Hướng dẫn Xây dựng và Huấn luyện Mạng Nơ-ron với MNIST

1. Chuẩn bị Dữ liệu

1.1. Tải Tập Dữ liệu MNIST

Giới thiệu về MNIST:

- Tập dữ liệu gồm 70,000 hình ảnh chữ số viết tay (0-9)
- Mỗi hình có kích thước 28x28 pixels
- Chia thành 60,000 ảnh cho huấn luyện và 10,000 ảnh cho kiểm tra

Nguồn tải:

- Yann LeCun's website
- Các thư viện như TensorFlow hoặc PyTorch (hàm tích hợp sẵn)

1.2. Chuẩn Hóa Giá Trị Pixel về Khoảng [0,1]

Quy trình chuẩn hóa:

- Giá trị pixel ban đầu: 0 đến 255
- Phương pháp: Chia mỗi giá trị pixel cho 255
- Ví dụ: Pixel giá trị 128 → 128/255 ≈ 0.502

Lợi ích:

- Tăng tốc quá trình huấn luyện
- Cải thiện hiệu suất mô hình

1.3. Chuyển Labels Thành One-Hot Encoding

Khái niệm One-Hot Encoding:

- Chuyển mỗi nhãn thành vector có kích thước bằng số lớp (10 lớp)
- Vị trí tương ứng với nhãn là 1, các vị trí khác là 0
- Ví dụ: Nhãn "3" → [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]

Mục đích:

- Giúp mô hình dễ dàng tính toán hàm mất mát
- Hỗ trợ cập nhật trọng số trong quá trình huấn luyện

1.4. Chia Tập Train/Test Theo Tỷ Lệ 8:2

Phân chia dữ liệu:

- Tổng số: 70,000 hình ảnh
- Tập huấn luyện: 56,000 hình ảnh (80%)

- Tập kiểm tra: 14,000 hình ảnh (20%)

Phương pháp chia:

- Sử dụng `train_test_split` của scikit-learn
- Đảm bảo tính ngẫu nhiên và đại diện của các lớp

2. Xây dựng Mạng Nơ-ron 3 Lớp

2.1. Thiết Kế Kiến Trúc Mạng Nơ-ron

Lớp Input:

- Số lượng neuron: 784 (28x28 pixels)
- Chức năng: Nhận và truyền dữ liệu đến lớp ẩn

Lớp Ẩn (Hidden Layer):

- Số lượng neuron: 128
- Hàm kích hoạt: ReLU
- Mục đích: Học các đặc trưng phi tuyến tính

Lớp Output:

- Số lượng neuron: 10 (cho 10 chữ số)
- Hàm kích hoạt: Softmax
- Chức năng: Chuyển đổi giá trị thành xác suất các lớp

2.2. Khởi Tạo Trọng Số và Bias

Trọng số (Weights):

- Khởi tạo ngẫu nhiên
- Phương pháp: Xavier Initialization hoặc He Initialization
- Mục đích: Tránh giá trị quá lớn hoặc nhỏ

Bias:

- Khởi tạo bằng 0 hoặc giá trị nhỏ
- Đảm bảo không lệ thuộc vào bias ban đầu

2.3. Xác Định Hàm Mất Mát và Hàm Tối Ưu

Hàm mất mát:

- Sử dụng Cross-Entropy Loss
- Phù hợp cho phân loại đa lớp

Hàm tối ưu:

- Gradient Descent hoặc các biến thể
- Các lựa chọn: SGD, Adam

3. Huấn luyện và Đánh giá

3.1. Huấn luyện Mô Hình (10 Epochs)

Quy trình mỗi epoch:

1. Forward Pass: Tính toán đầu ra
2. Tính Hàm Mất Mát: So sánh với nhãn thực tế
3. Backward Pass: Tính gradient
4. Cập Nhật Trọng Số và Bias: Sử dụng Gradient Descent

3.2. Sử dụng Gradient Descent

Công thức cập nhật:

```
θ = θ - η·∇θJ(θ)
```

Trong đó:

- θ: trọng số và bias
- η: tốc độ học
- ∇θJ(θ): gradient của hàm mất mát

Lưu ý quan trọng:

- Chọn tốc độ học phù hợp
- Xem xét sử dụng mini-batch
- Tránh các cực tiểu cục bộ

3.3. Vẽ Đồ Thị Loss Function

Quy trình:

1. Ghi lại giá trị loss mỗi epoch
2. Vẽ đồ thị bằng Matplotlib
3. Theo dõi quá trình hội tụ

Phân tích:

- Kiểm tra sự giảm của loss
- Phát hiện overfitting
- Đánh giá tốc độ hội tụ

3.4. Đánh giá Độ Chính Xác

Công thức tính accuracy:

```
Accuracy = (Số lượng dự đoán đúng / Tổng số dự đoán) × 100%
```

Quy trình đánh giá:

1. Forward Pass trên tập test
2. So sánh với nhãn thực tế
3. Tính tỷ lệ chính xác
4. Báo cáo và so sánh kết quả