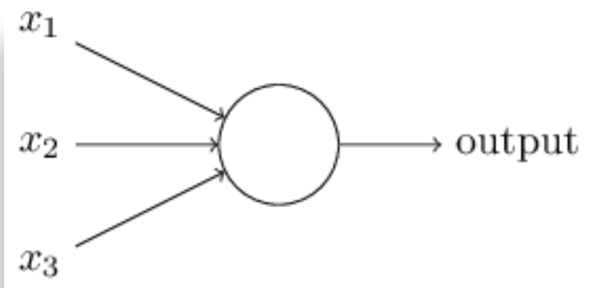
# PERCEPTRON/NEURON

# PERCEPTRON

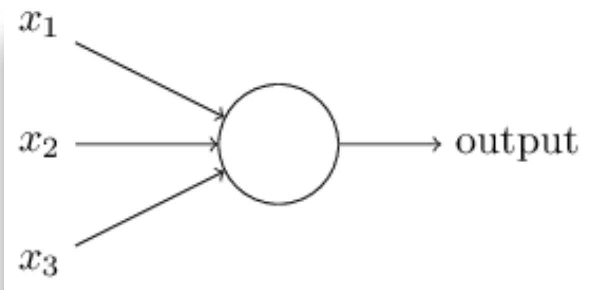
- The basic unit of a NN is a perceptron:




- Perceptron was introduced by Frank Rosenblatt in 1957.

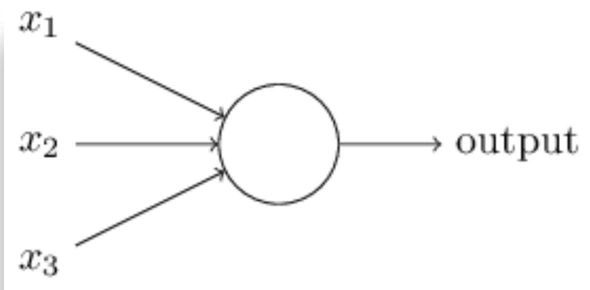
# PERCEPTRON

- How do we relate input with output?



# PERCEPTRON

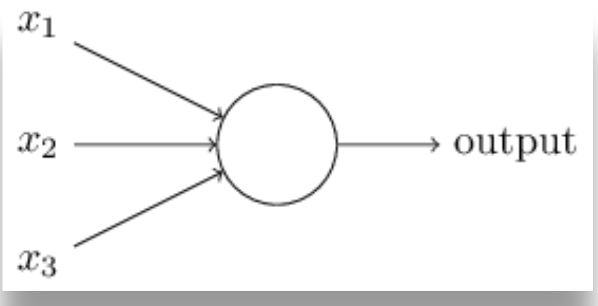
- How do we relate input with output?




  - H1: output depends on a threshold TH
    - Example: if $x_1 + x_2 + x_3 > TH$ return 1; 0 otherwise

# PERCEPTRON

- How do we relate input with output?

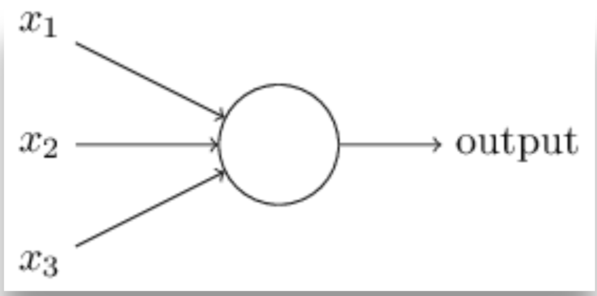



  - H2: H1 + weighted input
    - Example: if $x_1 * w_1 + x_2 * w_2 + x_3 * w_3 >$ TH return 1; 0 otherwise

But… this function will always be 0 in (0, 0, …).

# PERCEPTRON

- How do we relate input with output?

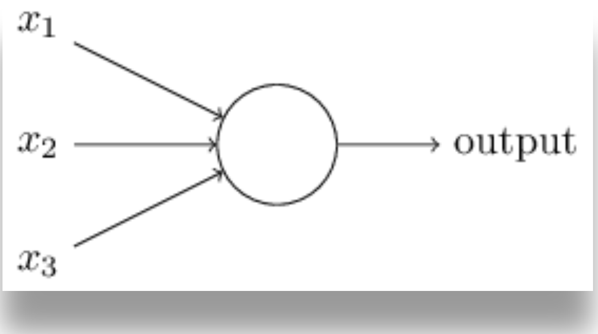



  - H3: H2 + bias (b)
    - Example: $x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + 1 * b$

Without b => poorer fit. But, even with b => linear

# PERCEPTRON => NEURON

- How do we relate input with output?

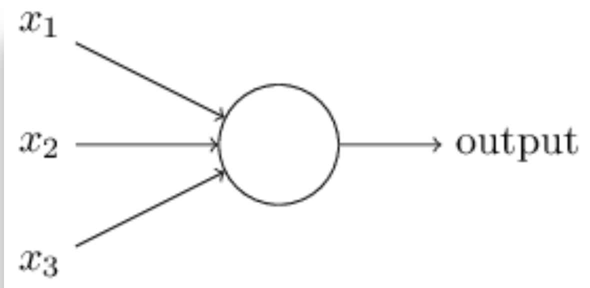



  - H4: H3 + activation function

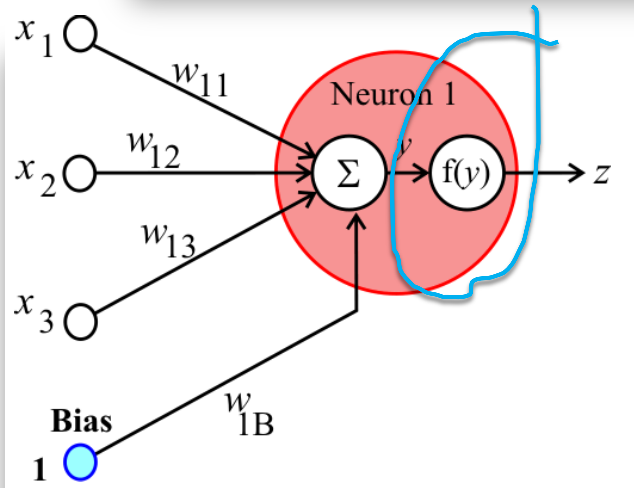
Mostly set to make a non-linear transformation which allows to fit nonlinear hypotheses

# NEURON

- How do we relate input with output?



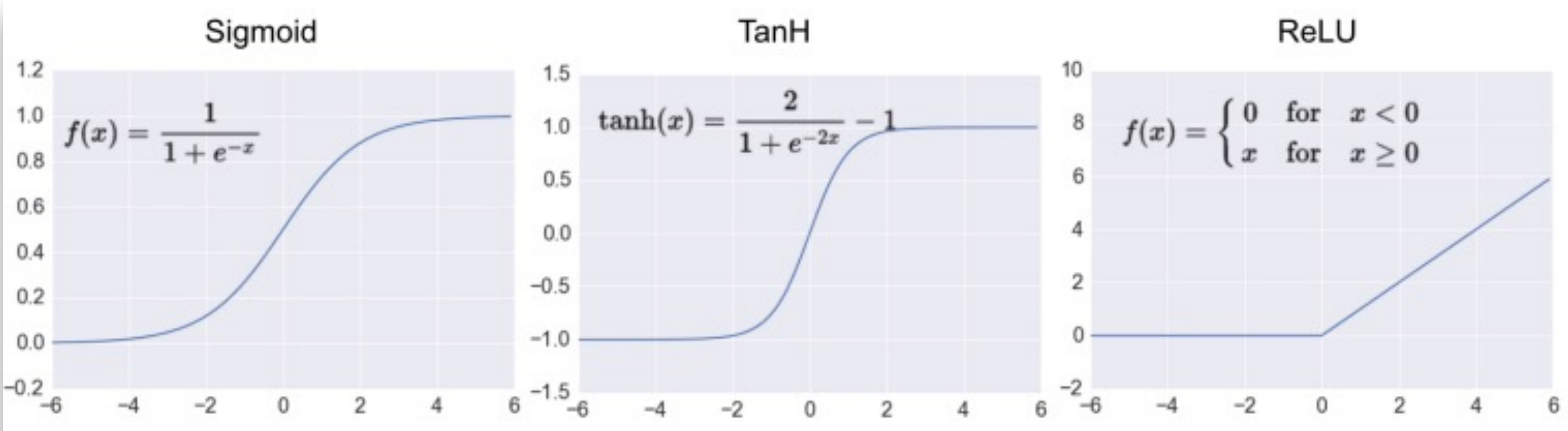

  - H4: H3 + activation function




http://sintesis.ugto.mx/WintemplaWeb/01Neural%20Lab/00Introduction/02Activation%20Function/index.htm

# NEURON

- Multiple activation functions are available.
- Some examples:

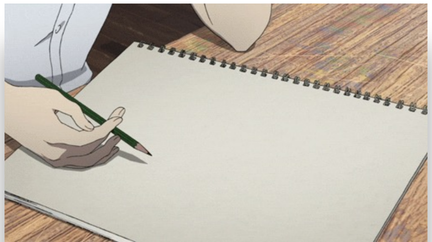



https://www.kdnuggets.com/2017/09/neural-network-foundations-explained-activation-function.html

# EXAMPLE (Jurafsky)

- Let
  - $W = [0.2, 0.3, 0.9]$
  - $x = [0.5, 0.6, 0.1]^T$
  - b (bias) = 0.5
  - $y = \sigma$ (W.x + b) = ($\sigma$ (sigma = sigmoid function)) = $1/(1+e^{-(W.x + b)})$

What is the value of y?



https://giphy.com/gifs/writing-11ikeVaUfcXLWM

# EXAMPLE (Jurafsky)

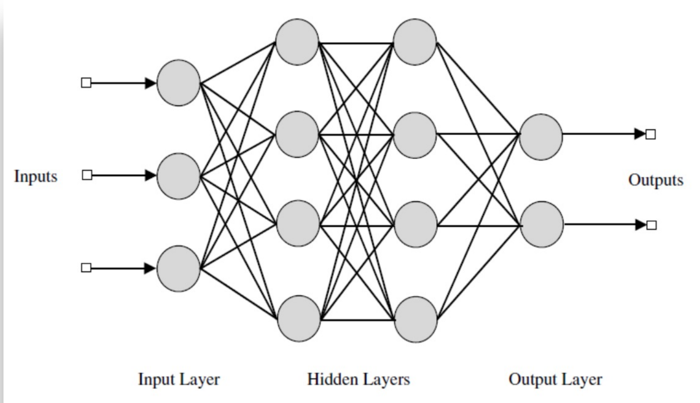
- Let
  - $W = [0.2, 0.3, 0.9]$
  - $x = [0.5, 0.6, 0.1]^T$
  - b (bias) = 0.5
  - $y = \sigma\,(W.x + b) = (\sigma$ (sigma = sigmoid function)) $= 1/(1+e^{-(W.x + b)})$

    - $W.x = [0.2, 0.3, 0.9].\ [0.5, 0.6, 0.1]^T = 0.2*0.5 + 0.3*0.6 + 0.9*0.1 = 0.37$
    - $W.x + b = 0.37+0.5 = 0.87$
    - $y = 1/(1 + e^{-0.87}) = 0.7$

# FEED-FORWARD NEURAL NETWORK (or MULTI-LAYER NEURAL NETWORK)

# FEED-FORWARD NEURAL NETWORK

- We increase the expressive power of the network by adding intermediate layers of neurons before the final output layer, yielding more complex, non-linear classifiers.
  - Note: In the standard architecture, each layer is fully-connected (Jurafsky)




http://cse22-iiith.vlabs.ac.in/exp4/index.html

# FEED-FORWARD NEURAL NETWORK

- How do we make the perfect cookies?
  - We make various tests, varying the recipe
    - First step: follow recipe, taste, if perfect we stop, otherwise:
      - Second step: we change the recipe (we change ingredients or some quantities of the ingredients), taste, if perfect we stop, otherwise:
        - Third step:...
- Neural Networks (NN) work in a similar manner

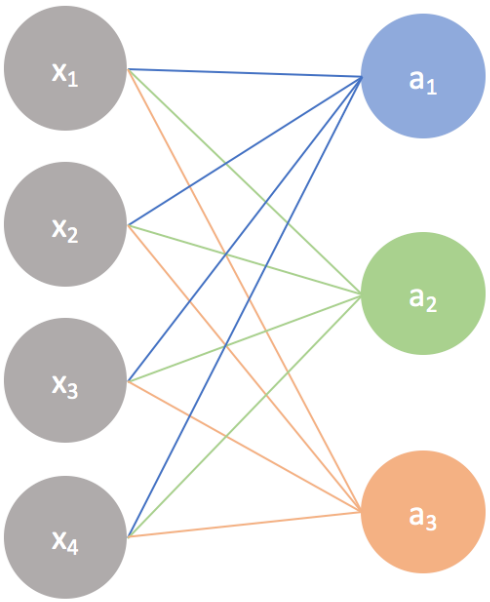
# FEED-FORWARD NEURAL NETWORK

- Neural Networks:
  - Initialization
  - Take input and process it (forward propagation)
  - Result is compared with the desired output => the error is obtained
  - Try to minimize the error, by changing some elements in the network (backward propagation).
    - We update each weight, so that the actual output becomes closer to the target output (minimizing the error). A learning rate is used ($\eta$).

# FEED-FORWARD NEURAL NETWORK

Input layer Output layer

## A simple neural network

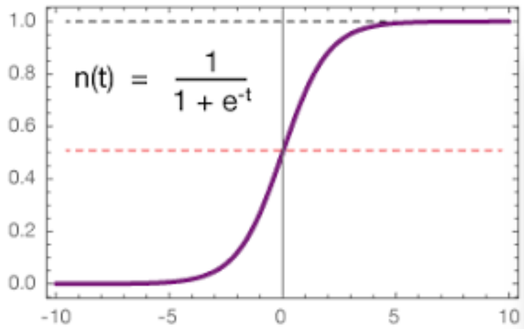


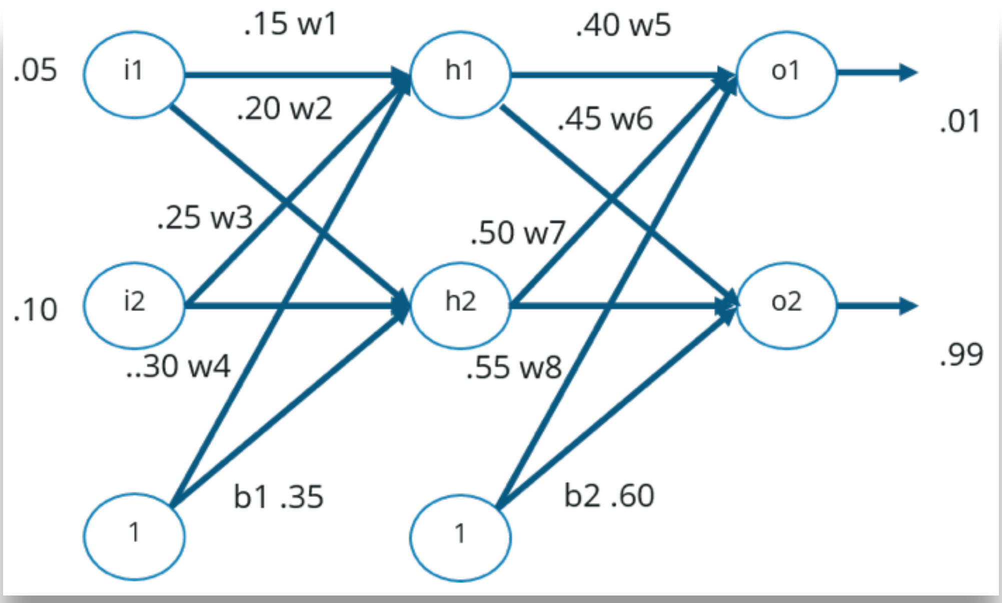

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \end{bmatrix} \xrightarrow[activation]{} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

# FEED-FORWARD NEURAL NETWORK

Input: 0.05,  0.1      Output:  0.01, 0.99

Initial weights and bias:    random, Activation function:      logistic function



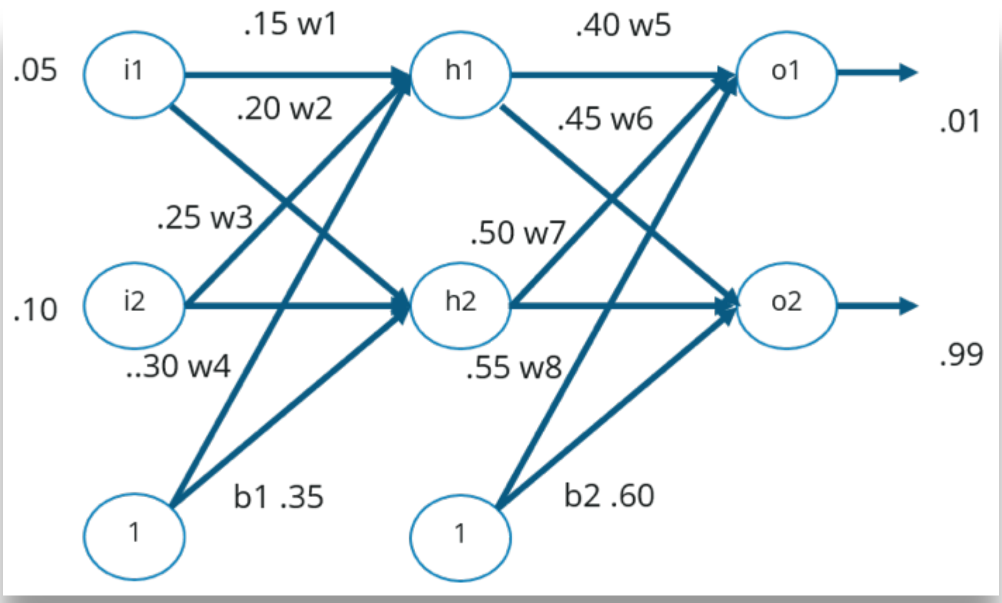

Example from: https://www.edureka.co/blog/backpropagation/

# FEED-FORWARD NEURAL NETWORK
## (FORWARD PROPAGATION)

$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1 = 0.15*0.05+0.2*0.1+0.35*1 = 0.3775$

Plus activation on $net_{h1}$: $out_{h1} = 0{,}593269992$

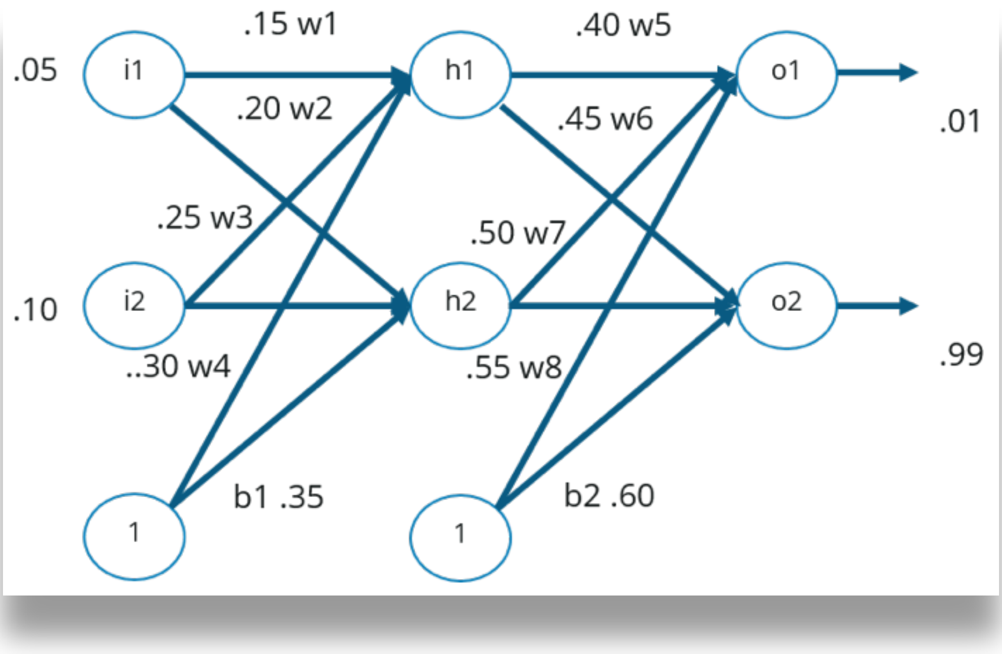
By the same token, calculate: $net_{h1}$, $out_{h2}$, $net_{o1}$, $out_{o1}$, $net_{o2}$, $out_{o2}$



# FEED-FORWARD NEURAL NETWORK
## (ERROR)

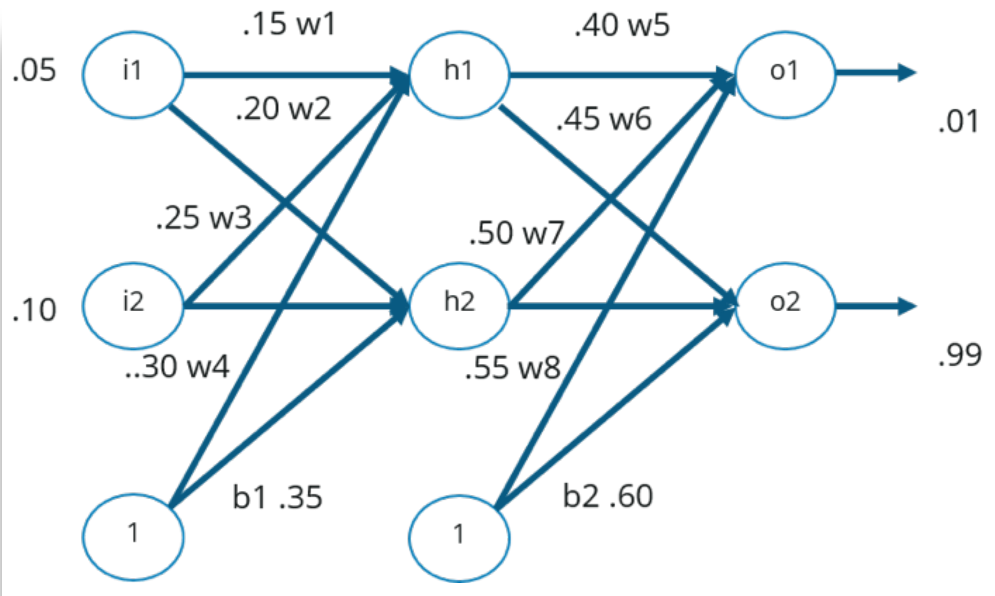
$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = 0.274811083$, $Eo_2 = 0.023560026$

$E_{total} = E_{o1} + E_{o2} = 0.298371109$



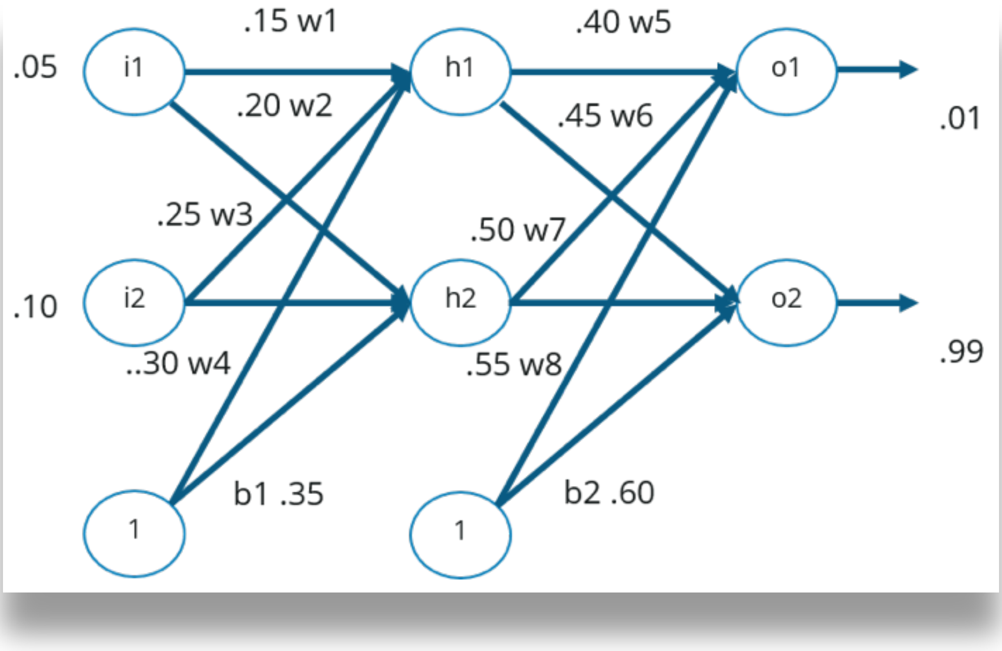
# FEED-FORWARD NEURAL NETWORK
## (BACKWARD PROPAGATION)

How much a change in $w_5$ affects the total error?
=> partial derivative of $E_{total}$ with respects to $w_5$ = the gradient with respect to $w_5$



# FEED-FORWARD NEURAL NETWORK
## (BACKWARD PROPAGATION)

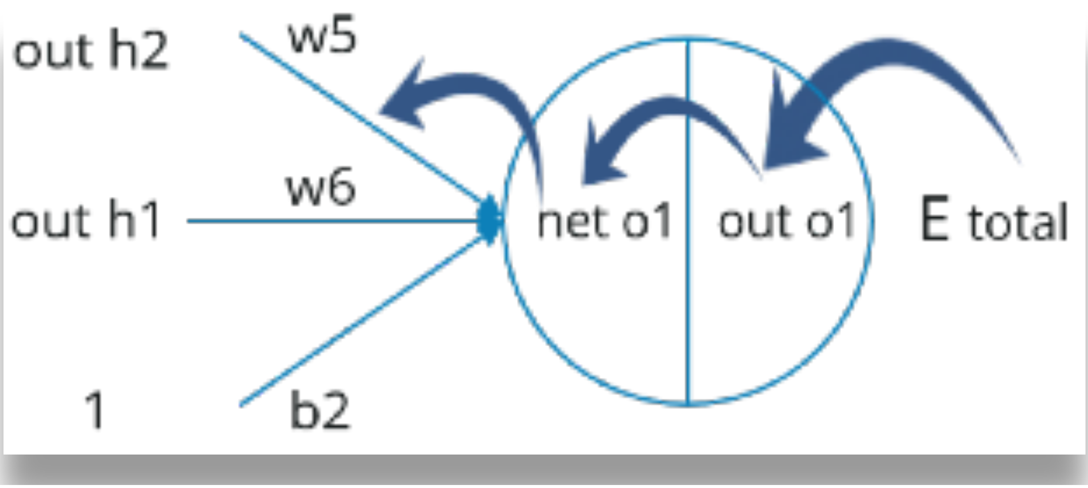
Need do calculate: $\dfrac{\partial E_{total}}{\partial w_5}$



# FEED-FORWARD NEURAL NETWORK
## (BACKWARD PROPAGATION)

CHAIN RULE:

$$\frac{\delta Etotal}{\delta w5} = \frac{\delta Etotal}{\delta out\ o1} * \frac{\delta out\ o1}{\delta net\ o1} * \frac{\delta net\ o1}{\delta w5}$$



# FEED-FORWARD NEURAL NETWORK
## (BACKWARD PROPAGATION)

- As $E_{Total} = ½ (target_{o1} – out_{o1})^2 + ½ (target_{o2} – out_{o2})^2$:

$\frac{dE_{Total}}{dout_{o1}} = -(target_{o1} - out_{o1}) = -(0.01\text{-}0.75136507) = 0.74136507$

- As $out_{o1} = 1/(1+e^{-neto1})$:

$\frac{dout_{o1}}{dnet_{o1}} = out_{o1}(1 - out_{o1}) = 0.186815602$

- As $net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$:

$\frac{dnet_{o1}}{dw_5} = out_{h1} = 0.593269992$

# FEED-FORWARD NEURAL NETWORK
## (BACKWARD PROPAGATION)

$$\frac{dE_{Total}}{dout_{o1}} = -(target_{o1} - out_{o1}) = -(0.01-0.75136507) = 0.74136507$$

$$\frac{dout_{o1}}{dnet_{o1}} = out_{o1}(1 - out_{o1}) = 0.186815602$$

$$\frac{dnet_{o1}}{dw_5} = out_{h1} = 0.593269992$$

$$\boxed{\frac{\delta Etotal}{\delta w5} = \frac{\delta Etotal}{\delta out\ o1} * \frac{\delta out\ o1}{\delta net\ o1} * \frac{\delta net\ o1}{\delta w5}}$$

# FEED-FORWARD NEURAL NETWORK
## (BACKWARD PROPAGATION)

$$\frac{dE_{Total}}{dw_5} = 0.74136507*0.186815602*0.593269992=0.082167041$$

$$\frac{\delta Etotal}{\delta w5} = \frac{\delta Etotal}{\delta out\ o1} * \frac{\delta out\ o1}{\delta net\ o1} * \frac{\delta net\ o1}{\delta w5}$$

# FEED-FORWARD NEURAL NETWORK
## (BACKWARD PROPAGATION)

Being η the Learning rate:

$$w_5 = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5}$$

And we update $w_5$