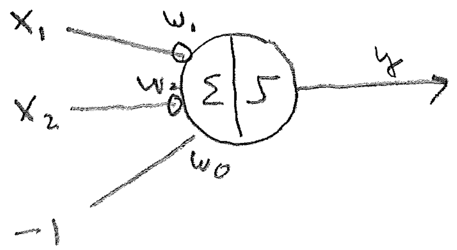
RECAP SOFAR

* model of a single neuron



$$y = \text{logsig}(w_1 x_1 + w_2 x_2 - w_0)$$

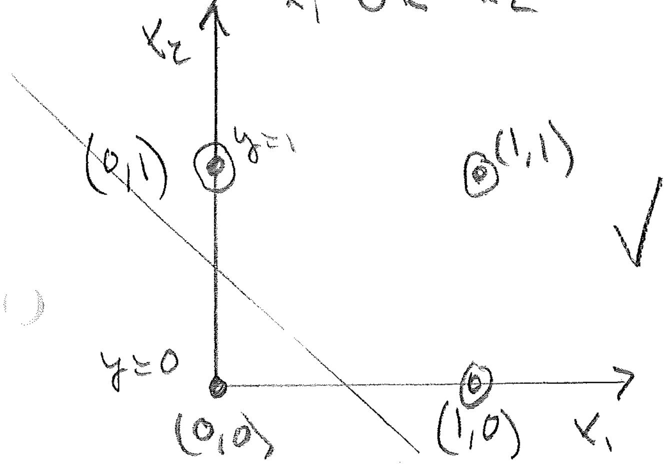
$$\text{class} = \begin{cases} 1 & \text{if } y = 0 \ (<0.5) \\ 2 & \text{if } y = 1 \ (>0.5) \end{cases}$$

* Training using gradient descent (GD)

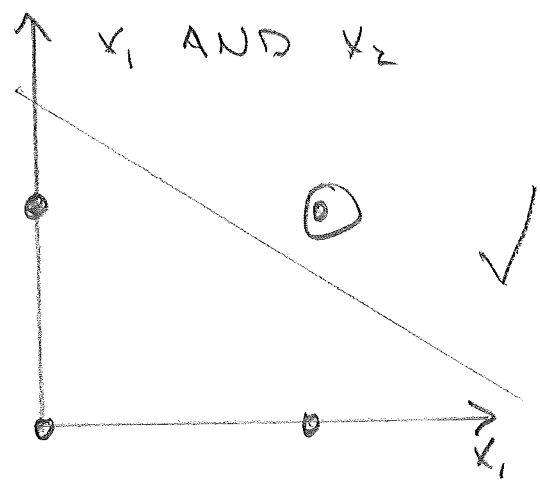
$$w_i^{t+1} = w_i^{t} - \mu \frac{\partial \mathcal{L}_{MSE}}{\partial w_i} \qquad \mathcal{L}_{MSE} : \text{loss}$$

SINGLE NEURON EXAMPES

x_1 OR x_2



x_1 AND x_2



$\text{logsig}(-10) \sim 0$, $\text{logsig}(+10) \sim 0$

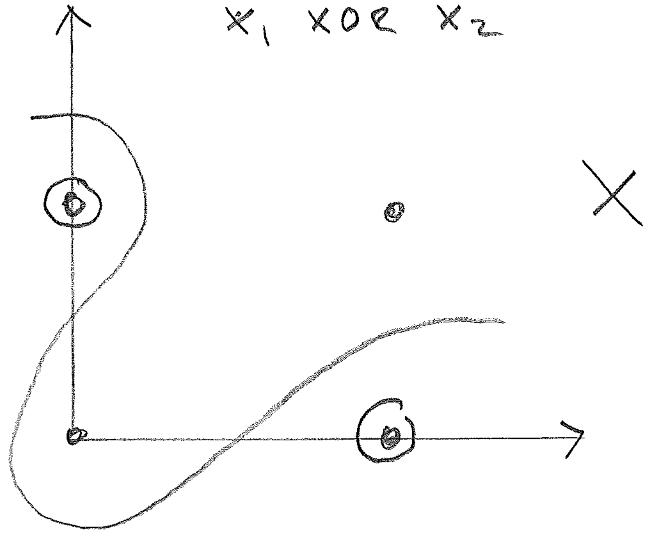
<NOTEBOOK>

$$\underbrace{w_1}_{30} x_1 + \underbrace{w_2}_{30} x - \underbrace{w_0}_{20}$$

$$\underbrace{w_1}_{15} x_1 + \underbrace{w_2}_{15} x_2 - \underbrace{w_0}_{20}$$

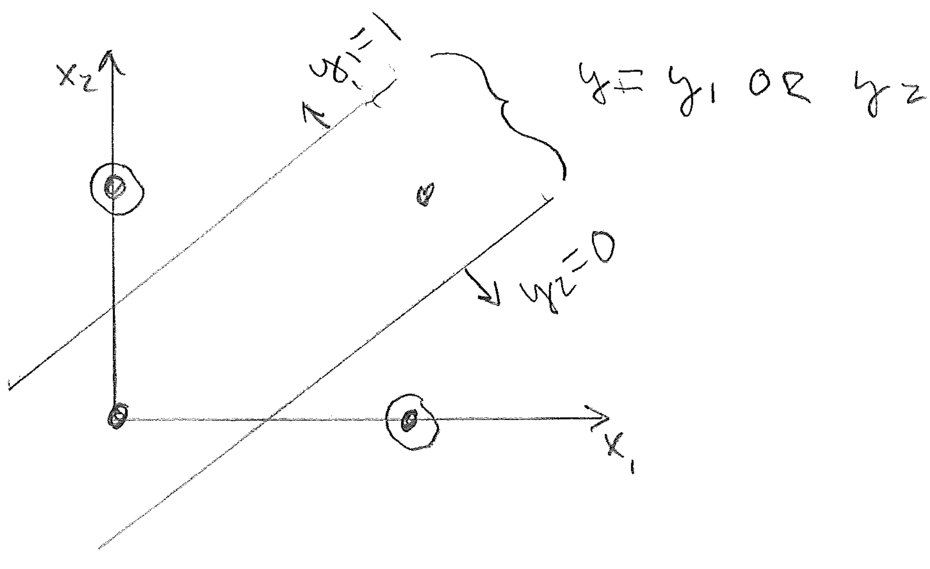
<NOTEBOOK>

x_1 XOR x_2



$\Rightarrow$ The first AI winter

Neural Network (multi-layer perceptron)



y_1: $\underbrace{w_{11}}_{-20} x_1 + \underbrace{w_{12}}_{30} x_2 - \underbrace{w_{10}}_{20}$

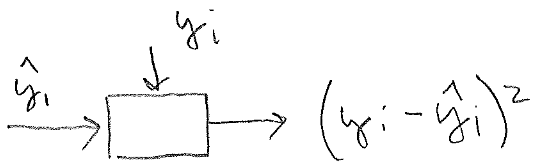
y_2: $\underbrace{w_{21}}_{30} x_1 + \underbrace{w_{22}}_{-20} x_2 - \underbrace{w_{20}}_{20}$

y: $\underbrace{w_1}_{30} y_1 + \underbrace{w_2}_{30} y_2 - \underbrace{w_0}_{20}$

<NOTEBOOK>

Gradient Descent for MLP

Loss function is minimized

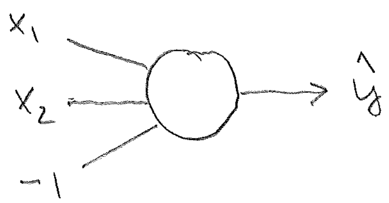


Forward: $\mathcal{L}_{MSE} = (y_i - \hat{y}_i)^2$

Backward (to input): $\frac{\partial \mathcal{L}_{MSE}}{\partial \hat{y}} = 2(y - \hat{y}) \cdot \underline{-1}$

$$\frac{\partial\, y - \hat{y}}{\partial \hat{y}} = -1$$

Output neuron



Forward: $\hat{y} = \mathrm{logsig}(w_1 x_1 + w_2 x_2 - w_0)$

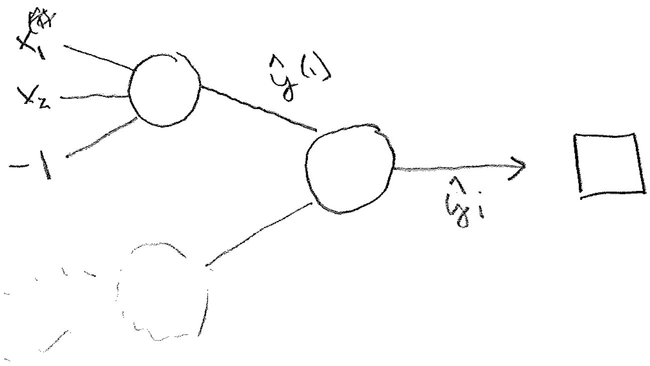
Backward: $$\frac{\partial \mathcal{L}_{MSE}}{\partial x_i} = \underbrace{\frac{\partial \mathcal{L}_{MSE}}{\partial \hat{y}}}_{\text{previous}} \cdot \frac{\partial \hat{y}}{\partial x_i} = \frac{\partial \mathcal{L}_{MSE}}{\partial \hat{y}} \cdot \mathrm{logsig}(\dots)(1 - \mathrm{logsig}(\dots)) \cdot \underline{w_i}$$

$\cancel{-w_i}$ -1 for $i=0$

update: $$w_i^{(t+1)} = w_i^{(t)} - \mu \frac{\partial \mathcal{L}_{MSE}}{\partial w_i}$$

$$= w_i^{(t)} - \mu \cdot \frac{\partial \mathcal{L}_{MSE}}{\partial \hat{y}} \cdot \mathrm{logsig}(\dots)(1 - \mathrm{logsig}(\dots)) \cdot x_i \quad (-1 \text{ for } i=0)$$

hidden layer neuron



Forward: $\hat{y}^{(1)} = \text{logsig}(w_1^{(1)} x_1 + w_2^{(1)} x_2 - w_0)$

Backward: $$\frac{\partial \mathcal{L}_{MSE}}{\partial x_i} = \underbrace{\frac{\partial \mathcal{L}_{MSE}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \hat{y}^{(1)}}}_{\text{previous}} \cdot \frac{\partial \hat{y}^{(1)}}{\partial x_i}$$

Update: $$\frac{\partial \mathcal{L}_{MSE}}{\partial w_i^{(1)}} = \frac{\partial \mathcal{L}_{MSE}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \hat{y}^{(1)}} \cdot \frac{\partial \hat{y}^{(1)}}{\partial w_i^{(1)}}$$

$$w_i^{(t+1),(1)} = w_i^{(t+1),(1)} - \mu \frac{\partial \mathcal{L}_{MSE}}{\partial w_i^{(1)}}$$

⇒ No matter how many layers and neurons all computation is local:

1. Forward pass: compute output for given inputs
2. Backward pass: compute gradient, concatenate w/ received gradient and pass forward
3. Update: update weights given the current gradient

why chain rule work? [extra material]

$$\frac{\partial f(g(h(x)))}{\partial x} = f'(g(h(x))) \cdot \frac{\partial g(h(x))}{\partial x}$$

$$= \underbrace{f'(g(h(x)))}_{\substack{f' \text{ : loss bw} \\ g(h(x)) \text{ : output fw}}} \cdot \underbrace{g'(h(x))}_{\substack{g' \text{ : output bw} \\ h(x) \text{ : hidden fw}}} \cdot \underbrace{h'(x)}_{\substack{h' \text{ : hidden bw} \\ x \text{ : input fw}}}$$