ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

LỚP TRÍ TUỆ NHÂN TẠO 2023 - 23TNT1

Báo cáo Đồ án #2

Đề tài: Xây dựng chương trình AI

Môn học: Thực hành - Giới thiệu ngành Trí tuệ nhân tạo

Sinh viên thực hiện:
Đinh Đức Anh Khoa (23122001)
Nguyễn Đình Hà Dương (23122002)
Nguyễn Lê Hoàng Trung (23122004)
Đinh Đức Tài (23122013)

Giáo viên hướng dẫn:
CN. Nguyễn Bảo Long

Ngày 1 tháng 1 năm 2024

Mục lục

1 Giới thiệu

Đây là bài báo cáo Đồ án #2, môn Thực hành Giới thiệu ngành Trí tuệ Nhân tạo.
Đồ án #2 có đề tài: Xây dựng chương trình AI.
Đồ án #2 được thực hiện bởi nhóm các thành viên:

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

2 Phân công nhiệm vụ

Sau đây là bảng phân công nhiệm vụ cho từng thành viên:

- **Tuần 2:**

Họ và tên	MSSV	Chức vụ	Nhiệm vụ
Đinh Đức Anh Khoa	23122001	Nhóm trưởng	- Tìm hiểu Loss Function - Tìm hiểu quá trình tiền xử lý dữ liệu
Nguyễn Đình Hà Dương	23122002	Thành viên	- Tìm hiểu hàm kích hoạt (Activation Function) - Tìm hiểu cách mô hình dự đoán mẫu
Nguyễn Lê Hoàng Trung	23122004	Thành viên	- Tìm hiểu thuật toán tối ưu (Optimizer)
Đinh Đức Tài	23122013	Thành viên	- Tìm hiểu cách đánh giá mô hình (Metrics) - Soạn slide trình chiếu, viết báo cáo đồ án

- **Tuần 3:**

Họ và tên	MSSV	Chức vụ	Nhiệm vụ
Đinh Đức Anh Khoa	23122001	Nhóm trưởng	- Viết phần dữ liệu, phân chia dữ liệu - Viết quá trình lan truyền tiến, lan truyền ngược, hàm Softmax - Quay demo
Nguyễn Đình Hà Dương	23122002	Thành viên	- Viết kiến trúc neural network
Nguyễn Lê Hoàng Trung	23122004	Thành viên	- Viết về optimizer SGD, quá trình hội tụ, kiểm thử mô hình - Chỉnh sửa và tải lên video demo
Đinh Đức Tài	23122013	Thành viên	- Viết các tham số của neural network - Soạn slide trình chiếu, viết báo cáo đồ án

3 Tổng quan công việc

3.1 Tuần 1: Các bài toán chính trong AI

Nội dung chính: (đã tìm hiểu trên lớp)

- Bài toán phân lớp
- Bài toán hồi quy
- Bài toán gom cụm

3.2 Tuần 2: Tìm hiểu quy trình xây dựng một chương trình AI

Nội dung chính:

- Quy trình xây dựng chương trình AI
- Tiền xử lý dữ liệu (Data preprocessing)
- Huấn luyện mô hình (Train model/Tune model)
- Đánh giá mô hình (Test model)

3.3 Tuần 3: Giải quyết bài toán tự chọn

Nội dung chính:

- Xác định bài toán
- Dữ liệu
- Xây dựng mạng neutral dựa trên dữ liệu
- Thiết kế, huấn luyện mạng neutral
- Quá trình hội tụ, kết quả kiểm thử mô hình

4 Tuần 1

4.1 Bài toán phân lớp

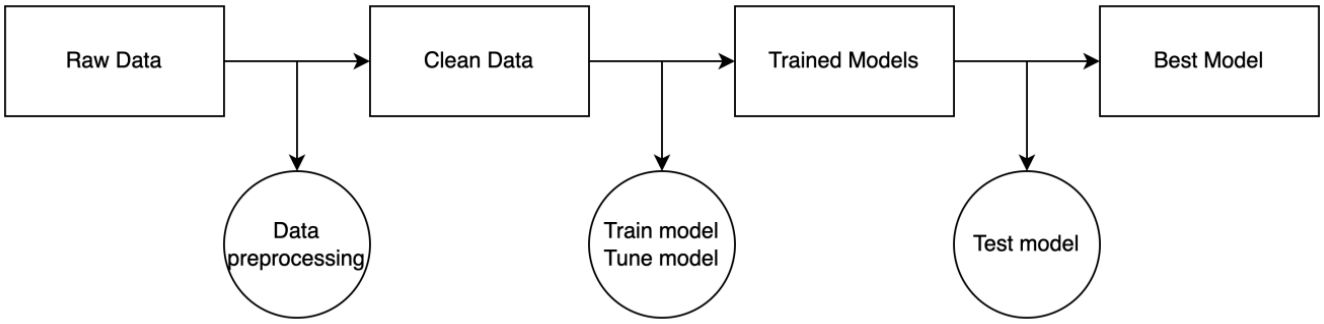
4.2 Bài toán hồi quy

4.3 Bài toán gom cụm

5 Tuần 2

5.1 Quy trình xây dựng chương trình AI

- Quy trình xây dựng một chương trình AI bắt đầu từ việc thu thập dữ liệu thô. Sau đó **tiền xử lý** để thu được dữ liệu sạch. Tiếp đến, **huấn luyện mô hình** trên dữ liệu đó và **đánh giá mô hình** thu được. Hình 1 tóm tắt quy trình này:



Hình 1: Quy trình xây dựng một chương trình AI

5.2 Tiền xử lý (Data preprocessing)

- Quá trình tiền xử lý dữ liệu [1] là bước quan trọng để biến đổi dữ liệu thô trở thành dữ liệu phù hợp cho quá trình huấn luyện của một mô hình AI. Một số quy trình tiền xử lý dữ liệu có thể kể đến bao gồm:

- Làm sạch dữ liệu (Data cleaning)
- Tích hợp dữ liệu (Data integration)
- Thu giảm dữ liệu (Data reduction)
- Biến đổi dữ liệu (Data transform)

1. **Làm sạch dữ liệu (Data cleaning):** Là quy trình loại bỏ các dữ liệu sai lệch, thiếu sót hoặc không phù hợp khỏi tập dữ liệu. Dữ liệu thô thường có xu hướng không đầy đủ, bị nhiễu và không toàn vẹn dữ liệu. Việc làm sạch dữ liệu sẽ giúp ta xử lý vấn đề trên. Ví dụ như xóa hàng loạt cột dữ liệu trống hoặc không có giá trị hay xóa các giá trị ngoại lai (các giá trị quá cao hoặc quá thấp so với các giá trị khác).

2. **Tích hợp dữ liệu (Data integration):** Là quy trình hợp nhất các dữ liệu khác nhau từ nguồn khác nhau thành một tập dữ liệu duy nhất nhằm tránh việc dư thừa và làm mất đi tính toàn vẹn của dữ liệu trong tập dữ liệu. Ví dụ với một công ty có hai loại dữ liệu, một loại để theo dõi bán hàng và một loại dùng để theo dõi khách hàng. Để phân tích hiệu quả bán hàng, ta cần phải kết hợp cả hai dữ liệu này thành một tập dữ liệu duy nhất.

3. **Thu giảm dữ liệu (Data reduction):** Là quy trình loại bỏ những thuộc tính dư thừa của dữ liệu thô mà không làm ảnh hưởng hay làm mất những thông tin quan trọng. Ví dụ với

tập dữ liệu của một siêu thị gồm các danh mục sản phẩm trong siêu thị và số lượng giao dịch hàng ngày. Có thể thấy rằng tập dữ liệu này khá lớn và việc khai phá cũng tốn rất nhiều thời gian, dẫn đến việc phân tích dữ liệu này trở nên bất khả thi. Như vậy quy trình thu giảm dữ liệu là rất cần thiết.

4. **Biến đổi dữ liệu (Data transform):** Là quy trình biến đổi dữ liệu từ dạng này sang dạng khác để phù hợp với quá trình khai phá dữ liệu. Ví dụ như chuyển đổi nhiệt độ từ độ C sang độ F hay chuyển đổi số điểm từ thang điểm 10 sang thang điểm 4 của một học sinh.

- Sau khi tiền xử lý, tùy thuộc vào từng bài toán mà ta sẽ chia dữ liệu đã xử lý thành nhiều tập dữ liệu để sử dụng cho các bước tiếp theo. Một số cách chia thường được sử dụng:

- Tập huấn luyện (Training sets), tập xác thực (Validation sets) và tập kiểm thử (Test sets)
- Chia dữ liệu theo thời gian (time series segmentation)

1. **Tập huấn luyện (Training sets), tập xác thực (Validation sets) và tập kiểm thử (Test sets):** Đây là cách chia dữ liệu rất phổ biến. Ta chia dữ liệu thành 3 phần khác nhau gồm: Tập huấn luyện dùng để huấn luyện mô hình, tập xác thực dùng để điều chỉnh các tham số của mô hình và tập kiểm thử dùng để đánh giá hiệu suất của mô hình.

2. **Chia dữ liệu theo thời gian (time series segmentation):** Với cách chia dữ liệu này, ta sẽ chia dữ liệu thành các tập dữ liệu khác nhau theo thời gian. Điều này sẽ giúp ích cho việc phân tích liên quan đến các xu hướng hoặc các biến động theo thời gian.

5.3 Huấn luyện mô hình (Train model/Tune model)

5.3.1 Cách mô hình dự đoán

- Quy trình dự đoán của mô hình là một quy trình toán học được xây dựng để dự đoán hoặc ước lượng các giá trị đầu ra dựa trên đầu vào cụ thể bằng cách phân tích các mẫu trong một tập hợp dữ liệu đầu vào nhất định.
- Quá trình huấn luyện của một mô hình dự đoán thông thường bao gồm các bước sau:

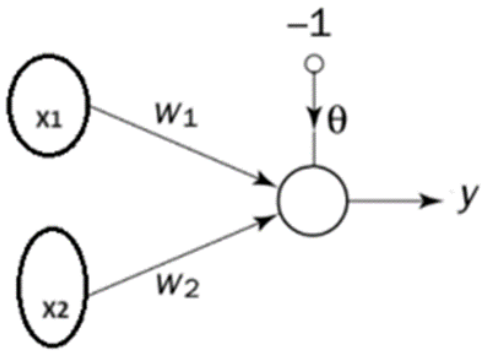
- **Chuẩn bị dữ liệu đầu vào:** Trước khi mô hình có thể dự đoán kết quả, ta cần chuẩn bị dữ liệu đầu vào, khởi tạo trọng số ngẫu nhiên cho các kết nối giữa các nút. Bước đầu cần phải thu thập dữ liệu đầu vào, điều này bao gồm hoạt động tiền xử lý dữ liệu (chuẩn hóa, mã hóa và chuyển đổi dữ liệu thành định dạng phù hợp) và sắp xếp các dữ liệu thành một tập dữ liệu duy nhất.
- **Truyền thuận** (Forward propagation): Dữ liệu được truyền qua các lớp (lớp đầu vào, lớp ẩn và lớp đầu ra) trong neural network, sử dụng activation function để tính toán đầu ra dựa vào dữ liệu và trọng số hiện tại.
- **Tính toán đầu ra:** Các dữ liệu được đưa qua neural network để huấn luyện mô hình qua nhiều vòng lặp (gọi là epochs), trọng số của mô hình được cập nhật để giảm thiểu hàm mất mát (Loss function).
- **Đánh giá đầu ra:** Đầu ra được đánh giá bằng cách so sánh với giá trị thực tế hoặc nhãn đã biết trước (đối với bài toán có giám sát). Điều này giúp đánh giá hiệu suất của mô hình và xác định mức độ chính xác của dự đoán

5.3.2 Ví dụ về mạng neural network cơ bản

- **Đặt vấn đề:** Xây dựng 1 mạng neural network để dự đoán đầu ra của toán tử AND, với 2 input là chân trị của A và B.
  Bảng chân trị của A, B và A AND B:

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

- **Mô hình neural network:** Ta sẽ xây dựng một neural network (thực chất là một perceptron) với input layer có 2 nút và output layer có 1 nút như sau:



Hình 2: Neural network biểu diễn được toán tử AND

- **Các bước huấn luyện mô hình neural network:**

  1. **Khởi tạo:** Ban đầu, **trọng số** (weight) (w_1, w_2) và **ngưỡng** (threshold)(θ) được gán các giá trị ngẫu nhiên (thông thường là trong đoạn [-0,5; 0,5])
  2. **Kích hoạt:** Ở lần học thứ p với đầu vào x_1, x_2, ta tính toán output như sau:

$$Y_p = \sigma(x_1 w_1 + x_2 w_2 - \theta)$$

  Trong đó:

  - x_1, x_2: đầu vào
  - w_1, w_2: trọng số tương ứng với mỗi đầu vào
  - θ: ngưỡng phân loại (classification threshold)
  - $\sigma(x)$: hàm kích hoạt - ta dùng hàm bước (step function)

$$\sigma(x) = \begin{cases} 1 & if \quad x \geq 0 \\ 0 & if \quad x > 0 \end{cases}$$

3. **Cập nhật trọng số:**

$$w_{i(p+1)} = w_{i(p)} + \Delta w_{i(p)}$$

Trong đó:

- $w_{i(p+1)}$: trọng số của lần học thứ `p+1`
- $w_{i(p)}$: trọng số của lần học thứ `p`
- $\Delta w_{i(p)}$: sự thay đổi khi cập nhật trọng số

Quy tắc điều chỉnh trọng số:

$$\Delta w_{i(p)} = a \times x_i \times e(p)$$

Trong đó:

- a: tốc độ học (learning rate) $(0 < a < 1)$
- $e(p)$: $e(p) = Y_{d(p)} - Y_p$ với Y_p là giá trị mô hình đưa ra cho lần học thứ p, $Y_{d(p)}$ là giá trị thực tế.

4. **Lặp lại:** Tăng lần lặp thứ `p` lên 1 và quay lại bước 2 cho đến khi thu được kết quả thích hợp.

- **Huấn luyện mô hình**

- Với $\theta = 0.2$, learning rate $a = 0.1$, ta có bảng sau:

Epoch	**Inputs**		**Ouput thực tế Y_d**	**Trọng số ban đầu**		**Output dự đoán Y**	**Error e**	**Cập nhập trọng số**	
	x_1	x_2	Y_d	W_1	W_2			W_1	W_2
1	0	0	0	0.3	-0.1	0	0	0.3	-0.1
	0	1	0	0.3	-0.1	0	0	0.3	-0.1
	1	0	0	0.3	-0,1	1	-1	0.2	-0.1
	1	1	1	0.2	0.1	0	1	0.3	0.0
2	0	0	0	0.3	0.0	0	0	0.3	0.0
	0	1	0	0.3	0.0	0	0	0.3	0.0
	1	0	0	0.3	0.0	1	-1	0.2	0.0
	1	1	1	0.3	0.0	1	0	0.2	0.0
3	0	0	0	0.2	0.0	0	0	0.2	0.0
	0	1	0	0.2	0.0	0	0	0.2	0.0
	1	0	0	0.2	0.0	1	-1	0.1	0.0
	1	1	1	0.1	0.0	0	1	0.2	0.1
4	0	0	0	0.2	0.1	0	0	0.2	0.1
	0	1	0	0.2	0.1	0	0	0.2	0.1
	1	0	0	0.2	0.1	1	-1	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1
5	0	0	0	0.1	0.1	0	0	0.1	0.1
	0	1	0	0.1	0.1	0	0	0.1	0.1
	1	0	0	0.1	0.1	0	0	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1

- Khi đó, ta được bộ trọng số là $w_1 = 0.1$, $w_2 = 0.1$.

5.3.3 Activation function

- **Activation function** (Hàm kích hoạt) của một nút trong mạng neutral nhân tạo là một hàm dùng để tính toán đầu ra của nút đó (dựa trên đầu vào, trọng số và bias của từng đầu vào).
- Activation function dùng để xác định đầu ra của mạng neutral. Activation function ánh xạ các giá trị vào khoảng từ 0 đến 1 hoặc -1 đến 1,...(tùy thuộc vào hàm).
- **Ý nghĩa:** Trong mô hình neural network không sử dụng activation function, phép nhân tuyến tính có thể tạo ra giá trị đầu ra vô cùng nhỏ hoặc vô cùng lớn, gây ra vấn đề tính toán và khó hội tụ. Dùng activation function sẽ giới hạn đầu ra trong khoảng giá trị cụ thể, giúp ổn định mô hình.
- Activation function được chia thành 2 loại:

- Linear Activation Function
- Non-linear Activation Functions

- Một số activation function phổ biến:

- Sigmoid (Logistic)
- Tanh (hyperbolic tangent)
- ReLU (Rectified Linear Unit)
- Leaky ReLU

1. **Sigmoid (Logistic):** Hàm sigmoid có dạng đường cong. Hàm Sigmoid nhận đầu vào là một số thực và chuyển thành một giá trị trong khoảng (0; 1).

$$S(x) = \frac{1}{1+e^{-x}}$$

   - Ưu điểm:
     - Có đạo hàm đẹp, dễ xây dựng mô hình và cập nhật tham số dựa trên back-propagation.
   - Nhược điểm:
     - Bão hòa và triệt tiêu gradient (vanishing gradient).
     - Không có trung tâm là 0 gây khó khăn cho việc hội tụ.
     - Sử dụng hàm mũ, tính toán lâu.

2. **Tanh (hyperbolic tangent):** Hàm tanh nhận đầu vào là một số thực và chuyển thành một giá trị trong khoảng (-1; 1).

$$\tanh x = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

   - Ưu điểm:
     - Đối xứng qua 0 - khắc phục được một nhược điểm của Sigmoid.

- Nhược điểm:
  - Hàm Tanh bị bão hoà ở 2 đầu, gây hiện tượng vanishing gradient.

3. **ReLU (Rectified Linear Unit)**

$$R(x) = max(0, x)$$

- Ưu điểm:
  - Tốc độ hội tụ nhanh hơn hàm Tanh và Sigmoid.
  - Tính toán nhanh hơn do có công thức đơn giản hơn hàm Tanh và Sigmoid.
- Nhược điểm:
  - Với các nút có giá trị nhỏ hơn 0, qua hàm ReLU sẽ thành 0, gây ra hiện tượng "Dying ReLU".
  - Hàm ReLu không được chặn trên nên có thể gây ra hiện tượng exploding gradient.

4. **Leaky ReLU**

$$R(x) = max(0, 01x, x)$$

- Ưu điểm:
  - Khắc phục hiện tượng "Dying ReLU"

5.3.4 Loss function

- **Loss function** [2] [3] là hàm số được sử dụng để đo lường mức độ sai lệch giữa dự đoán của mô hình học máy và giá trị thực tế, giá trị của loss function càng nhỏ thì mô hình dự đoán càng chính xác. Từ đó ta có thể điều chỉnh các tham số của mô hình để cải thiện hiệu suất của mô hình.
- Một số loss function phổ biến:

- Mean Squared Error (MSE)
- Mean Absolute Error (MAE)
- Cross-Entropy Loss function

1. **Mean Squared Error (MSE):** Là một loss function phổ biến được sử dụng cho các mô hình hồi quy. Về cơ bản, hàm này đánh giá mức độ sai lệch giữa các giá trị được dự đoán và các giá trị thực bằng cách tính trung bình các giá trị bình phương của các sai lệch đó. MSE có công thức:

$$MSE = \frac{1}{N}\sum_{j=1}^{N}(y_i - \hat{y}_i)^2$$

Trong đó:

- y_i: giá trị thực tế

- $\hat{y}_i$: giá trị được dự đoán bởi mô hình
- N: số lượng dữ liệu

2. **Mean Absolute Error (MAE):** Là một loss function thường được sử dụng cho các mô hình hồi quy. Hàm này đánh giá mức độ sai lệch giữa các giá trị được dự đoán và các giá trị thực bằng cách tính trung bình các giá trị tuyệt đối của các sai lệch đó. MAE có công thức:

$$MAE = \frac{1}{N}\sum_{j=1}^{N}|y_i - \hat{y}_i|$$

Trong đó:

- y_i: giá trị thực tế
- $\hat{y}_i$: giá trị được dự đoán bởi mô hình
- N: số lượng dữ liệu

3. **Cross-Entropy Loss function:** Là một loss function thường được sử dụng trong các bài toán phân loại bao gồm cả các mô hình neural network hoặc bài toán dự đoán xác suất. Hàm này đo lường mức độ sai lệch giữa hai phân phối xác suất: phân phối dự đoán của mô hình và phân phối thực tế. Công thức chung của Cross-Entropy Loss function cho hai phân phối xác suất P và Q là:

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

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

5.3.5 Optimizer

- Về cơ bản, optimizers là cơ sở để xây dựng mô hình neural network với mục đích huấn luyện mô hình được các features (hay pattern) của dữ liệu đầu vào, sử dụng thuật toán hoặc phương pháp thay đổi sự điều chỉnh của neural network như weights (trọng số) và learing rate (tốc độ học) để đáp ứng đầu ra của loss function.
- Một vài thuật toán tối ưu:

- Gradient Descent (GD)
- Stochastic Gradient Descent (SGD)
- Momentum
- Adagrad
- RMSprop

- Adam

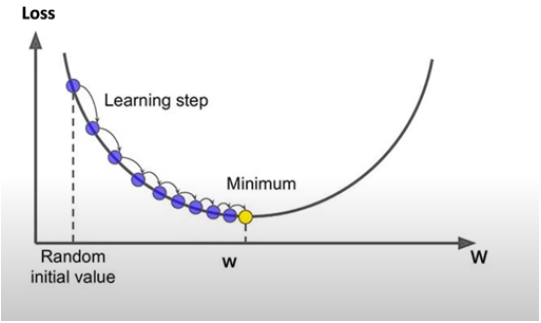
1. **Gradient Descent (GD)**

   - Gradient Descent là thuật toán tìm điểm gần với điểm cực tiểu nhất (hay minimum) và xem đó là nghiệm bài toán. Hướng tiếp cận của thuật toán ở đây là chọn 1 nghiệm ngẫu nhiên, cứ sau mỗi vòng lặp (hay epoch) thì cho nó tiến dần đến điểm cần tìm để chắc chắn rằng cho việc huấn luyện mô hình đi đúng như mong muốn.

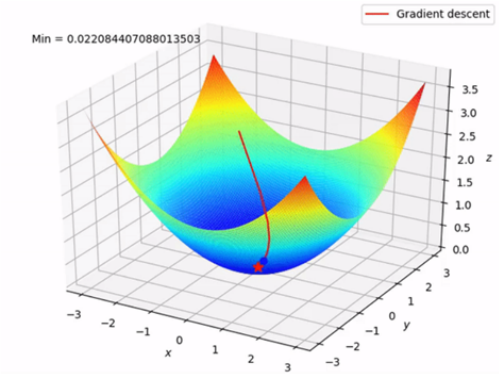
   $$x_{new} = x_{old} - learing\_rate \times gradient(x)$$

   Trong đó:

   – $gradient(x)$: đạo hàm của hàm f theo x
   – Nếu đạo hàm của hàm số < 0 thì $x_{new} > x_{old}$ nên nghiệm sẽ di chuyển về bên phải tiến gần tới điểm cực tiểu
   – Nếu đạo hàm của hàm số < 0 thì $x_{new} < x_{old}$ nên nghiệm sẽ di chuyển về bên trái tiến gần tới điểm cực tiểu

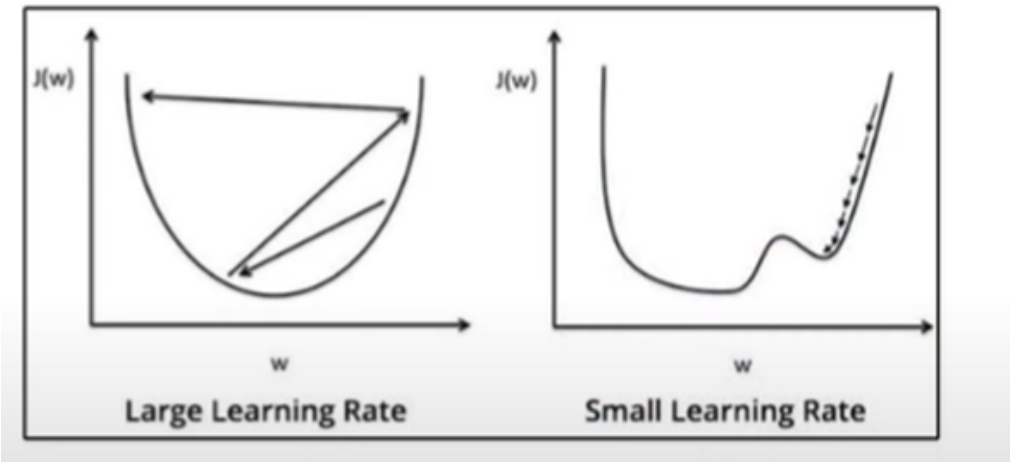


Hình 3: GD cho hàm 1 biến



Hình 4: GD cho hàm nhiều biến

- Với learing rate lớn thì ta sẽ tìm được điểm minimum nhanh chóng nhưng với độ chính xác không cao (như hình bên trái chỉ mất 3 epochs)
- Với learing rate nhỏ thì ta sẽ tìm được điểm minimum có độ chính xác cao nhưng sẽ mất khá nhiều thời gian.

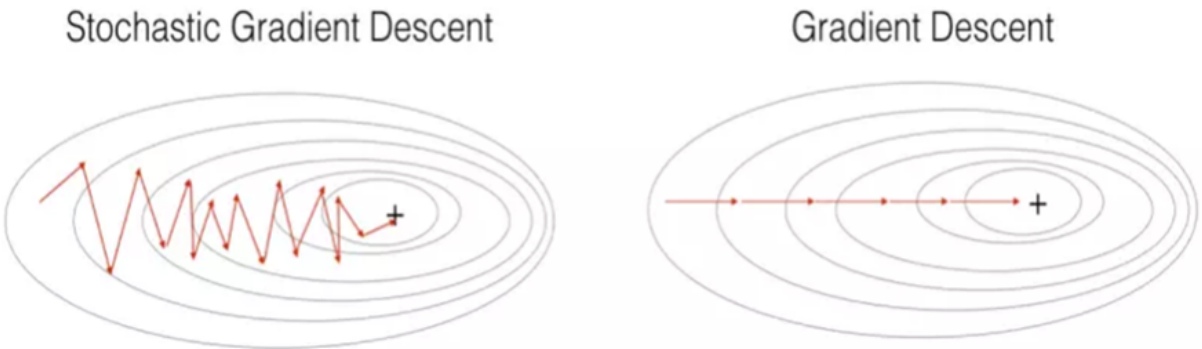


Hình 5: Large và Small Learning Rate

- Ưu điểm: thuật toán cơ bản, dễ hiểu, giải quyết được vấn đề tối ưu model bằng cách cập nhật trong số sau mỗi epoch.
- Nhược điểm: vì đơn giản và lâu đời nên GD còn nhiều hạn chế như phụ thuộc vào nghiệm khởi tạo ban đầu và learing rate. Ngoài ra tốc độ học quá lớn sẽ khiến cho thuật toán không hội tụ hoặc tốc độ học nhỏ ảnh hưởng đến tốc độ training.

2. **Stochastic Gradient Descent (SGD)**

- Stochastic Gradient Descent là một biến thể của Gradient Descent. Thay vì sau mỗi epoch chúng ta sẽ cập nhật trọng số một lần, thì trong mỗi epoch có N điểm dữ liệu, chúng ta sẽ cập nhật trọng số N lần. Nhờ vậy nên SGD sẽ hội tụ rất nhanh chỉ sau vài epoch, ngoài ra SGD có thể phù hợp với online learning.

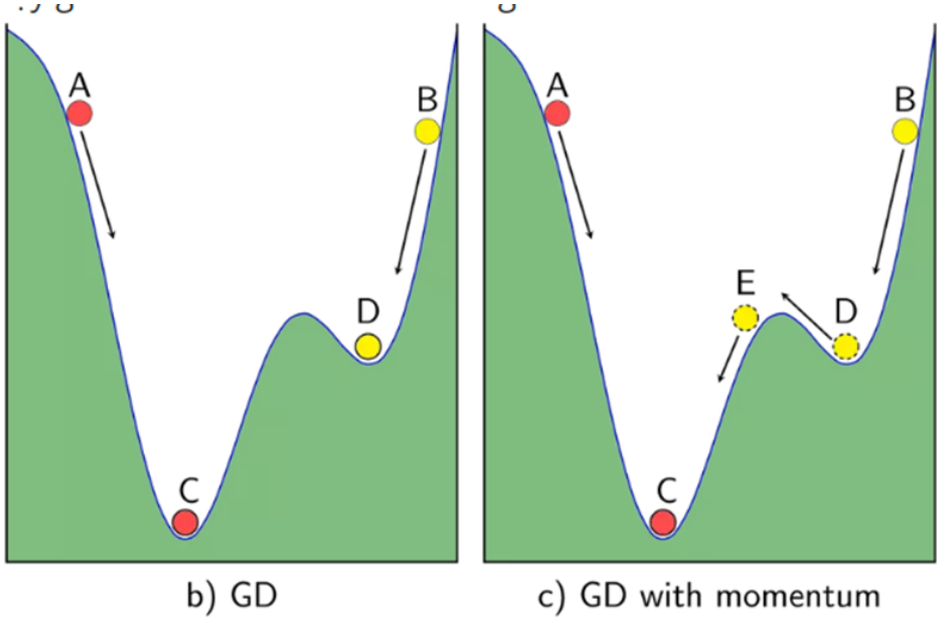


Hình 6: SGD có đường đi khá zigzag bởi vì 1 điểm dữ liệu không thể đại diện cho toàn bộ dữ liệu

- Ưu điểm: Thuật toán giải quyết được đối với cơ sở dữ liệu lớn mà GD không làm được.
- Nhược điểm: Thuật toán vẫn chưa giải quyết được 2 nhược điểm lớn của GD nên ta phải kết hợp SGD với một số thuật toán khác (Momentum, AdaGrad,...)

3. **Momentum**

- Để khắc phục thuật toán của GD, người ta sử dụng Momentum. Để giải thích được Gradient with Momentum thì trước tiên ta nên nhìn dưới góc độ vật lí: Như hình phía dưới, nếu ta thả 2 viên bi tại 2 điểm khác nhau A và B thì viên bị A sẽ trượt xuống điểm C còn viên bi B sẽ trượt xuống điểm D, nhưng ta lại không mong muốn viên bi B sẽ dừng ở điểm D (local minimum) mà sẽ tiếp tục lăn tới điểm C (global minimum). Để thực hiện được điều đó ta phải cấp cho viên bi B 1 vận tốc ban đầu đủ lớn để nó có thể vượt qua điểm E tới điểm C. Dựa vào ý tưởng này người ta xây dựng nên thuật toán Momentum (tức là theo đà tiến tới).



Hình 7: Momentum

$$x_{new} = x_{old} - (\gamma \times v + learing\_rate \times gradient)$$

Trong đó:

  - x_{new}: tọa độ mới
  - x_{old}: tọa độ cũ
  - γ: parameter, thường bằng 0.9
  - $learning\_rate$: tốc độ học
  - gradient : đạo hàm của hàm f

- Ưu điểm: thuật toán tối ưu giải quyết được vấn đề tiến tới global minimum của GD
- Nhược điểm: khi tới gần đích mất khá nhiều thời gian dao động qua lại trước khi dừng hẳn.

4. **Adagrad**

- Không giống như các thuật toán trước đó với learning rate là hằng số trong quá trình training. Thuật toán Adagrad lại coi learning rate là tham số biến thiên sau mỗi thời điểm t.

$$\theta_{t+1} = \theta_t - \frac{n}{\sqrt{G_t + \epsilon}} \times g_t$$

Trong đó:

- n: hằng số
- g_t: gradient tại thời điểm t
- ϵ: hệ số tránh lỗi (chia cho mẫu bằng 0)
- G: ma trận chéo mà mỗi phần tử trên đường chéo (i,i) là bình phương của đạo hàm vector tham số tại thời điểm t

- Vấn đề ở đây là giá trị G tăng đơn điệu qua các lần lặp, nên tham số learning rate sẽ giảm dần đến mức không còn cập nhật được nữa.
- Ưu điểm: Tránh được việc điều chỉnh learning rate thủ công, chỉ cần để learning rate default là 0.01 thì thuật toán sẽ tự động điều chỉnh.
- Nhược điểm: Vì tổng bình biến thiên sẽ lớn dần theo thời gian cho đến khi tốc độ học cực kì nhỏ, làm việc training trở nên đóng băng.

5. **RMSprop**

- RMSprop là thuật toán giải quyết vấn đề tỷ lệ học giảm dần của Adagard bằng cách chia tỷ lệ học cho trung bình của bình phương gradient

$$E[g^2]_t = 0,9E[g^2]_{t-1} + 0,1g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{n}{\sqrt{E[g^2]_t + \epsilon}} \times g_t$$

- Ưu điểm: giải quyến được vấn đề tốc độ học giảm dần của Adagrad
- Nhược điểm: có thể cho kết quả nghiệm chỉ là local minimum chứ không đạt được global minimum.

6. **Adam**

- Adam là thuật toán chủ đạo đang được sử dụng ngày này. Để giải quyết nhược điểm của thuật toán RMSprop, ta có sự kết hợp hoàn hảo với thuật toán Momentum khi có thể đưa kết quả nghiệm đạt được global minimum. Ngược lại, vì thuật toán Momentum có thể đưa ra kết quả nghiệm đạt được global minimum nhưng mất khá nhiều thời gian dao động qua lại trước khi dừng hẳn, ta có thể kết hợp với thuật toán RMSprop để giải quyết được vấn đề tốc độ học giảm dần. Có thể ví Adam như một quả cầu rất nặng và có ma sát, vì có khối lượng nặng nên quả cầu Adam dễ dàng vượt qua local minimum tới global minimum, khi tới global minimum vì có lực ma sát nên nó sẽ dễ dàng dừng lại hơn.

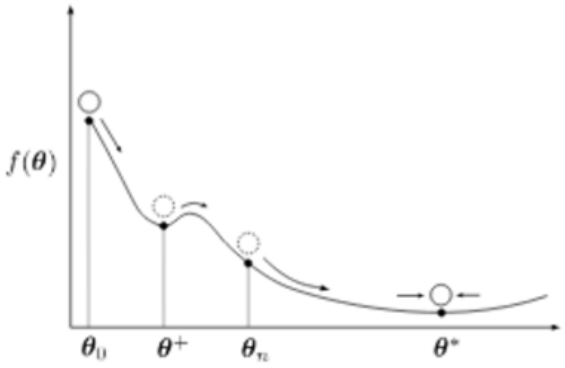


Figure 2: Heavy Ball with Friction, where the ball with mass overshoots the local minimum θ^+ and settles at the flat minimum θ^*.

Hình 8: Quả cầu Adam

- Ưu điểm: tính toán nhanh, tiết kiệm bộ nhớ, rất phù hợp đối với các tập dữ liệu lớn, siêu tham số trực quan và thường yêu cầu ít hoặc không cần điều chỉnh

5.4 Đánh giá mô hình (Test model)

- Đo lường hiệu suất để đánh giá mô hình là một phần trong quy trình xây dựng máy học. Tất cả các mô hình máy học đều cần **metric** để đánh giá hiệu suất.
- Với **supervised learning**, có 2 bài toán lớn là **Regression** và **Classification**. Sau đây là các **metric** phổ biến đối với 2 bài toán này: [1]

5.4.1 Regression metrics

Mô hình Regression có đầu ra liên tục (continuous output). Do đó, chúng ta cần một metric dựa trên việc tính toán khoảng cách giữa **giá trị được dự đoán** (predicted) và **giá trị thực tế** (ground truth). Các regression metric chủ yếu: [4]

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)

1. **Mean Absolute Error (MAE)**

   - Giá trị chênh lệch trung bình giữa giá trị thực tế và giá trị được dự đoán bởi mô hình Regression.

$$MAE = \frac{1}{N}\sum_{j=1}^{N}|y_i - \hat{y}_i|$$

   Trong đó:

[1] Các metric phổ biến dùng trong test model. Cách tính, ý nghĩa từng metric

  - y_i: giá trị thực tế
  - $\hat{y}_i$: giá trị được dự đoán bởi mô hình Regression
  - N: số lượng dữ liệu
- Ý nghĩa :
  - MAE có miền giá trị từ $[0, +\infty]$.
  - Trên cùng tập dữ liệu, MAE càng nhỏ thì có độ chính xác càng cao.
  - MAE không nhạy cảm với giá trị ngoại lệ (outliers) do việc sử dụng giá trị tuyệt đối.
  - MAE không phản ánh mức độ sai số cụ thể của mô hình. Nó không phân biệt được các lỗi dương và âm, chỉ cho ta biết lỗi trung bình.

2. **Mean Squared Error (MSE)**

- Giá trị trung bình của bình phương độ chênh lệch giữa giá trị mục tiêu và giá trị được dự đoán bởi mô hình Regression.

$$MSE = \frac{1}{N}\sum_{j=1}^{N}(y_i - \hat{y}_i)^2$$

  Trong đó:
  - y_i: giá trị thực tế
  - $\hat{y}_i$: giá trị được dự đoán bởi mô hình Regression
  - N: số lượng dữ liệu
- Ý nghĩa:
  - MSE có miền giá trị từ $[0, +\infty]$.
  - Trên cùng tập dữ liệu, MSE càng nhỏ thì có độ chính xác càng cao.
  - Vì lấy bình phương sai số nên đơn vị của MSE khác với đơn vị của kết quả dự đoán.
  - MSE nhạy cảm với các giá trị ngoại lệ (outliers) do tính bình phương. Các giá trị lớn hơn sẽ có ảnh hưởng lớn hơn đến MSE

3. **Root Mean Squared Error (RMSE)**

- Căn bậc hai giá trị trung bình của bình phương độ chênh lệch giữa giá trị mục tiêu và giá trị được dự đoán bởi mô hình Regression.

$$RMSE = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(y_i - \hat{y}_i)^2}$$

  Trong đó:
  - y_i: giá trị thực tế
  - $\hat{y}_i$: giá trị được dự đoán bởi mô hình Regression
  - N: số lượng dữ liệu
- Ý nghĩa:
  - RMSE có miền giá trị từ $[0, +\infty]$.
  - Trên cùng tập dữ liệu, RMSE càng nhỏ thì có độ chính xác càng cao.

– Việc lấy căn bậc 2 giúp RMSE có cùng đơn vị với kết quả dự đoán, đồng thời làm giá trị RMSE không quá lớn khi số lượng điểm dữ liệu lớn.
– RMSE cũng như MSE, nhạy cảm với các giá trị ngoại lệ (outliers) do tính bình phương.

5.4.2 Classification metrics

- Các mô hình classification có **đầu ra rời rạc** (discrete output), vì vậy chúng ta cần metric so sánh các lớp rời rạc dưới một dạng nào đó.
- Các classification metric đều đánh giá hiệu suất của mô hình và cho biết mức độ phân loại tốt hay xấu, nhưng mỗi metric lại đánh giá mô hình theo một cách khác nhau.
- Các classification metric chủ yếu:

- Accuracy
- Precision and Recall
- F1-score
- AU-ROC

Confusion matrix [5] [6] là một kỹ thuật đánh giá hiệu năng của mô hình cho các bài toán phân lớp. Confusion matrix là một ma trận thể hiện số lượng điểm dữ liệu thuộc vào một class và được dự đoán thuộc vào class.
Confusion matrix cung cấp thêm thông tin về tỉ lệ phân lớp đúng giữa các lớp, hay giúp phát hiện các lớp có tỉ lệ phân lớp nhầm cao nhờ vào các khái niệm True (False) Positive (Negative)

		Actual Class y		
		Positive	Negative	
$h_\theta(x)$ **Test outcome**	Test outcome positive	**True positive** (TP)	**False positive** (FP, Type I error)	Precision = $\frac{\#\text{TP}}{\#\text{TP} + \#\text{FP}}$
	Test outcome negative	**False negative** (FN, Type II error)	**True negative** (TN)	Negative predictive value = $\frac{\#\text{TN}}{\#\text{FN} + \#\text{TN}}$
		Sensitivity = $\frac{\#\text{TP}}{\#\text{TP} + \#\text{FN}}$	Specificity = $\frac{\#\text{TN}}{\#\text{FP} + \#\text{TN}}$	Accuracy = $\frac{\#\text{TP} + \#\text{TN}}{\#\text{TOTAL}}$

Hình 9: Confusion matrix

- **True positive (TP):** Đối tượng ở lớp Positive, mô hình phân đối tượng vào lớp Positive (dự đoán đúng).

- **True negative (TN):** Đối tượng ở lớp Negative, mô hình phân đối tượng vào lớp Negative (dự đoán đúng).
- **False positive (FP):** Đối tượng ở lớp Negative, mô hình phân đối tượng vào lớp Positive (dự đoán sai).
- **False negative (FN):** Đối tượng ở lớp Positive, mô hình phân đối tượng vào lớp Negative (dự đoán sai).

1. **Accuracy** [7]
   - Accuracy (độ chính xác) đánh giá mô hình thường xuyên dự đoán đúng đến mức nào.
   - Accuracy là tỉ lệ giữa số điểm dữ liệu được dự đoán đúng và tổng số điểm dữ liệu.
   - Accuracy lộ rõ hạn chế khi được sử dụng trên bộ dữ liệu không cân bằng (imbalanced dataset). Mô hình có thể đạt accuracy cao khi phân loại dữ liệu vào các lớp đa số.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

2. **Precision** [8]
   - **Precision** (Positive Predictive Value): (True Positives) / (All Positive Predictions)

$$P = Precision = \frac{TP}{TP+FP}$$

   - Ý nghĩa: "Khi mô hình dự đoán một mẫu là Positive, thì có bao nhiêu tỉ lệ mẫu đó thực sự là Positive?" Nếu Precision cao, tức là tỷ lệ dự đoán Positive đúng của mô hình là cao, và mô hình có khả năng phân loại Positive tốt.

3. **Recall** [8]
   - **Recall** (True Positive Rate): (True Positives) / (All Actual Positives)

$$R = Recall = \frac{TP}{TP+FN}$$

   - Ý nghĩa: "Khi có một mẫu Positive, thì có bao nhiêu tỉ lệ mẫu đó được mô hình dự đoán đúng?" Nếu Recall cao, tức là mô hình có khả năng phát hiện các mẫu Positive tốt.

4. **F Measure** [9]
   - **F-score**: là phép đo đánh giá hiệu suất của mô hình bằng cách kết hợp giữa precision P và recall R

$$F_\beta = \frac{(\beta^2+1)\times P\times R}{\beta^2\times P+R}$$

     - Khi β tiến đến 0, độ ảnh hưởng của P sẽ càng lớn.
     - Khi β tiến đến ∞, độ ảnh hưởng của R sẽ càng lớn.
     - Ý nghĩa: Sử dụng cả Precision và Recall để đánh giá hiệu suất mô hình.

- **F_1-score:** Khi $\beta = 1$, ta được F_1-score

$$F_1 = 2\frac{P \times R}{P + R}$$

  - F_1-score: là trung bình điều hòa của precision P và recall R
  - Ý nghĩa: Precision và Recall có mức độ quan trọng tương đương và cần metric đánh giá hiệu suất tổng thể của mô hình phân loại.

5. **ROC Curve, AUC** [10]

- **ROC curve:** Đường cong ROC (Receiver Operating Characteristic curve) là biểu đồ hiển thị hiệu suất của mô hình classification ở tất cả các ngưỡng phân loại (classification thresholds). Đường cong này vẽ hai tham số:

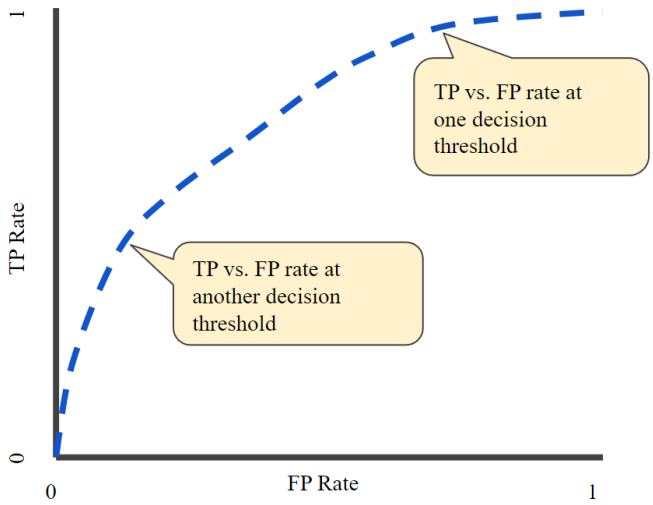
  (a) True Positive Rate (TPR) - hay còn được gọi là Recall:

$$TPR = \frac{TP}{TP + FN}$$

  (b) False Positive Rate (FPR):

$$FPR = \frac{FP}{FP + TN}$$

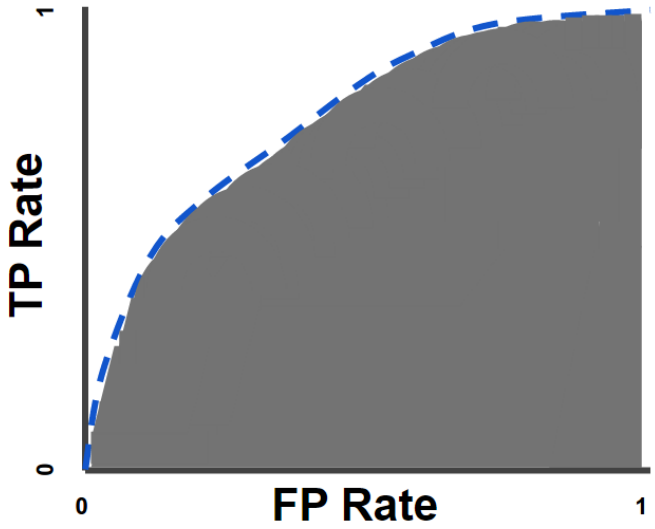
  Đường cong ROC vẽ đồ thị TPR so với FPR ở các ngưỡng phân loại khác nhau. Việc giảm ngưỡng phân loại sẽ phân loại nhiều mục hơn thành Positive, do đó làm tăng cả kết quả False Positive và kết quả True Positives thực sự. Hình dưới đây cho thấy một đường cong ROC điển hình:



Hình 10: ROC curve

- **AUC:** Diện tích dưới đường cong ROC (Area Under the ROC Curve). AUC đo toàn bộ diện tích hai chiều bên dưới toàn bộ đường cong ROC từ (0,0) đến (1,1) - hay là phép tính tích phân của đường cong ROC từ 0 đến 1.

$$\int_0^1 ROC(x)dx$$



Hình 11: AUC

- **Ý nghĩa:** AUC có giá trị từ 0 đến 1. Với $AUC = 0$, mô hình dự đoán sai 100%; với $AUC = 1$, mô hình dự đoán đúng 100%.
  - AUC không phụ thuộc vào tỷ lệ **(scale-invariant)**. AUC đo lường mức độ xếp hạng chính xác của các dự đoán, không phụ thuộc vào giá trị tuyệt đối của chúng.
  - AUC không phụ thuộc vào ngưỡng phân loại **(classification-threshold-invariant)**. AUC đo lường chất lượng của các dự đoán mô hình bất kể ngưỡng phân loại được chọn

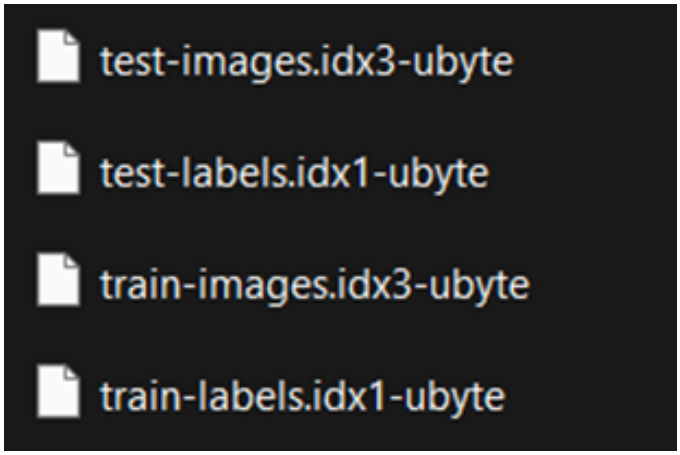
6 Tuần 3

6.1 Bài toán cần giải quyết

- **Bài toán:** Nhận diện các chữ số viết tay.
- **Sơ lược về mô hình:** [11] 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.

6.2 Dữ liệu và phân chia dữ liệu

- Dữ liệu được sử dụng trong mô hình này là tập dữ liệu **MNIST** [12] được lấy từ website kaggle.com bao gồm 4 file:



Hình 12: Các tập dữ liệu

- Trong đó:
  - 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.

- 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.

- 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:

Hình 13: Ví dụ một hình ảnh của tập dữ liệu thể hiện chữ số 3

- **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.

6.3 Thiết kế, huấn luyện mạng neutral

6.3.1 Kiến trúc neural network

1. **Xây dựng mô hình 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.
   - Trong quá trình huấn luyện: kết quả từ lớp Output sẽ thông qua hàm mất mát (trong mô hình này là Cross-Entropy) kết hợp với hàm softmax để tính toán sai số giữa đầu ra dự đoán của mô hình và đầu ra thực tế.
   - Sau khi đi qua tất cả các lớp và tính toán được sai số của mô hình. Ta sẽ cập nhật các tham số (Weight và Bias) bằng thuật toán lan truyền ngược (back propagation) và Stochastic Gradient Descen (SGD) optimizer.

2. **Khởi tạo các tham số của neural network:**

   - `Imput_dim = 784`: Kích thước của lớp input
   - `Hidden_dim = [10]`: Một danh sách gồm các kích thước của từng lớp ẩn
   - `Output_dim = 10`: Kích thước của lớp output
   - `Regularization`: Hằng số được thêm vào quá trình tính gradient để giảm hiện tượng overfitting
   - `Learning Rate`: Tốc độ học của mô hình (lr)

3. **Xây dựng các class cơ bản:**

   - `Linear`: là một hàm tuyến tính được biểu diễn qua phương trình

     $$y = w \times x + b$$

     Trong đó:
     - y: kết quả đầu ra của hàm linear
     - w: trọng số (Weight)
     - x: giá trị đầu vào của hàm
     - b: hệ số bias
   - `ReLu`: Nếu không có các hàm kích hoạt phi tuyến, thì mạng neural của chúng ta dù có nhiều lớp vẫn sẽ có hiệu quả như một lớp tuyến tính. Vì vậy cần có sự xuất hiện của một hàm phi tuyến trong mô hình. Trong mô hình này, ta sẽ sử dụng hàm ReLu. Hàm ReLu được định nghĩa như sau:

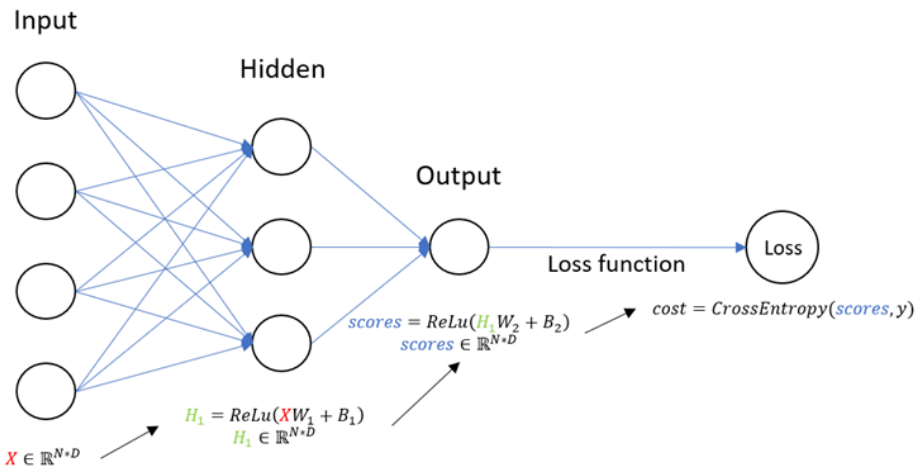
     $$R(x) = max(0, x)$$

     Trong đó
     - x: giá trị đầu vào của hàm.

6.3.2 Quá trình huấn luyện mô hình

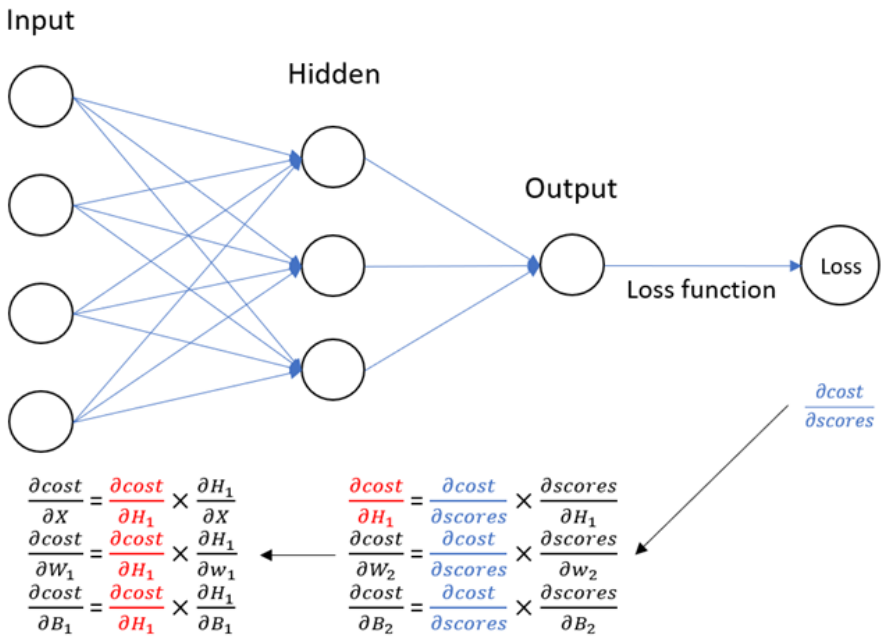
- Mô hình được huấn luyện thông qua 2 quá trình: **lan truyền tiến** (Forward propagation) và **lan truyền ngược** (Back propagation)

1. **Lan truyền tiến:**
   - Trong quá trình lan truyền tiến, giá trị đầu vào từ lớp `Input` là ma trận `X` sẽ được sử dụng để tính toán giá trị của lớp `Hidden` là H_1, sau đó giá trị H_1 sẽ tiếp tục được sử dụng làm giá trị đầu vào để tính toán giá trị của lớp `Output` là `scores`. Cuối cùng, ta sẽ sử dụng hàm một hàm mất mát là hàm `Cross-Entropy` để tính toán sai số `cost` giữa đầu ra dự đoán của mô hình là `scores` và đầu ra thực tế là `y`.



Hình 14: Mô hình mô phỏng lan truyền tiến

2. **Lan truyền ngược:**
- Sau khi tính toán được sai số giữa đầu ra dự đoán của mô hình và đầu ra thực tế, ta sẽ sử dụng sai số này để **điều chỉnh các tham số của từng lớp** sao cho giá trị đầu ra dự đoán gần với đầu ra thực tế nhất bằng cách lan truyền ngược.
- Các bước lan truyền ngược: Đầu tiên, ta **tính toán giá trị đạo hàm** của hàm mất mát đối với giá trị đầu ra dự đoán của mô hình. Sau đó ta sử dụng giá trị này kết hợp với quy tắc dây chuyền (chain rule) để tính toán đạo hàm của lớp `Output` rồi tiếp tục lấy đạo hàm của lớp `Output` để tính toán đạo hàm của lớp trước nữa. Nói cách khác, đạo hàm của lớp sau sẽ được lan truyền lại để tính đạo hàm của lớp trước thông qua quy tắc dây chuyền (chain rule).



Hình 15: Mô hình mô phỏng lan truyền ngược

- Sau khi tính toán được đạo hàm của từng tham số (Weight và Bias) trong từng lớp, ta sẽ sử dụng phương pháp gradient descent để **cập nhật lại các tham số** này nhằm giảm giá trị của cost.

$$W_1 := W_1 - lr * \frac{\partial cost}{\partial W_1} \qquad W_2 := W_2 - lr * \frac{\partial cost}{\partial W_2}$$
$$B_1 := B_1 - lr * \frac{\partial cost}{\partial B_1} \qquad B_2 := B_2 - lr * \frac{\partial cost}{\partial B_2}$$

Hình 16: Công thức cập nhật tham số (Weight và Bias) của từng lớp

6.3.3 Loss function: Cross-Entropy (CE) và hàm Softmax

- **Cross-Entropy Loss function:** Là một loss function thường được sử dụng trong các bài toán phân loại bao gồm cả các mô hình neural network hoặc bài toán dự đoán xác suất. Hàm này đo lường mức độ sai lệch giữa hai phân phối xác suất: phân phối dự đoán của mô hình và phân phối thực tế. Công thức chung của Cross-Entropy Loss function cho hai phân phối xác suất P và Q là:

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

Trong đó:

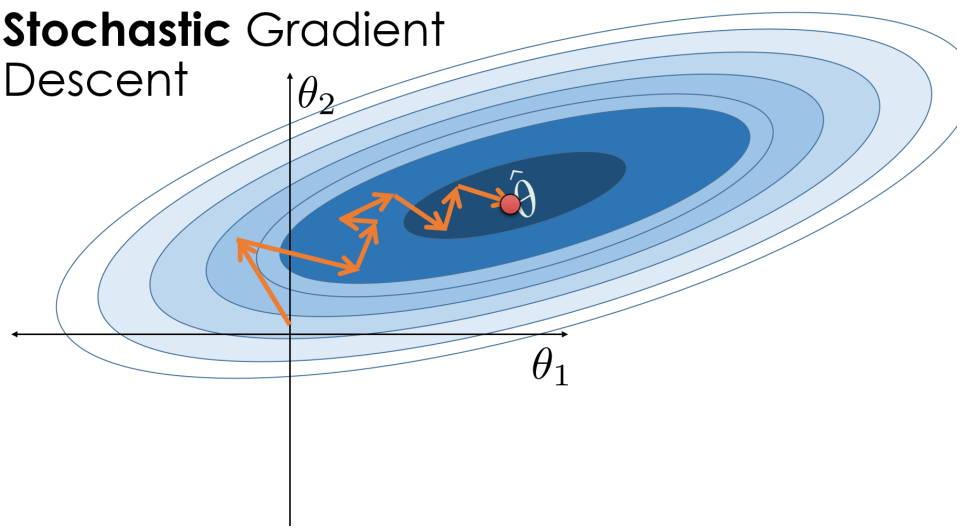
- C: số lượng các class cần phân lớp.
- Q_i: xác suất thực tế của lớp thứ i
- P_i: xác suất dự đoán của lớp thứ i bởi mô hình

- 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:

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

6.3.4 Optimizer: Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent là một biến thể của Gradient Descent. Thay vì sau mỗi epoch chúng ta sẽ cập nhật trọng số một lần, thì trong mỗi epoch có N điểm dữ liệu, chúng ta sẽ cập nhật trọng số N lần. Nhờ sử dụng SGD nên tốc độ hội tụ của mô hình khá nhanh.



Hình 17: Stochastic Gradient Descent

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

- Learning rate, số epoch, batch-size và regularization là các thông số quan trọng trong việc quyết định tốc độ học và độ chính xác của mô hình.

1. **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.
   - Các giá trị liên quan đến **learning rate** được khởi tạo như sau:
     - $learning\_rate = 0.001$
     - $decay\_after = 50$
     - $learning\_rate\_decay = 0.99$
2. **Số epoch:** Số lần truyền tập train để huấn luyện cho neural network
     - $epoch = 200$
3. **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.
     - $batch - size = 200$
4. **Regularization:** Hằng số được thêm vào quá trình tính gradient để giảm hiện tượng overfitting
     - $Regularization = 5 \times 10^{-6}$

6.4 Quá trình hội tụ, kết quả kiểm thử mô hình

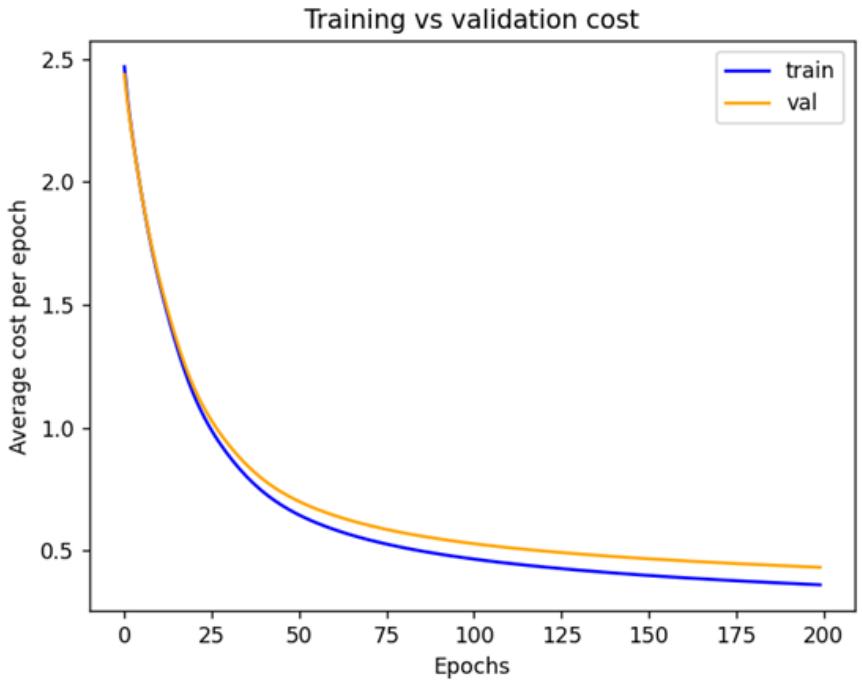
6.4.1 Quá trình hội tụ

- Để tiện quan sát các giá trị cost sau mỗi epoch của tập train set và validation set, cứ sau mỗi 10 epoch thì ta hiển thị giá trị trung bình của hàm loss (average cost), độ chính xác của mô hình với tập train (train accuracy) và độ tương thích của mô hình đối với các tập dữ liệu mới (val accuracy).
- Giá trị cứ sau mỗi 10 epoch khi train mô hình ghi lại được:

- Epoch 10 – Train cost: 1.65, Validation Cost: 1.74
- Epoch 20 – Train cost:1.16, Validation Cost: 1.33
- Epoch 30 – Train cost: 0.9, Validation Cost: 1.03
- Epoch 40 – Train cost: 0.75, Validation Cost: 0.89

- Qua đó ta có thể thấy được:

- Train cost và Validation cost giảm dần theo sự tăng dần của epoch.
- Đạt được mục tiêu của việc huấn luyện mô hình: giảm thiểu các giá trị của hàm loss, tăng độ chính xác của mô hình.
- Huấn luyện dựa trên quá trình lan truyền tiến và ngược trong neural network giúp mô hình học được các mối quan hệ giữa input và output để cho ra dự đoán chính xác với những bộ dữ liệu mới ngoài tập train set.



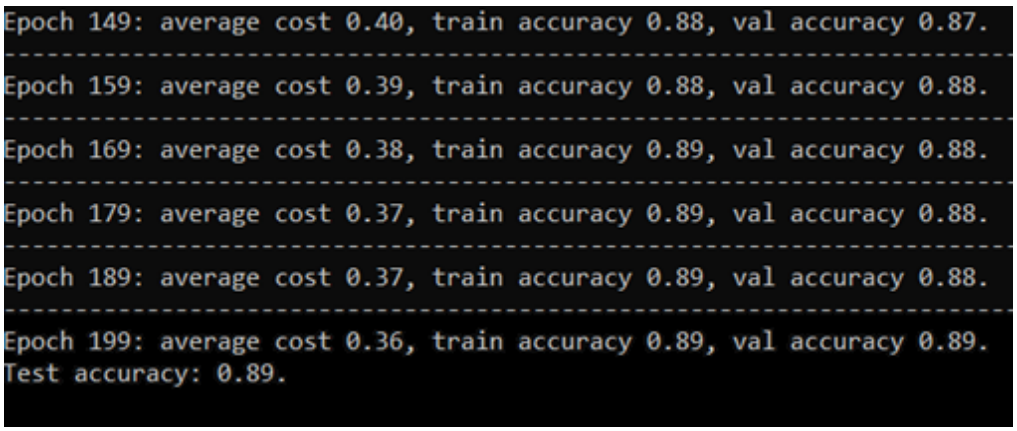
Hình 18: So sánh Training và Validation Cost

- Dựa vào biểu đồ minh họa sự thay đổi của 2 giá trị Train cost và Validation cost trên cả tập train set và validation set sau mỗi epoch, ta có nhận xét:

- Nếu cả hai cost giảm dần, cuối cùng hội tụ ổn định thì quá trình train của mô hình học tốt.
- Nếu cost trên tập train set giảm dần trong khi cost trên tập validation tăng lên, có thể đây là dấu hiệu của overfitting.

6.4.2 Kết quả kiểm thử

- 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%**

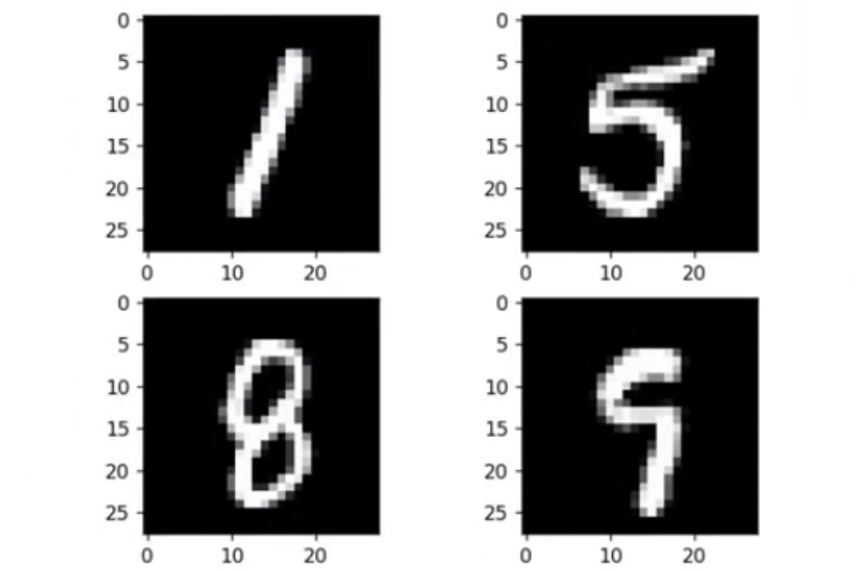


Hình 19: Accuracy trên test set

6.5 Demo

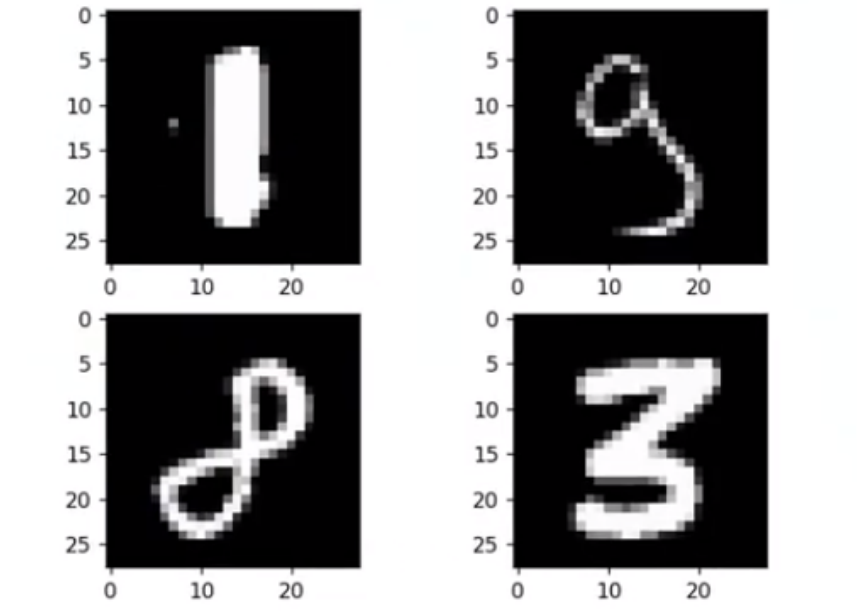
- Link demo [13]: https://www.youtube.com/watch?v=UQExhzgt-6E

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



Hình 20: Các mẫu mô hình dự đoán đúng

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



Hình 21: Các mẫu mô hình dự đoán sai

7 Tổng kết - Nhận xét

Tổng kết

- Sau đồ án này, nhóm em đã có thể hiểu rõ cách xây dựng một chương trình AI

Nhận xét

- Trong thời gian thực hiện đồ án, các thành viên có tinh thần trách nhiệm và năng lực tốt, hoàn thành nhiệm vụ được giao đúng với thời hạn.
- Có sự nỗ lực lớn, tinh thần đoàn kết giúp đỡ nhau giữa các thành viên.

Tài liệu

[1] Tổng quan về tiền xử lý dữ liệu.
https://gizsolution.wordpress.com/2017/01/06/tong-quan-ve-tien-xu-ly-du-lieu/.

[2] Hàm mất mát (loss function).
https://khanh-personal.gitbook.io/ml-book-vn/chapter1/ham-mat-mat.

[3] Code Cross Entropy Loss: Intro, Applications.
https://www.v7labs.com/blog/cross-entropy-loss-guide.

[4] Performance Metrics in Machine Learning [Complete Guide].
https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide.

[5] Evaluation of Binary Classifiers.
http://mlwiki.org/index.php/Evaluation_of_Binary_Classifiers.

[6] [Machine Learning] Làm thế nào để đánh giá một mô hình Máy học?
http://tutorials.aiclub.cs.uit.edu.vn/index.php/2021/05/18/evaluation/.

[7] Classification: Accuracy.
https://developers.google.com/machine-learning/crash-course/classification/accuracy.

[8] Classification: Precision and Recall.
https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall.

[9] Precision and Recall: F Measure.
http://mlwiki.org/index.php/Precision_and_Recall#F_Measure.

[10] Classification: ROC Curve and AUC.
https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc.

[11] Creating a neural network from scratch.
https://github.com/SonPhatTran/Neural-Network-from-scratch.

[12] The MNIST database of handwritten digits.
https://www.kaggle.com/datasets/hojjatk/mnist-dataset.

[13] Nhóm 4 23TNT1 HCMUS | ĐỒ ÁN 2: XÂY DỰNG CHƯƠNG TRÌNH AI | Video demo huấn luyện mô hình.
https://www.youtube.com/watch?v=UQExhzgt-6E.