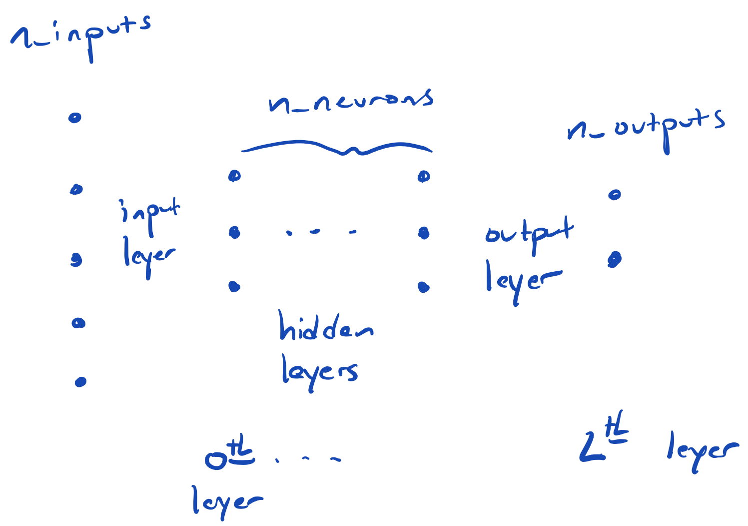
Network Convention



① input layer

$$\vec{a}^1 = f(\vec{z}^1) \quad \text{where} \quad \vec{z}^1 = w^1 \vec{x} + \vec{b}^1$$

- $a_i^1 = f(z_i^1)$
- $\vec{a}, \vec{z}$ ~ column vectors $\Rightarrow$ $\vec{x}, \vec{b}$ ~ column vectors
- w^1 ~ n_neurons x n_inputs

② hidden layers : $\vec{a}^l = f(\vec{z}^l = w^l \vec{a}^{l-1} + \vec{b}^{l-1})$

$\vec{a}^l, \vec{z}^l$ ~ (n_neurons x 1)

w^l ~ (n_neurons x n_neurons)

③ output layer :

$\vec{y} = \vec{a}^L = f(\vec{z}^L)$

$\vec{a}^L, \vec{z}^L$ ~ (n_outputs x 1)

w^L ~ (n_outputs x n_neurons)

Some Matrix Derivative Notes (Convention)

$$\vec{y} \sim n,\ \vec{x} \sim m \Rightarrow \frac{\partial \vec{y}}{\partial \vec{x}} \sim \begin{pmatrix} \partial y_1/\partial x_1 & \cdots & \partial y_1/\partial x_m \\ \vdots & & \\ \partial y_n/\partial x_1 & \cdots & \partial y_n/\partial x_m \end{pmatrix} \sim n \times m \sim \vec{y}\,\vec{x}^T$$

General: ① If $a \sim$ scalar, $X \sim n\times m$, $\frac{\partial a}{\partial X} \sim m \times n$

② If $\vec{z} \sim n\times 1$, $w \sim n \times m$, $\partial \vec{z}/\partial w \sim m \times 1$ ✓

$$\frac{\partial z_i}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}} \sum_b w_{ib} x_b = \sum_b x_b \delta_{ij} \delta_{kb} = x_k \delta_{ij}$$

Backpropagation

Given a loss function $\mathcal{L}$, determine gradients with respect to all w^i & b^i. Then use

$C = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i \sim$ sample average

Denote $\mathcal{L} = \mathcal{L}(\vec{y}, \vec{t})$ where $\vec{t}$ is the target data.

L^{th} layer

$$\vec{z}^L = w^L a^{L-1} + \vec{b}^L$$

$$\frac{\partial \mathcal{L}}{\partial w^L} = \frac{\partial \mathcal{L}}{\partial \vec{y}} \frac{\partial \vec{y}}{\partial \vec{z}^L} \frac{\partial \vec{z}^L}{\partial w^L} \quad ; \quad \frac{\partial \vec{z}^L}{\partial w^L} \sim \text{n\_outputs} / (\text{n\_outputs} \times \text{n\_neurons}) \sim (\text{n\_neurons} \times 1)$$

$$= \frac{\partial \mathcal{L}}{\partial \vec{y}} f'(\vec{z}^L)\, \vec{a}^{L-1}$$

$\llcorner$ (n_nuerons x 1)

$$\frac{\partial y_i}{\partial z_j} = \partial y_i/\partial z_i \, \delta_{ij} = \begin{pmatrix} \ddots & & 0 \\ & \ddots & \\ 0 & & \ddots \end{pmatrix} \longrightarrow \begin{pmatrix} \partial y_1/\partial z_1 \\ \vdots \\ \partial y_n/\partial z_n \end{pmatrix}^T \sim (1, n\_outputs)$$

$$\frac{\partial \mathcal{L}}{\partial \vec{y}} \sim (1, n\_outputs)$$

$$\frac{\partial \mathcal{L}}{\partial w^L} \sim (w^L)^T \sim (n\_neurons \times n\_output) = (1 \times n\_outputs) \cdot (n\_neurons \times 1)$$

$$= n\_neurons \times n\_outputs$$

$$\Longrightarrow \frac{\partial \mathcal{L}}{\partial w^L} = \left(\frac{\partial \mathcal{L}}{\partial \vec{y}} \circ f'(\vec{z}^L) \right) \cdot \vec{a}^{L-1}$$

matrix row vector × column vector → needs column vector × row vector

$$\underline{\text{Define}} \quad (\delta^L)^T = \frac{\partial \mathcal{L}}{\partial \vec{y}} \circ f'(\vec{z}^L)$$

Hademerd

- $(\delta^L) \sim (n\_outputs \times 1)$
- once we flip to row vectors, this will make senses

<u>Note in Python</u>

$\delta^L \sim (1 \times n\_outputs)$

$w_L \sim (n\_output \times n\_neurons)$

$$\frac{\partial \mathcal{L}}{\partial w^L} = \vec{a}^{L-1} (\delta^L)^T \sim (n\_neurons \times n\_output)$$

$(n\_neurons \times 1) \times (1 \times n\_outputs)$
$= (n\_neurons \times n\_outputs)$

$$\frac{\partial \mathcal{L}}{\partial \vec{b}^L} = (\delta^L)^T$$

<u>Hidden-Layers</u>

$n\_output \times n\_neurons$

$\cancel{(\delta^L)^T \vec{a}^{L-1}}$

$$\frac{\partial \mathcal{L}}{\partial w^{L-1}} = \underline{\frac{\partial \mathcal{L}}{\partial \vec{y}} \frac{\partial \vec{y}}{\partial \vec{z}^L}} \frac{\partial \vec{z}^L}{\partial \vec{a}^{L-1}} \frac{\partial \vec{a}^{L-1}}{\partial \vec{z}^{L-1}} \frac{\partial \vec{z}^{L-1}}{\partial w^{L-1}}$$

$$= (\delta^L)^T (w^L) \circ f'(\vec{z}^{L-1}) (\vec{a}^{L-2})$$

$$\begin{array}{c}n\_neurons \\ \times \\ n\_neurons\end{array} = (1 \times n\_outputs)(n\_outputs \times n\_neurons)(1 \times n\_neurons)(n\_neurons \times 1)$$

$$= (1 \times n\_neurons) \odot (1 \times n\_neurons) \times (n\_neurons \times 1)$$

$$= (n\_neurons \times 1) \times [(1 \times n\_neurons) \odot (1 \times n\_neurons)]$$

$$= (n\_neurons) \times (n\_neurons)$$

$$\frac{\partial \mathcal{L}}{\partial w^{L-1}} = \vec{a}^{L-2}\left((\delta^{L})^{T} w^{L} \odot f'(z^{L-1})\right)$$

$$= \vec{a}^{L-2}(\delta^{L-1})^{T}$$

$$\boxed{\frac{\partial \mathcal{L}}{\partial w^{L-1}} = (\delta^{L-1})^{T}}$$

$$\boxed{(\delta^{l})^{T} = (\delta^{l+1})^{T} w^{l+1} \odot f'(z^{l})}$$

$$\Downarrow$$

$$\frac{\partial \mathcal{L}}{\partial w^{l}} = \vec{a}^{l-1}(\delta^{l})^{T}$$

$$\frac{\partial \mathcal{L}}{\partial b^{l}} = (\delta^{l})^{T}$$

Note

$$\frac{\partial \mathcal{L}}{\partial b^{l}} \sim (1 \times n\_neurons) \sim (\delta^{l})^{T}$$

Using Coding Conventions

Typical vectors are row vectors

this ensures dimensions for products still work

$\Longrightarrow$ transpose everything! (And keep orders!)

Note: This is not globally consistent: $\delta^L \sim$ row vector, $b^L \sim$ row vector

but $\frac{\partial \mathcal{L}}{\partial b^L} \sim$ column

But, it works for the computations

$$\boxed{\begin{aligned}
&\frac{\partial \mathcal{L}}{\partial w^L} = (a^{L-1})^T \delta^L \quad \& \quad \frac{\partial \mathcal{L}}{\partial b^L} = \delta^L \\
&\frac{\partial \mathcal{L}}{\partial w^\ell} = (a^{\ell-1})^T \delta^\ell \quad \& \quad \frac{\partial \mathcal{L}}{\partial b^\ell} = \delta^\ell \\
&\delta^L = \frac{\partial \mathcal{L}}{\partial \vec{y}^L} \odot f'(\vec{z}^L) \\
&\delta^\ell = f'(\vec{z}^\ell) \odot \left(\delta^{\ell+1} (w^{\ell+1})^T\right)
\end{aligned}}$$

Check: $z^1 = \vec{x}_{1 \times n\_inputs}\, w^1_{n\_inputs \times n\_neurons} + \vec{b}^1_{1 \times n\_inputs}$

$L-1$: $\delta^L \sim (1 \times n\_outputs)$

$w^L \sim (n\_neurons \times n\_outputs)$

$$\delta^{L-1} = f'(\vec{z}^{L-1}) \odot \left(\delta^L (w^L)^T\right)$$

$$(1 \times n\_neurons) \odot \left[(1 \times n\_outputs) \times (n\_outputs \times n\_neurons)\right]$$

$$= (1 \times n\_neurons) \checkmark$$

$$(\delta^{l})^{T} = (\delta^{l+1})^{T} w^{l+1} \odot f'(z^{l})$$

$$\downarrow$$

$$\delta^{l} = \left[(\delta^{l+1})^{T} w^{l+1} \odot f'(z^{l})\right]^{T} =$$

Adding in Python data

- For training, we would like to feed forward all the training data simultaneously ~ vectorization.

features

$$\vec{x} \longrightarrow X \sim (\text{n\_samples}, \text{n\_inputs})$$

$$\vec{z}^{\,1}_{\text{n\_samples} \times \text{n\_neurons}} = \overleftrightarrow{X}_{\text{n\_samples} \times \text{n\_inputs}} \times w_{\text{n\_inputs} \times \text{n\_neurons}} + \vec{b}_{1 \times \text{n\_neurons}}$$

Python adds $\vec{b}$ to each row of Xw

Using Softmax Output Activation

If we change the output activation to a softmax,

$$y_i = a_i^L = s(z_i^L) = \frac{e^{z_i^L}}{\sum_k e^{z_k^L}}$$

we only affect δ_L. Furthermore, let's take our NN to do max likelihood on the log likelihood. See "Softmax ..." note for details

<u>Note</u>: $\mathcal{L} = -\log$ likelihood

now a non-diagonal matrix

$$\frac{\partial \mathcal{L}}{\partial w^L} = -\frac{\partial \mathcal{L}}{\partial \vec{y}} \frac{\partial \vec{y}}{\partial \vec{z}} \frac{\partial \vec{z}}{\partial w^L}$$

$$= -\sum_{i=1}^{N} (\vec{a}^{L-1}(x^{(i)}))^T (\vec{y}^{(i)} - s(\vec{z}_s^L ; x^{(i)}))$$

$$\frac{\partial C}{\partial w^L} = -(\vec{a}^{L-1})^T (\vec{y} - s(\vec{z}_s^L))$$

$$= (\vec{a}^{L-1})^T (s(\vec{z}_s^L) - \vec{y})$$

$$= (\vec{a}^{L-1})^T \delta^L \quad \text{where} \quad \delta^L = s(\vec{z}_s^L) - \vec{y}$$

hidden

$\vec{a}^{L-1} \sim$ (n_samples x n_neurons)

$\vec{Y} \sim$ (n_samples x n_classes)

$S(\vec{z}_s^{\,L}) \sim$ (n_samples, n_classes)

$\frac{\partial C}{\partial w^L} \sim$ (n_neurons x n_samples) x (n_samples, n_classes)

= (n_neurons x n_classes)

(n_neuron × n_output)

$\omega^L \sim$ (n_output , n_neuron)

$$\frac{\partial L}{\partial w^L} = \frac{\partial L}{\partial \vec{y}^L}\underline{\frac{\partial \vec{y}^L}{\partial \vec{z}}\frac{\partial \vec{z}}{\partial w^L}} - \frac{\text{n\_output}}{\text{n\_output} \times \text{n\_neuron}} \sim (1 \times \text{n\_neuron})$$

n-row

n_output × n_out

$$\partial w_{ij}\left(z^L_l = \sum_k w_{lk} a_k^{L-1} \right) = \sum_k a_k^{L-1} \delta^i{}_l \delta^j{}_k$$

$$= \underline{a_j^{L-1} \delta^{il}}$$

$$\sum_{i,j} \frac{\partial L}{\partial y_i} \frac{\partial y_i}{\partial z_j} \frac{\partial z_j}{\partial w_{ab}} = \sum_{ij} \frac{\partial L}{\partial y_i} \frac{\partial y_i}{\partial z_j} a_b^{L-1} \delta^{aj}$$

$$\frac{\partial L}{\partial w_{ab}} = \sum_i \frac{\partial L}{\partial y_i} \frac{\partial y_i}{\partial z_a} a_b^{L-1}$$

n×n n

row colum row

Matrix colum vector

row

column

row vector

$$\vec{z}^{1} = X\omega + b$$

$(1 \times n\text{-neurons})$ $\quad$ $(1 \times n\text{-input})(n\text{-input} \times n\text{-neurons})$

$(N \times n\text{-neurons}) = (N \times n\text{-input})(n\text{-input} \times n\text{-neurons})$

$(N \times n\text{-neurons})^T$

$$\frac{\partial f}{\partial \omega^L} = (a^{L-1})^T \delta^L \quad \& \quad \frac{\partial f}{\partial b^L} = \delta^L$$

$$\frac{\partial f}{\partial \omega^l} = (a^{l-1})^T \delta^l \quad \& \quad \frac{\partial f}{\partial b^l} = \delta^l$$

$(N \times n\text{-outputs})$

$(n\text{-outputs} \times N)$

$$\delta^L = \frac{\partial f}{\partial \vec{y}^L} \circ f'(\vec{z}^L)$$

$$\delta^l = f'(\vec{z}^l) \circ \left(\delta^{l+1} (\omega^{l+1})^T\right)$$

$(\delta^L)^T =$ row-vector

$\downarrow$ transpose

$\delta^L =$

$(a^{L-1})(\delta^L)^T$

$\downarrow$

$\boxed{(a^{L-1})^T}\delta^L \qquad (\delta^L)(a^{L-1})^T$

row vector