

Hàm số, mũ và log

☑ Trắc nghiệm 6 ⬇ PDF bài

Một model nhận ảnh chữ số viết tay và trả về mười con số, mỗi số cho một chữ số từ 0 đến 9, chẳng hạn $[1.2, -0.4, 6.3, 0.8, \dots]$. Mười số này chưa phải xác suất: có số âm, và tổng không bằng 1. Để đổi chúng thành xác suất, model lấy hàm mũ của từng số rồi chia cho tổng. Khi chấm model, ta lấy log của xác suất mà model dành cho đáp án đúng. Bài này học hai hàm đó, hàm mũ và log, cùng hai hàm dựng từ chúng là sigmoid và softmax.

Bài chỉ cần toán THCS: lũy thừa, phân số và đọc đồ thị. Mọi công thức đều có một đoạn NumPy in ra số để đối chiếu với phép tính tay.

MỤC TIÊU

- Đọc hàm số qua bảng giá trị và đồ thị.
- Tính e^x , $\log x$; dùng $\log(ab) = \log a + \log b$.
- Tính sigmoid và softmax.
- Tính log-loss; biết vì sao đoán sai mà tự tin bị phạt nặng.

Hàm số: đưa vào một số, nhận ra một số

Một hàm số là một quy tắc: mỗi giá trị đầu vào cho đúng một giá trị đầu ra. Đổi nhiệt độ từ độ C sang độ F là một hàm: $F = 1.8C + 32$. Trong Python, một `def` nhận số và trả về số cũng là một hàm theo đúng nghĩa ấy.

Dạng đơn giản nhất mà học máy dùng là hàm bậc nhất

$$y = wx + b.$$

w là **hệ số góc** (slope): x tăng 1 thì y tăng đúng w . b là giá trị của y khi $x = 0$. Đồ thị là một đường thẳng. Bài linear regression ở chương 2 học đúng hai số w và b này từ dữ liệu.

Cách kiểm nhanh một hàm có phải bậc nhất không là chia độ tăng của y cho độ tăng của x giữa các điểm liên tiếp. Với hàm bậc nhất, tỉ số này luôn bằng w . `np.diff` trả về hiệu của hai phần tử liền nhau trong mảng.

Python

```
import numpy as np

def to_f(c):
    return 1.8 * c + 32

c = np.array([-10, 0, 10, 20, 30])
f = to_f(c)
print(f)
print(np.diff(f) / np.diff(c))
```

KẾT QUẢ

```
[14.  32.  50.  68.  86.]
[1.8  1.8  1.8  1.8]
```

Bốn tỉ số đều bằng 1.8, đúng hệ số góc. 0 độ C cho 32 độ F, đúng hệ số tự do. Hàm cong như $y = x^2$ thì tỉ số này đổi theo từng đoạn; phần sau của bài gặp nhiều hàm như thế.

HỎI NHANH

Hàm $y = 3x - 2$. Khi x tăng từ 4 lên 6, y tăng bao nhiêu?

A 2

B 3

C 6

D 16

Lũy thừa và hàm mũ

2^{10} là 2 nhân với chính nó 10 lần, bằng 1024. Ba quy tắc đủ dùng cho cả bài:

$$a^m \cdot a^n = a^{m+n}, \quad a^0 = 1, \quad a^{-n} = \frac{1}{a^n}.$$

Học máy hay dùng cơ số $e \approx 2.71828$. Đây là một hằng số như π . Lý do chọn nó sẽ rõ ở bài 0.6: hàm e^x có đạo hàm bằng chính nó, nên các phép tính về sau rất gọn. Hàm e^x có ký hiệu khác là $\exp(x)$, trong NumPy là `np.exp`.

Ba điều cần nhớ về e^x . Nó **luôn dương**, kể cả khi x âm. $e^0 = 1$. x âm càng nhiều thì e^x càng sát 0; x dương thì e^x tăng rất nhanh, mỗi khi x tăng 1 thì giá trị nhân thêm khoảng 2.7 lần.

Python

```
x = np.array([-5, -1, 0, 1, 2, 5, 10])
print([round(float(v), 4) for v in np.exp(x)])
print(np.exp(2 + 3), np.exp(2) * np.exp(3))
```

KẾT QUẢ

```
[0.0067, 0.3679, 1.0, 2.7183, 7.3891, 148.4132, 22026.4658]
148.4131591025766 148.4131591025766
```

Dòng đầu cho thấy $e^{-5} \approx 0.0067$ đã rất nhỏ, còn e^{10} đã hơn hai vạn. Dòng thứ hai kiểm tra quy tắc $e^{2+3} = e^2 \cdot e^3$: hai cách tính cho cùng một số.

Hàm mũ tăng nhanh đến mức máy tính không chứa nổi. Số thực float64 lớn nhất khoảng $1.8 \cdot 10^{308}$, nên e^{710} đã vượt quá và thành `inf` (vô cùng). Phần softmax ở dưới sẽ gặp lại chuyện này.

Python

```
with np.errstate(over="ignore"):
    print(np.exp(np.array([700.0, 710.0])))
```

KẾT QUẢ

```
[1.01423205e+304      inf]
```

HỎI NHANH

$e^3 \cdot e^{-3}$ bằng bao nhiêu?

A 0

B 1

C e^9

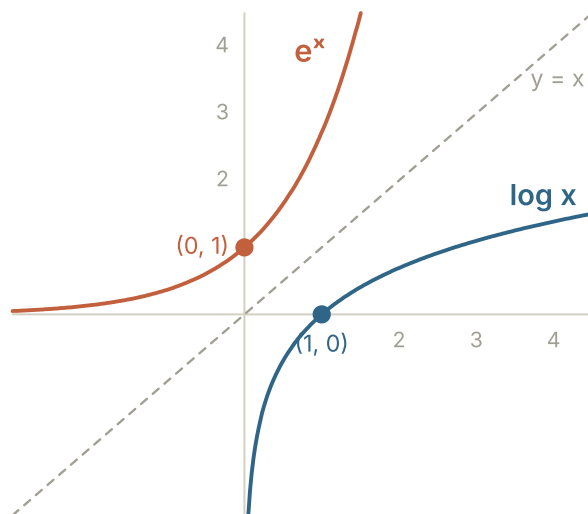
D e^{-9}

Log: hỏi ngược lại phép mũ

$2^3 = 8$, nên “2 mũ mấy thì bằng 8?” có câu trả lời là 3. Ta viết $\log_2 8 = 3$. Log là phép hỏi ngược của phép mũ: $\log_a x$ là số mũ cần đặt lên a để được x .

Cơ số e cho **log tự nhiên**, viết $\ln x$. Trong sách học máy và trong NumPy, `log` không ghi cơ số nghĩa là cơ số e : `np.log` là $\ln$, còn `np.log2` và `np.log10` cho cơ số 2 và 10. Bài này cũng viết `log` cho log tự nhiên.

Đồ thị của $\log x$ là đồ thị của e^x lật qua đường chéo $y = x$, vì hai hàm làm ngược việc của nhau.



e^x và $\log x$ đối xứng qua đường $y = x$.

Từ hình đọc được ba điều. $\log x$ chỉ có nghĩa khi $x > 0$. $\log 1 = 0$. Với $0 < x < 1$, như một xác suất, $\log x$ là số âm, và x càng sát 0 thì $\log x$ càng âm nhiều.

Quy tắc quan trọng nhất của log đến từ quy tắc nhân lũy thừa:

$$\log(ab) = \log a + \log b, \quad \log \frac{a}{b} = \log a - \log b, \quad \log a^k = k \log a.$$

Log biến phép nhân thành phép cộng. Học máy cần điều này vì xác suất hay được nhân với nhau. Giả sử model chấm 1000 câu, mỗi câu nó cho đáp án đúng xác suất 0.5. Xác suất model đoán đúng cả 1000 câu là 0.5^{1000} , một số cỡ 10^{-302} . Thêm 100 câu nữa thì máy làm tròn thành đúng 0, gọi là tràn dưới (underflow). Cộng log thì không có vấn đề gì.

Python

```
print(np.log2(8), np.log10(1000), np.log(np.e))
print(np.log(3 * 7), np.log(3) + np.log(7))
p = np.full(1100, 0.5)
print(np.prod(p[:1000]), np.prod(p))
print(np.sum(np.log(p[:1000])), np.sum(np.log(p)))
```

KẾT QUẢ

```
3.0 3.0 1.0
3.044522437723423 3.044522437723423
9.332636185032189e-302 0.0
-693.1471805599454 -762.4618986159398
```

Tích của 1100 xác suất đã thành 0.0, nên mọi so sánh dựa trên nó đều mất. Tổng log của chúng là -762.46 , một số bình thường, và vẫn so sánh được: tổng log lớn hơn nghĩa là tích lớn hơn, vì log là hàm tăng.

HỎI NHANH

Biết $\log 2 \approx 0.693$ và $\log 5 \approx 1.609$. $\log 10$ xấp xỉ bao nhiêu?

A 0.916

B 1.115

C 2.302

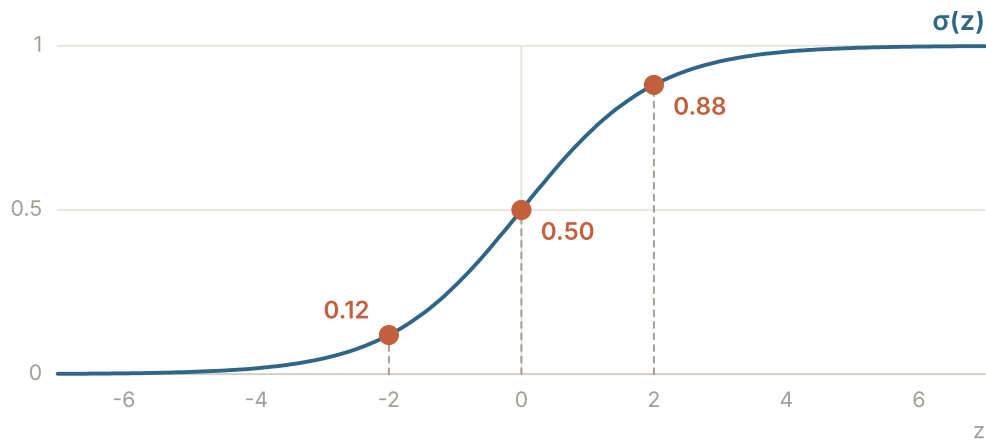
D 3.218

Sigmoid: ép một số bất kỳ vào khoảng (0, 1)

Một bộ lọc thư rác tính cho mỗi thư một điểm z bằng hàm bậc nhất của vài con số mô tả thư, như số đường link hay số chữ in hoa. z có thể là bất kỳ số thực nào, còn điều ta muốn là xác suất thư là rác, một số nằm giữa 0 và 1. Hàm **sigmoid** làm việc ấy:

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

Đọc công thức bằng ba trường hợp. $z = 0$ thì $e^0 = 1$, nên $\sigma(0) = 1/2$. z dương lớn thì e^{-z} sát 0, nên $\sigma(z)$ sát 1. z âm nhiều thì e^{-z} rất lớn, nên $\sigma(z)$ sát 0. Đồ thị là một đường chữ S, không bao giờ chạm 0 hay 1.



Sigmoid đưa mọi z vào khoảng $(0, 1)$; $\sigma(0) = 0.5$.

Hình chữ S đối xứng qua điểm $(0, 0.5)$, tức là $\sigma(-z) = 1 - \sigma(z)$. Điểm $z = 2$ cho xác suất thư rác 0.88 thì $z = -2$ cho 0.12.

Hỏi ngược lại: xác suất p ứng với điểm z nào? Giải $\sigma(z) = p$ ra

$$z = \log \frac{p}{1 - p}.$$

Tỉ số $p/(1 - p)$ gọi là **odds**: $p = 0.8$ cho odds bằng 4, nghĩa là “khả năng xảy ra gấp 4 lần không xảy ra”. Vì vậy z còn được gọi là log-odds, hay **logit**. Từ logit sẽ gặp lại ở mọi bài phân loại.

Python

```
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

z = np.array([-6, -2, 0, 2, 6])
print(np.round(sigmoid(z), 4))
print(np.round(sigmoid(z) + sigmoid(-z), 4))
logit = np.log(0.8 / 0.2)
print(round(logit, 4), sigmoid(logit))
```

KẾT QUẢ

```
[0.0025 0.1192 0.5      0.8808 0.9975]
[1.  1.  1.  1.  1.]
1.3863 0.8
```

Dòng thứ hai kiểm $\sigma(z) + \sigma(-z) = 1$ ở cả năm điểm. Dòng thứ ba đi một vòng: từ $p = 0.8$ sang logit 1.3863, rồi sigmoid trả lại đúng 0.8.

HỎI NHANH

Model cho một thư điểm z và $\sigma(z) = 0.8$. Với một thư khác có điểm $-z$, sigmoid bằng bao nhiêu?

A -0.8

B 0.2

C 0.8

D 1.25

Softmax: từ nhiều điểm thành một phân phối

Với mười chữ số, model cho mười điểm $z_1, \dots, z_{10}$, gọi là **logits**. Softmax đổi chúng thành mười xác suất:

$$p_k = \frac{e^{z_k}}{\sum_j e^{z_j}}.$$

Hai bước: lấy mũ để mọi số thành dương, rồi chia cho tổng để các số cộng lại bằng 1. Logit lớn hơn vẫn cho xác suất lớn hơn, vì e^x là hàm tăng.

Cộng cùng một số c vào mọi logit thì softmax **không đổi**: $e^{z_k+c} = e^c \cdot e^{z_k}$, và thừa số e^c xuất hiện ở cả tử lẫn mẫu nên triệt tiêu. Tính chất này cứu ta khỏi chuyện tràn số. Logits như [1000, 999, 998] làm `np.exp` thành `inf`, và `inf / inf` cho `nan`. Trừ đi logit lớn nhất trước khi lấy mũ thì số lớn nhất thành $e^0 = 1$, không còn gì tràn, mà kết quả vẫn y như cũ.

Python

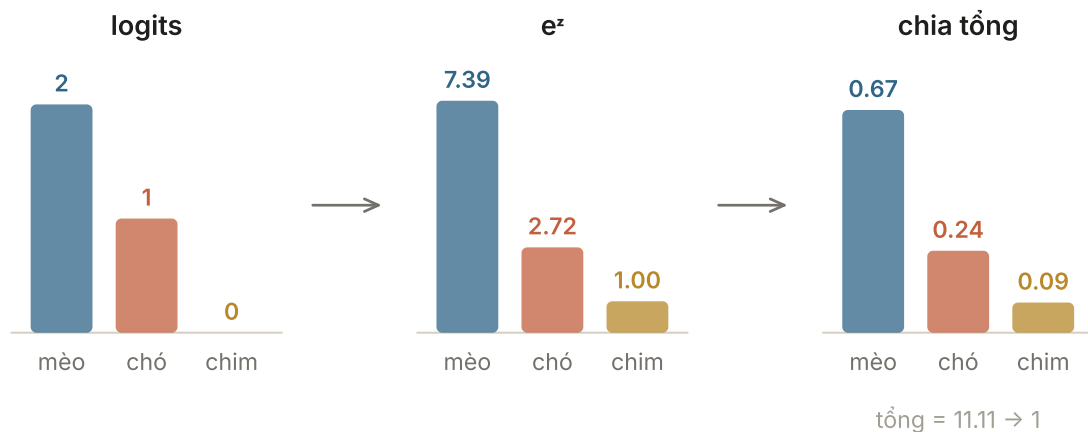
```
def softmax(z):
    e = np.exp(z - z.max())
    return e / e.sum()

z = np.array([2.0, 1.0, 0.0])
print(np.round(softmax(z), 4), round(softmax(z).sum(), 6))
print(np.round(softmax(z + 100), 4))
big = np.array([1000.0, 999.0, 998.0])
print(np.round(softmax(big), 4))
with np.errstate(over="ignore", invalid="ignore"):
    print(np.exp(big) / np.exp(big).sum())
```

KẾT QUẢ

```
[0.6652 0.2447 0.09 ] 1.0
[0.6652 0.2447 0.09 ]
[0.6652 0.2447 0.09 ]
[nan nan nan]
```

Ba dòng đầu cho cùng một phân phối, vì ba bộ logits chỉ khác nhau một hằng số cộng. Dòng cuối là cách viết thẳng theo công thức, không trừ max: kết quả hỏng hoàn toàn.



Softmax: lấy mũ từng logit rồi chia cho tổng.

Với hai lớp, softmax của logits $[z, 0]$ cho lớp đầu xác suất $e^z / (e^z + 1)$. Chia cả tử và mẫu cho e^z được $1 / (1 + e^{-z}) = \sigma(z)$. Sigmoid là softmax hai lớp.

HỎI NHANH

Logits ba lớp là $[0, \log 3, 0]$. Xác suất của lớp giữa là bao nhiêu?

- A $1/3$
- B $3/4$
- C $3/5$
- D $\log 3/3$

HỎI NHANH

Model cho ba logits $[3, 3, 3]$. Softmax trả về gì?

- A $[1, 1, 1]$
- B $[1/3, 1/3, 1/3]$
- C $[0.95, 0.95, 0.95]$
- D $[3/9, 3/9, 3/9]$ nhưng chỉ khi logits dương

HỎI NHANH

Cộng 5 vào cả ba logits $[2, 1, 0]$, thành $[7, 6, 5]$. Xác suất của lớp đầu thay đổi thế nào?

- A Không đổi, vẫn khoảng 0.665
- B Tăng lên, vì logit của nó lớn hơn
- C Giảm xuống, vì các lớp kia cũng tăng
- D Không tính được nếu không biết nhân thật

Vì sao loss có log

Model đã cho xác suất. Để chấm, ta nhìn xác suất p mà model dành cho **đáp án đúng** và tính

$$\text{loss} = -\log p.$$

Dấu trừ để loss không âm, vì $\log p \leq 0$ khi $p \leq 1$. Đoán đúng chắc chắn, $p = 1$, cho loss 0. p càng nhỏ thì loss càng lớn, và lớn không giới hạn khi p tiến về 0. Ta gọi đây là **log-loss**, hay **cross-entropy** khi có nhiều lớp.

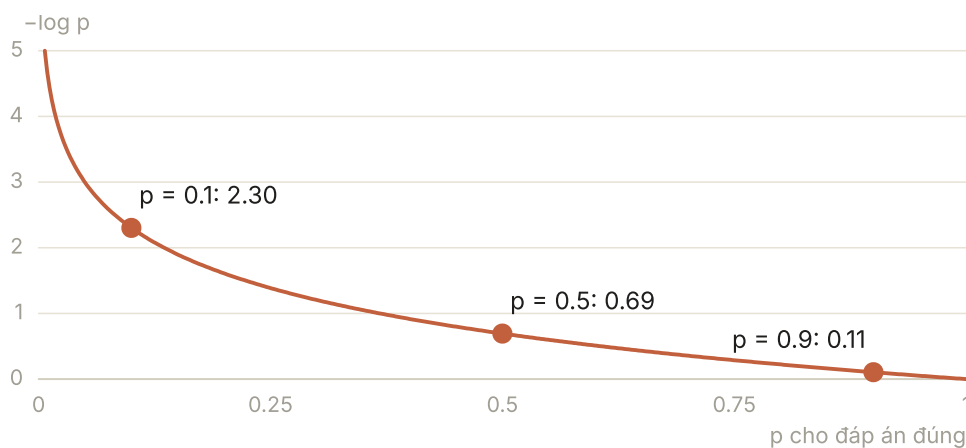
Python

```
p_true = np.array([0.99, 0.9, 0.5, 0.1, 0.01])
print(np.round(-np.log(p_true), 3))
```

KẾT QUẢ

```
[0.01  0.105 0.693 2.303 4.605]
```

Từ 0.99 xuống 0.9, loss chỉ tăng thêm chưa tới 0.1. Từ 0.1 xuống 0.01, loss tăng thêm 2.3. Giảm p đi một nửa thì loss luôn tăng thêm đúng $\log 2 \approx 0.693$, vì $-\log(p/2) = -\log p + \log 2$. Model nói “chắc 99% là mèo” mà ảnh là chó thì bị phạt rất nặng.



$-\log p$: gần 0 khi p sát 1, tăng vọt khi p tiến về 0.

Lấy trung bình log-loss trên cả tập dữ liệu:

$$-\frac{1}{n} \sum_{i=1}^n \log p_i = -\frac{1}{n} \log \prod_{i=1}^n p_i.$$

Vế phải là log của tích các xác suất, tức xác suất model gán cho **toàn bộ** đáp án đúng của tập, nếu coi các mẫu độc lập. Làm loss trung bình nhỏ nhất cũng chính là làm tích ấy lớn nhất. Ta tính bằng tổng log thay vì tích vì lý do đã thấy ở trên: tích của nghìn xác suất tràn dưới thành 0.

Với hai lớp, nhãn $y \in \{0, 1\}$ và model cho $p = P(y = 1)$, xác suất của đáp án đúng là p khi $y = 1$ và $1 - p$ khi $y = 0$. Viết gộp lại:

$$\text{loss} = -[y \log p + (1 - y) \log(1 - p)].$$

Với mỗi mẫu, chỉ một trong hai số hạng khác 0. Đây là binary cross-entropy của các bài phân loại chương 1 và 2.

Log-loss nhìn thấy điều accuracy không thấy. Hai model dưới đây cùng đúng 3 trên 4 câu khi lấy ngưỡng 0.5, nhưng model B sai một câu với độ tự tin rất cao.

Python

```
A = np.array([0.9, 0.8, 0.6, 0.4])    # xác suất model A dành cho đáp án đúng
B = np.array([0.6, 0.6, 0.6, 0.05])
for name, p in [("A", A), ("B", B)]:
    print(name, "accuracy", (p > 0.5).mean(), "log-loss", round(-np.log(p).mean(), 3))
```

KẾT QUẢ

```
A accuracy 0.75 log-loss 0.439
B accuracy 0.75 log-loss 1.132
```

Cùng accuracy 0.75, nhưng log-loss của B gấp hơn hai lần A. Câu B cho đáp án đúng chỉ 0.05 đóng góp $-\log 0.05 \approx 3.0$ vào tổng. Nếu đề chấm bằng log-loss, model B thua xa.

Giờ nối softmax với log-loss. Model phân loại ảnh thành mèo, chó, chim, cho logits $[2, 1, 0]$; như mục trên, softmax cho 0.665, 0.245, 0.090. Ảnh thật là chó, nên loss là $-\log 0.245 \approx 1.408$. Nếu ảnh là mèo thì loss là 0.408, là chim thì 2.408. Ba số cách nhau đúng 1, bằng khoảng cách giữa các logit, vì lấy log của công thức softmax được

$$-\log p_k = \log \sum_j e^{z_j} - z_k.$$

Với cùng một bộ logits, số hạng đầu chung cho mọi lớp, nên lớp có logit lớn hơn 1 thì loss nhỏ hơn đúng 1.

Giờ model sửa logit của chó từ 1 lên 2, thành $[2, 2, 0]$. Chó được xác suất 0.468, loss còn $-\log 0.468 \approx 0.759$. Huấn luyện ở các chương sau làm đúng việc này hàng triệu lần: chỉnh logits để loss trên đáp án đúng giảm dần.

Python

```
z = np.array([2.0, 1.0, 0.0])
p = softmax(z)
print(np.round(p, 3), np.round(-np.log(p), 3))
z2 = np.array([2.0, 2.0, 0.0])
print(np.round(softmax(z2), 3), round(-np.log(softmax(z2)[1]), 3))
```

KẾT QUẢ

```
[0.665 0.245 0.09 ] [0.408 1.408 2.408]
[0.468 0.468 0.063] 0.759
```

HỎI NHANH

Model dành cho đáp án đúng xác suất 0.8. Sau một lần sửa, xác suất ấy còn 0.4. Log-loss của mẫu này thay đổi thế nào?

- A Tăng gấp đôi
- B Tăng thêm 0.4
- C Không đổi, vì đáp án vẫn là lớp có xác suất cao nhất
- D Tăng thêm khoảng 0.693

Khi đi thi

- Kiểm đầu ra của model là logits hay xác suất: xác suất thì không âm và mỗi dòng cộng lại bằng 1.
- Tự tính softmax thì trừ max trước khi lấy mũ; tự tính log-loss thì kẹp `np.clip(p, 1e-15, 1 - 1e-15)` để không gặp $\log(0)$.
- Xem đề chấm bằng accuracy hay log-loss: hai metric có thể xếp hạng hai model ngược nhau.

Tóm tắt

- Hàm bậc nhất $y = wx + b$: w là độ tăng của y khi x tăng 1, b là giá trị tại $x = 0$.
- e^x luôn dương, $e^0 = 1$, tăng rất nhanh; $e^{a+b} = e^a e^b$.
- $\log$ là phép hỏi ngược của mũ, chỉ có nghĩa khi $x > 0$, âm khi $x < 1$; $\log(ab) = \log a + \log b$ biến tích thành tổng.
- Sigmoid $\sigma(z) = 1/(1 + e^{-z})$ đưa một số bất kỳ vào $(0, 1)$; $\sigma(-z) = 1 - \sigma(z)$; ngược lại là logit $\log \frac{p}{1-p}$.
- Softmax lấy mũ rồi chia tổng; cộng cùng hằng số vào mọi logit không đổi kết quả, nên trừ max để tránh tràn số.
- Log-loss $-\log p$ của đáp án đúng: 0 khi chắc chắn đúng, rất lớn khi tự tin mà sai; trung bình log-loss nhỏ nhất tương đương xác suất của cả tập lớn nhất.

Kiểm tra cuối bài

Mười câu dưới đây bao quát cả bài. Làm hết rồi xem điểm; câu nào sai thì đọc lại mục tương ứng.

CÂU 1

Hàm sigmoid trả về giá trị nằm trong khoảng nào?

A $(-1, 1)$

B $(-\infty, \infty)$

C $(0, 1)$

D $[0, \infty)$

ĐỀ THI NOAI 2026 Singapore · Vòng sơ loại · Câu 175

CÂU 2

Lớp cuối của một mạng neural dùng softmax. Mạng này nhiều khả năng làm bài toán gì?

A Dự đoán một số thực, như giá nhà

B Phân loại nhiều lớp, đổi logits thành xác suất

C Chỉ phân loại hai lớp

D Không dự đoán gì, softmax chỉ dùng ở lớp ẩn

ĐỀ THI NOAI 2026 Singapore · Vòng sơ loại · Câu 193

CÂU 3

Cross-entropy dùng để chấm model phân loại. Nó đo điều gì?

- A Bình phương khoảng cách giữa nhãn và lớp được chọn
- B Trị tuyệt đối của hiệu hai nhãn
- C Góc giữa vector dự đoán và vector nhãn one-hot
- D Độ lệch giữa phân phối dự đoán và phân phối của nhãn

ĐỀ THI NOAI 2026 Singapore · Vòng sơ loại · Câu 194

CÂU 4

Biết $\log a = 1.2$ và $\log b = 0.5$. $\log(a^2b)$ bằng bao nhiêu?

- A 2.9
- B 1.7
- C 0.72
- D 1.94

CÂU 5

Một model trả về vector $[0.7, 0.5, -0.2]$ cho ba lớp. Vector này là gì?

- A Logits, cần qua softmax mới thành xác suất
- B Xác suất của ba lớp, vì giá trị lớn nhất nhỏ hơn 1
- C Xác suất, nhưng cần làm tròn số âm lên 0
- D Log-loss của ba lớp

CÂU 6

Nhãn thật là $y = 0$, model cho $P(y = 1) = 0.9$. Binary cross-entropy của mẫu này xấp xỉ bao nhiêu?

- A 0.105
- B 0.9
- C 0, vì nhãn bằng 0
- D 2.303

CÂU 7

Một học sinh nhân xác suất của 2000 mẫu, mỗi xác suất khoảng 0.6, và máy in ra 0.0. Vì sao, và nên sửa thế nào?

- A Code sai; tích của các số dương không thể bằng 0
- B Tích quá nhỏ nên tràn dưới thành 0; cộng log thay vì nhân
- C Do 0.6 không biểu diễn chính xác được; đổi sang 0.5
- D Do NumPy giới hạn 1000 phần tử; chia thành hai lần nhân

CÂU 8

`np.exp(np.array([800.0, 801.0]))` cho `[inf, inf]`. Muốn tính softmax của hai logits này mà không bị nan thì làm thế nào?

- A Chia cả hai logits cho 800 trước khi lấy mũ
- B Dùng `np.exp2` thay cho `np.exp`
- C Làm tròn logits về số nguyên rồi lấy mũ
- D Trừ cả hai cho 801 trước khi lấy mũ

CÂU 9

Model A và model B cùng có accuracy 0.80 trên tập validation. Log-loss của A là 0.45, của B là 0.95. Vì sao có chênh lệch ấy?

- A B có lỗi trong code vì cùng accuracy thì log-loss phải bằng nhau
- B A đoán đúng nhiều câu hơn B nên loss thấp hơn
- C Log-loss lớn hơn nghĩa là model B tốt hơn
- D B gán xác suất rất thấp cho đáp án đúng ở vài câu

CÂU 10

Với logits $[z_1, \dots, z_K]$, log-loss khi lớp đúng là k bằng $\log \sum_j e^{z_j} - z_k$. Tăng riêng z_k lên một chút thì log-loss thay đổi thế nào?

- A Giảm, vì z_k tăng nhanh hơn số hạng log
- B Tăng, vì tổng $\sum_j e^{z_j}$ tăng
- C Không đổi, vì softmax không đổi khi cộng hằng số
- D Giảm về đúng 0 sau một lần tăng

Đọc thêm

- Khan Academy — [Exponential and logarithmic functions](#). Ôn lũy thừa và log từ đầu, nhiều bài tập ngắn có lời giải.
- 3Blue1Brown — [What's so special about Euler's number \$e\$?](#) Video giải thích vì sao e^x là hàm có đạo hàm bằng chính nó, chuẩn bị cho bài 0.6.
- Google — [Machine Learning Crash Course: Logistic regression](#). Sigmoid, log-odds và log-loss trong một bài phân loại thật.
- Goodfellow, Bengio, Courville — [Deep Learning](#), chương 4.1. Đọc phần tràn số và cách tính softmax ổn định.