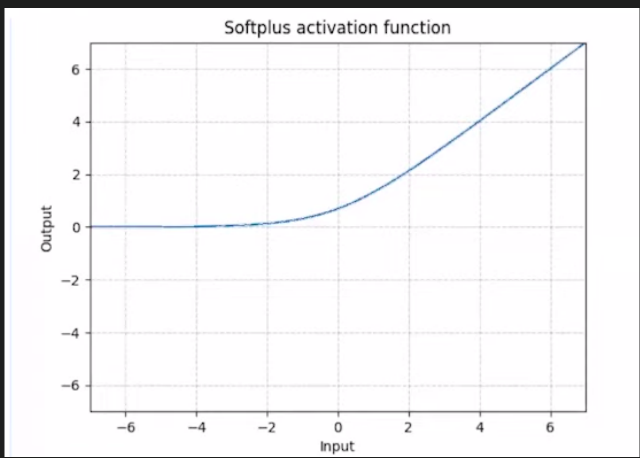
- ReLU
  - ↳ Variations : Leaky, etc.
  - ↳ 1 $\underbrace{\text{kink}}_{\text{esquina, corner}}$: equivariant to scalar product (scale invariant)

- Softplus

  ↙ Large $\beta \approx$ ReLU

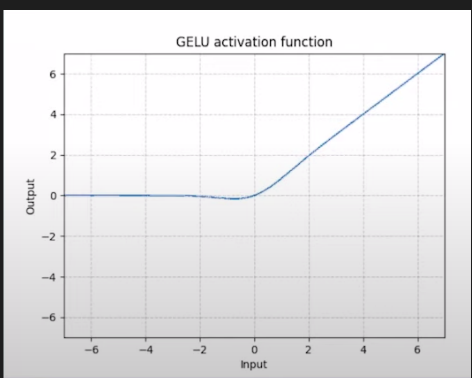
$$\text{Softplus}(x) = \frac{1}{\beta} * \log(1 + \exp(\beta * x))$$



- ELU

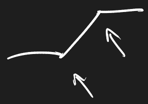
- CELU

- GELU : non-monotonic



$$\text{GELU}(x) = x * \Phi(x)$$

where $\Phi(x)$ is the Cumulative Distribution Function for Gaussian Distribution.

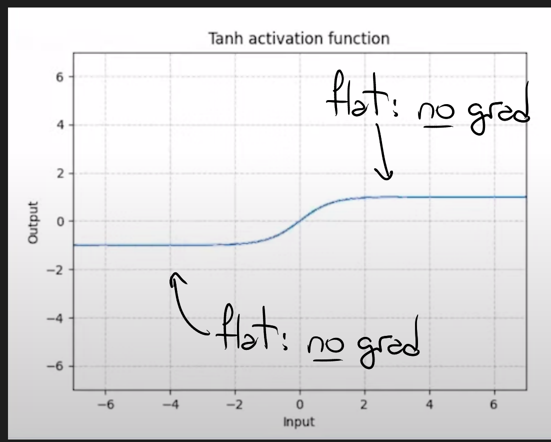
- ReLU 6  2 kinks

- Good ol' Sigmoid

- Tanh :

$$\frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

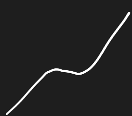
Problem:

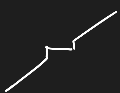


→ ReLUs are better for deeper networks (not clear why)

- Soft sign 

- Hard tanh 

- Tanh shrink 

- Hard shrink 

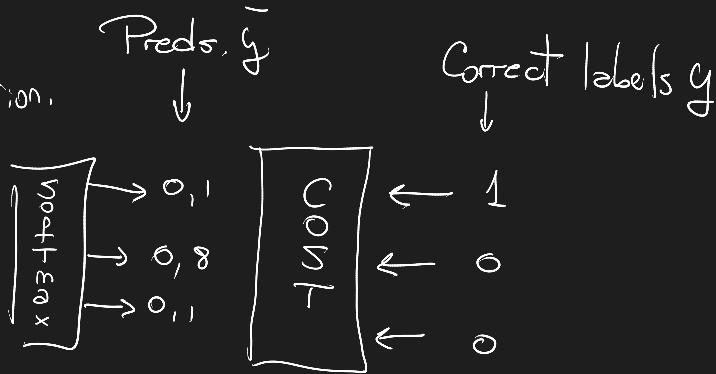
- LogSigmoid: $\log(\text{sigmoid}(x))$

Softmax : "soft argmax"
Softmin : "soft argmin"

} Invariant to translations.

↳ softmax $(-x)$

Cost Function for Classification.



- MSE
- Binary cross entropy
  - ↳ Negative log likelihood (main way)

↳ $-\log(\bar{y}_c)$

$\bar{y}$ of correct category.

Note: the others $\bar{y}_i$ are not used.

↳ We need the log softmax for this.

- L1 norm: not used very often

Architectures

↳ Different ways of arranging modules to build Neural Networks

Multiplicative Modules

- Quadratic modules

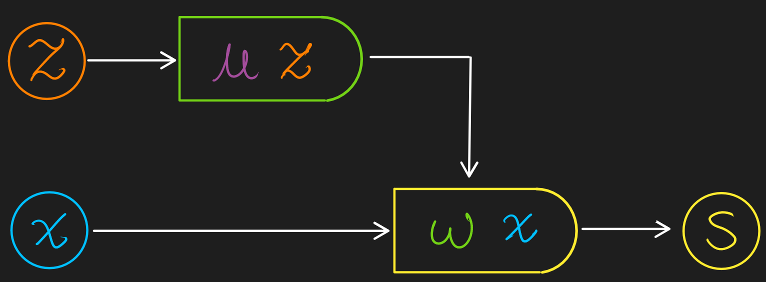
output ... weighted sum of inputs (linear function)

$$s_i = \sum_j w_{ij} x_j$$

with each w_{ij} given by

$$w_{ij} = \sum_k u_{ijk} z_k$$

linear function of inputs z_k



equiv:

if $z_k = x_j \Rightarrow$ "quadratic form"

$$S_i = \sum_{jk} u_{ijk} z_k x_j$$

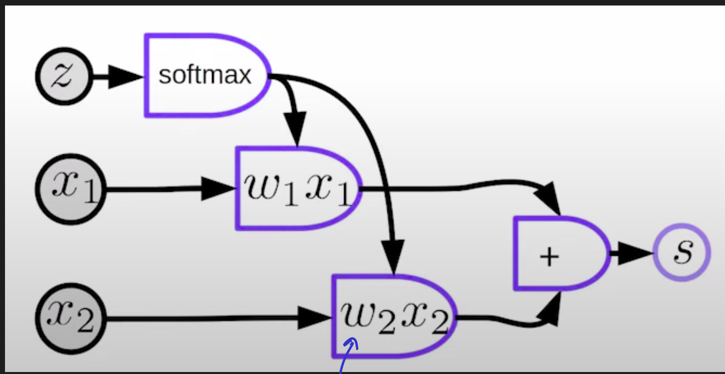
2nd degree monomial.

- Attention module

$$S_i = \sum_j w_j x_{ij}$$

with

$$w_j = \frac{e^{z_j}}{\sum_k e^{z_k}}$$



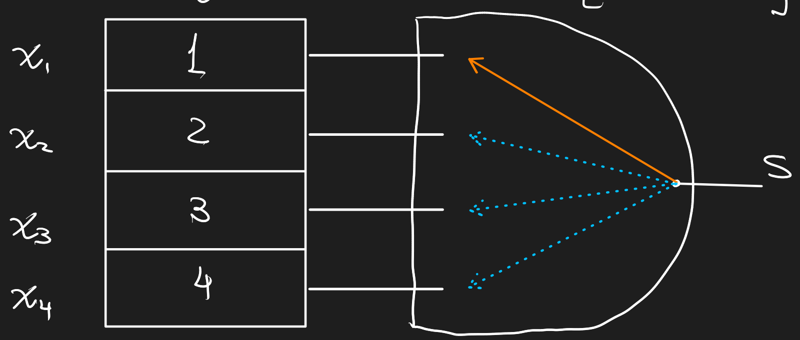
w_i : scalar
x_i : scalar

pays attention to these part of the input

i	w		x	
0	0,7	→	1	← important
1	0	→	2	← non important
2	0,2	→	3	← a little
3	0,1	→	4	← less than a little

If w is 1-hot vector ⇒ it is a switch (a hard switch)

x $\qquad w = [1\ 0\ 0\ 0]$

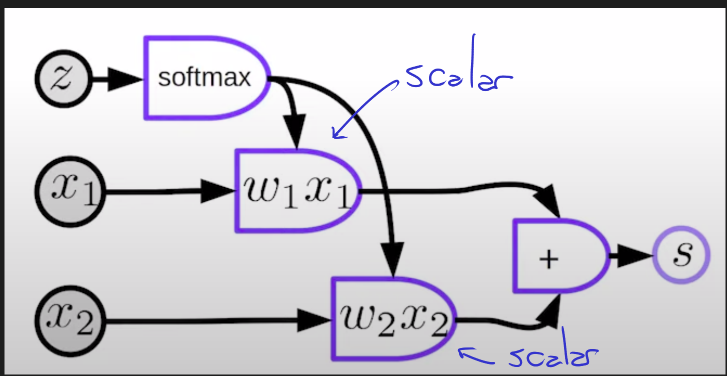


Backprop is trivial for switches.

↳ same gradient from the top goes through the only way possible.

no gradient through the non-selected inputs.

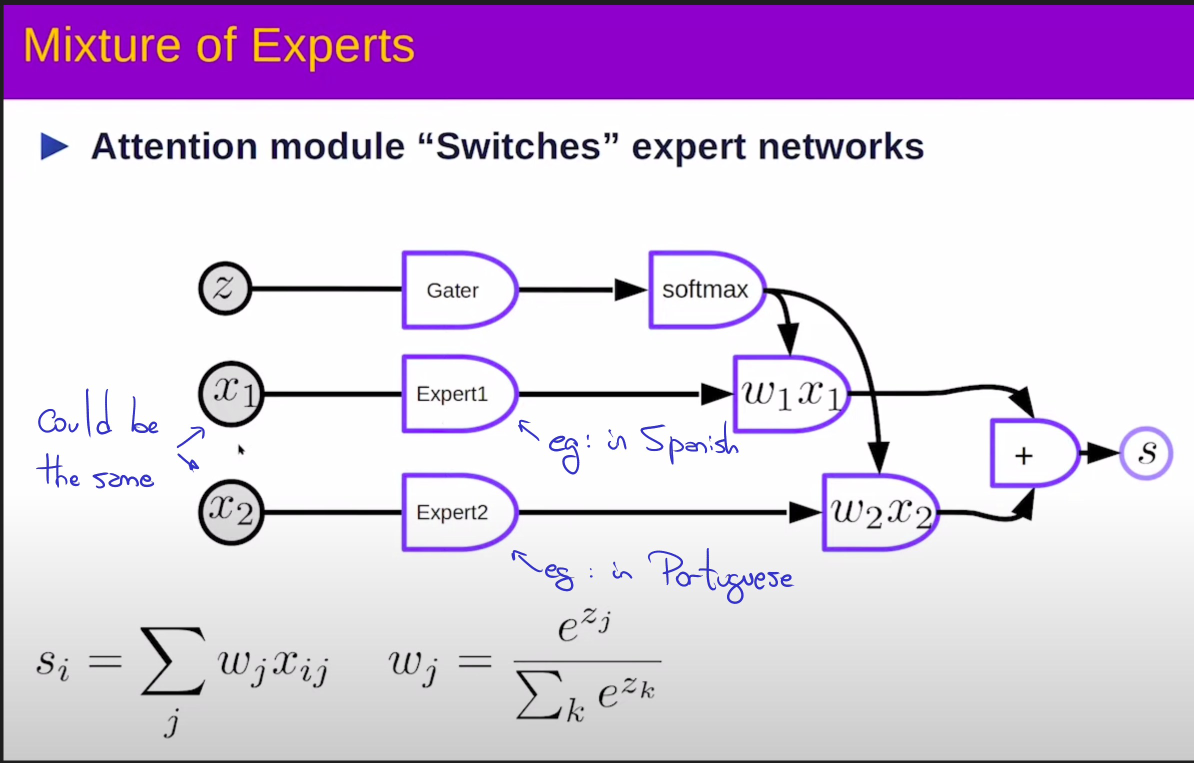
Soft switch



$$S = W_1 \cdot x_1 + W_2 \cdot x_2$$

with $w_1 + w_2 = 1$

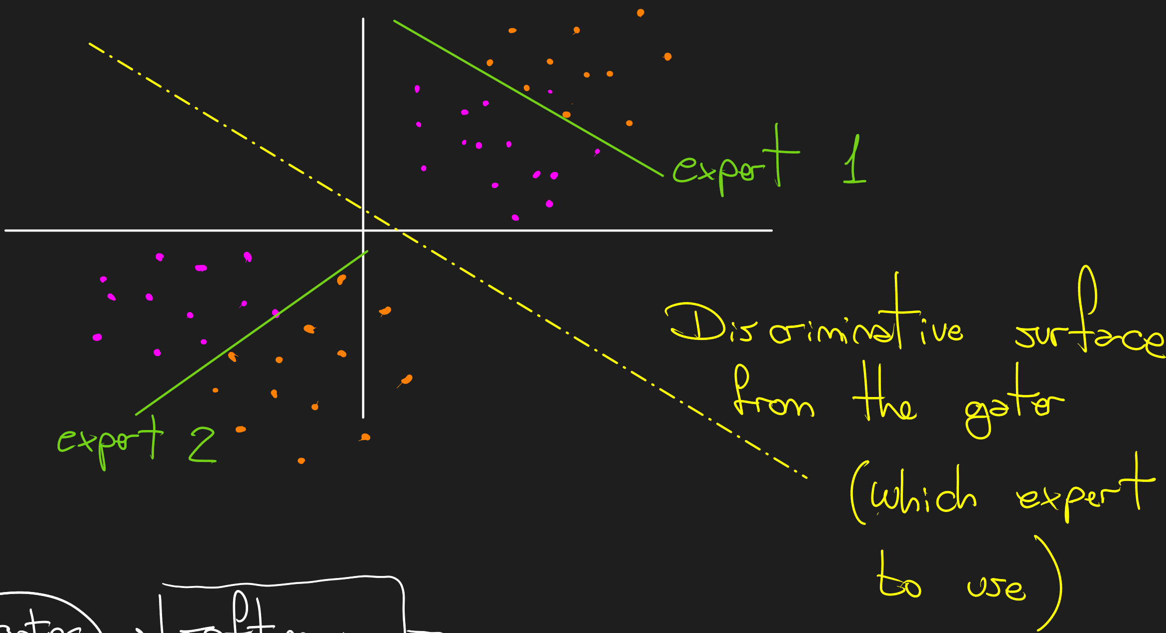
z = Attention the system pays to x_1 and x_2.

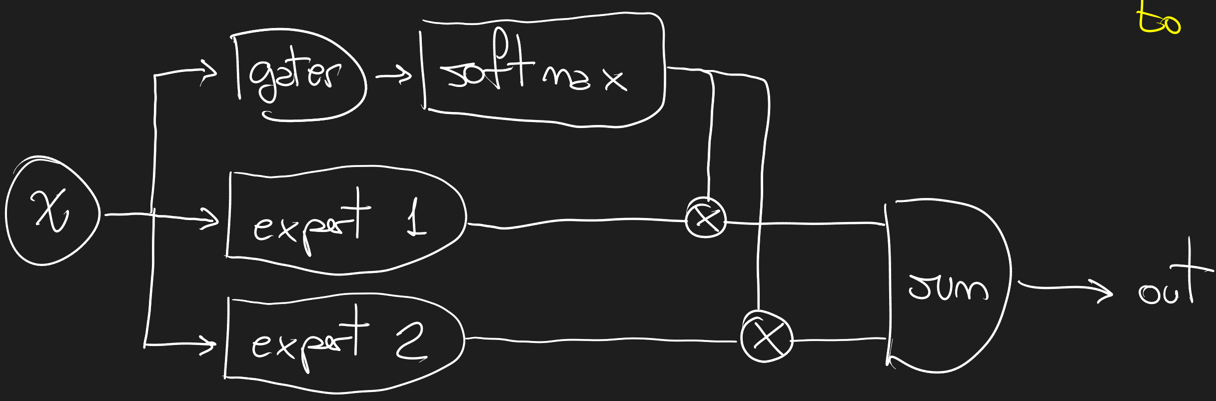


If the input is in some mixed dialect (eg: Catalan), the attention module will switch between experts in a smart way.

Example: Linear classifiers to Nonlinear one

Imagine one of the experts could linearly classify a portion of the input space, and the other expert, classify another portion of the input space.





Training: Simple Backprop.

Generalization.

Parameter transformations

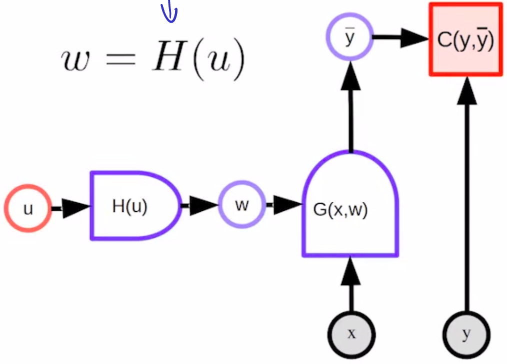
Y. LeCun

- When the parameter vector is the output of a function

neural net

cost function

$$w = H(u)$$



$$u \leftarrow u - \eta \frac{\partial H}{\partial u}^T \frac{\partial C}{\partial w}^T$$

$$w \leftarrow w - \eta \frac{\partial H}{\partial u} \frac{\partial H}{\partial u}^T \frac{\partial C}{\partial w}^T$$

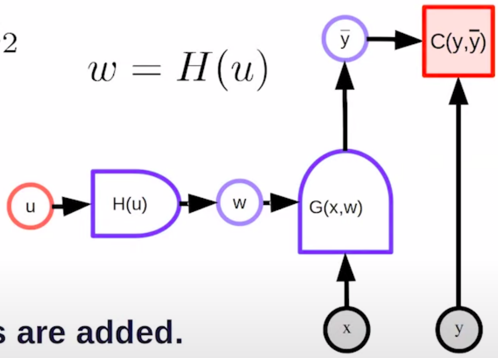
$$[N_w \times N_u]\ [N_u \times N_w]\ [N_w \times 1]$$

Simple parameter transform: weight sharing

Y. LeCun

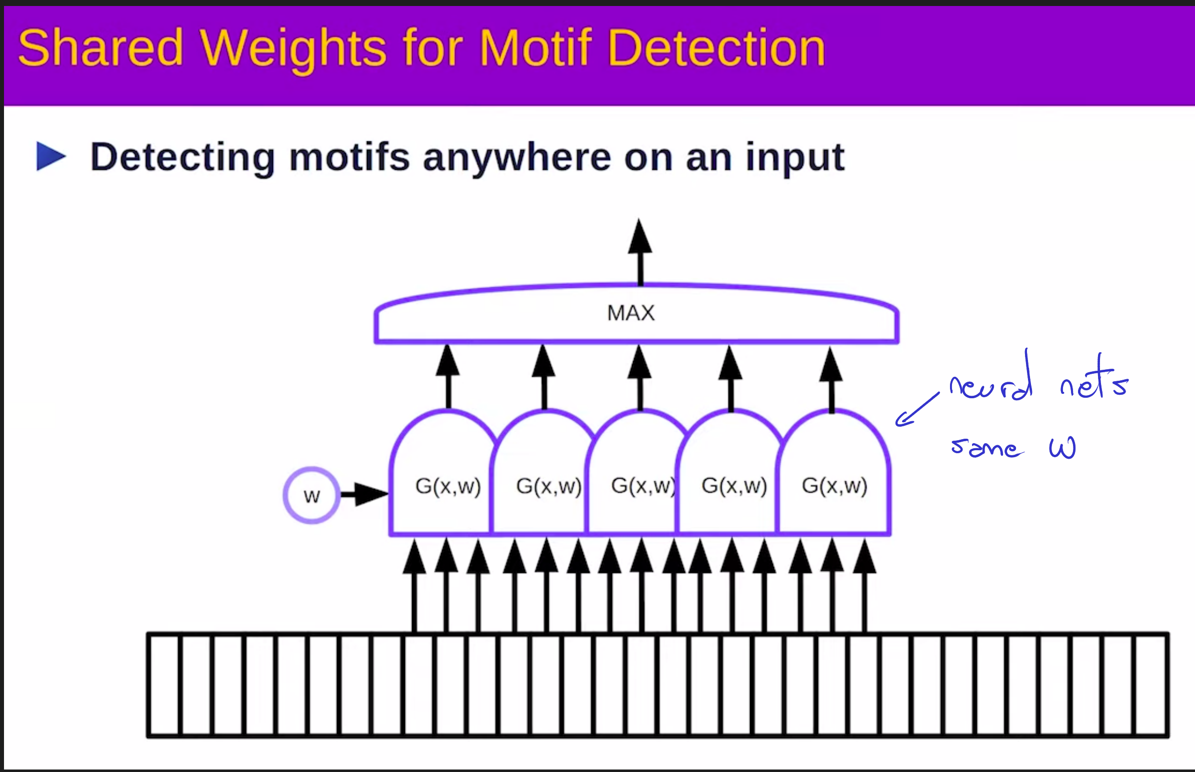
- Function H(u) replicates one component of u into multiple components of w
  - $w_1, = w_2 = u_1 \quad w_3 = w_4 = u_2$
- H is like a “Y” branch.
  - Gradients are summed in the backprop
- The gradients w.r.t. shared parameters are added.

$$w = H(u)$$



Special Case.

"face on image"; "alexa"/"hey google"



neural nets
same w

Exercize:

- Back prop through softmax

$$\sigma_i(x) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

$$\text{Jacobian}(\sigma(x)) = \begin{bmatrix} \frac{\partial \sigma_1}{\partial x_1} & \frac{\partial \sigma_1}{\partial x_2} & \cdots & \frac{\partial \sigma_1}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \sigma_n}{\partial x_1} & \frac{\partial \sigma_n}{\partial x_2} & \cdots & \frac{\partial \sigma_n}{\partial x_n} \end{bmatrix}$$

First, we apply log (all positive values)

$$\log\left(\sigma_i(x)\right) = \log\left(\frac{e^{x_i}}{\sum_j e^{x_j}}\right)$$

$$= \underbrace{\log e^{x_i}}_{=x_i} - \underbrace{\log\left(\sum_j e^{x_j}\right)}_{\text{same for all}}$$

$$\log \sigma_i(x) = x_i - \log\left(\sum_k e^{x_k}\right)$$

Differentiating

$$\frac{\partial}{\partial x_j} \log \sigma_i(x) = \frac{\partial x_i}{\partial x_j} - \frac{\partial}{\partial x_j} \cdot \log\left(\sum_k e^{x_k}\right)$$

Where

$$\frac{\partial x_i}{\partial x_j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

So

$$\frac{\partial x_i}{\partial x_j} = \mathbb{1}\{i = j\}$$

The other part

$$\frac{\partial}{\partial x_j} \log\left(\sum_k e^{x_k}\right) = \frac{1}{\sum_k e^{x_k}} \cdot \frac{\partial}{\partial x_j}\left(\sum_k e^{x_k}\right)$$

where

$$\frac{\partial}{\partial x_j}\left(\sum_k e^{x_k}\right) = \sum_k \frac{\partial e^{x_k}}{\partial x_j}$$

$$= 0 + 0 + \frac{\partial e^{x_j}}{\partial x_j} + 0 + 0 + \cdots$$

$$= \frac{\partial e^{x_j}}{\partial x_j}$$

$$= e^{x_j}$$

All together

$$\frac{\partial}{\partial x_j} \log \sigma_i(x) = \frac{\partial x_i}{\partial x_j} - \frac{\partial}{\partial x_j} \log\left(\sum_k e^{x_k}\right)$$

$$= \mathbb{1}\{i=j\} - \underbrace{\frac{1}{\sum_k e^{x_k}} \cdot e^{x_j}}$$

$$= \mathbb{1}\{i=j\} - \sigma_j(x)$$

Multiplying both sides by $\sigma_i(x)$, we get (1)

$$\sigma_i(x)\left(\frac{\partial}{\partial x_j} \log \sigma_i(x)\right) = \sigma_i(x) \cdot \left(\mathbb{1}\{i=j\} - \sigma_j(x)\right)$$

If we take partial derivative of $\log(\text{softmax})$

$$\frac{\partial}{\partial x_j} \log\left(\sigma_i(x)\right) = \frac{1}{\sigma_i(x)} \cdot \frac{\partial \sigma_i(x)}{\partial x_j}$$

deriv. of log

$$\frac{\partial \sigma_i(x)}{\partial x_j} = \sigma_i(x) \cdot \frac{\partial}{\partial x_j} \log\left(\sigma_i(x)\right)$$

$$\overset{(1)}{\downarrow}$$

$$= \sigma_i(x) \cdot \left(\mathbb{1}\{i=j\} - \sigma_j(x)\right)$$

To conclude, the Jac. matrix becomes:

$$J(\sigma(x)) = \begin{bmatrix} \sigma_1(1-\sigma_1) & \sigma_1 \cdot (-\sigma_2) & \cdots & \sigma_1(-\sigma_n) \\ \sigma_2(-\sigma_1) & \sigma_2(1-\sigma_2) & \cdots & \sigma_2(-\sigma_n) \\ \vdots & & \ddots & \vdots \\ \sigma_n(-\sigma_1) & \sigma_n(-\sigma_1) & \cdots & \sigma_n(1-\sigma_n) \end{bmatrix}$$