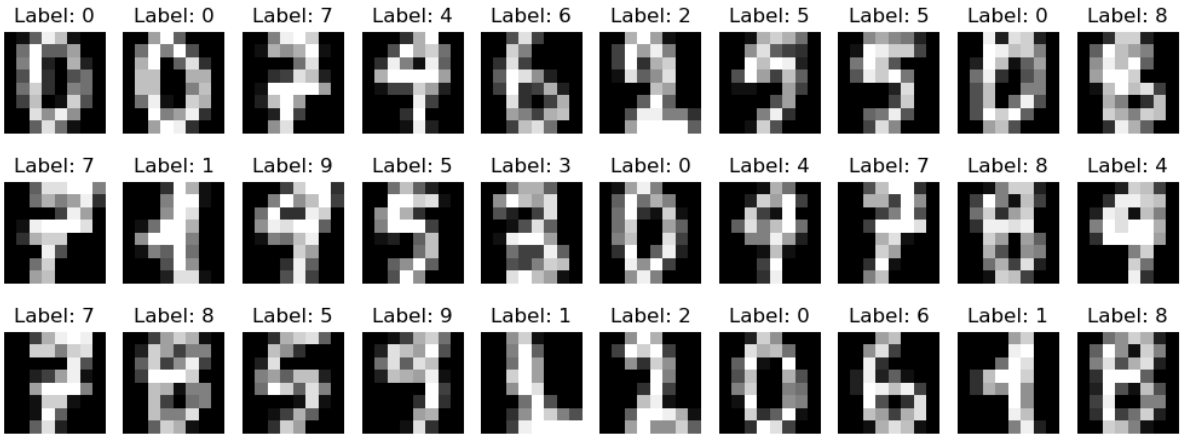


Fig. 1: Dataset Samples

2 Multi-Class MLP

2.1 Forward Propagation

Forward propagation refers to the process of passing inputs through the network to obtain predictions. In this model, the input is passed through each layer, and at each hidden layer, the following steps occur:

- Compute the weighted sum:
$$z^{[l]} = a^{[l-1]}W^{[l]} + b^{[l]}$$
where $a^{[l-1]}$ is the activation from the previous layer, $W^{[l]}$ is the weight matrix, and $b^{[l]}$ is the bias vector.
- Apply the ReLU activation function (except for the output layer):
$$a^{[l]} = \text{ReLU}(z^{[l]}) = \max(0, z^{[l]})$$

- Dropout is applied to the hidden layer activations during training:

$$\hat{a}^{[l]} = \frac{a^{[l]}}{1-p} \cdot \text{mask}$$

  where p is the dropout rate and mask is a randomly generated binary mask.

For the output layer, the softmax function is applied:

$$\hat{y} = \text{Softmax}(z^{[L]}) = \frac{e^{z^{[L]}}}{\sum_{i=1}^{C} e^{z_i^{[L]}}}$$

where C is the number of output classes.

2.2 Loss Function

The model uses cross-entropy loss with L2 regularization. The cross-entropy loss is computed as:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{m} \sum_{i=1}^{m} \sum_{c=1}^{C} y_{i,c} \log(\hat{y}_{i,c})$$

where $y_{i,c}$ is the true label for input i, $\hat{y}_{i,c}$ is the predicted probability, and m is the batch size.
L2 regularization is applied to the weights to prevent overfitting:

$$\mathcal{L}_{\text{L2}} = \lambda \sum_{l=1}^{L-1} \sum_{i,j} W_{i,j}^{[l]}$$

where λ is the regularization strength.
The total loss function is the sum of the cross-entropy loss and the L2 regularization term:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{L2}}$$

2.3 Backpropagation and Optimization

Backpropagation is used to compute the gradients of the loss with respect to the weights and biases. The gradient of the loss with respect to the output layer is:

$$\delta^{[L]} = \hat{y} - y$$

For hidden layers, the gradient is computed by backpropagating the error:

$$\delta^{[l]} = (\delta^{[l+1]} W^{[l+1]T}) \cdot \text{ReLU}'(z^{[l]})$$

where $\text{ReLU}'(z^{[l]}) = 1$ if $z^{[l]} > 0$, else 0.
The weights and biases are updated using the Adam optimizer:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) \nabla_\theta J(\theta)$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) \nabla_\theta J(\theta)^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\theta = \theta - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

where m_t and v_t are the first and second moment estimates, β_1 and β_2 are decay rates, η is the learning rate, and ϵ is a small value to avoid division by zero.

2.4 Training Process

The model is trained using mini-batch gradient descent with the Adam optimizer. The training process involves the following steps:

- Shuffle the training data and split it into mini-batches.
- For each mini-batch, perform a forward pass to compute predictions.
- Perform a backward pass to compute gradients and update parameters.
- Track the training and test losses over epochs.

The learning rate decays during training as follows:

$$\eta_{\text{new}} = 0.999 \times \eta_{\text{old}}$$

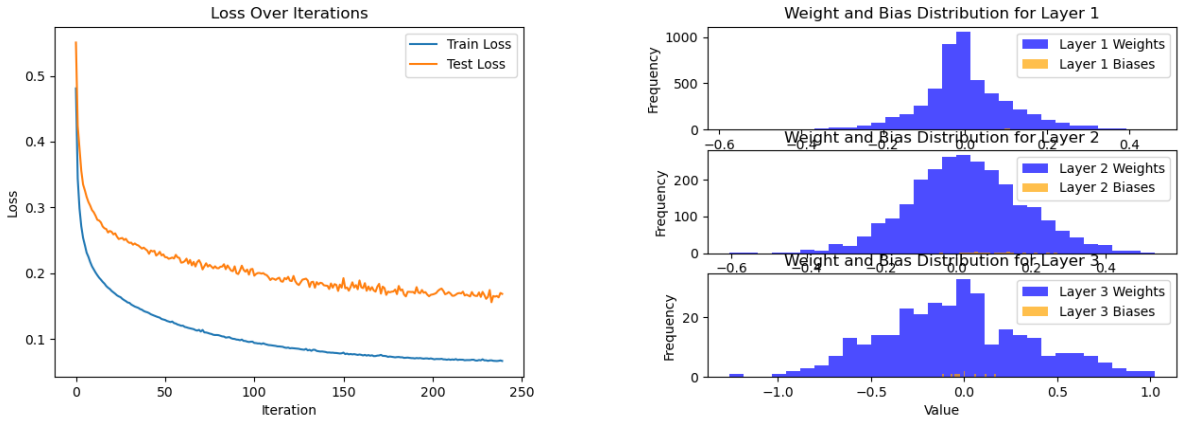
This helps the model converge more smoothly toward the optimal solution.

3 Result

We use the model to train the optical digits dataset, and got perfect results:

The parameters we use are:

```
mlp = MultiLayerPerceptron([64, 80, 32, 10], n_iter=240, lr=1e-3, batch_size=32)
```



(a) Loss Trend

(b) Weights and Biases

Fig. 2: Training Course

Epoch 1/240, Train Loss: 0.4809, Test Loss: 0.5506
Epoch 40/240, Train Loss: 0.1412, Test Loss: 0.2363
Epoch 80/240, Train Loss: 0.1059, Test Loss: 0.2076
Epoch 120/240, Train Loss: 0.0867, Test Loss: 0.1943
Epoch 160/240, Train Loss: 0.0762, Test Loss: 0.1888
Epoch 200/240, Train Loss: 0.0703, Test Loss: 0.1670
Epoch 240/240, Train Loss: 0.0664, Test Loss: 0.1684
Test accuracy: 0.9682804674457429

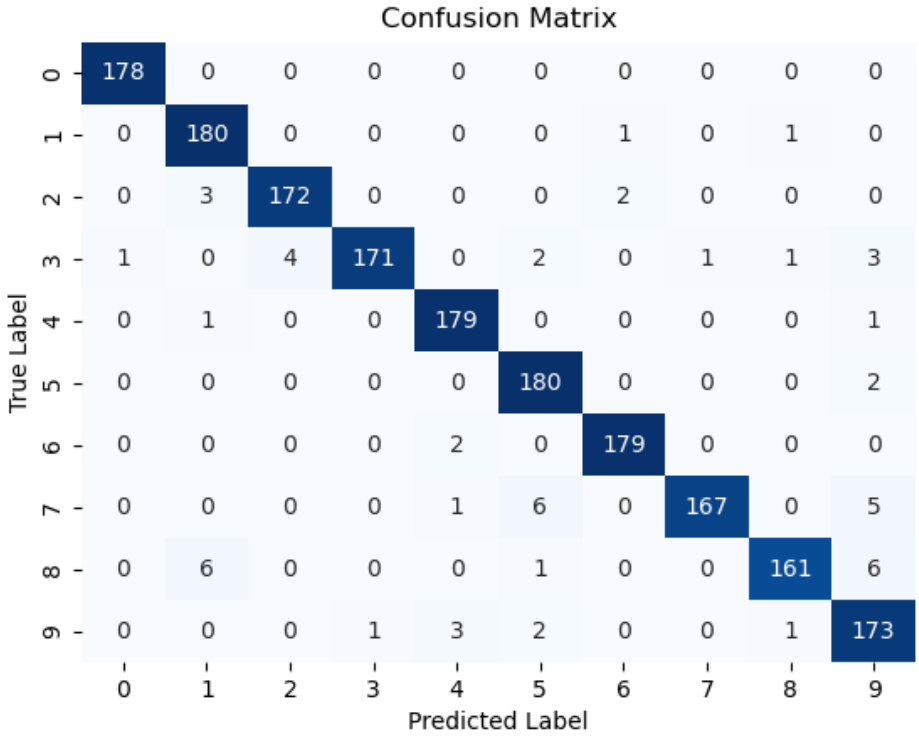


Fig. 3: Confusion Matrix

Seen from the confusion matrix, our model is very well.

4 Conclusion

In this report, we implemented and evaluated a Multilayer Perceptron (MLP) model for handwritten digit recognition using the Optical Digits dataset. The model demonstrated exceptional performance, achieving a test accuracy of 96.83% with a minimal test loss of 0.1684. The use of ReLU activations, dropout regularization, and the Adam optimizer effectively enhanced the model's robustness and generalization capabilities.
Key observations from this project include:

- The choice of architecture, particularly the intermediate hidden layer sizes (80 and 32 neurons), struck a balance between model complexity and computational efficiency.
- The regularized cross-entropy loss, combined with L2 regularization, successfully mitigated overfitting and stabilized the training process.
- The model's consistent reduction in training and test losses across epochs, as visualized in Fig. 2, highlights the efficacy of the learning rate decay schedule in fine-tuning the parameters.
- The confusion matrix (Fig. 3) shows that the model excels at classifying digits accurately, with minimal misclassification, reflecting its strong predictive capability.

Several factors may influence the performance of the MLP model in the handwritten digit recognition task. These factors include data quality, model design, and training processes, as detailed below:

- **Data Quality:** The quality of the training data is a critical factor. Noisy, incomplete samples or mislabeled data may lead the model to learn inaccurate patterns, reducing classification accuracy. Additionally, errors or inconsistencies in feature extraction and preprocessing could further degrade model performance.
- **Data Size:** The size of the training dataset directly affects the model's ability to learn. An insufficient dataset may prevent the model from capturing the relationship between inputs and outputs, leading to underfitting. Conversely, an overly large dataset could impose heavy computational demands. Although the Optical Digits dataset (5,620 samples) is moderate in size, it may still be insufficient for training more complex models.
- **Data Imbalance:** Imbalanced data, where some digit classes have significantly more samples than others, may cause the model to bias its predictions toward the majority class, reducing generalization ability. Addressing this issue may require techniques like oversampling, undersampling, or class-weight adjustment.
- **Number of Hidden Layers and Neurons:** The number of hidden layers and neurons determines the model's capacity. Too few layers or neurons may limit the model's ability to represent data complexity, leading to underfitting. Conversely, excessive layers or neurons may increase the risk of overfitting and computational cost. The current architecture (64-80-32-10) balances complexity and performance but could benefit from further tuning.
- **Hyperparameter Settings:** Key hyperparameters, such as learning rate, regularization strength, and batch size, greatly impact training. An excessively high learning rate can destabilize training, while a low rate may slow convergence. Improper regularization may fail to prevent overfitting effectively.
- **Risk of Gradient Explosion:** Deep networks may experience gradient explosion during backpropagation, leading to training instability. While using ReLU activations and regularization mitigates this risk, careful initialization and gradient clipping strategies are essential for ensuring stable training.
- **Limited Generalization Ability:** The diversity of the training data and the complexity of the model together determine its ability to generalize. If the model fails to adapt adequately to data variations, it may perform poorly on unseen data. Additionally, the lack of data augmentation techniques could further limit generalization.