Formule RMS

$$rms\ proportion_e = \frac{rms_e = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(o_{i,e} - d_i)^2}}{max(rms_e),\ \forall e \in epochs}$$

$$with \begin{cases} n : number\ of\ neurons\ on\ the\ output\ layer \\ o_{i,e} : value\ obtained\ for\ the\ i^{th}\ neuron\ at\ the\ e^{th}\ epoch \\ d_i : value\ desired\ for\ the\ i^{th}\ neuron \end{cases}$$

$$rms\ proportion_e = \frac{rms_e = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(o_{i,e} - d_i)^2}}{max(rms_e),\ \forall e \in epochs}$$

$$with \begin{cases} n : number\ of\ neurons\ on\ the\ output\ layer \\ o_{i,e} : value\ obtained\ for\ the\ i^{th}\ neuron\ at\ the\ e^{th}\ epoch \\ d_i : value\ desired\ for\ the\ i^{th}\ neuron \end{cases}$$

function DISCRETIZE($hiddenNeuron[], piece$)
    $result \leftarrow 0$
    for $i = 0 \rightarrow hiddenNeuron.length$ **do**
        $result \leftarrow result + piece^i \times cutting(hiddenNeuron[i], piece)$
        $i \leftarrow i + 1$
    end for
    return result
end function

-

$first\_order.calc\_hidden\_layer(samples.inputs)$
$high\_order.calc\_output\_layer(first\_order.hidden\_layer)$
$first\_order.calc\_output\_layer(first\_order.hidden\_layer,\ [0, ..., 0])$

$h\_output \leftarrow ampli(high\_order.output\_layer)$
$right\_houtput \leftarrow [0,\ 0]$
if $good\_answer(first\_order)$ **then**
    $right\_houtput[1] \leftarrow 1$
else
    $right\_houtput[0] \leftarrow 1$
end if
$first\_order.calc\_output\_layer(first\_order.hidden\_layer,\ h\_output)$

$calc\_stats()$

$high\_order.train(first\_order.hidden\_layer,\ right\_houtput)$
$first\_order.train(samples.inputs,\ samples.outputs,\ h\_output)$

function $train(inputs,\ outputs,\ add)$
$y \leftarrow build\_error\_vector(...)$
$update\_weights\_hidden\_layer(...)$

for $i = 0 \rightarrow output\_neurons.length$ **do**
$output\_neurons[i].update\_weights\_gradient(y[i],\ hidden\_neurons,\ add)$
$output\_neurons[i].update\_weights\_perceptron(outputs[i],\ hidden\_neurons,\ add)$
end for
end function

function $update\_weights\_gradient(error,\ intputs,\ add)$
$calc\_output(inputs + add)$

for $j = 0 \rightarrow inputs.length$ **do**
$dw \leftarrow weights[j] - last\_weights[j]$
$p \leftarrow error \times inputs[j]$
$weights[j] \leftarrow weights[j] + learning\_rate \times p + momentum \times dw$
end for
end function

function $update\_weights\_perceptron(goal,\ intputs,\ add)$
$calc\_output(inputs + add)$

for $j = inputs.length \rightarrow inputs.length + add.length$ **do**
$dw \leftarrow weights[j] - last\_weights[j]$
$p \leftarrow (goal - state) \times add[inputs.length - j]$
$weights[j] \leftarrow weights[j] + \frac{learning\_rate \times p + momentum \times dw}{add.length}$
end for
end function