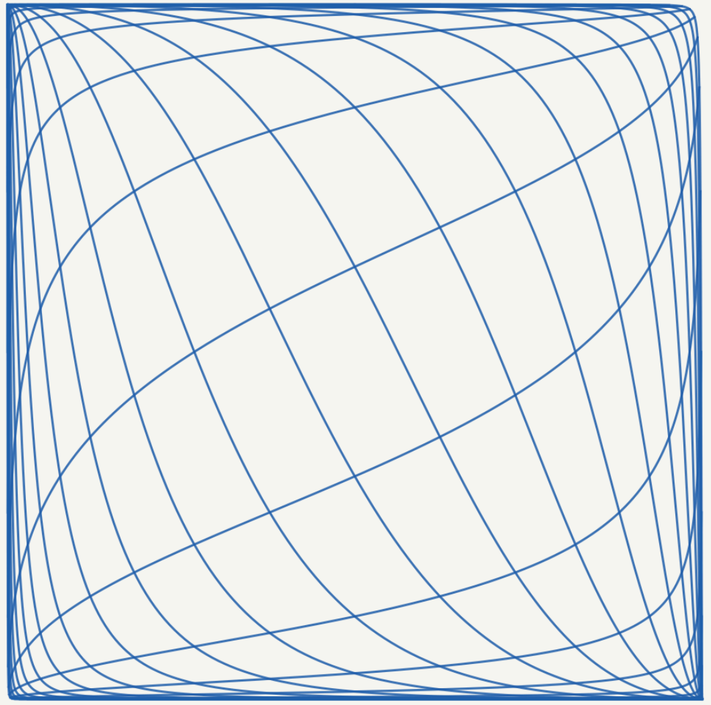
SIENA, NIPS-CERN, UFJF

Limiar

Redes neurais, do neurônio ao fractal

Aula 4: derivadas parciais e o Jacobiano

13/11/2026. C∴ Chrysthofer A. A. Afonso



Uma grade regular do plano, levada pela camada com a tanh do exemplo da aula: as retas viram curvas, que se apertam nas bordas, onde a tanh se achata.

FAPEMIG CNPq Conselho Nacional de Desenvolvimento Científico e Tecnológico CAPES

1 · REVISÃO

O que ficou das aulas 2 e 3, de memória

1 O que uma camada calcula, escrita como matriz?

2 Por que duas camadas lineares valem uma?

3 O que é a derivada parcial?

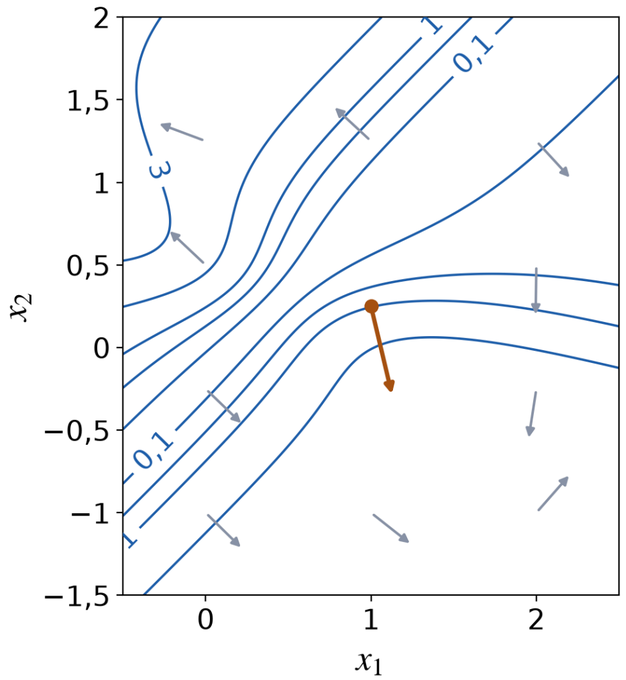
4 Para onde aponta o gradiente?

1 · REVISÃO

O fio do curso, da aula 1 à 10

1	**O neurônio e o perceptron**	um neurônio desenha uma reta, e nenhuma reta separa o XOR	25/09, presencial
2	**Derivada e gradiente**	o passo contra a derivada; a taxa grande demais faz o treino divergir	02/10, presencial
3	**Matrizes e camadas**	duas camadas separam o XOR, com pesos escolhidos à mão	13/11, on-line
4	**O Jacobiano**	a regra da cadeia com vetores, uma camada de cada vez	13/11, on-line
5	**Retropropagação**	a regra da cadeia, de trás para a frente, acha os pesos do XOR	13/11, on-line
6	**O MNIST em NumPy**	um MLP em 70 mil dígitos escritos à mão, com treino, validação e teste	27/11, on-line
7	**PyTorch**	a biblioteca faz a retropropagação; o erro final contra a taxa	27/11, on-line
8	**Os dados do CERN**	um MLP estima a massa dos dois múons, conferida pela fórmula	04/12, on-line
9	**O fractal e o projeto**	a fronteira entre a taxa que converge e a que diverge	04/12, on-line
10	**A apresentação**	a apresentação do projeto e a revisão do curso inteiro	11/12, on-line

Da aula 2: derivada parcial e gradiente



A perda da rede desta aula e o gradiente.

DERIVADA PARCIAL: AS OUTRAS VARIÁVEIS PARADAS

$$\frac{\partial f}{\partial x_j} \approx \frac{f(\ldots, x_j + h, \ldots) - f(\ldots, x_j - h, \ldots)}{2h}$$

GRADIENTE: PARA ONDE A FUNÇÃO MAIS CRESCE

$$\vec{\nabla} f = \left(\frac{\partial f}{\partial x_1}; \frac{\partial f}{\partial x_2} \right)$$

Até aqui, a função devolvia um número só, a perda. Uma camada devolve um vetor.

2 · O Jacobiano

E quando a saída é um vetor?

Uma camada de duas entradas e duas saídas tem quatro derivadas parciais: cada saída depende das duas entradas.

Uma linha por saída, uma coluna por entrada

$$J = \begin{pmatrix} \dfrac{\partial h_1}{\partial x_1} & \dfrac{\partial h_1}{\partial x_2} \\ \dfrac{\partial h_2}{\partial x_1} & \dfrac{\partial h_2}{\partial x_2} \end{pmatrix}$$

$$J_{ij} = \frac{\partial h_i}{\partial x_j}$$

a linha *i* é da saída *i*; a coluna *j*, da entrada *j*

$$\Delta \vec{h} \approx J \, \Delta \vec{x}$$

Perto do ponto, um passo pequeno na entrada muda a saída de J vezes o passo: J é a melhor aproximação linear da função.

Jacobiano, no curso: a matriz $m \times n$ das derivadas parciais de uma função que recebe n números e devolve m. Com $m = 1$, é o gradiente, deitado.

FAPEMIG CNPq Conselho Nacional de Desenvolvimento Científico e Tecnológico CAPES

2 · O JACOBIANO

Numa camada linear, o Jacobiano é W

$$z_k = \sum_j w_{kj}\, x_j + b_k$$

$$\frac{\partial z_k}{\partial x_j} = w_{kj}$$

$$J = W$$

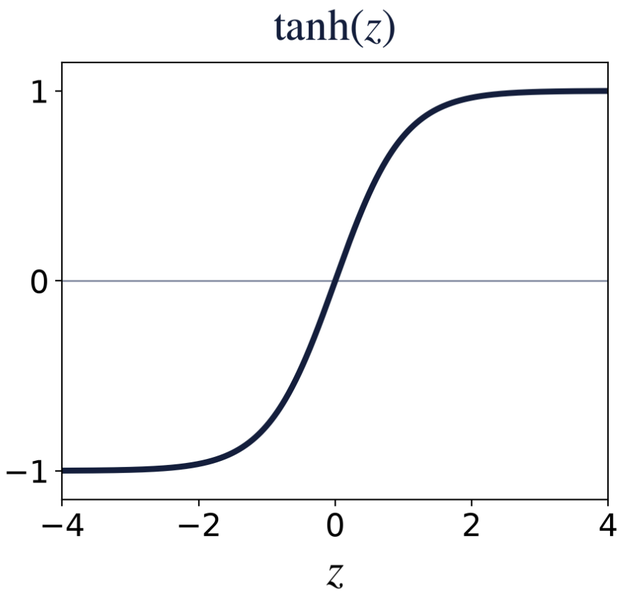
NA AULA 2, A RETA

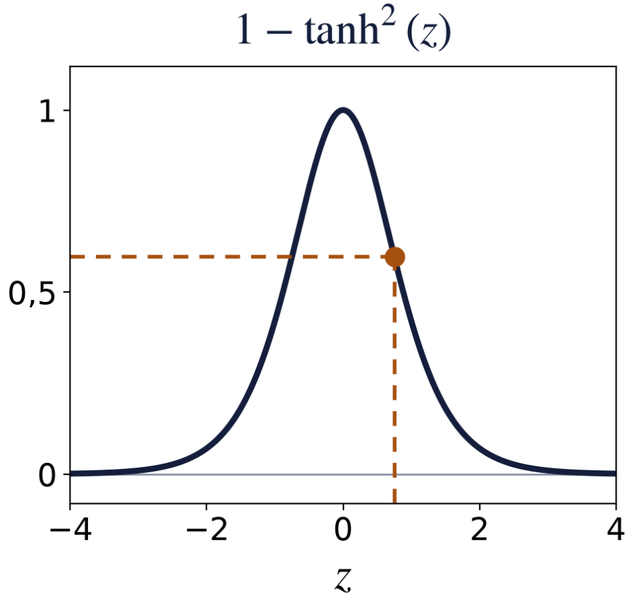
$$y = w\,x + b$$

A derivada em relação a *x* é *w*, em qualquer ponto. Uma função linear tem a mesma inclinação em todo lugar.

Camada linear, no curso: a camada sem ativação, só a soma ponderada mais o viés.

A tangente hiperbólica e a inclinação dela





$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

tanh, a tangente hiperbólica

$$\tanh'(z) = 1 - \tanh^{2}(z)$$

$$1 - \tanh^{2}(0{,}75) = 0{,}5966$$

Caixa-preta: entra z, sai a inclinação. A conta pede a derivada da exponencial e não se abre agora; o código a confere pela derivada numérica.

O Jacobiano de uma camada com tanh

$$h_k = \tanh(z_k)$$

$$z_k = \sum_j w_{kj}\, x_j + b_k$$

$$\frac{\partial h_k}{\partial x_j} = (1 - h_k^2)\, w_{kj}$$

A linha k de J é a linha k de W vezes um número só, a inclinação da tanh no neurônio k. Muda de ponto para ponto.

UM EXEMPLO

$$W = \begin{pmatrix} 1 & -1 \\ 0{,}5 & 2 \end{pmatrix}$$

$$\vec{z} = W\vec{x} + \vec{b} = \begin{pmatrix} 0{,}75 \\ 0 \end{pmatrix}$$

$$\vec{h} = \tanh(\vec{z}) = \begin{pmatrix} 0{,}6351 \\ 0 \end{pmatrix}$$

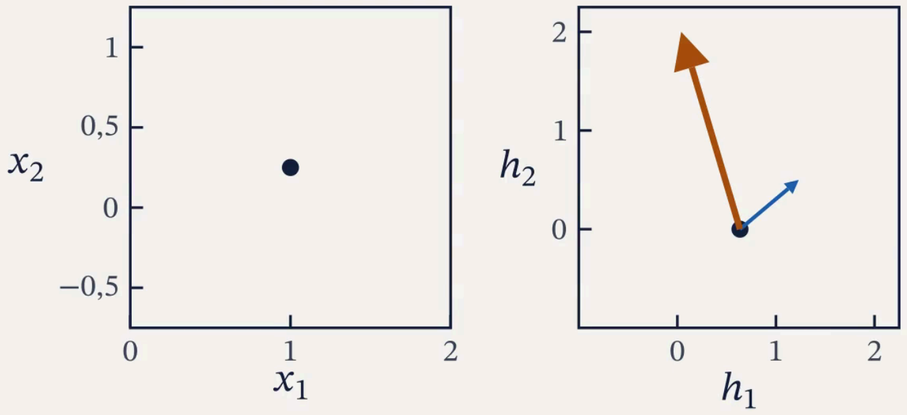
$$J = \begin{pmatrix} 0{,}5966 & -0{,}5966 \\ 0{,}5 & 2 \end{pmatrix}$$

com o vetor b = (0; −1) e o ponto x = (1; 0,25); a derivada numérica dá o mesmo J, com diferença de $7{,}0 \cdot 10^{-11}$

O Jacobiano coluna a coluna

As duas colunas, lado a lado, são o Jacobiano

$$\vec{h} = \tanh(W\vec{x} + \vec{b})$$



coluna 1 ≈ (0,5966; 0,5000)

coluna 2 ≈ (−0,5966; 2,0000)

$$J = \begin{pmatrix} 0{,}5966 & -0{,}5966 \\ 0{,}5000 & 2{,}0000 \end{pmatrix}$$

a conta à mão dá o mesmo

O passo encolhe, e a coluna chega à conta

Δ	coluna 1	coluna 2
0,5	(0,6034; 0,4898)	(−0,6034; 1,5232)
0,25	(0,5990; 0,4974)	(−0,5990; 1,8485)
0,05	(0,5967; 0,4999)	(−0,5967; 1,9934)
0,001	(0,5966; 0,5000)	(−0,5966; 2,0000)
à mão	(0,5966; 0,5000)	(−0,5966; 2,0000)

A COLUNA J, PELA DIFERENÇA CENTRAL

$$\frac{\vec{h}(\vec{x} + \Delta\vec{e}_j) - \vec{h}(\vec{x} - \Delta\vec{e}_j)}{2\Delta}$$

o vetor e de índice *j*: um passo só na entrada *j*

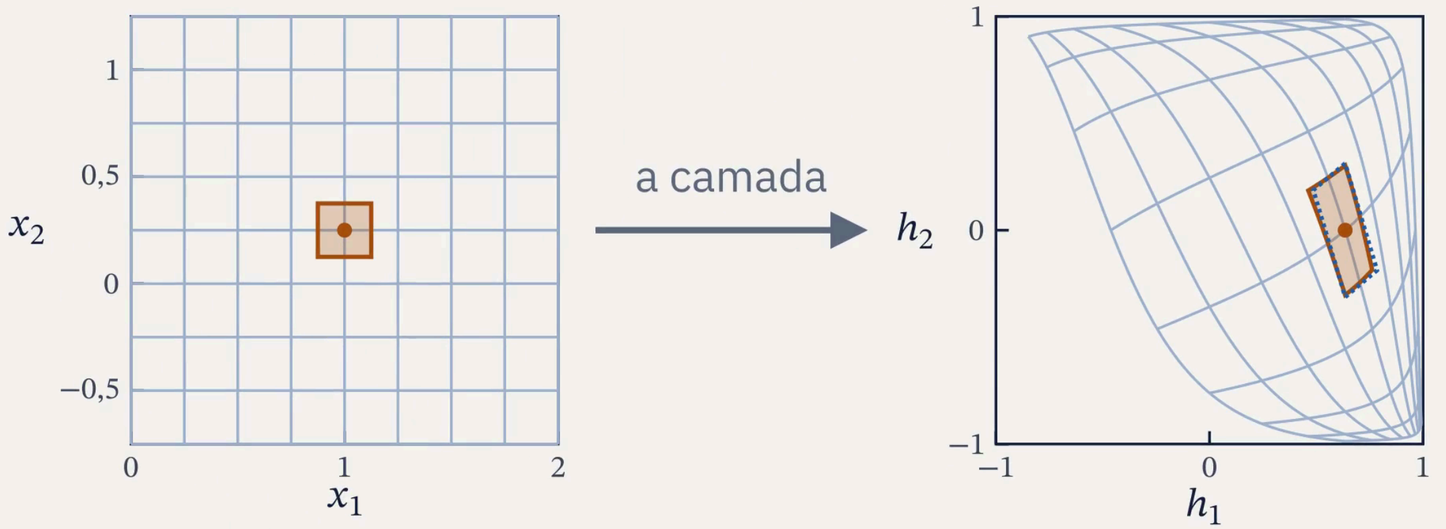
No código, $\Delta = 10^{-6}$, e a diferença para a conta à mão é $7 \cdot 10^{-11}$.

Perto do ponto, a camada age como J

o quadradinho e a imagem dele o quadradinho levado por J

$$\vec{h} = \tanh(W\vec{x} + \vec{b}), \quad W = \begin{pmatrix} 1 & -1 \\ 0{,}5 & 2 \end{pmatrix}, \quad \vec{b} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$$

$$J = \begin{pmatrix} 0{,}60 & -0{,}60 \\ 0{,}50 & 2{,}00 \end{pmatrix}$$



3 · A REGRA DA CADEIA COM VETORES

O Jacobiano da composição é o produto

$$\vec{x} \xrightarrow{f} \vec{u} \xrightarrow{g} \vec{y}$$

$$J_{g\circ f}(\vec{x}) = J_g\big(f(\vec{x})\big)\, J_f(\vec{x})$$

∘, bola: $g \circ f$ é f e depois g

POR QUÊ

$$\Delta\vec{u} \approx J_f\, \Delta\vec{x}$$

$$\Delta\vec{y} \approx J_g\, \Delta\vec{u} \approx (J_g\, J_f)\, \Delta\vec{x}$$

$$\underbrace{J_g}_{m\times k}\ \underbrace{J_f}_{k\times n} = \underbrace{J_{g\circ f}}_{m\times n}$$

NO EXEMPLO, DUAS CAMADAS

$$J_2 = \begin{pmatrix} 0{,}5418 & 0{,}2709 \\ -0{,}9820 & 0{,}9820 \end{pmatrix}$$

$$J_2\, J_1 = \begin{pmatrix} 0{,}4587 & 0{,}2186 \\ -0{,}0948 & 2{,}5497 \end{pmatrix}$$

o Jacobiano numérico das duas camadas dá o mesmo, com diferença de $1{,}0 \cdot 10^{-10}$

FAPEMIG CNPq Conselho Nacional de Desenvolvimento Científico e Tecnológico CAPES

3 · A REGRA DA CADEIA COM VETORES

Até a perda, o gradiente é uma linha

$$\vec{x} \xrightarrow{J_1} \vec{h}_1 \xrightarrow{J_2} \vec{h}_2 \xrightarrow{\vec{v}} y \xrightarrow{2(y-t)} L$$

$$\vec{\nabla}_{\vec{x}} L = \underbrace{2\,(y-t)}_{1\times 1}\ \underbrace{\vec{v}}_{1\times 2}\ \underbrace{J_2}_{2\times 2}\ \underbrace{J_1}_{2\times 2}$$

o gradiente de *L* em relação ao vetor x: o produto dá 1 × 2, uma linha

A SAÍDA E A PERDA

$$y = \vec{v} \cdot \vec{h}_2 + c$$

$$L = (y - t)^2$$

com o vetor v = (1; −1), $c = 0$ e $t = 0$

OS NÚMEROS

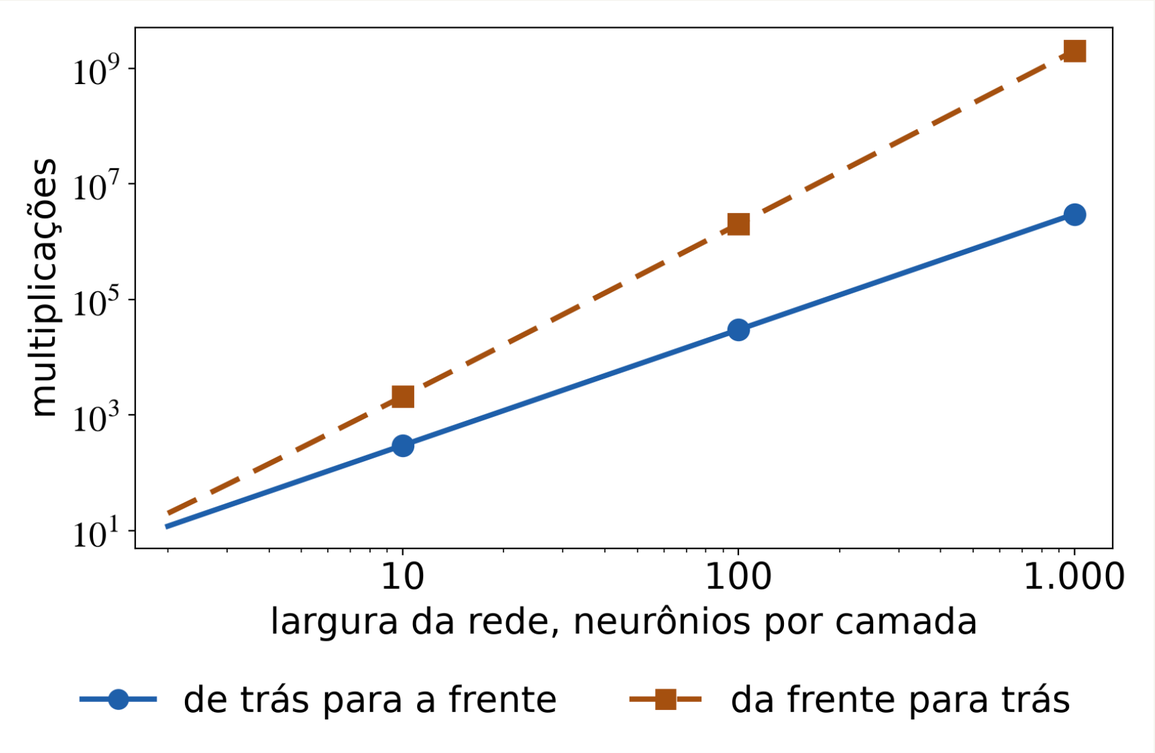
$$L = 0{,}9766$$

$$\vec{\nabla}_{\vec{x}} L = (1{,}0940;\ -4{,}6074)$$

pela cadeia e pela derivada numérica, com diferença de $1{,}4 \cdot 10^{-10}$

FAPEMIG CNPq Conselho Nacional de Desenvolvimento Científico e Tecnológico CAPES

De trás para a frente custa menos



DE TRÁS PARA A FRENTE

$$\left((\vec{g}\,J_3)\,J_2\right)J_1$$

linha vezes matriz $3m^2$

DA FRENTE PARA TRÁS

$$\vec{g}\left(J_3\left(J_2\,J_1\right)\right)$$

matriz vezes matriz $2m^3 + m^2$

3 camadas de $m = 1.000$: 3 milhões de multiplicações contra 2 bilhões.

As duas ordens, conta a conta

Com 100 neurônios por camada: 30 mil contas contra 2 milhões

de trás para a frente



o gradiente

48
multiplicações

da frente para trás



o gradiente

144
multiplicações

3 · A regra da cadeia com vetores

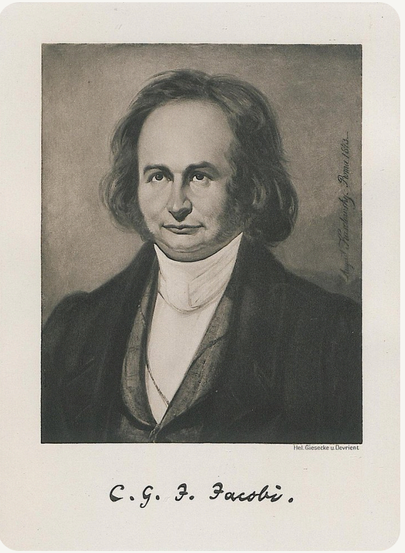
A retropropagação é esse produto, de trás para a frente

Uma linha vezes uma matriz, camada por camada,
começando pela perda: a conta da aula 5.

4 · DE ONDE VEM

Da regra da cadeia ao gradiente barato

1676	A regra da cadeia aparece numa memória de Leibniz, ao que parece pela primeira vez.
1841	Jacobi publica “De determinantibus functionalibus”, sobre o determinante da matriz das derivadas parciais, que hoje leva o nome dele.
1970	Seppo Linnainmaa descreve, na dissertação de mestrado, o modo reverso da diferenciação automática: a conta de trás para a frente.
1983	Baur e Strassen mostram que o gradiente custa um múltiplo pequeno do custo da própria função.



Jacobi em 1843, retrato de August Kaselowsky. Wikimedia Commons, domínio público.

Fontes: Wikipedia, verbetes “Chain rule”, “Jacobian matrix and determinant” e “Automatic differentiation”; Andreas Griewank, 2012, “Who invented the reverse mode of differentiation?”.

5 · O CÓDIGO

O Jacobiano em NumPy, passo a passo

1. A camada com a tangente hiperbólica, com os pesos e o ponto do exemplo
2. O Jacobiano à mão: cada linha de W vezes a inclinação da tanh
3. O Jacobiano numérico: uma coluna por entrada, mexendo só ela
4. As duas matrizes comparadas, e o produto $J_2\,J_1$ contra o numérico das duas camadas
5. As multiplicações contadas nas duas ordens

O NUMPY QUE RESOLVE

```
np.tanh(W @ x + b)
(1 - h**2)[:, None] * W
np.column_stack(colunas)
linha @ J
```

[:, None] faz do vetor uma coluna, e cada linha de W vai vezes o seu número

5 · O CÓDIGO

O que a entrega pede

À MÃO

$$J = \begin{pmatrix} 0{,}596586 & -0{,}596586 \\ 0{,}500000 & 2{,}000000 \end{pmatrix}$$

NUMÉRICO

$$J \approx \begin{pmatrix} 0{,}596586 & -0{,}596586 \\ 0{,}500000 & 2{,}000000 \end{pmatrix}$$

$$\text{maior diferença} = 7{,}0 \cdot 10^{-11}$$

Uma função que calcula o Jacobiano numérico de uma camada e o compara com o calculado à mão. Cada um escolhe a própria camada e o próprio ponto.

6 · FECHAMENTO

Para estudar e entregar

Estudo

- 3Blue1Brown, redes neurais, capítulo 3
- 3Blue1Brown, redes neurais, capítulo 4
- Nielsen, capítulo 2, “How the backpropagation algorithm works”

Entrega

- o Jacobiano numérico de uma camada, comparado com o calculado à mão
- uma pergunta sobre o estudo

A seguir

- aula 5: a retropropagação acha os pesos do XOR
- aula 6: o MNIST em NumPy, com treino, validação e teste