Tuần 1 - Tổng hợp kiến thức Buổi học số 2

Time-Series Team

Ngày 7 tháng 6 năm 2025

Buổi học số 2 (Thứ 4, 04/06/2025) bao gồm hai nội dung chính:

- ***Phần I: Giới thiệu ngôn ngữ Python: Cấu trúc dữ liệu Python, Function, Conditions***
- ***Phần II: Khái niệm toán học và ứng dụng thực tế***

Phần I: Giới thiệu ngôn ngữ Python

1 Bản chất ngôn ngữ Python

1.1 Lý do hình thành và đặc điểm của ngôn ngữ Python

Python là một ngôn ngữ lập trình bậc cao (high-level), thông dịch (interpreted), đa mục đích (general-purpose) với cú pháp rõ ràng và dễ đọc. Mục tiêu khi tạo Python là giúp việc lập trình trở nên dễ tiếp cận, nhanh chóng, và ít lỗi hơn, nhất là với người mới học hoặc nhà nghiên cứu.

Đặc điểm	**Giải thích**
Dễ học & dễ đọc	Python gần giống tiếng Anh, dễ tiếp cận cho người mới
Thông dịch	Không cần biên dịch trước, chương trình được chạy trực tiếp
Đa mô hình lập trình	Hỗ trợ hướng đối tượng, lập trình hàm, thủ tục
Kiểu động	Không cần khai báo kiểu dữ liệu khi tạo biến
Thư viện phong phú	Có sẵn rất nhiều thư viện cho khoa học dữ liệu, AI, web, automation…
Đa nền tảng	Chạy tốt trên Windows, macOS, Linux, và cả điện toán đám mây

1.2 Lịch sử phát triển của Python

Hành Trình Lịch Sử của Python

1980s	Guido van Rossum làm việc tại Centrum Wiskunde & Informatica (CWI), Hà Lan, bắt đầu phát triển Python.
1991	Python 0.9.0 ra mắt – hỗ trợ exception, function, module.
2000	Python 2.0 ra đời – hỗ trợ garbage collection, Unicode.
2008	Python 3.0 ra đời – phá vỡ tương thích với Python 2 nhưng cải tiến mạnh về cấu trúc và unicode.
2020	Python 2 chính thức dừng hỗ trợ.
Hiện tại	Python là một trong các ngôn ngữ phổ biến nhất thế giới, dùng trong AI, khoa học dữ liệu, web, tự động hóa...

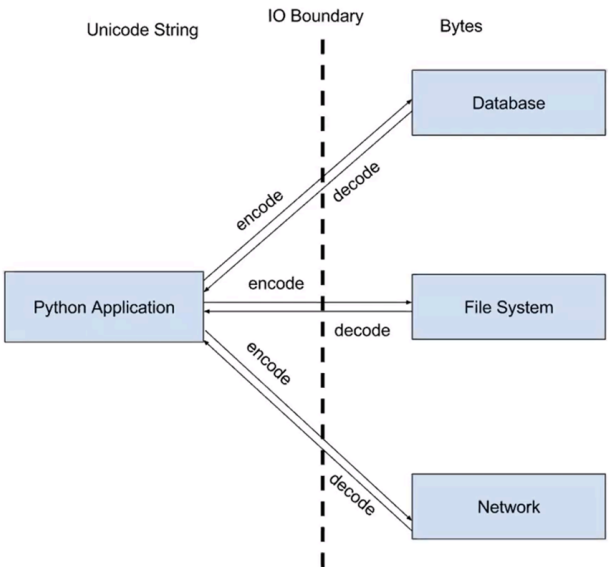
*T*ên "Python" không xuất phát từ con rắn mà từ chương trình hài **"Monty Python's Flying Circus"**.

2 Biểu diễn dữ liệu trong Python

2.1 Kiểu dữ liệu cơ bản

Kiểu	`Ví dụ`	**Mô tả**
int	`x = 10`	Số nguyên
float	`y = 3.14`	Số thực (dấu chấm động)
bool	`is_ok = True`	Boolean (True/False)
str	`name = "Khoa"`	Chuỗi văn bản

Với dữ liệu String: Các kí tự String hay số đều được encode bằng **UNICODE** (ví dụ: một giá trị byte nào đó), nên khi dữ liệu truyền qua internet nó sẽ được truyền dưới dạng những con số. Khi hiển thị dữ liệu cho người dùng, giá trị byte sẽ được dịch trở lại dạng text.



Hình 1: Hình minh họa dữ liệu String/Number được encode bằng UNICODE

2.2 Cấu trúc dữ liệu cơ bản

- **List** – Danh sách có thể thay đổi, có thứ tự, cho phép phần tử trùng lặp. Bao gồm *value* (giá trị trong list) và *index* (vị trí của giá trị trong list, bắt đầu từ 0).

```
fruits = ["apple", "banana", "cherry"]
fruits.append("orange")

```

- **Tuple** – Danh sách không thay đổi (bất biến), có thứ tự, cho phép phần tử trùng lặp.

```
coords = (10, 20)

```

- **Set** – Tập hợp không có thứ tự, không cho phép phần tử trùng lặp.

```
unique_values = {1, 2, 3}

```

- **Dict** – Từ điển (key-value pairs). Giống như từ điển gồm từ và ý nghĩa, dictionary trong Python gồm *key* và *value*. `Key` là duy nhất, dùng để tra cứu nhanh.
  - *Value* có thể là một giá trị hoặc một danh sách các giá trị.
  - *Key* đại diện cho toàn bộ giá trị chứa trong nó.
  - Khái niệm `key:value` có thể hiểu như trong một cuốn sách, *key* là tiêu đề trang và *value* là nội dung trang.

```
student = {"name": "Linh", "age": 20}

```

2.3 Overflow & Underflow

- **Overflow (tràn số)**: xảy ra khi một số lớn hơn giới hạn của kiểu dữ liệu được khai báo. Tuy nhiên, Python có thể tự động điều chỉnh các số nguyên rất lớn mà không bị tràn, ví dụ làm tròn về `inf`.
- **Underflow (dưới số)**: xảy ra khi một số nhỏ hơn giới hạn của kiểu dữ liệu được khai báo, dẫn đến giá trị bị "làm tròn về không", gây mất chính xác và kết quả không mong muốn.

Hình 2: Hình minh họa cho Overflow và UnderFlow

2.4 Cấu trúc dữ liệu đặc biệt

Dạng biểu diễn	Ứng dụng	Ví dụ
Chuỗi JSON	Giao tiếp web/API	`{"name": "An", "score": 90}`
Pandas DataFrame	Phân tích dữ liệu	Bảng dữ liệu dạng Excel
Numpy Array	Xử lý số học ma trận	`np.array([[1, 2], [3, 4]])`
Tensor	Deep Learning	`torch.tensor([1.0, 2.0])`

3 Function trong Python

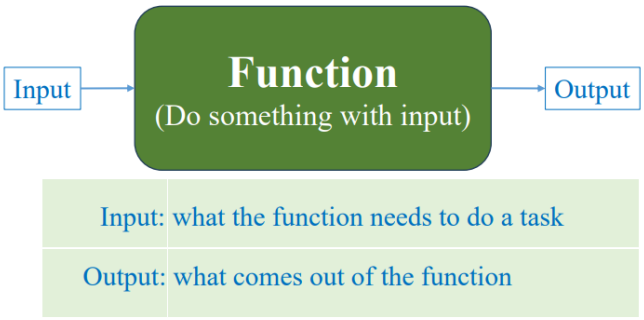
3.0.1 Đặc điểm của Function

Hàm là khối mã có thể tái sử dụng, giúp tổ chức chương trình logic rõ ràng, giảm thiểu lặp lại, và dễ dàng mở rộng hoặc bảo trì.

```
def ten_ham(tham_so_1, tham_so_2=mac_dinh):
    """ Function Description. """
    # Code here
    return ket_qua
```

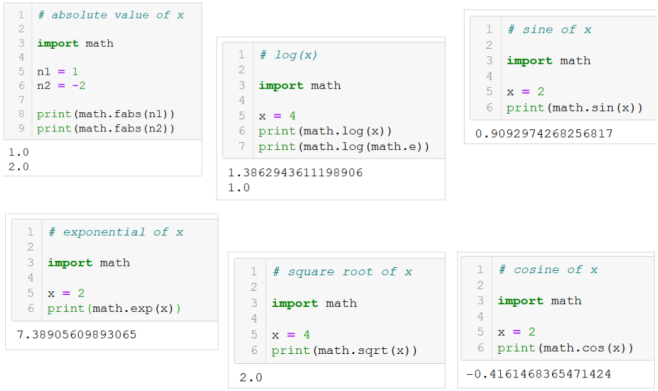
Code Listing 1: Cấu trúc hàm trong Python

Lưu ý đặt tên hàm: Nên đặt tên bắt đầu bằng động từ thể hiện hành động của hàm như `calculate_loss()`, `train_model()`, `predict_output()`.



Hình 3: Mô tả cách Function hoạt động

Ngoài các hàm mình tự khai báo(*User-defined function*) thì Python có bao gồm rất nhiều các hàm có sẵn (*Build-in function*), trong đó các hàm liên quan đến nhau thường được gọi thông qua thư viện. *Ví dụ: random , sqrt functions từ thư viện math.*



Hình 4: Một số Function trong Math Module

3.0.2 Built-in function

Built-in Function (hàm tích hợp sẵn) trong Python là những hàm đã được định nghĩa sẵn bởi ngôn ngữ, người dùng có thể sử dụng trực tiếp **mà không cần định nghĩa lại hoặc import từ thư viện bên ngoài**.

Đặc điểm

- Có thể gọi trực tiếp ở bất cứ đâu trong chương trình.
- Được tối ưu hoá sẵn giúp tăng hiệu suất khi thực thi.
- Bao phủ nhiều chức năng phổ biến như: xử lý dữ liệu, toán học, kiểm tra kiểu, v.v.

Một số hàm built-in phổ biến

Hàm	Mô tả
`print()`	In ra màn hình
`len()`	Trả về độ dài của chuỗi, list, tuple, v.v.
`type()`	Trả về kiểu dữ liệu của một đối tượng
`int()`, `float()`, `str()`	Ép kiểu dữ liệu
`sum()`	Tính tổng các phần tử trong iterable
`max()`, `min()`	Trả về giá trị lớn nhất, nhỏ nhất
`abs()`	Trả về giá trị tuyệt đối
`range()`	Tạo một dãy số
`sorted()`	Trả về một list đã sắp xếp
`input()`	Nhận dữ liệu đầu vào từ bàn phím

Ví dụ minh họa

```
numbers = [3, 7, 2, 9]
print("Sum:", sum(numbers))
print("Max:", max(numbers))
print("Data Type:", type(numbers))
print("Sort:", sorted(numbers))
```

Tham khảo đầy đủ

Danh sách đầy đủ các hàm built-in có thể xem tại tài liệu chính thức của Python:

- https://docs.python.org/3/library/functions.html

Ví dụ mở rộng

Bạn có thể phân loại các hàm built-in theo nhóm chức năng:

- Nhóm hàm toán học: `abs()`, `pow()`, `round()`
- Nhóm xử lý chuỗi và list: `len()`, `sorted()`, `reversed()`
- Nhóm kiểm tra kiểu: `type()`, `isinstance()`
- Nhóm chuyển đổi kiểu dữ liệu: `int()`, `float()`, `str()`, `bool()`

3.0.3 User-defined function

Trong Python, **user-defined function** là hàm do người dùng định nghĩa nhằm tái sử dụng các đoạn mã, giúp chương trình tổ chức tốt hơn, đơn giản hóa việc mở rộng và bảo trì.

Cấu trúc tổng quát của một hàm

```
def function_name(parameters):
    '''
    Do docstring here.
    '''
    # Code here
    return result
```

Giải thích cấu trúc:

- `def`: từ khóa để khai báo một hàm.
- **function_name**: tên hàm, nên dùng chữ thường, dấu gạch dưới và bắt đầu bằng động từ (ví dụ: `calculate_sum`).
- **parameters**: các tham số truyền vào hàm để xử lý.
- **Docstring** (dòng mô tả trong dấu nháy ba): mô tả chức năng của hàm, giúp người khác dễ hiểu.
- **Code xử lý**: nội dung của hàm, được viết cách lề (indent) bằng 4 dấu cách.
- **return result**: trả về kết quả của hàm.

Lưu ý khi xây dựng hàm

- **Đặt tên hàm**: Dùng chữ thường, dấu gạch dưới (_), bắt đầu bằng động từ để mô tả hành động.
- **Thụt lề (indentation)**: Dùng 4 dấu cách để thụt lề phần nội dung trong hàm.
- **Xác định tham số đầu vào**: Dữ liệu đầu vào giúp hàm thực hiện nhiệm vụ cụ thể.
- **Viết chuỗi mô tả (docstring)**: Mô tả mục đích và cách sử dụng hàm.
- **Giá trị trả về**: Sử dụng `return` để trả về kết quả xử lý.

Ví dụ về hàm Python định nghĩa bởi người dùng (user-defined function) để tính diện tích hình tròn đơn vị

Dưới đây là một ví dụ về hàm Python do người dùng định nghĩa để tính diện tích hình tròn đơn vị (unit circle), đồng thời lồng ghép kiến thức về giới hạn (limit) và tích phân (integration) – những khái niệm toán học quan trọng trong AI/Machine Learning khi xử lý liên tục hoặc tối ưu hóa.

Ý tưởng toán học

Diện tích hình tròn đơn vị (bán kính $r = 1$) là:

$$A = \pi r^2 = \pi$$

Diện tích cũng có thể được tính bằng tích phân:

$$A = \int_{-1}^{1} \sqrt{1 - x^2}\, dx$$

Công thức này xuất phát từ phương trình đường tròn:

$$x^2 + y^2 = 1 \implies y = \sqrt{1 - x^2}$$

Khi số đoạn chia (số lát cắt) $n \to \infty$, tổng diện tích các hình chữ nhật nhỏ tiến gần đến diện tích thực, thể hiện kiến thức *limit* (giới hạn).

Hàm Python tính diện tích hình tròn đơn vị bằng tích phân số

```
import math

def compute_unit_circle_area(n=100000):
    """
    Calculate the area of the unit circle using the numerical integration method.

    n: number of slices (number of small rectangles), the larger the number, the more
    accurate the result.
    """
    a, b = -1, 1
    dx = (b - a) / n
    total_area = 0.0

    for i in range(n):
        x = a + i * dx
        y = math.sqrt(1 - x**2)
        total_area += y * dx
    return 2 * total_area
# Call function
print(compute_unit_circle_area())  # Output: 3.14159
```

Giải thích kết quả và ứng dụng

Khi n đủ lớn (ví dụ $100,000$), kết quả xấp xỉ π. Phương pháp này cho ta thấy mối liên hệ giữa hình học và giải tích, ứng dụng trong AI khi cần tối ưu hàm mục tiêu (loss), hoặc xử lý liên tục dữ liệu.

4 Conditions trong Python và Ứng dụng

4.1 Câu lệnh điều kiện (Conditions) trong Python

- **Điều kiện** kiểm tra giá trị Boolean (`True` hoặc `False`).
- Câu lệnh: `if`, `elif`, `else`.

```
if condition:
    # implement when condition is True
elif another_condition:
    # implement when another_condition is True
else:
    # implement when all conditions are False
```

4.2 Ứng dụng điều kiện trong hàm ReLU (Rectified Linear Unit)

ReLU là hàm kích hoạt rất phổ biến trong mạng neural, định nghĩa:

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} x & \text{nếu } x > 0 \\ 0 & \text{ngược lại} \end{cases}$$

Cách dùng điều kiện trong Python để định nghĩa hàm ReLU:

```
def relu(x):
    if x > 0:
        return x
    else:
        return 0
```

Ý nghĩa: ReLU giúp mạng neural học được các đặc trưng phi tuyến tính, tránh vấn đề gradient biến mất.

4.3 Ứng dụng điều kiện trong chatbot

Chatbot thường cần xử lý nhiều tình huống hội thoại khác nhau dựa trên điều kiện nhận diện từ khóa, câu hỏi, hay trạng thái người dùng. Ví dụ điều kiện kiểm tra câu hỏi người dùng:

```
def chatbot_response(user_input):
    user_input = user_input.lower()
    if "hello" in user_input or "hi" in user_input:
        return "Hello! How can I help you today?"
    elif "price" in user_input:
        return "Our products range from $10 to $100."
    elif "bye" in user_input or "goodbye" in user_input:
        return "Goodbye! Have a nice day."
    else:
        return "Sorry, I didn't understand that. Could you please rephrase?"
```

Ý nghĩa:

- Điều kiện giúp chatbot đưa ra phản hồi phù hợp dựa trên từ khóa hoặc ý định người dùng.
- Tăng trải nghiệm tương tác tự nhiên và thông minh hơn.

Phần II: Khái niệm toán học và ứng dụng thực tế

1 Hàm Log

1.1 Tính chất quan trọng của hàm logarit

1.1.1 Hàm đơn điệu (Monotonic Function)

Hàm log là một **hàm đơn điệu tăng** trên miền xác định $(0, +\infty)$. Điều đó có nghĩa là:

$$x_1 < x_2 \Rightarrow \log(x_1) < \log(x_2)$$

Tính chất này rất quan trọng trong Machine Learning vì giúp **giữ nguyên thứ tự** (ranking) sau khi áp dụng hàm log.

1.1.2 Các tính chất khác

Chuyển đổi phép nhân thành phép cộng

$$\log(a \cdot b) = \log(a) + \log(b)$$

Ứng dụng: xử lý các giá trị rất nhỏ như xác suất bằng cách **chuyển phép nhân thành phép cộng**:

$$\log(p_1 \cdot p_2 \cdot \ldots \cdot p_n) = \log(p_1) + \log(p_2) + \ldots + \log(p_n)$$

Chuyển đổi phép chia thành phép trừ

$$\log\left(\frac{a}{b}\right) = \log(a) - \log(b)$$

Lũy thừa thành phép nhân

$$\log(a^n) = n \cdot \log(a)$$

Ứng dụng: thường dùng trong tối ưu hàm mất mát như cross-entropy.

Đạo hàm của hàm log

$$\frac{d}{dx}\log(x) = \frac{1}{x}$$

Miền xác định của log

Hàm log chỉ xác định khi đầu vào là số dương:

$$x > 0$$

1.2 Ví dụ ứng dụng của hàm Log

1.2.1 Thiết kế hàm mất mát (Loss Function)

Hàm mất mát (*loss function*) đo lường sự sai lệch giữa giá trị dự đoán và giá trị thực tế. Mục tiêu của quá trình huấn luyện mô hình là tìm tham số để **giảm thiểu hàm mất mát**. Một cách hiểu trực quan, ta có khái niệm:

$$\text{Surprise}(E) = \frac{1}{P(E)}$$

Điều này có nghĩa: sự kiện càng ít xảy ra (xác suất $P(E)$ nhỏ), thì mức độ "bất ngờ" của nó càng cao. Để tiện cho việc tính toán và xử lý, ta lấy logarit:

$$\log(\text{Surprise}(E)) = \log\left(\frac{1}{P(E)}\right) = -\log(P(E))$$

Từ đó, ta dẫn đến khái niệm nền tảng trong **Lý thuyết Thông tin (Information Theory)**:

$$\text{Information}(x) = -\log(P(x)) \tag{1}$$

Ý nghĩa:

- Nếu $P(x)$ nhỏ (sự kiện hiếm), thì $-\log(P(x))$ lớn: thông tin chứa trong sự kiện đó nhiều.
- Nếu $P(x)$ gần 1 (sự kiện chắc chắn), thì $-\log(P(x))$ gần 0: thông tin thu được gần như không có.

Ví dụ minh họa:

- Nếu $P(x) = 1$ (xảy ra chắc chắn), thì $\text{Information}(x) = -\log(1) = 0$
- Nếu $P(x) = 0.01$, thì $\text{Information}(x) = -\log(0.01) \approx 4.605$

1.2.2 Công thức Shannon cho Entropy

Claude Shannon — cha đẻ của lý thuyết thông tin — định nghĩa entropy là kỳ vọng của thông tin:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log P(x_i) \tag{2}$$

Trong đó:

- X là biến ngẫu nhiên rời rạc với n giá trị khả dĩ.
- $P(x_i)$ là xác suất xảy ra của sự kiện x_i.
- $H(X)$ đo **mức độ bất định trung bình** của X.

Giải thích toán học:

- Nếu tất cả các sự kiện đều có xác suất bằng nhau, entropy đạt giá trị lớn nhất $\Rightarrow$ tính bất định cao nhất.
- Nếu một sự kiện xảy ra chắc chắn, entropy bằng 0 $\Rightarrow$ không có gì bất ngờ.

Ví dụ: Với một biến nhị phân (như tung đồng xu):

- Nếu $P(\text{ngửa}) = 0.5$ và $P(\text{sấp}) = 0.5$, thì

$$H(X) = -(0.5 \log 0.5 + 0.5 \log 0.5) = \log 2 = 1 \text{ bit}$$

- Nếu $P(\text{ngửa}) = 1$, $P(\text{sấp}) = 0$, thì

$$H(X) = -(1 \cdot \log 1 + 0 \cdot \log 0) = 0$$

Ứng dụng trong học máy: Entropy và hàm log được sử dụng trong:

- Cross-Entropy Loss
- KL-Divergence (Độ lệch giữa hai phân phối xác suất)
- Decision Tree (Dựa trên thông tin thu được để chia nhánh)

1.2.3 Xử lý các số rất nhỏ: Sử dụng log để tránh mất chính xác

Khi nhân nhiều xác suất nhỏ với nhau:

$$v = v_1 \times v_2 \times \cdots \times v_n$$

kết quả có thể rất nhỏ dẫn đến mất độ chính xác hoặc bị làm tròn về 0 trên máy tính.

Giải pháp: Tính log:

$$\log v = \log v_1 + \log v_2 + \cdots + \log v_n$$

Giúp tránh mất mát độ chính xác và vẫn giữ nguyên thứ tự so sánh vì hàm log là đơn điệu tăng.

2 Hàm căn bậc hai

Hàm căn bậc hai ($\sqrt{x}$) là một công cụ toán học quan trọng trong nhiều bài toán học máy, đặc biệt là khi đánh giá sự thay đổi tương đối, tính toán khoảng cách hoặc xử lý dữ liệu có sự chênh lệch lớn về độ lớn.

2.1 Chuẩn hóa ảnh hưởng giữa các vật thể

Khi đo lường sự thay đổi hoặc dịch chuyển, kích thước của vật thể ảnh hưởng rất lớn đến việc đánh giá mức độ quan trọng của sự thay đổi đó. Cụ thể:

- Một vật thể nhỏ có chiều dài khoảng $3\,cm$, nếu di chuyển $3\,cm$, thì đã dịch chuyển 100% kích thước ban đầu.
- Một vật thể lớn hơn như chiếc ô tô dài $3\,m = 300\,cm$, nếu cũng di chuyển $3\,cm$, thì tỉ lệ thay đổi chỉ là 1%.

Nếu không xử lý, các mô hình sẽ đánh giá sai tầm quan trọng của những dịch chuyển này. Khi đó, căn bậc hai đóng vai trò quan trọng trong việc **giảm ảnh hưởng quá lớn từ những chênh lệch tỷ lệ**.

2.2 Cân bằng ảnh hưởng bằng căn bậc hai

Khi đánh giá khoảng cách hoặc độ thay đổi, sử dụng căn bậc hai giúp:

- Giảm tác động quá lớn từ các giá trị lớn (giống như log).
- Làm mịn sự khác biệt giữa các vật thể có độ lớn chênh lệch.
- Giữ nguyên tính chất không âm và tăng đơn điệu, giúp dễ diễn giải.

Căn bậc hai thường xuất hiện trong các công thức đo khoảng cách như:

$$\text{Euclidean Distance: } d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{3}$$

Tức là căn bậc hai được dùng để tính **khoảng cách thực tế trong không gian** — một khái niệm rất quan trọng trong:

- Học không giám sát: k-Means, DBSCAN
- Học có giám sát: KNN
- Giảm chiều dữ liệu: PCA, t-SNE

3 Hàm mũ e^x

Hàm mũ cơ bản được định nghĩa như sau:

$$y = e^x$$

Trong đó:

- $x \in (-\infty, +\infty)$ là tập giá trị đầu vào (có thể âm, dương hoặc bằng 0).
- $y = e^x \in (0, +\infty)$ là tập giá trị đầu ra luôn dương và không bao giờ bằng 0.

3.1 Tính chất của hàm e^x

- Hàm **luôn dương**: $e^x > 0$ với mọi x.
- Hàm **đơn điệu tăng**: nếu $x_1 < x_2$ thì $e^{x_1} < e^{x_2}$.
- Đạo hàm của hàm mũ là chính nó: $\frac{d}{dx}e^x = e^x$.
- Khi $x = 0$, ta có $e^0 = 1$.

3.2 Ứng dụng trong học máy và trí tuệ nhân tạo

Hàm e^x thường được sử dụng trong các tình huống cần:

- **Chuyển đổi các giá trị thực** (bao gồm cả âm) thành **giá trị dương**, để có thể xử lý như xác suất hoặc trọng số.
- **Khuếch đại sự khác biệt giữa các giá trị**: Giá trị lớn hơn sẽ tạo ra e^x lớn vượt trội, rất quan trọng khi đưa ra quyết định trong mô hình.

Ví dụ: Hàm Softmax – chuẩn hóa xác suất

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

Hàm Softmax có các tính chất sau:

- Biến đổi một vector $x = [x_1, x_2, \ldots, x_n]$ thành một vector xác suất: tất cả phần tử đều dương và tổng bằng 1.
- Khi x_i lớn vượt trội so với các phần tử khác, thì $\text{Softmax}(x_i) \to 1$ và các phần tử còn lại gần như bằng 0.
- Tăng độ nhạy khi phân loại — giá trị lớn nhất sẽ chiếm ưu thế.

Ví dụ cụ thể:
Giả sử đầu ra mạng là:

$$x = [1.2,\ 0.9,\ -1.0] \Rightarrow e^x = [e^{1.2},\ e^{0.9},\ e^{-1.0}] \approx [3.32,\ 2.46,\ 0.37]$$

Khi đó:

$$\text{Softmax}(x) = \left[\frac{3.32}{6.15}, \frac{2.46}{6.15}, \frac{0.37}{6.15}\right] \approx [0.54,\ 0.40,\ 0.06]$$

Giải thích: Dù giá trị ban đầu cách biệt không nhiều, sau khi qua hàm mũ và chuẩn hóa, phần tử lớn nhất (1.2) đã chiếm ưu thế rõ ràng (54%). Không dùng trực tiếp giá trị x để chuẩn hóa xác suất là vì:

- Vì x có thể âm nên không thể dùng làm xác suất (xác suất không thể âm).
- Tổng của các x_i có thể không bằng 1, không đảm bảo là phân phối xác suất.
- Hàm mũ giúp giữ thứ tự và làm trơn quyết định, thuận tiện cho học sâu (deep learning).