Đồ án #2 - Xây dựng chương trình AI

Đinh Đức Anh Khoa - 23122001
Nguyễn Lê Hoàng Trung - 23122002
Nguyễn Đình Hà Dương - 23122004
Đinh Đức Tài - 23122013

FIT@HCMUS

TPHCM, tháng 1 năm 2024

Tổng quan

1. Mở đầu

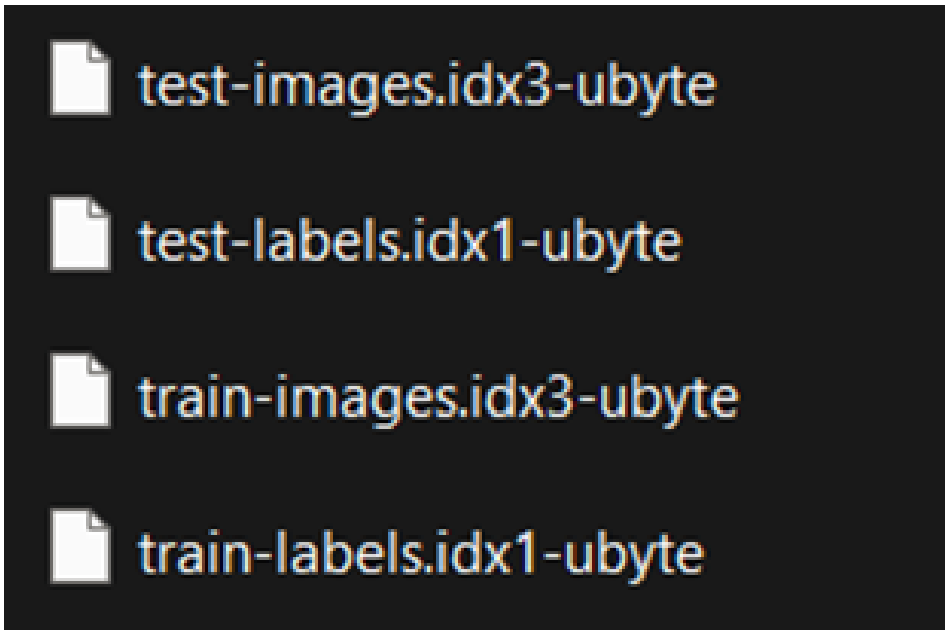
- Ở bài thuyết trình này, chúng ta sẽ tìm hiểu cách xây dựng mô hình neural network để giải quyết một bài toán cụ thể.

2. Bài toán cần giải quyết

- **Bài toán:** Nhận diện các chữ số viết tay.
- **Sơ lược về mô hình:** Mô hình nhận đầu vào (input) là hình ảnh có kích thước **28x28 pixel** chứa một chữ số bất kỳ từ **0 đến 9**, sau đó dự đoán và đưa ra kết quả là một số nguyên tương ứng với chữ số xuất hiện trong hình ảnh nhận được ban đầu.

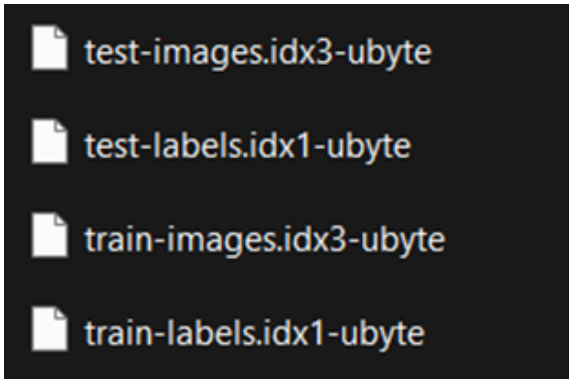
3. Dữ liệu và phân chia dữ liệu

- Dữ liệu được sử dụng trong mô hình này là tập dữ liệu **MNIST** được lấy từ website kaggle.com bao gồm 4 file:



Hình: Các tập dữ liệu

3. Dữ liệu và phân chia dữ liệu



- File `test-images.idx3-ubyte` chứa 60.000 hình ảnh có kích thước 28x28 pixel là các chữ số viết tay của tập huấn luyện (train set).
- File `train-labels.idx1-ubyte` chứa các nhãn là các chữ số tương ứng với từng hình ảnh của tập huấn luyện.
- File `test-images.idx3-ubyte` chứa 10.000 hình ảnh có kích thước 28x28 pixel là các chữ số viết tay của tập kiểm tra (test set).
- File `test-labels.idx1-ubyte` chứa các nhãn là các chữ số tương ứng với từng hình ảnh của tập kiểm tra.

3. Dữ liệu và phân chia dữ liệu

- Các hình ảnh trong tập dữ liệu đều có nền là màu đen và nét màu trắng thể hiện một chữ số bất kỳ từ 0 đến 9

Hình: Ví dụ một hình ảnh của tập dữ liệu thể hiện chữ số 3

3. Dữ liệu và phân chia dữ liệu

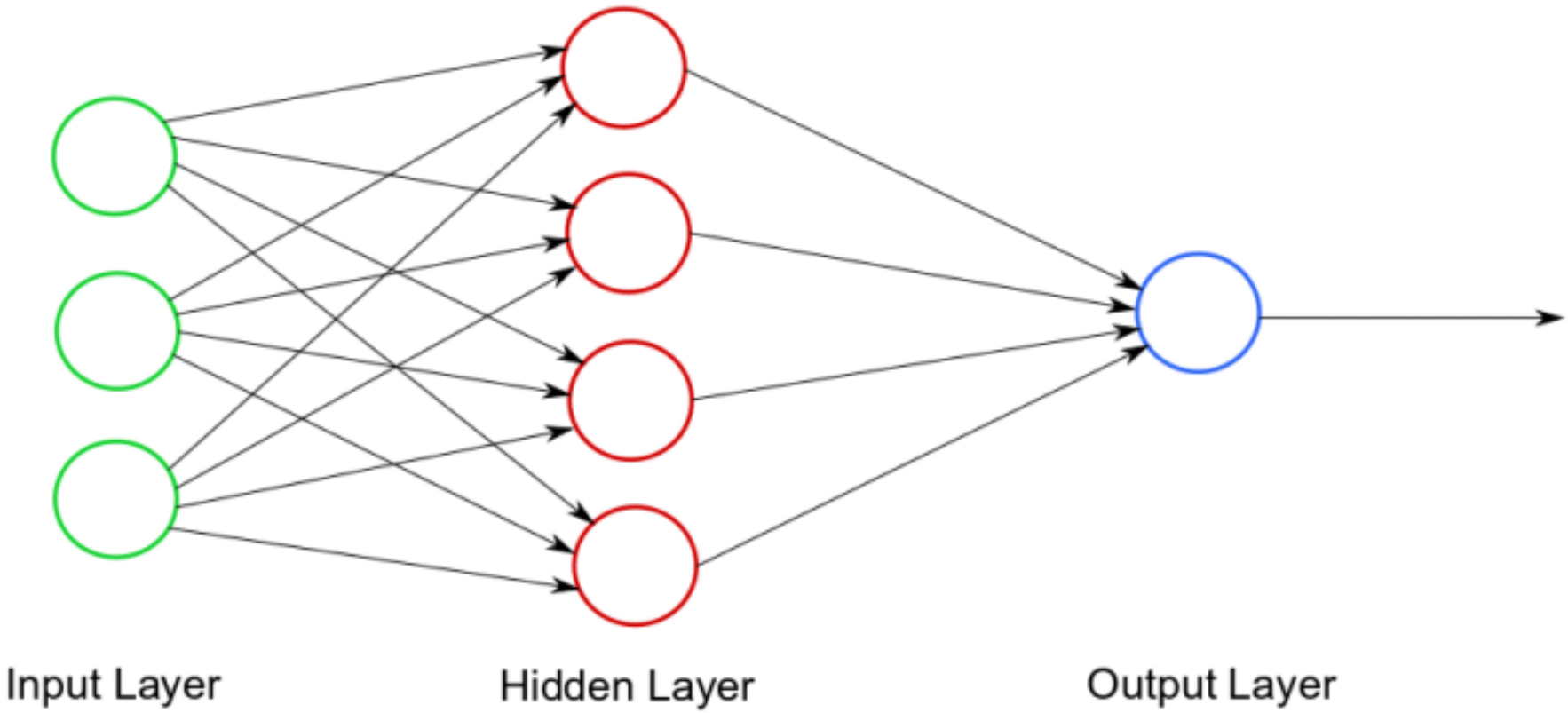
- Tiếp theo, sử dụng phương thức `train_test_split` từ thư viện `scikit-learn` lên train set để tạo ra được một tập dữ liệu mới là validation set. Lúc này, train set sẽ được tách ra thành hai tập ngẫu nhiên với 5% là validation set và phần còn lại là train set.
- Vậy ta sẽ có được ba tập dữ liệu sẽ được sử dụng cho các bước sau của mô hình:

- **Train set**: Sử dụng để huấn luyện và tinh chỉnh các tham số của mô hình.
- **Validation set**: Sử dụng để kiểm tra độ chính xác của mô hình trong quá trình học xem mô hình có thích nghi tốt với tập dữ liệu chưa từng thấy bao giờ hay không, từ đó có thể giảm thiểu vấn đề overfitting.
- **Test set**: Sử dụng để kiểm tra độ hiệu quả của mô hình sau khi được huấn luyện.

4. Thiết kế, huấn luyện neural network

- Kiến trúc neural network
- Lan truyền tiến, lan truyền ngược
- Loss function
- Optimizer
- Learning rate, số epoch, batch-size, regularization
- Độ chính xác và những vấn đề khác

4.1 Kiến trúc neural network



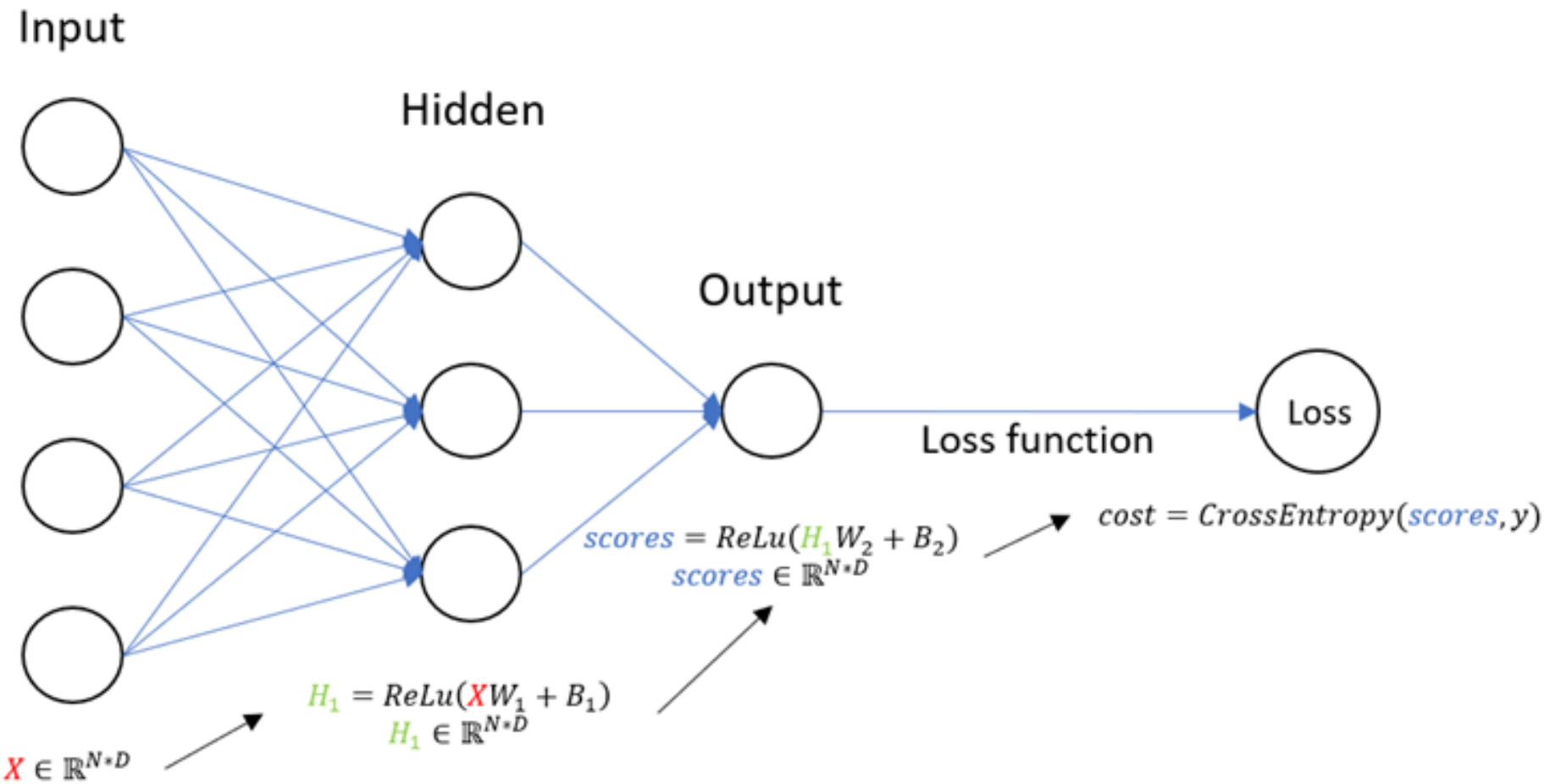
Hình: Mô hình neural network

4.1 Kiến trúc neural network

Kiến trúc: Mô hình neural network được sử dụng là một multilayer perceptron gồm một lớp đầu vào (input layer), một lớp ẩn (hidden layer) và một lớp đầu ra (output layer).

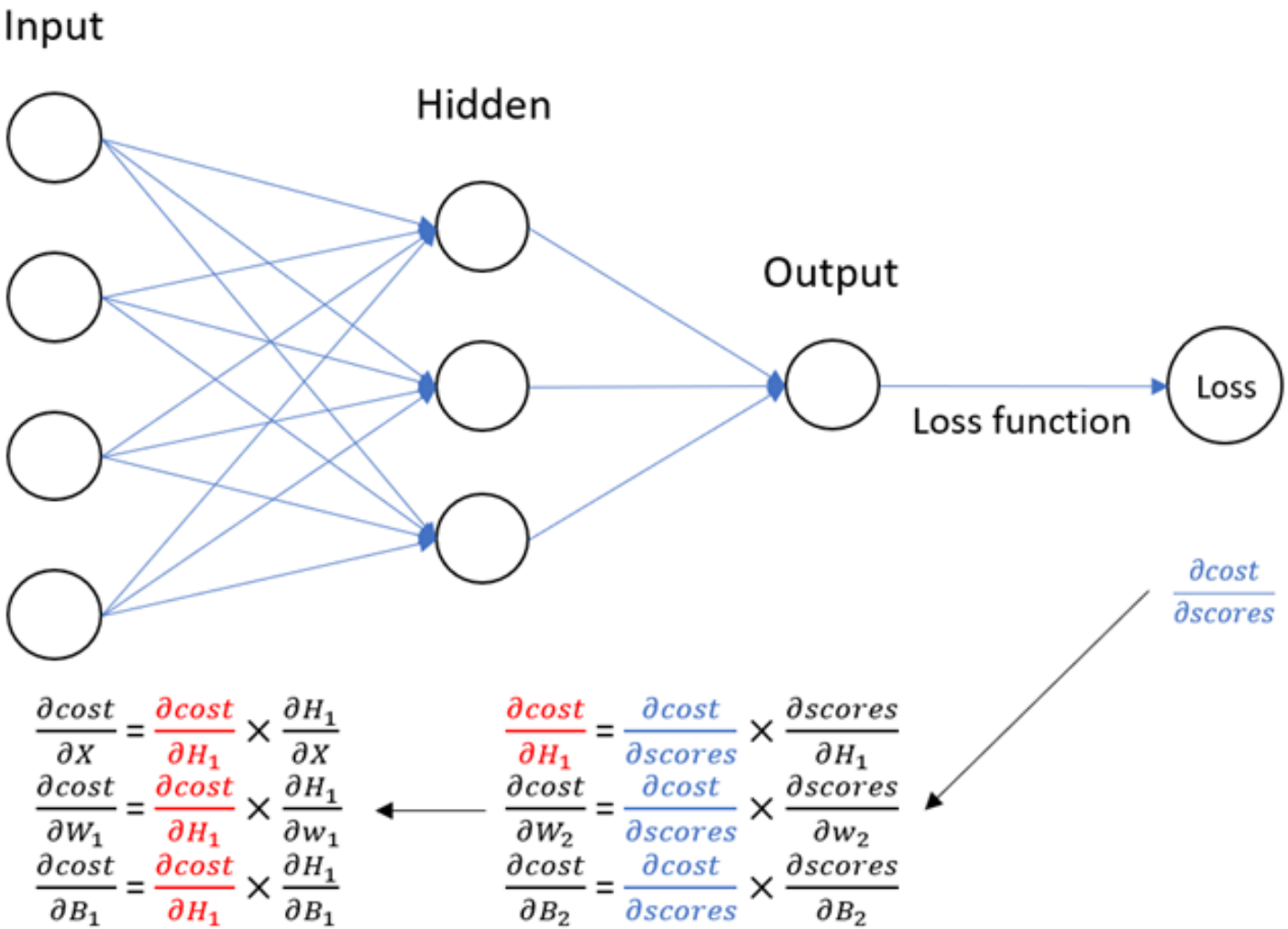
- Input layer: Gồm 784 nút, mỗi nút thể hiện một pixel trên ma trận ảnh 28x28
- Hidden layer(s): Mô hình này sẽ chỉ gồm 1 lớp ẩn duy nhất gồm 10 nút. Dữ liệu từ lớp Input sẽ thông qua hàm Linear và hàm ReLu để tính toán kết quả đầu ra cho lớp ẩn.
- Output Layer: Gồm 10 nút thể hiện các chữ số từ 0 đến 9. Kết quả đầu ra từ lớp ẩn sẽ thông qua hàm Linear để có được kết quả đầu ra cho lớp Output. Giá trị của mỗi nút là xác suất mà hình ảnh ban đầu chứa chữ số tương ứng mà nút đó thể hiện. Kết quả dự đoán của mô hình là nút có xác suất cao nhất.

4.2 Lan truyền tiến và lan truyền ngược



Hình: Mô hình mô phỏng lan truyền tiến

4.2 Lan truyền tiến và lan truyền ngược



Hình: Mô hình mô phỏng lan truyền ngược

4.2 Lan truyền tiến và lan truyền ngược

$$W_1 := W_1 - lr * \frac{\partial cost}{\partial W_1} \qquad W_2 := W_2 - lr * \frac{\partial cost}{\partial W_2}$$

$$B_1 := B_1 - lr * \frac{\partial cost}{\partial B_1} \qquad B_2 := B_2 - lr * \frac{\partial cost}{\partial B_2}$$

Hình: Công thức cập nhật tham số (Weight và Bias) của từng lớp.

4.3 Loss function: CE và hàm Softmax

- Đối với mô hình này, kết quả đầu ra của lớp Output có thể là các giá trị âm. Do đó, ta không thể trực tiếp sử dụng các giá trị này để tính toán sai số thông qua hàm Cross-Entropy được. Thay vào đó ta sẽ **chuẩn hóa các giá trị này bằng hàm Softmax** có hai tính chất là các xác suất luôn nằm trong khoảng (0, 1] và tổng các xác suất bằng 1. Sau đó sử dụng các giá trị đã chuẩn hóa này để tính toán sai số thông qua hàm Cross-Entropy.
- Giả sử có một vector đầu vào $z = (z_1, z_2, ..., z_k)$, hàm Softmax sẽ tính xác suất p_i cho mỗi phần tử theo công thức:

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}, \forall i = 1, 2, ..., k$$

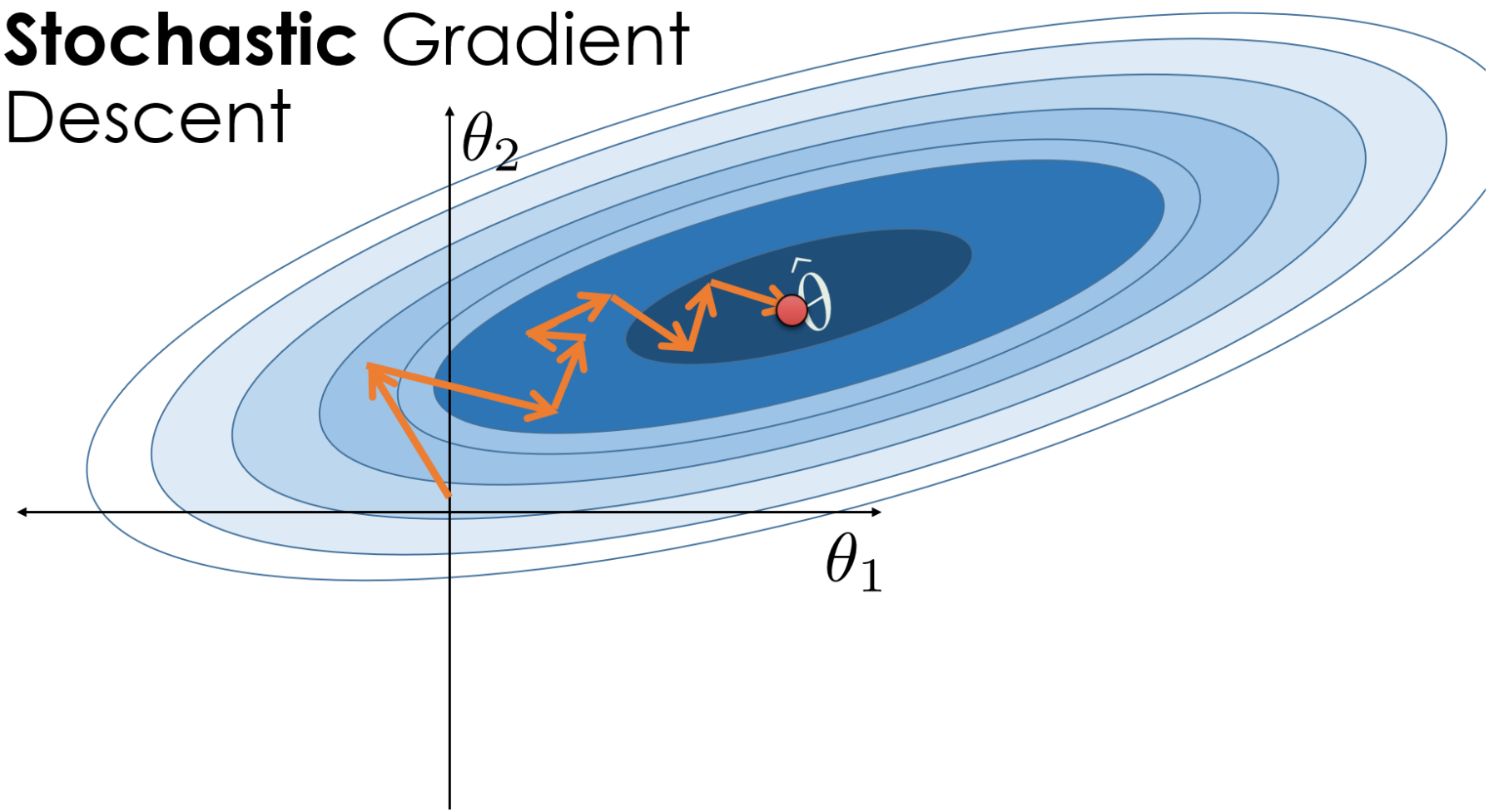
4.3 Loss function: CE và hàm Softmax

Cross-Entropy Loss function:

$$CE = -\sum_{i=1}^{C} P_i \times \log(Q_i)$$

Trong đó:

- C: số lượng các class cần phân lớp.
- Q_i: xác suất thực tế của lớp thứ i
- P_i: xác suất dự đoán của lớp thứ i bởi mô hình



Hình: Stochastic Gradient Descent

4.5 Learning rate, số epoch, batch-size, regularization

- **Learning rate:**

- Ban đầu khởi tạo giá trị `learning_rate` thủ công, cứ sau mỗi `decay_after` lần lặp thì ta sẽ cập nhật giá trị `learning_rate` mới bằng cách nhân với một lượng `learning_rate_decay` (với `learning_rate_decay` < 1).
- Điều này sẽ giúp tăng dần tính chính xác với `learning_rate` giảm dần khi tiến lại gần điểm hội tụ, tránh trường hợp `learning_rate` quá lớn dẫn đến việc phân kì.

- $learning\_rate = 0.001$
- $decay\_after = 50$
- $learning\_rate\_decay = 0.99$

4.5 Learning rate, số epoch, batch-size, regularization

- **Số epoch:** Số lần truyền tập train để huấn luyện cho neural network
  - $epoch = 200$
- **Batch-size:** Tách bộ dữ liệu train thành các batch nhỏ hơn với kích thước là `batch-size` để tối ưu tốc độ huấn luyện.
  - $batch - size = 200$
- **Regularization:** Hằng số được thêm vào quá trình tính gradient để giảm hiện tượng overfitting
  - $Regularization = 5 \times 10^{-6}$

5. Quá trình hội tụ, kết quả kiểm thử mô hình

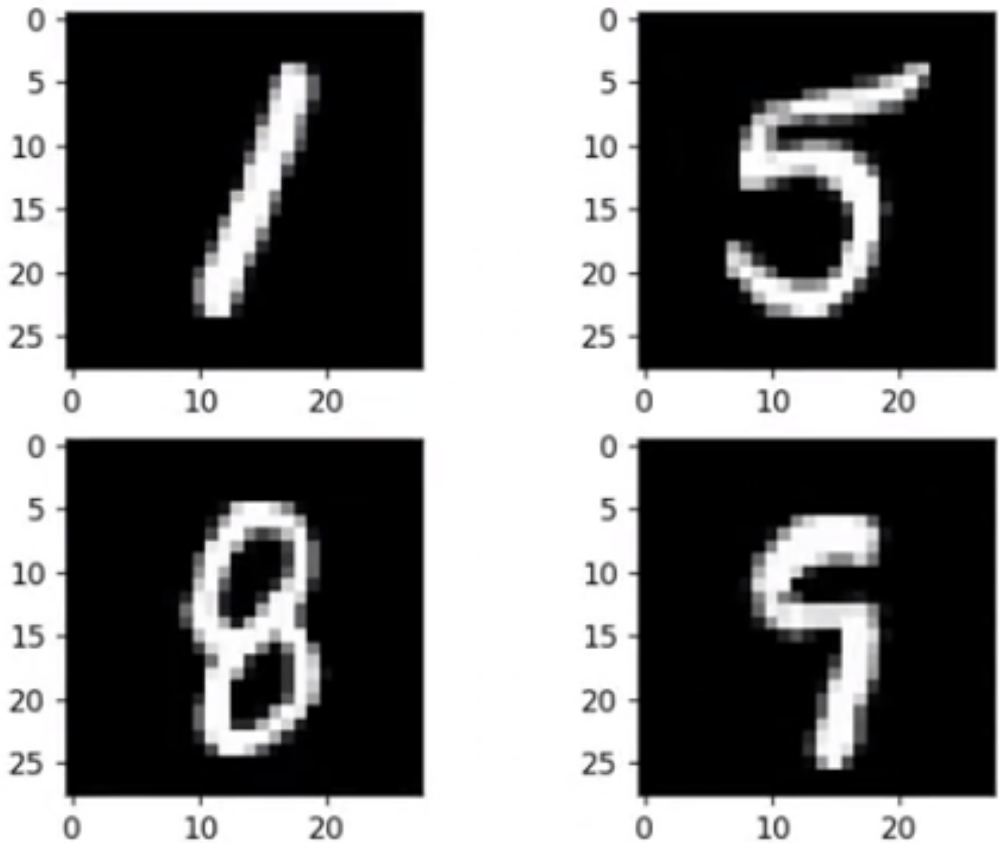
- Trong quá trình huấn luyện, giá trị của hàm loss (average cost) giảm dần và độ chính xác của train set và validation set tăng dần theo số lần lặp (epoch).
- Trong quá trình huấn luyện, ở **epoch cuối**:
  - Giá trị cost trung bình: 0.36
  - Độ chính xác (accuracy) trên train set: 89%
  - Độ chính xác (accuracy) trên validation set: 89%
- **Trong quá trình kiểm thử mô hình, độ chính xác (accuracy) trên test set: 89%**

```
Epoch 189: average cost 0.37, train accuracy 0.89, val accuracy 0.88.
----------------------------------------------------------------------
Epoch 199: average cost 0.36, train accuracy 0.89, val accuracy 0.89.
Test accuracy: 0.89.
```

Hình: Accuracy - test set

6. Demo

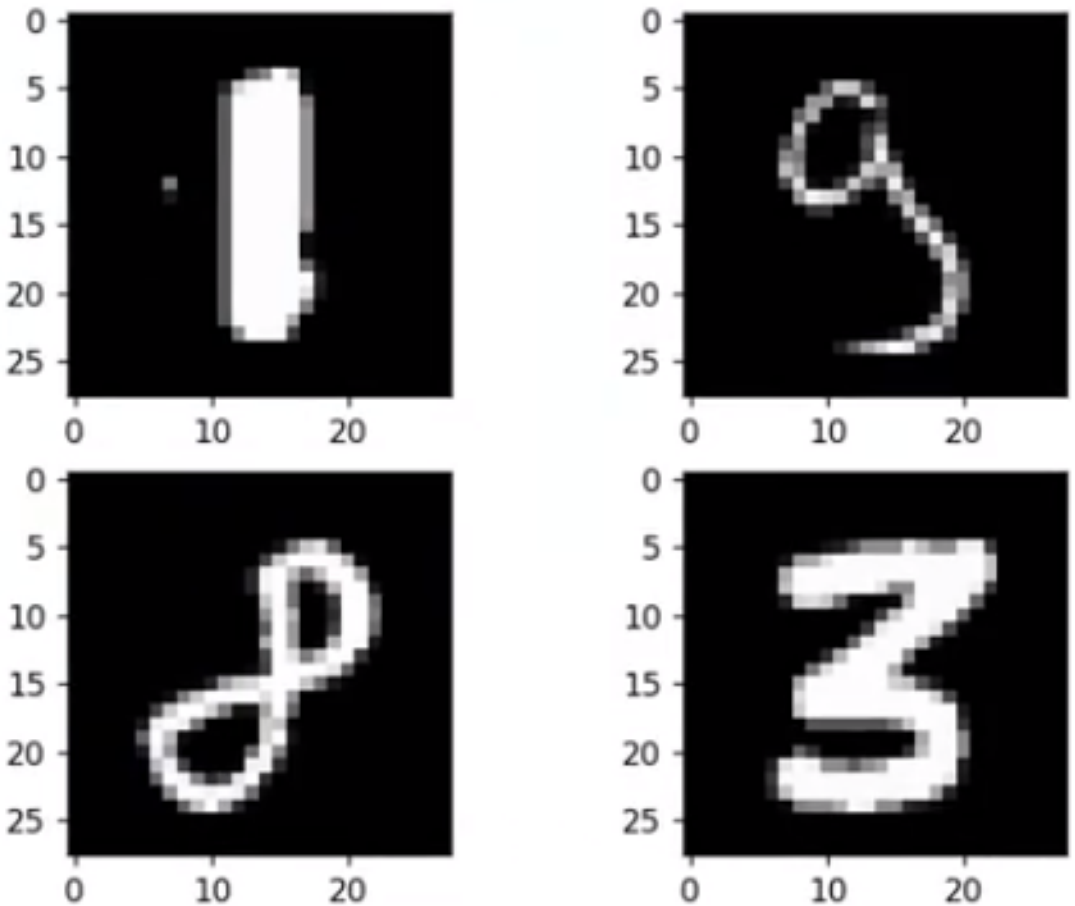
- 4 hình ảnh trong test set mà mô hình dự đoán đúng



Hình: Các mẫu mô hình dự đoán đúng

6. Demo

- 4 hình ảnh trong test set mà mô hình dự đoán sai



Tham Khảo

- The MNIST database of handwritten digits
  https://www.kaggle.com/datasets/hojjatk/mnist-dataset
- MLP Neural Network
  https://github.com/SonPhatTran/Neural-Network-from-scratch
- Demo
  https://www.youtube.com/watch?v=UQExhzgt-6E

The End