AI VIET NAM – AIO2025

Softmax

Ngày 4 tháng 6 năm 2025

Bài viết được thực hiện bởi Nguyễn Đình Tiềm, Bùi Minh Đức và Đinh Quang Vinh.

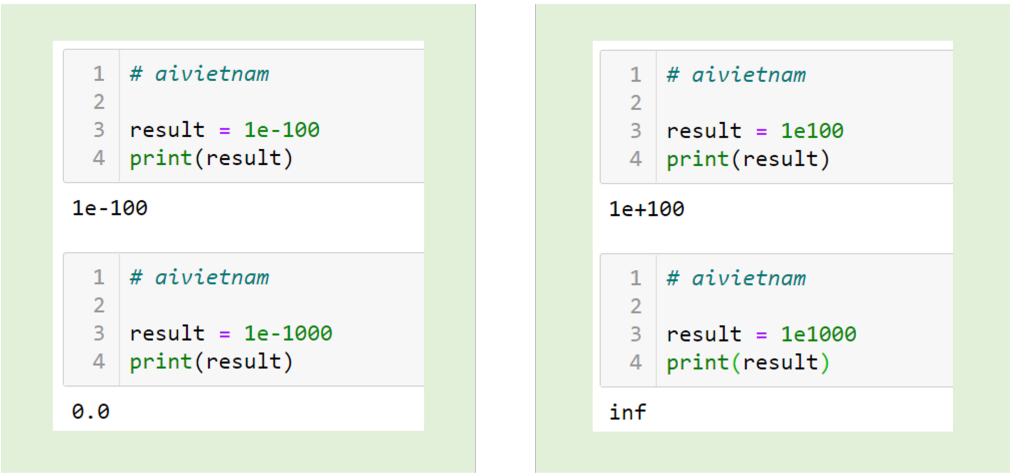
Bài viết này giới thiệu hàm softmax, giúp chuyển đổi dãy số thành phân phối xác suất. Nó cũng trình bày cách giải quyết vấn đề overflow và underflow khi xử lý số quá lớn hoặc quá nhỏ bằng stable softmax - một giải pháp hiệu quả. Stable softmax không chỉ ổn định tính toán softmax mà còn đảm bảo sự chính xác và độ tin cậy trong quá trình xử lý.

1. Overflow và Underflow

Trong lập trình học máy và học sâu, ta thường xuyên phải xử lý các số thực có phạm vi rộng, từ rất lớn đến rất nhỏ. Điều này gây ra nhiều khó khăn, điển hình là vấn đề overflow và underflow.

Overflow là hiện tượng xảy ra khi một số lớn hơn giới hạn của kiểu dữ liệu mà nó được lưu trữ. Underflow là hiện tượng khi một số nhỏ hơn giới hạn dưới của kiểu dữ liệu mà nó được lưu trữ. Ví dụ, trong Python, nếu một số thực vượt quá giới hạn trên của kiểu dữ liệu float, nó sẽ được chuyển đổi thành số vô cùng lớn, và ngược lại, một số quá nhỏ sẽ được chuyển thành giá trị 0. Điều này có thể dẫn đến các lỗi tính toán, hoặc thậm chí có thể làm chương trình dừng hoạt động.

Ví dụ về overflow và underflow có thể xuất hiện trong Python thông qua những trường hợp sau:



Hình 1: Ví dụ về overflow và underflow.

2. Phần Trăm

Phần trăm không chỉ là một khái niệm số học, mà còn là cách ta diễn đạt về sự phân phối và so sánh giữa các phần khác nhau của một tổng thể. Hãy tưởng tượng bạn có một hộp kẹo với 3 viên kẹo bạc

hà, 5 viên kẹo cafe và 2 viên kẹo chocolate. Thay vì nói rằng "trong 10 (3+5+2) viên kẹo có 2 viên kẹo chocolate", rõ ràng dùng phần trăm sẽ mang nhiều ý nghĩa hơn "số kẹo chocolate là 20%".

$$\text{Phần trăm chocolate} = \frac{2}{3+5+2} \times 100 = 20\%$$

Tính phần trăm giúp ta hiểu rõ hơn về sự phân phối và quan hệ giữa các phần khác nhau. Trong Học máy, đặc biệt là khi ta muốn dự đoán các khả năng, việc đo lường phần trăm trở nên quan trọng. Ví dụ, trong việc dự đoán loại kẹo phổ biến nhất, phần trăm giúp ta hiểu xác suất mà mỗi loại có thể xuất hiện.

3. Softmax

Qua các phần vừa rồi, ta đã hiểu về sự quan trọng của việc đo lường bằng phần trăm. Tuy nhiên, làm thế nào ta có thể áp dụng khái niệm phần trăm khi đối mặt với các số âm?

3.1 Tính phần trăm với số âm

Trong học máy và học sâu, đầu ra của mô hình thường là một dãy số không chỉ có các số dương mà còn có các số âm. Nếu ta cố ý tính phần trăm trực tiếp từ các số này, kết quả nhận được có thể sẽ không đúng với thực tế.

Để hiểu rõ hơn về vấn đề, hãy xem xét một ví dụ đơn giản liên quan đến dự đoán thời tiết. Giả sử ta có một mô hình dự đoán xác suất mưa (P_{rain}) và nắng (P_{sun}) dựa trên các yếu tố như nhiệt độ, độ ẩm, và áp suất không khí. Đầu ra của mô hình là một dãy số:

$$\text{Đầu ra của mô hình} = [1.2, -0.8, 3.5]$$

Trong đó:

- $P_{\text{rain}} = 2.0$
- $P_{\text{sun}} = -10.0$
- $P_{\text{cloudy}} = 0.1$

Ta hãy thử tính phần trăm của rain:

$$P_{\text{rain}} = \frac{2.0}{2.0 - 10.0 + 0.1} \times 100 = -25.32\%$$

Vấn đề xuất hiện ở đây P_{rain} là số âm, tổng của các xác suất là một số âm, dẫn đến kết quả phần trăm không có ý nghĩa.

3.2 Softmax

Giải pháp cho vấn đề trên là softmax, một phép toán giúp ta chuyển đổi một dãy số thành một phân phối xác suất. Nó có tổng của các xác suất bằng 1, và tất nhiên có thể tính toán chính xác với số âm.

Một trong những người tạo ra softmax là ông Geoffrey Hinton, một nhà nghiên cứu AI nổi tiếng. Hàm softmax do ông tạo ra được sử dụng rộng rãi trong các mô hình học sâu để đưa ra dự đoán xác suất chính xác cho nhiều lớp.

Công thức của softmax cho một dãy số z có K phần tử là:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Trong đó:

- $\text{Softmax}(z_i)$ là xác suất của phần tử thứ i trong dãy sau khi áp dụng softmax.
- e là số Euler, một hằng số tự nhiên có giá trị xấp xỉ 2.71828.
- $\sum_{j=1}^{K} e^{z_j}$ là tổng của e lũy thừa từng phần tử trong dãy z.

Ví dụ minh họa:

Giả sử ta có một dãy số z như sau: $z = [2.0, -10.0, 0.1]$, ta sẽ sử dụng công thức softmax để chuyển đổi z thành một phân phối xác suất.

$$\text{Softmax}([2.0, -10.0, 0.1]) = \left[\frac{e^{2.0}}{e^{2.0} + e^{-10.0} + e^{-0.1}}, \frac{e^{-10.0}}{e^{2.0} + e^{-10.0} + e^{0.1}}, \frac{e^{0.1}}{e^{2.0} + e^{-10.0} + e^{0.1}}\right]$$

$$\text{Softmax}([2.0, -10.0, 0.1]) \approx [0.86989, 0.00001, 0.13011]$$

Sau khi áp dụng softmax, ta nhận được một phân phối xác suất, trong đó xác suất lớn nhất ứng với phần tử đầu tiên (2.0) và giảm dần xuống cho các phần tử còn lại.

Ví dụ xây dựng hàm softmax với Python:

```python
import math

# Given three values
v1 = 1.0
v2 = 2.0
v3 = 3.0

# Compute softmax
total = math.exp(v1) + math.exp(v2) + math.exp(v3)

s1 = math.exp(v1)/total
s2 = math.exp(v2)/total
s3 = math.exp(v3)/total

# Print out
print(f"{s1:.5f}  {s2:.5f}   {s3:.5f}")

>> Output: 0.09003   0.24473    0.66524
```

Nhìn chung, softmax không chỉ giúp ta tính toán phần trăm mà còn giúp ta dễ dàng lựa chọn xác suất cao nhất, làm cho kết quả trở nên rõ ràng và dễ hiểu. Tuy nhiên softmax dễ bị ảnh hưởng bởi số lớn, có thể dẫn đến vấn đề overflow.

4. Stable Softmax

Ở phần 3, ta đã tìm hiểu về softmax trong việc biến đổi số thành xác suất, nhưng như mọi thuật toán, softmax cũng gặp phải các thách thức như overflow và underflow khi các số quá lớn hoặc quá nhỏ.

4.1 Overflow và underflow khi tính toán softmax

Khi ta thực hiện phép toán softmax trên các số lớn, máy tính có thể không xử lý được và gặp vấn đề overflow. Ngược lại, khi ta thực hiện phép toán softmax trên các số quá nhỏ, ta lại đối mặt với vấn đề underflow.

Ví dụ overflow:

```python
import math

# Given three values
v1 = 1001.0
v2 = 1002.0
v3 = 1003.0

# Compute softmax
total = math.exp(v1) + math.exp(v2) + math.exp(v3)

s1 = math.exp(v1)/total
s2 = math.exp(v2)/total
s3 = math.exp(v3)/total

# Print out
print(f"{s1:.5f}  {s2:.5f}  {s3:.5f}")
```

```
---------------------------------------------------------------------------
OverflowError                             Traceback (most recent call last)
C:\Users\DINHVI~1\AppData\Local\Temp/ipykernel_6404/1559505462.py in <module>
      7
      8 # compute softmax
----> 9 total = math.exp(v1) + math.exp(v2) + math.exp(v3)
     10
     11 s1 = math.exp(v1)/total

OverflowError: math range error
```

Ví dụ underflow:

```python
import math

# Given three values
v1 = -801.0
v2 = -805.0
v3 = -800.0

# Compute softmax
total = math.exp(v1) + math.exp(v2) + math.exp(v3)

s1 = math.exp(v1)/total
s2 = math.exp(v2)/total
s3 = math.exp(v3)/total

# Print out
print(f"{s1:.5f}  {s2:.5f}  {s3:.5f}")
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[17], line 11
      8 # compute softmax
      9 total = math.exp(v1) + math.exp(v2) + math.exp(v3)
---> 11 s1 = math.exp(v1)/total
     12 s2 = math.exp(v2)/total
     13 s3 = math.exp(v3)/total

ZeroDivisionError: float division by zero
```

Một cách hiệu quả để giải quyết overflow là **Stable Softmax**. Thay vì thực hiện phép toán trực tiếp trên các số, ta điều chỉnh chúng trước khi tính toán để tránh các giá trị quá lớn hoặc quá nhỏ.

Công thức của stable softmax cho một dãy số z có K phần tử là:

$$\text{Stable Softmax}(z_i) = \frac{e^{z_i - \max(z)}}{\sum_{j=1}^{K} e^{z_j - \max(z)}}$$

Trong đó:

- Stable Softmax(z_i) là xác suất của phần tử thứ i trong dãy sau khi áp dụng stable softmax.
- $\max(z)$ là giá trị lớn nhất trong dãy số z.
- $\sum_{j=1}^{K} e^{z_j - \max(z)}$ là tổng của e lũy thừa từng phần tử sau khi trừ đi giá trị lớn nhất trong dãy z.

Ví dụ minh họa:

```python
import math
# Given three values
v1 = 1.0
v2 = 1001.0
v3 = 1002.0

# get max
max_value = v3

# compute stable softmax
e_v1 = math.exp(v1 - max_value)
e_v2 = math.exp(v2 - max_value)
e_v3 = math.exp(v3 - max_value)

total = e_v1 + e_v2 + e_v3

s1 = e_v1/total
s2 = e_v2/total
s3 = e_v3/total

# print out
print(f"{s1:.5f}  {s2:.5f}   {s3:.5f}")

# Outphut: 0.00000 0.26894 0.73106
```

Ưu Điểm của stable softmax:

- **Tránh overflow**: Giữ cho các số ổn định trong quá trình tính toán softmax.
- **Kết quả đồng nhất**: Không làm thay đổi ý nghĩa của phân phối xác suất cuối cùng.

Nhìn chung, stable softmax là một bước quan trọng để đảm bảo tính ổn định và chính xác của softmax, giúp ta xử lý mọi dạng dữ liệu mà không phải lo lắng về vấn đề của số học.

Phần 5: Kết Luận

Trong bài viết này ta đã tìm hiểu về cách áp dụng hàm Softmax khi tính toán phần trăm với số âm và cách xử lý vấn đề overflow và underflow bằng stable softmax. Ngoài ra, bài viết cung cấp cái nhìn tổng quan về cách softmax hoạt động trong học máy và làm thế nào nó giải quyết các vấn đề về số học. Đặc biệt là Stable softmax không chỉ giải quyết vấn đề về tính toán, mà còn giúp tăng độ chính xác và ổn định của mô hình.