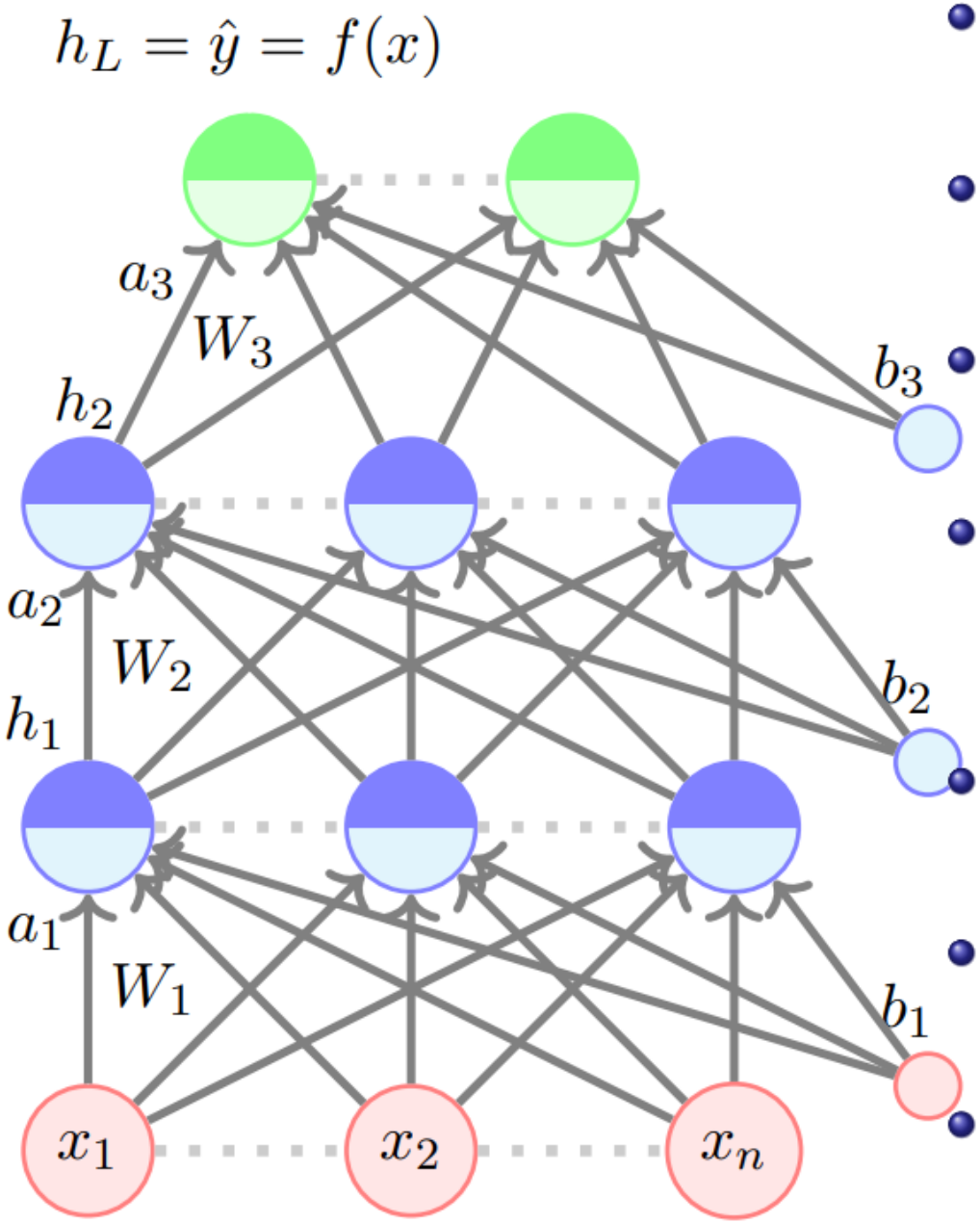
Feedforward Neural Networks And Back Propagation

Outline

- FFN
- Learning Parameters
- Loss Function
- Learning Algorithm
- Backpropagation
- Derive gradient w r t output neuron
- Derive gradient w r t hidden neuron
- Numerical Example

Feedforward Neural Networks



- The input to the network is an **n**-dimensional vector
- The network contains $\mathbf{L-1}$ hidden layers (2, in this case) having **n** neurons each
- Finally, there is one output layer containing **k** neurons (say, corresponding to **k** classes)
- Each neuron in the hidden layer and output layer can be split into two parts : pre-activation and activation (a_i and h_i are vectors)
- The input layer can be called the 0-th layer and the output layer can be called the (L)-th layer
- $W_i \in \mathbb{R}^{n \times n}$ and $b_i \in \mathbb{R}^n$ are the weight and bias between layers $i-1$ and i $(0 < i < L)$
- $W_L \in \mathbb{R}^{n \times k}$ and $b_L \in \mathbb{R}^k$ are the weight and bias between the last hidden layer and the output layer ($L = 3$ in this case)

- The pre-activation at layer i is given by

$$a_i(x) = b_i + W_i h_{i-1}(x)$$

- The activation at layer i is given by

$$h_i(x) = g(a_i(x))$$

where g is called the activation function (for example, logistic, tanh, linear, *etc.*)

- The activation at the output layer is given by

$$f(x) = h_L(x) = O(a_L(x))$$

where O is the output activation function (for example, softmax, linear, *etc.*)

- To simplify notation we will refer to $a_i(x)$ as a_i and $h_i(x)$ as h_i

Learning Parameters of Feedforward Neural Networks

- We can write it more concisely as

Algorithm: gradient_descent()

$t \leftarrow 0$;
$max\_iterations \leftarrow 1000$;
$Initialize \quad \theta_0 = [W_1^0, ..., W_L^0, b_1^0, ..., b_L^0]$;
while t++ $< max\_iterations$ **do**
  $\theta_{t+1} \leftarrow \theta_t - \eta \nabla \theta_t$;
end

- where $\nabla \theta_t = [\frac{\partial \mathscr{L}(\theta)}{\partial W_{1,t}}, .., \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,t}}, \frac{\partial \mathscr{L}(\theta)}{\partial b_{1,t}}, .., \frac{\partial \mathscr{L}(\theta)}{\partial b_{L,t}}]^T$
- Now, in this feedforward neural network, instead of $\theta = [w, b]$ we have $\theta = [W_1, W_2, .., W_L, b_1, b_2, .., b_L]$

$$\begin{bmatrix}
\frac{\partial \mathscr{L}(\theta)}{\partial W_{111}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{11n}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{211}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{21n}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,11}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,1k}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,1k}} & \frac{\partial \mathscr{L}(\theta)}{\partial b_{11}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial b_{L1}} \\
\frac{\partial \mathscr{L}(\theta)}{\partial W_{121}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{12n}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{221}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{22n}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,21}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,2k}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,2k}} & \frac{\partial \mathscr{L}(\theta)}{\partial b_{12}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial b_{L2}} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\frac{\partial \mathscr{L}(\theta)}{\partial W_{1n1}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{1nn}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{2n1}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{2nn}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,n1}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,nk}} & \frac{\partial \mathscr{L}(\theta)}{\partial W_{L,nk}} & \frac{\partial \mathscr{L}(\theta)}{\partial b_{1n}} & \cdots & \frac{\partial \mathscr{L}(\theta)}{\partial b_{Lk}}
\end{bmatrix}$$

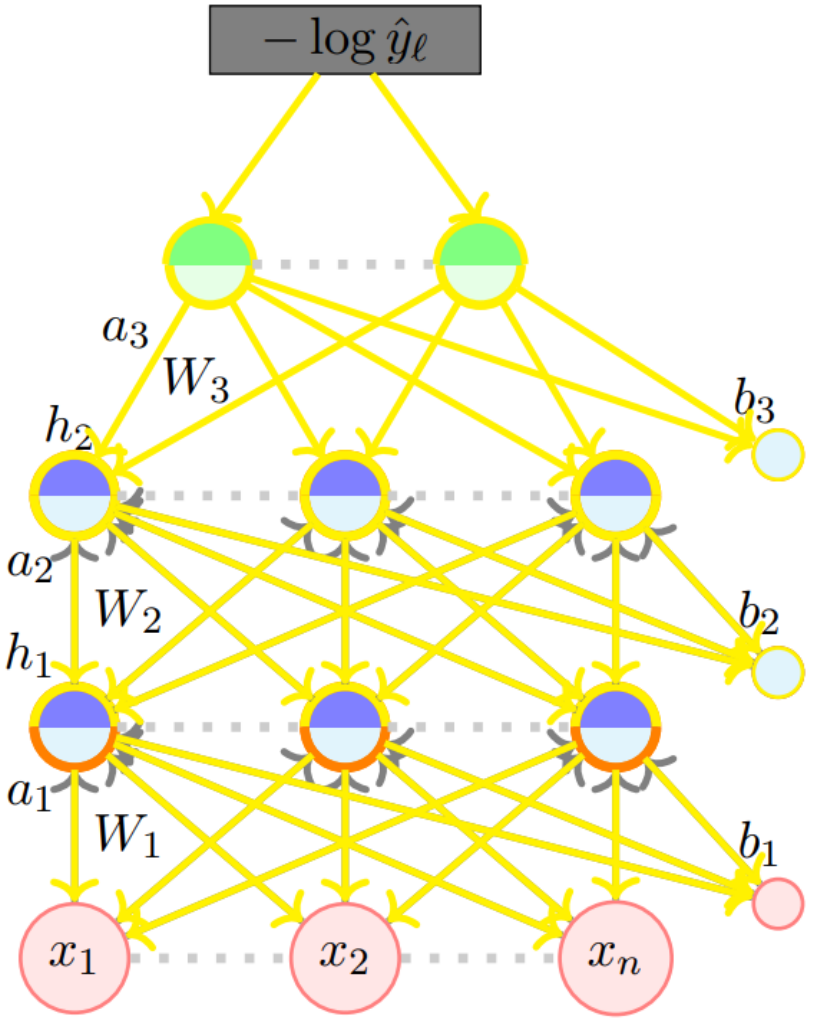
- $\nabla\theta$ is thus composed of
  $\nabla W_1, \nabla W_2, ...\nabla W_{L-1} \in \mathbb{R}^{n\times n}, \nabla W_L \in \mathbb{R}^{n\times k}$,
  $\nabla b_1, \nabla b_2, ..., \nabla b_{L-1} \in \mathbb{R}^n$ and $\nabla b_L \in \mathbb{R}^k$

Loss Functions

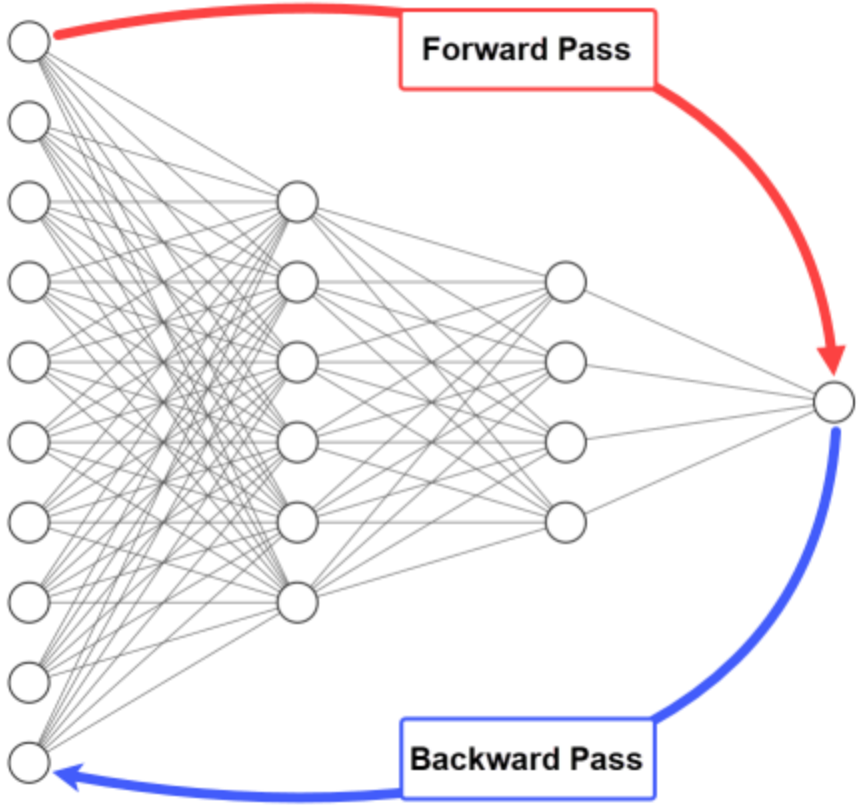
	Outputs	
	Real Values	Probabilities
Output Activation	Linear	Softmax
Loss Function	Squared Error	Cross Entropy

Backpropagation

$$\underbrace{\frac{\partial \mathscr{L}(\theta)}{\partial W_{111}}}_{\text{Talk to the weight directly}} = \underbrace{\frac{\partial \mathscr{L}(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_3}}_{\text{Talk to the output layer}} \underbrace{\frac{\partial a_3}{\partial h_2} \frac{\partial h_2}{\partial a_2}}_{\text{Talk to the previous hidden layer}} \underbrace{\frac{\partial a_2}{\partial h_1} \frac{\partial h_1}{\partial a_1}}_{\text{Talk to the previous hidden layer}} \underbrace{\frac{\partial a_1}{\partial W_{111}}}_{\text{and now talk to the weights}}$$



Backpropagation



- **Forward Propagation:** Pass inputs through the network to compute outputs and loss.
- **Compute Output Layer Error:** Calculate error at the output layer using the derivative of the loss function.
- **Backpropagate Errors:** Compute gradients for each layer using the chain rule, moving backward from output to input.
- **Update Weights:** Adjust weights and biases using gradient descent to minimize the loss.

Backpropagation Algorithm

- Create a feed-forward network with n_i inputs, n_{hidden} hidden units, and n_{out} output units.
- Initialize all network weights to small random numbers
- Until the termination condition is met, Do
  - For each (x, t), in training examples, Do
    - Propagate the input forward through the network:

      1. Input the instance x, to the network and compute the output o_u of every unit u in the network.

    - Propagate the errors backward through the network

      2. For each network unit k, calculate its error term δ_k

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

      3. For each network unit h, calculate its error term δ_h

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{h,k}\delta_k$$

4. Update each network weight w_{ji}

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

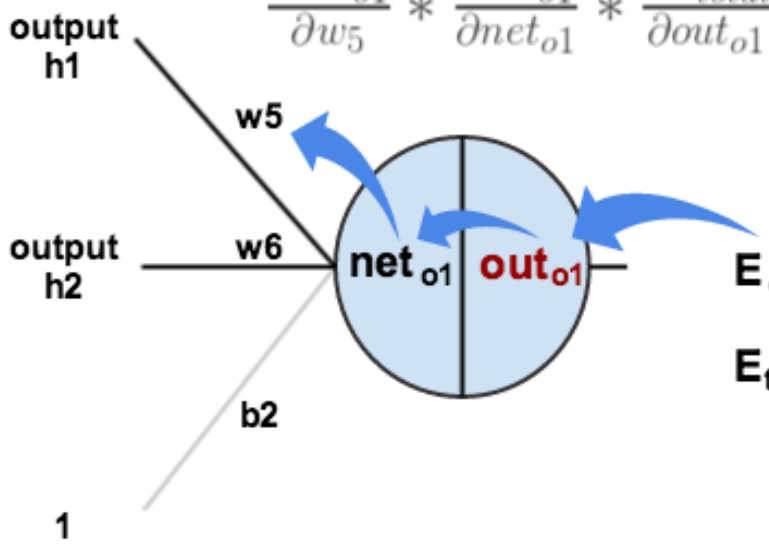
Where

$$\Delta w_{ji} = \eta \delta_j x_{ji}$$

Backwards Pass

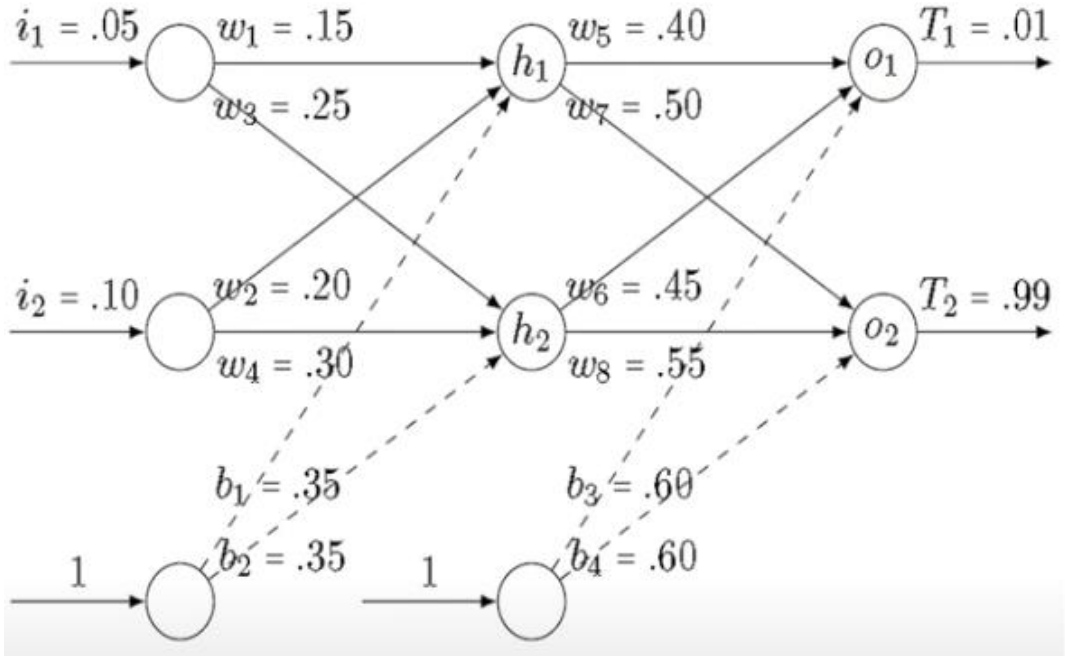
We want to know how much a change in w_5 affects the total error $\frac{\partial E_{total}}{\partial w_5}$

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



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

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



The Chain Rule

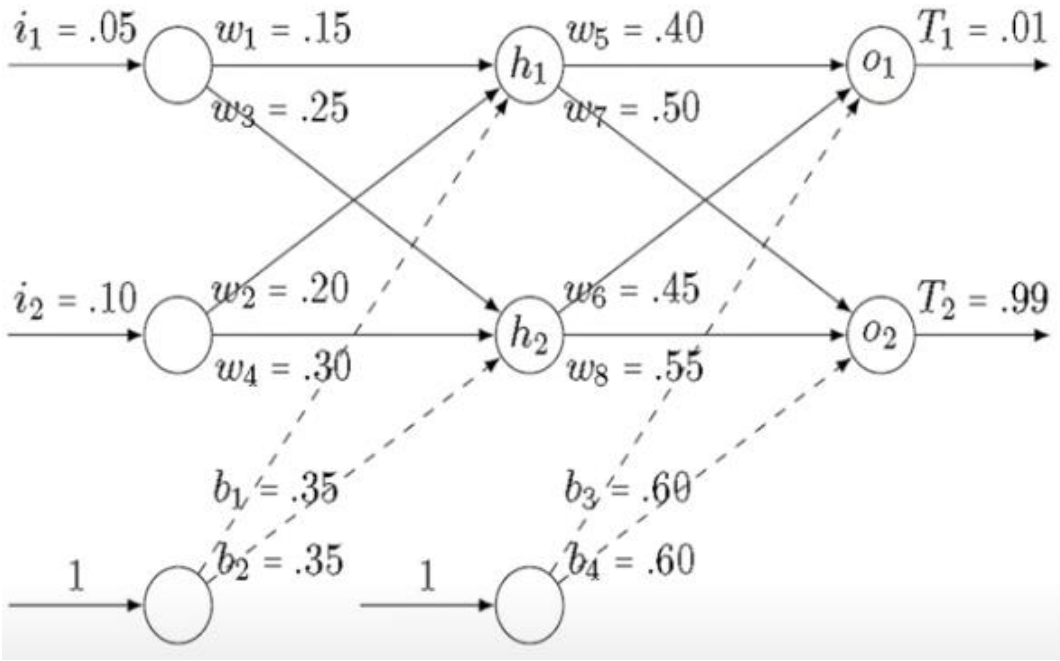
$$z = e^{xy} \quad x = 2u$$

$$\frac{\partial z}{\partial u} = \frac{\partial z}{\partial x} \cdot \frac{\partial x}{\partial u}$$

Update Rule in SGD

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$



Ed is the error on training example **d**

$$E_d(\vec{w}) \equiv \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2$$

Notations

x_{ji} = the i[th] input to unit j

w_{ji} = the weight associated with the i[th] input to unit j

$net_j = \sum_i w_{ji} X_{ji}$ (the weighted sum of inputs for unit j)

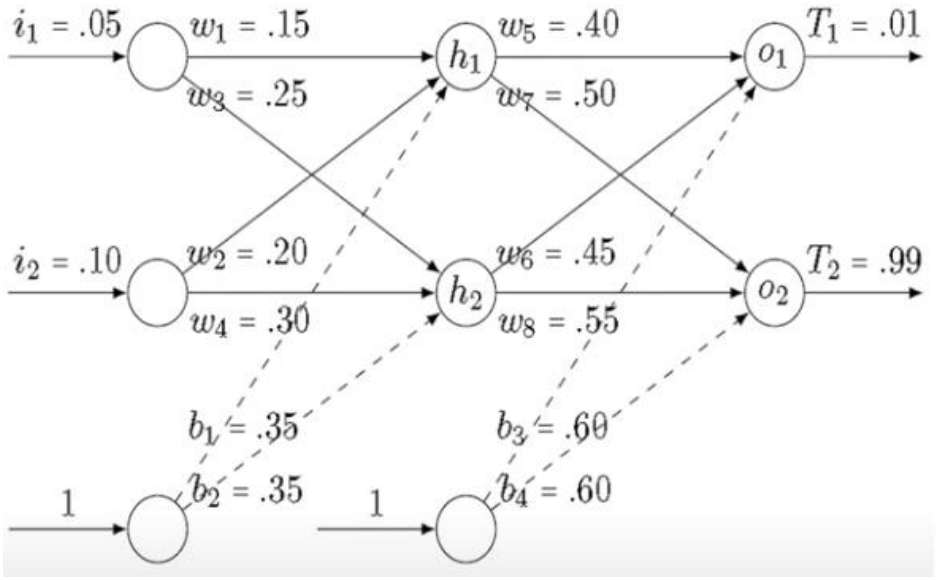
o_j = the output computed by unit j

t_j = the target output for unit j

σ = the sigmoid function

outputs = the set of units in the final layer of the network

Downstream(j) = the set of units whose immediate inputs include the output of unit j

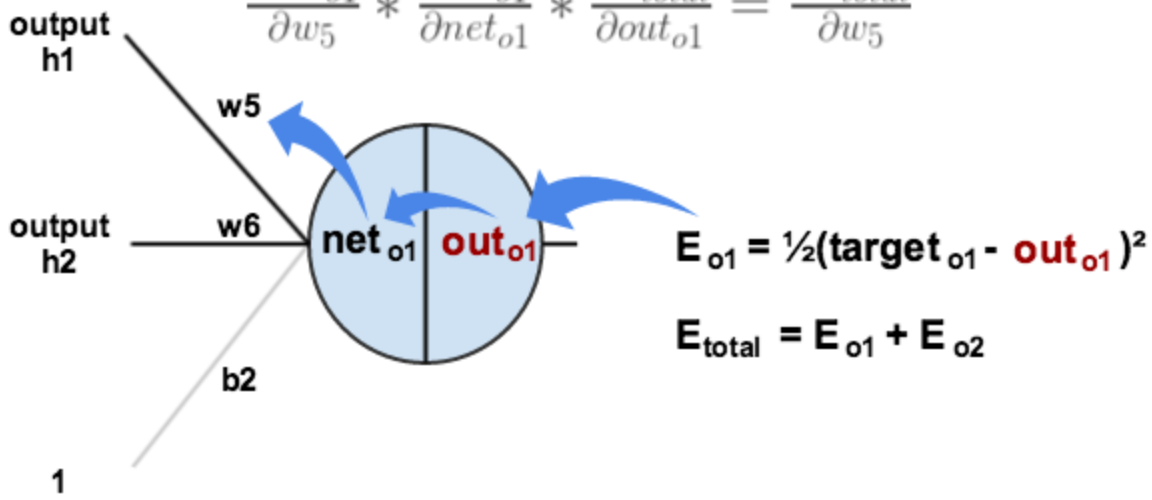


Case 1: unit j is an output unit

Wji can influence the network only through netj, netj can influence the network only through outj

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j}\frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



Case 1: unit j is an output unit

Wji can influence the network only through netj, netj can influence the network only through outj

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j}\frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j}\frac{1}{2}\sum_{k \in outputs}(t_k - o_k)^2$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j}\frac{1}{2}(t_j - o_j)^2$$

$$= \frac{1}{2}2(t_j - o_j)\frac{\partial(t_j - o_j)}{\partial o_j}$$

$$= -(t_j - o_j)$$

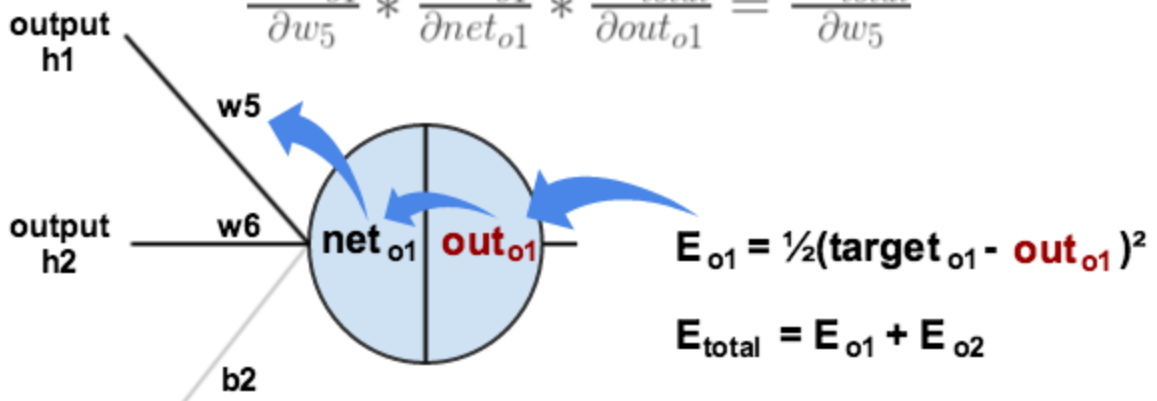
$$\frac{\partial o_j}{\partial(net_j)} = \frac{\partial \sigma(net_j)}{\partial(net_j)}$$

$$= \sigma(net_j)\,(1 - \sigma(net_j))$$

$$= o_j\,(1 - o_j)$$

$$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j)\, o_j(1 - o_j)$$

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



$$\frac{\partial \sigma(x)}{\partial(x)} = \sigma(x)\,(1 - \sigma(x))$$

Case 1: unit j is an output unit

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}$$

$$= \frac{\partial E_d}{\partial net_j} x_{ji}$$

$$net_j = \sum_i w_{ji} X_{ji}$$

$$\frac{\partial net_j}{\partial w_{ji}} = x_{ji}$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} x_{ji}$$

$$\Delta w_{ji} = \eta (t_j - o_j) o_j (1 - o_j) x_{ji}$$

$$\Delta w_{ji} = \eta \delta_j x_{ji}$$

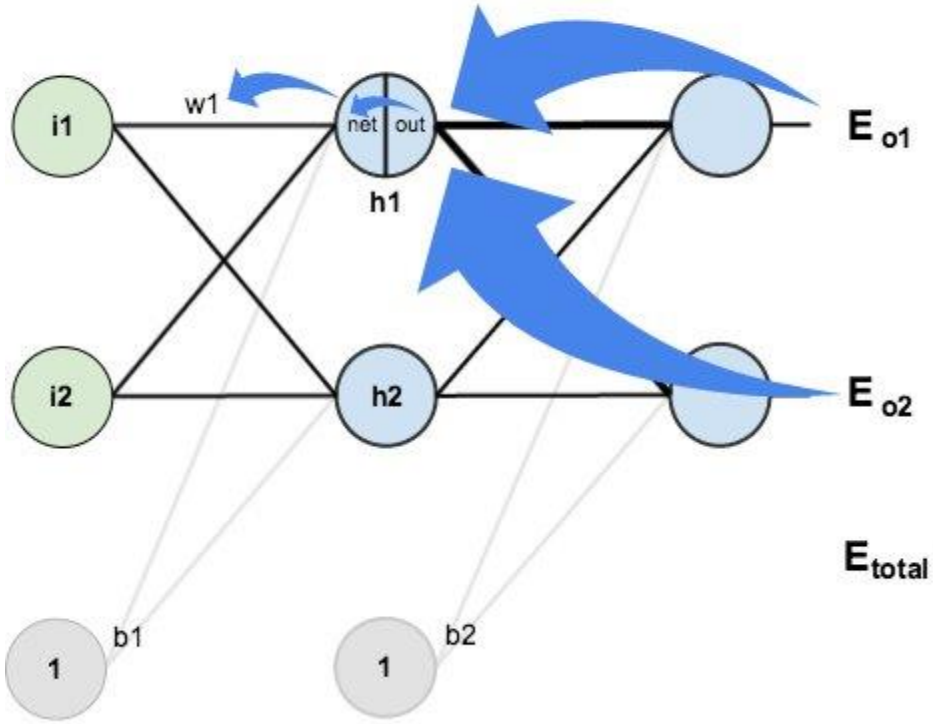
$$\delta_j = (t_j - o_j) o_j (1 - o_j)$$

Case 2: unit j is an internal (hidden) unit

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\downarrow$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



each hidden layer neuron contributes to the output (and therefore error) of multiple output neurons

out_{h1} affects both out_{o1} and out_{o2} therefore the $\frac{\partial E_{total}}{\partial out_{h1}}$ needs to take into consideration its effect on the both output neurons:

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

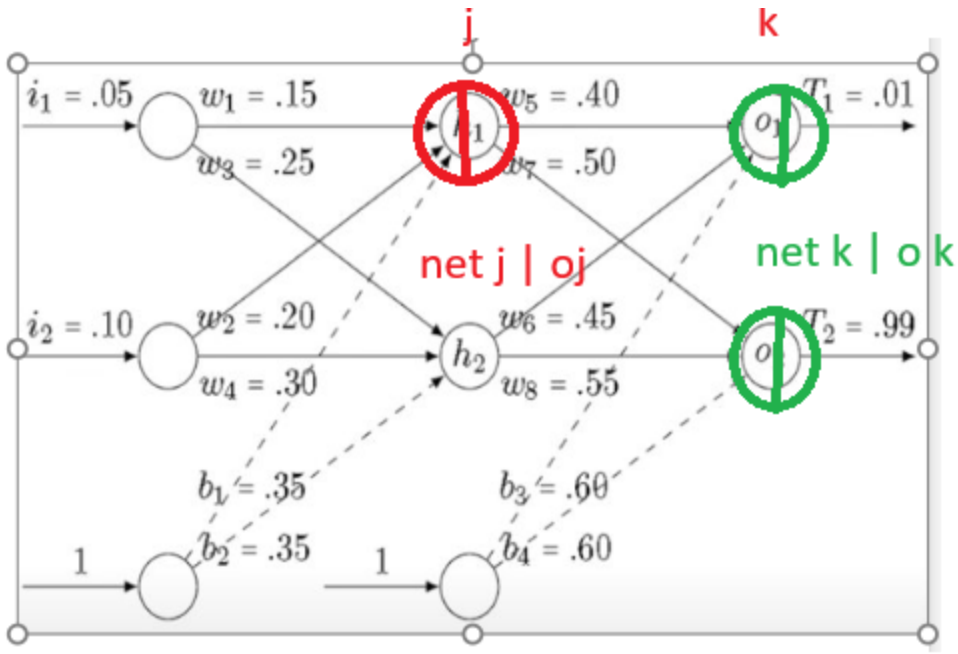
$$E_{total} = E_{o1} + E_{o2}$$

Case 2: unit j is an internal (hidden) unit

$$\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream(j)} \frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j}$$

$$= \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial net_j}$$

$$= \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial o_j} \frac{\partial o_j}{\partial net_j}$$



Case 2: unit j is an internal (hidden) unit

$$\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream(j)} \frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j}$$

$$= \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial net_j}$$

$$= \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial o_j} \frac{\partial o_j}{\partial net_j}$$

$$= \sum_{k \in Downstream(j)} -\delta_k \, w_{kj} \frac{\partial o_j}{\partial net_j}$$

$$= \sum_{k \in Downstream(j)} -\delta_k \, w_{kj} \, o_j(1 - o_j)$$

$$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j)\, o_j(1 - o_j)$$

$$\delta_j = (t_j - o_j)\, o_j(1 - o_j)$$

Xkj (input of output layer neuron) is the output fror hidden neuron oj

$$\frac{\partial\, net_k}{\partial o_j} = \frac{\partial x_{kj} w_{kj}}{\partial o_j} = \frac{\partial o_j w_{kj}}{\partial o_j}$$

$$\frac{\partial o_j}{\partial (net_j)} = \frac{\partial \sigma(net_j)}{\partial (net_j)}$$

$$= \sigma(net_j)\,(1 - \sigma(net_j))$$

$$= o_j\,(1 - o_j)$$

Case 2: unit j is an internal (hidden) unit

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} x_{ji} \qquad \frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream(j)} -\delta_k \, w_{kj} \, o_j(1 - o_j)$$

$$\Delta w_{ji} = \eta \, o_j(1 - o_j) \sum_{k \in Downstream(j)} \delta_k \, w_{kj} \, x_{ji}$$

$$\Delta w_{ji} = \eta \, \delta_j \, x_{ji}$$

$$\delta_j = o_j(1 - o_j) \sum_{k \in Downstream(j)} \delta_k \, w_{kj}$$

Algorithm: gradient_descent()

$t \leftarrow 0$;

$max\_iterations \leftarrow 1000$;

$Initialize \quad \theta_0 = [W_1^0, ..., W_L^0, b_1^0, ..., b_L^0]$;

while $t{+}{+} < max\_iterations$ **do**

$h_1, h_2, ..., h_{L-1}, a_1, a_2, ..., a_L, \hat{y} = forward\_propagation(\theta_t)$;

$\nabla\theta_t = backward\_propagation(h_1, h_2, ..., h_{L-1}, a_1, a_2, ..., a_L, y, \hat{y})$;

$\theta_{t+1} \leftarrow \theta_t - \eta\nabla\theta_t$;

end

Algorithm: forward_propagation(θ)

for $k = 1$ *to* $L - 1$ **do**

$a_k = b_k + W_k h_{k-1}$;

$h_k = g(a_k)$;

end

$a_L = b_L + W_L h_{L-1}$;

$\hat{y} = O(a_L)$;

Algorithm: back_propagation($h_1, h_2, ..., h_{L-1}, a_1, a_2, ..., a_L, y, \hat{y}$)

//Compute output gradient ;

$\nabla_{a_L} \mathscr{L}(\theta) = -(e(y) - \hat{y})$;

for $k = L$ *to* 1 **do**

  // Compute gradients w.r.t. parameters ;

  $\nabla_{W_k} \mathscr{L}(\theta) = \nabla_{a_k} \mathscr{L}(\theta) h_{k-1}^T$;

  $\nabla_{b_k} \mathscr{L}(\theta) = \nabla_{a_k} \mathscr{L}(\theta)$;

  // Compute gradients w.r.t. layer below ;

  $\nabla_{h_{k-1}} \mathscr{L}(\theta) = W_k^T (\nabla_{a_k} \mathscr{L}(\theta))$;

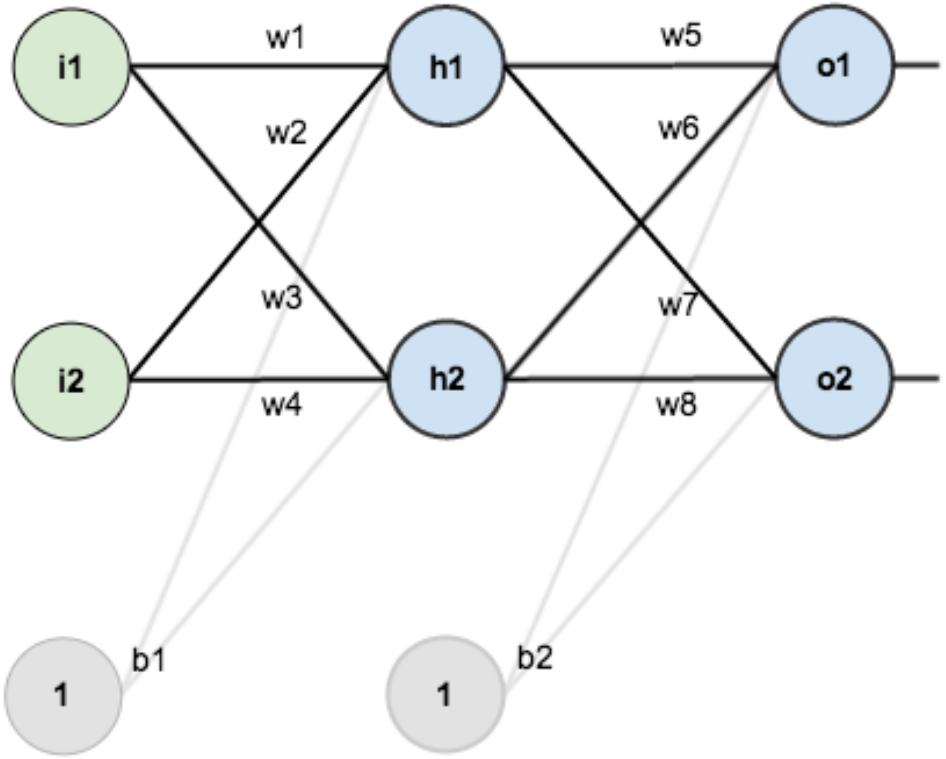
  // Compute gradients w.r.t. layer below (pre-activation);

  $\nabla_{a_{k-1}} \mathscr{L}(\theta) = \nabla_{h_{k-1}} \mathscr{L}(\theta) \odot [\ldots, g'(a_{k-1,j}), \ldots]$;

end

Numerical Example

Back propagation



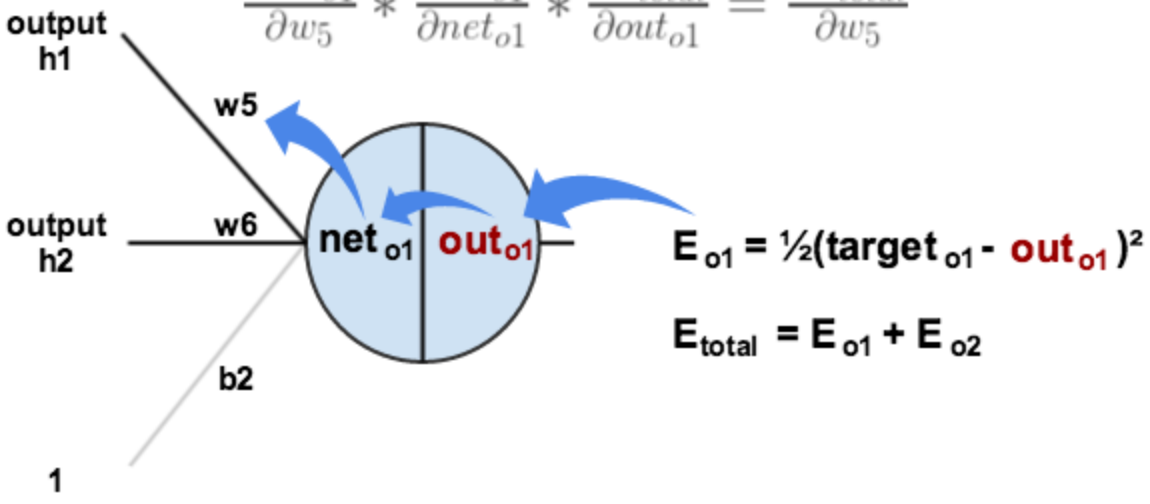
The goal of backpropagation is to optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs

Forward Pass: calculate *total net input* to each hidden layer neuron, *apply* an *activation function* and then repeat the process with the output layer neurons

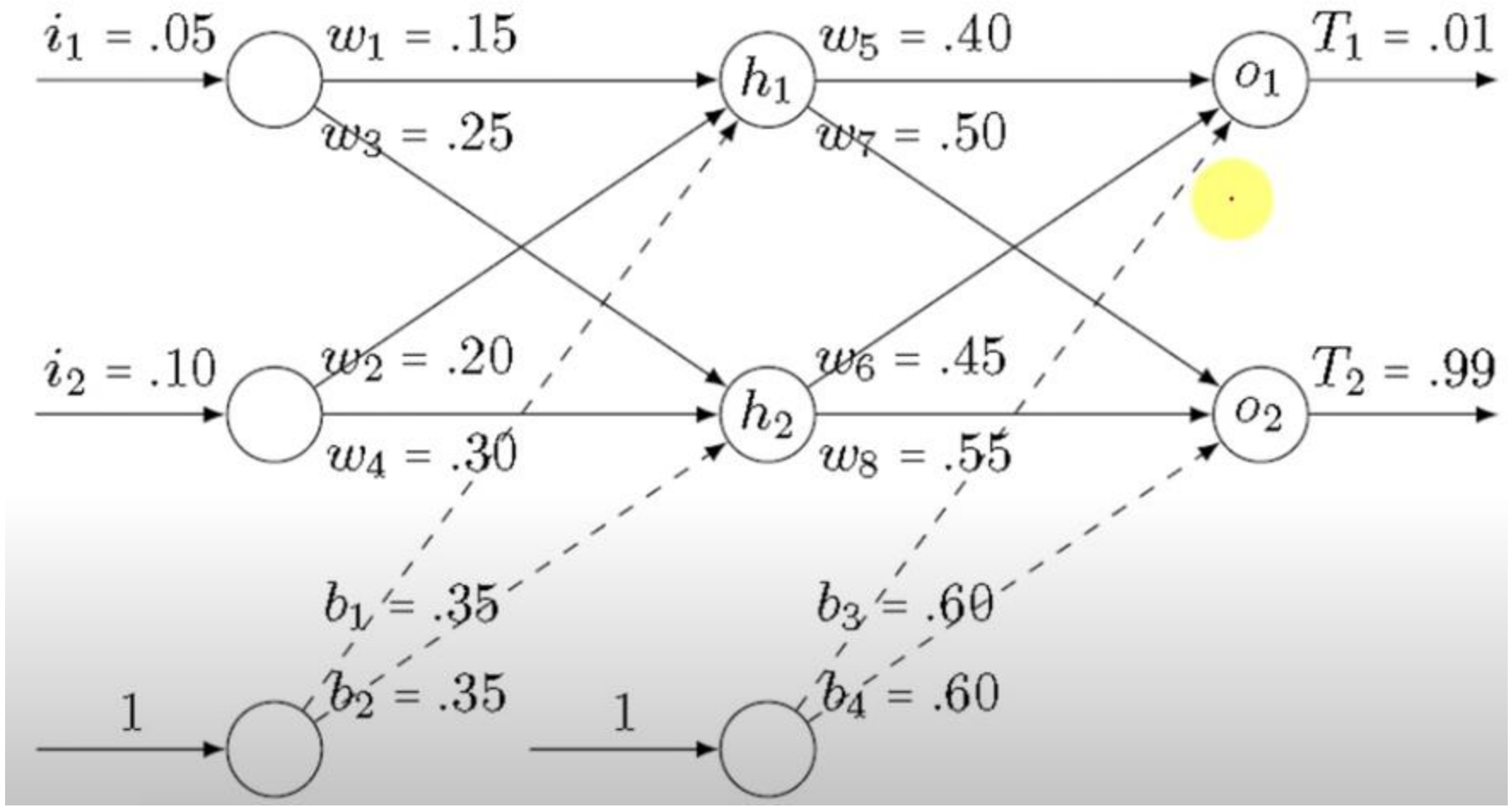
Backwards Pass: update the weights in the network so that they cause the actual output to be closer the target output, thereby minimizing the error

Compute the error

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$

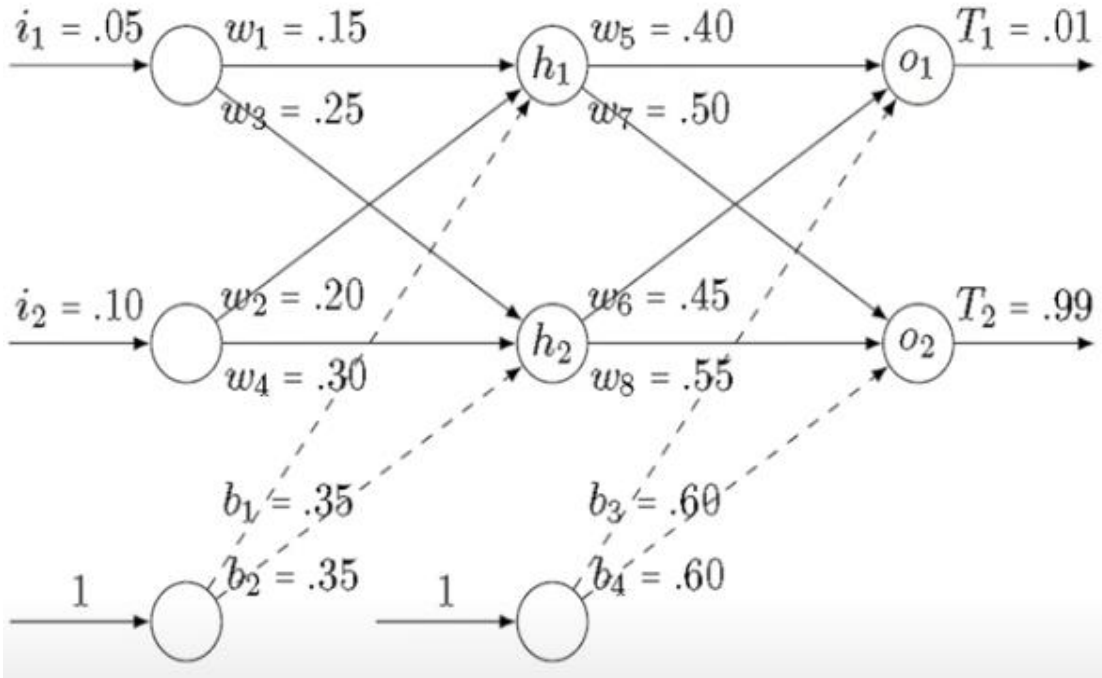


how much does the total net input of o1 change with respect to each weight?


$i_1 = .05$
$i_2 = .10$
$w_1 = .15$
$w_2 = .20$
$w_3 = .25$
$w_4 = .30$
h_1
h_2
$w_5 = .40$
$w_6 = .45$
$w_7 = .50$
$w_8 = .55$
o_1
o_2
$T_1 = .01$
$T_2 = .99$
$b_1 = .35$
$b_2 = .35$
$b_3 = .60$
$b_4 = .60$
1
1

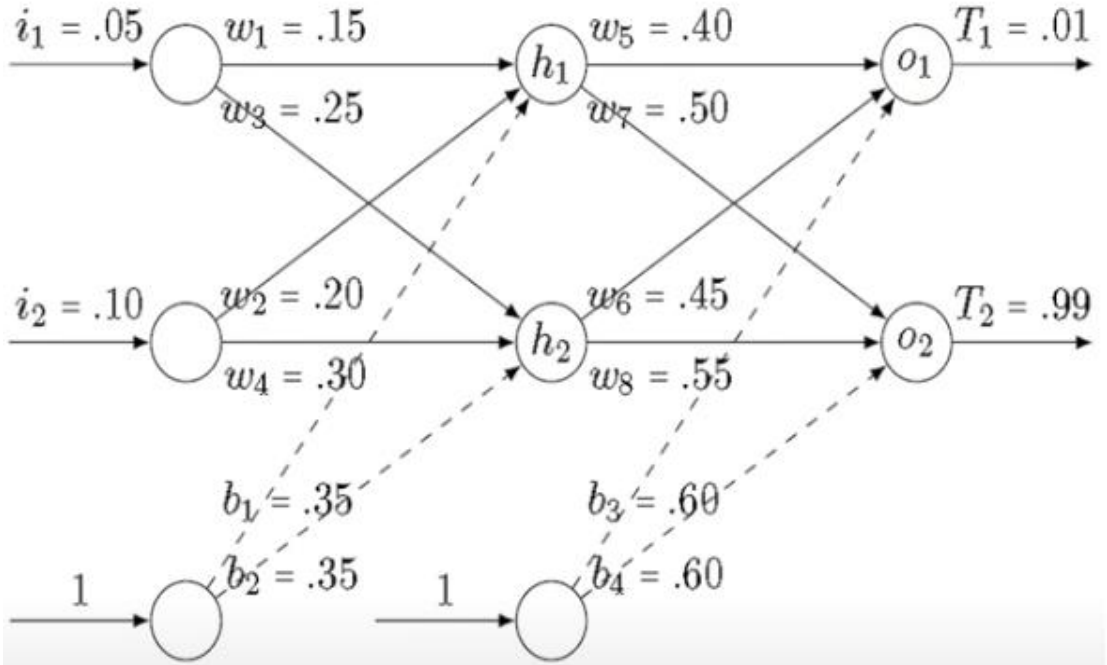
Output of h1 and h2

$$
\begin{aligned}
\text{out}_{h_1} &= f(w_1 \times i_1 + w_2 \times i_2 + b_1 \times 1) \\
&= f(0.15 \times 0.05 + 0.20 \times 0.10 + 0.35 \times 1) \\
&= f(0.3775) \\
&= \frac{1}{1 + e^{-0.3775}} \\
&= 0.59327
\end{aligned}
$$

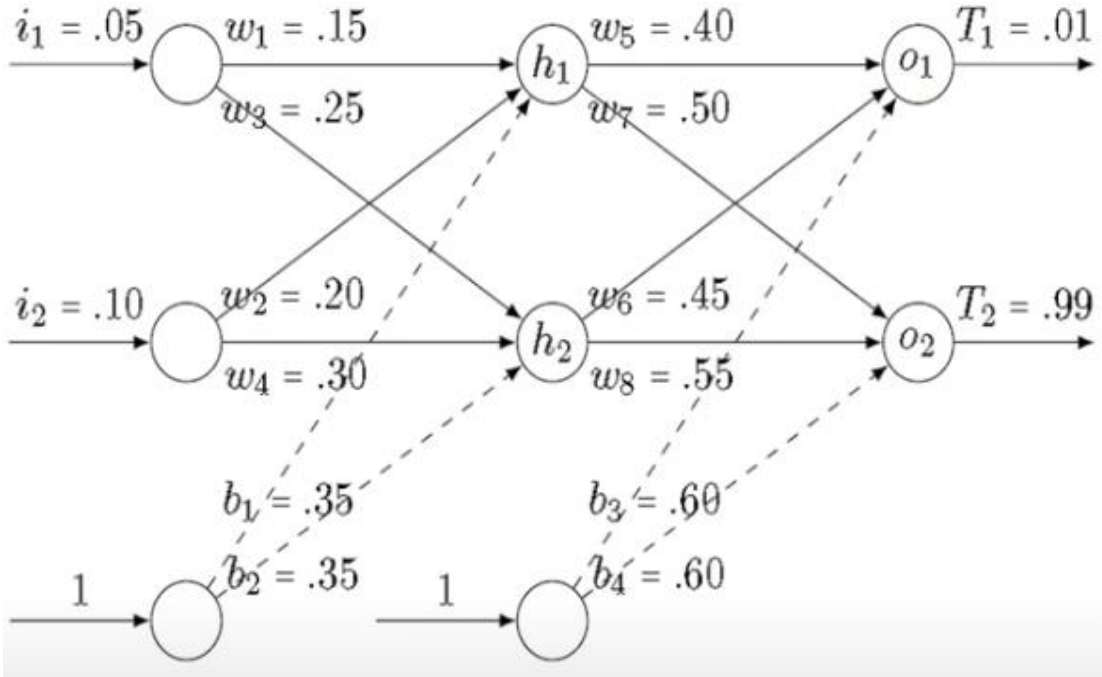


Output of h1 and h2

$$
\begin{aligned}
\text{out}_{h_2} &= f(w_3 \times i_1 + w_4 \times i_2 + b_2 \times 1) \\
&= f(0.25 \times 0.05 + 0.30 \times 0.10 + 0.35 \times 1) \\
&= f(0.3925) \\
&= \frac{1}{1 + e^{-0.3925}} \\
&= 0.59689
\end{aligned}
$$

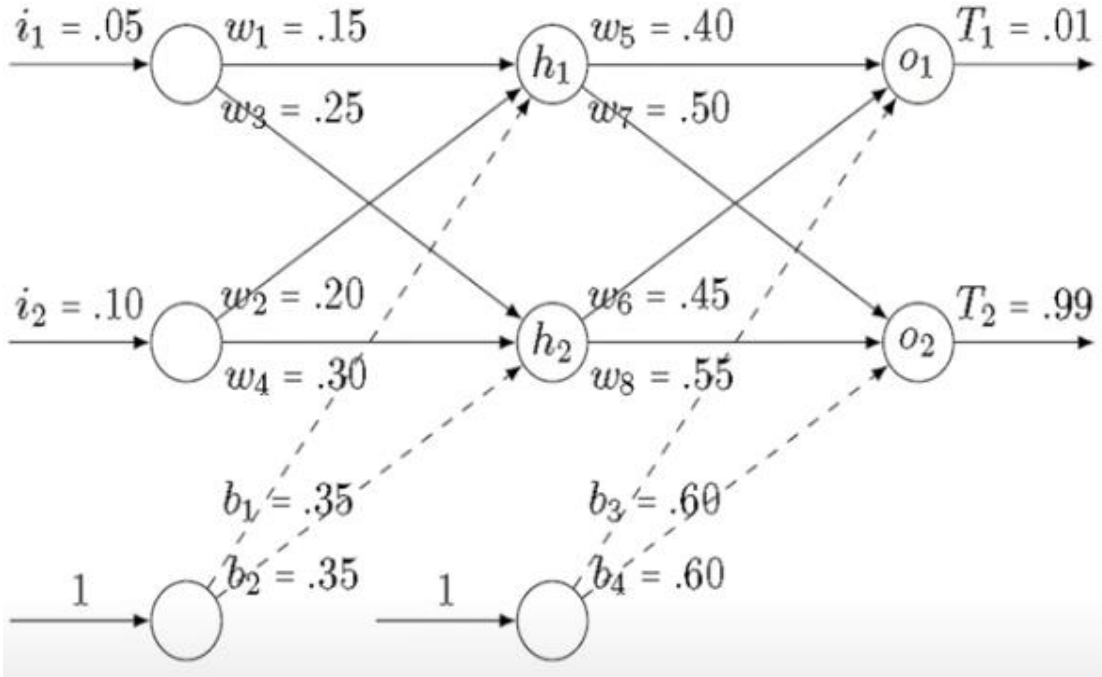


Output of O1 and O2



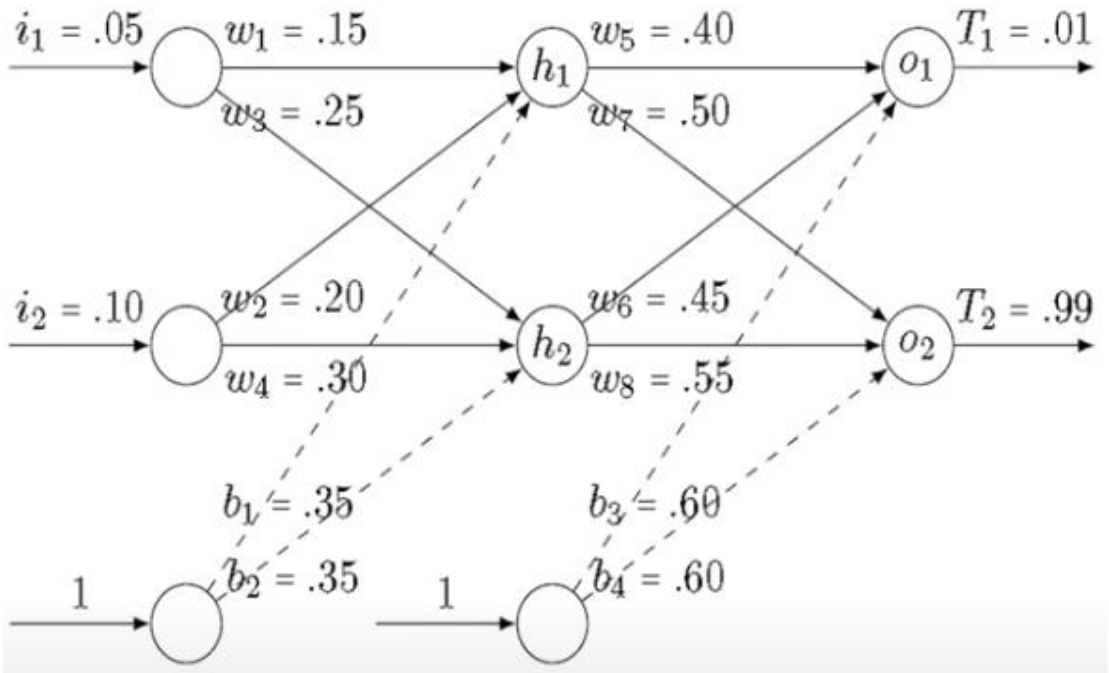
$$
\begin{aligned}
\text{out}_{o_1} &= f(w_5 \times \text{out}_{h_1} + w_6 \times \text{out}_{h_2} + b_3 \times 1) \\
&= f(0.40 \times 0.59327 + 0.45 \times 0.59689 + 0.60 \times 1) \\
&= f(1.10591) \\
&= \frac{1}{1 + e^{-1.10591}} \\
&= 0.75137
\end{aligned}
$$

Output of O1 and O2



$$
\begin{aligned}
\text{out}_{o_2} &= f(w_7 \times \text{out}_{h_1} + w_8 \times \text{out}_{h_2} + b_4 \times 1) \\
&= f(0.50 \times 0.59327 + 0.55 \times 0.59689 + 0.60 \times 1) \\
&= f(1.22492) \\
&= \frac{1}{1 + e^{-1.22492}} \\
&= 0.77293
\end{aligned}
$$

Error



$$E = \frac{1}{2}(T_1 - \text{out}_{o_1})^2 + \frac{1}{2}(T_2 - \text{out}_{o_2})^2$$

$$= (0.01 - 0.75137)^2 + (0.99 - 0.77293)^2$$

$$= 0.298371$$

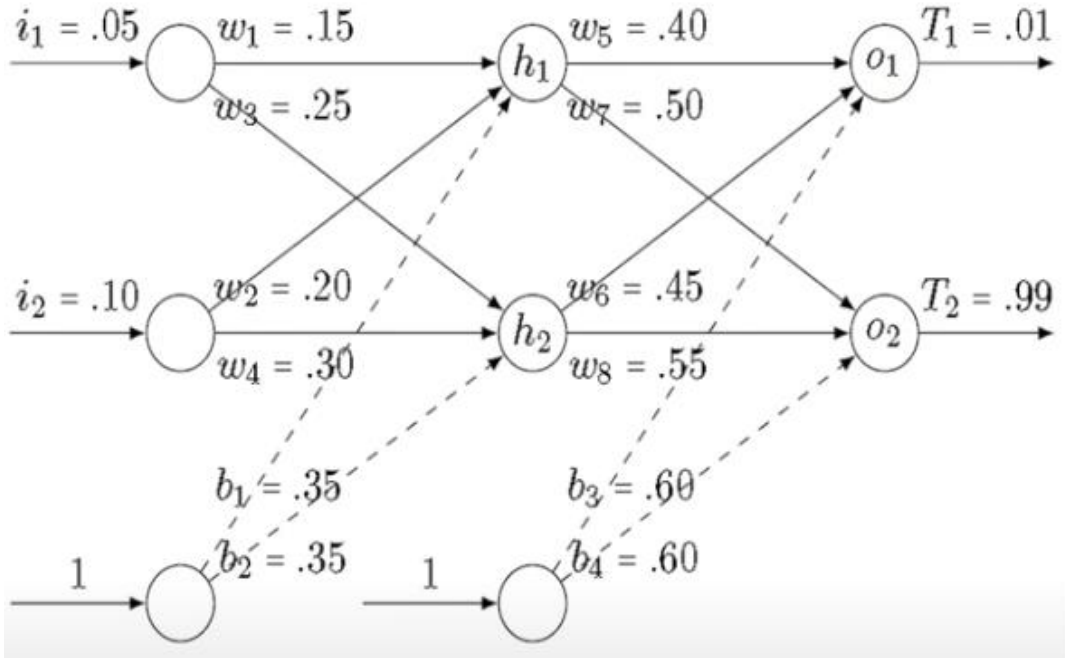
How to update weights?

$$\Delta w_{ji} = \eta \delta_j o_i$$

$$\delta_j = o_j(1-o_j)(t_j - o_j) \qquad \text{if } j \text{ is an output unit}$$

$$\delta_j = o_j(1-o_j)\sum_k \delta_k w_{kj} \qquad \text{if } j \text{ is a hidden unit}$$

Adjust the weights coming to O1



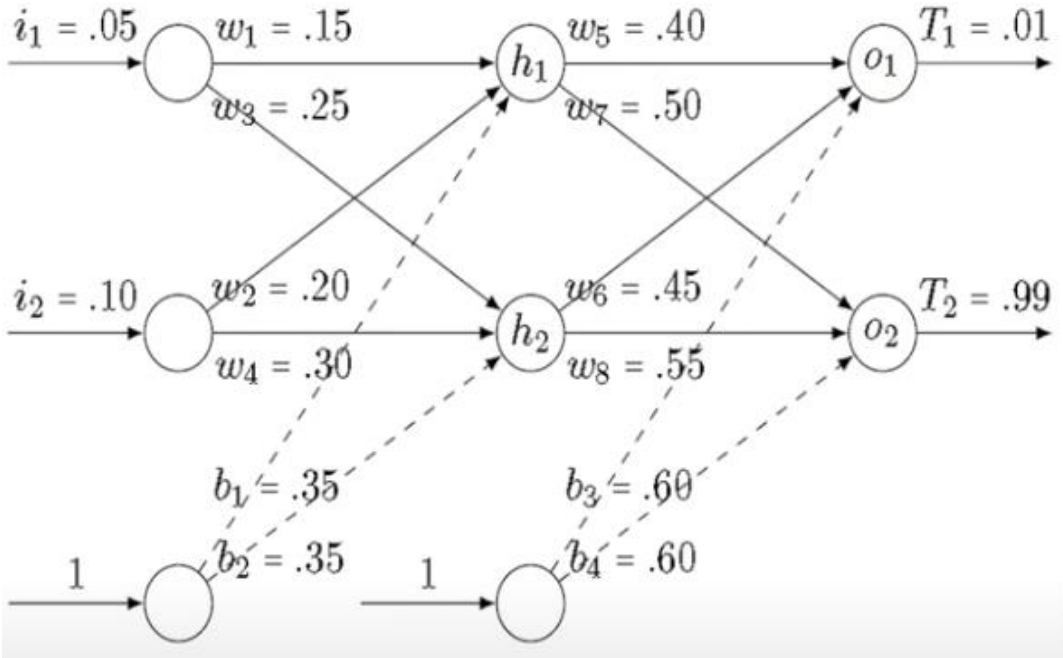
$$\begin{aligned}\delta_{o_1} &= (T_1 - \text{out}_{o_1}) \times \text{out}_{o_1} \times (1 - \text{out}_{o_1}) \\ &= (0.01 - 0.75137) \times 0.75137 \times (1 - 0.75137) \\ &= -0.13850\end{aligned}$$

$$\begin{aligned}w_5^+ &= w_5 + \eta \times \delta_{o_1} \times \text{out}_{h_1} \\ &= 0.40 + 0.5 \times (-0.13850) \times 0.59327 \\ &= 0.35892\end{aligned}$$

Adjust the weights coming to O1

$$\delta_{o_1} = (T_1 - \text{out}_{o_1}) \times \text{out}_{o_1} \times (1 - \text{out}_{o_1})$$
$$= (0.01 - 0.75137) \times 0.75137 \times (1 - 0.75137)$$
$$= -0.13850$$

$$w_5^+ = w_5 + \eta \times \delta_{o_1} \times \text{out}_{h_1}$$
$$= 0.40 + 0.5 \times (-0.13850) \times 0.59327$$
$$= 0.35892$$



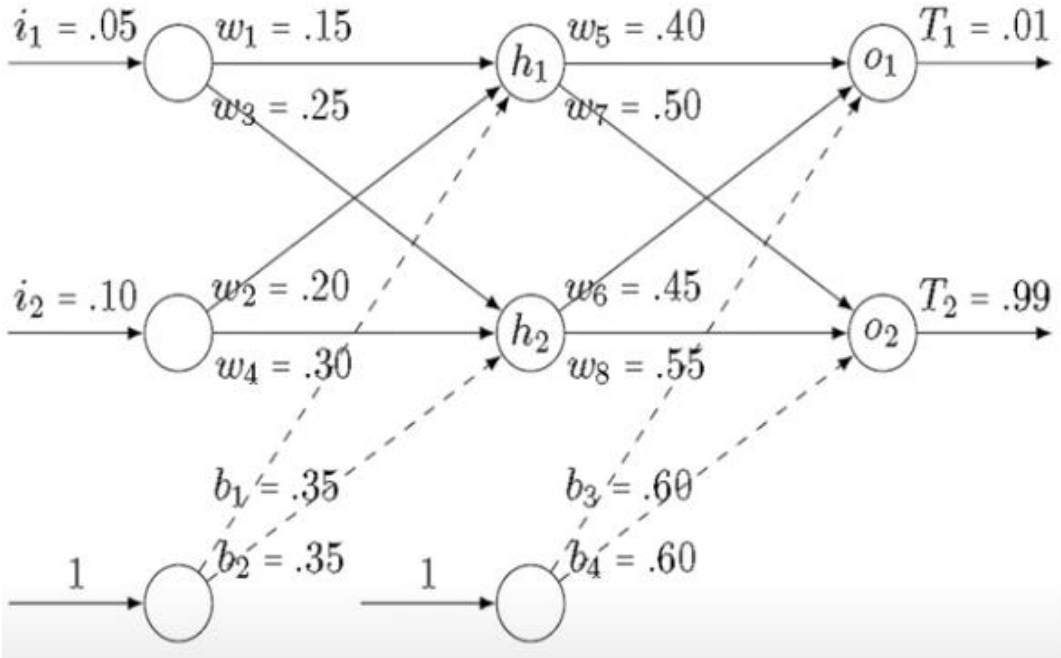
$$w_6^+ = w_6 + \eta \times \delta_{o_1} \times \text{out}_{h_2}$$
$$= 0.45 + 0.5 \times (-0.13850) \times 0.59689$$
$$= 0.40867$$

$$b_3^+ = b_3 + \eta \times \delta_{o_1} \times 1$$
$$= 0.60 + 0.5 \times (-0.13850) \times 1$$
$$= 0.53075$$

Adjust the weights coming to O2

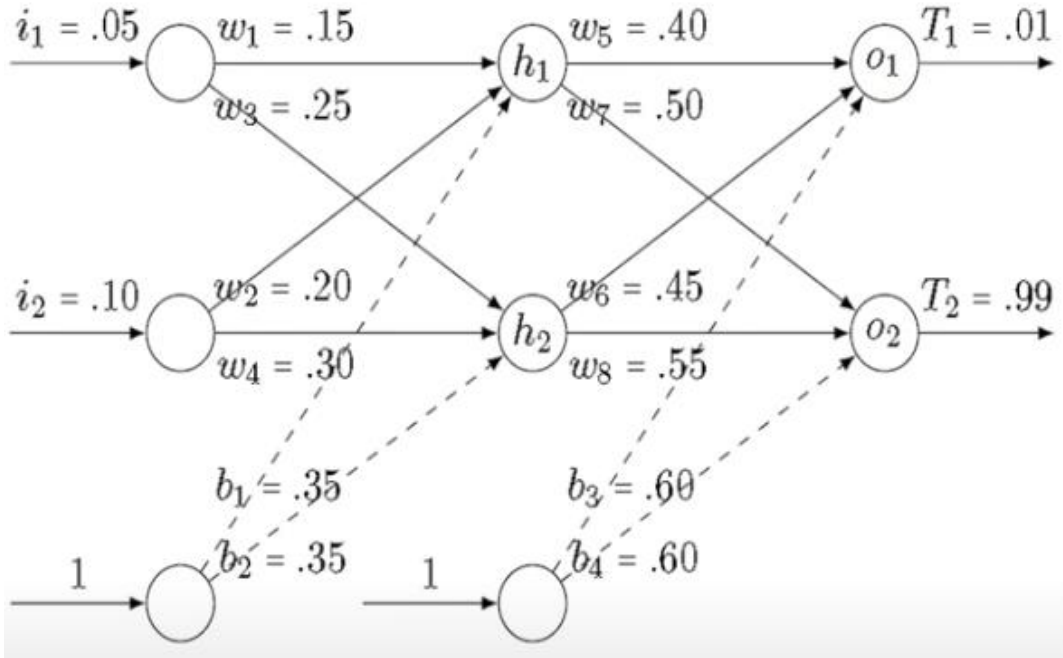
$$\delta_{o_2} = (T_2 - \text{out}_{o_2}) \times \text{out}_{o_2} \times (1 - \text{out}_{o_2})$$
$$= (0.99 - 0.77293) \times 0.77293 \times (1 - 0.77293)$$
$$= 0.03810$$

$$w_7^+ = w_7 + \eta \times \delta_{o_2} \times \text{out}_{h_1}$$
$$= 0.50 + 0.5 \times 0.03810 \times 0.59327$$
$$= 0.51130$$



Adjust the weights coming to O2

$$\begin{aligned}\delta_{o_2} &= (T_2 - \text{out}_{o_2}) \times \text{out}_{o_2} \times (1 - \text{out}_{o_2}) \\ &= (0.99 - 0.77293) \times 0.77293 \times (1 - 0.77293) \\ &= 0.03810\end{aligned}$$



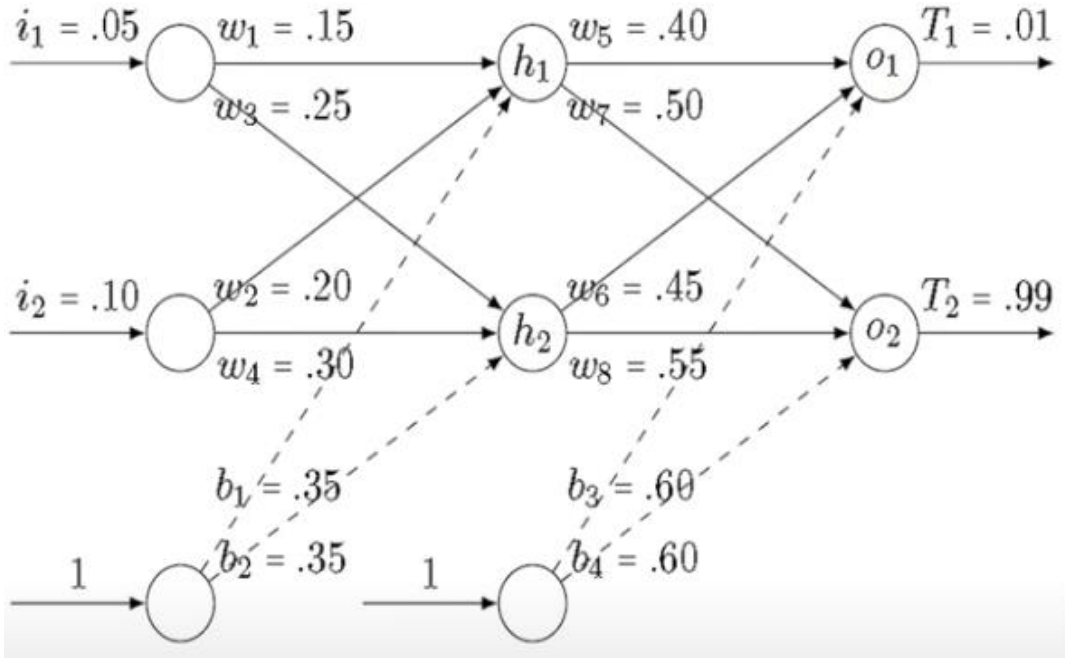
$$\begin{aligned}w_7^+ &= w_7 + \eta \times \delta_{o_2} \times \text{out}_{h_1} \\ &= 0.50 + 0.5 \times 0.03810 \times 0.59327 \\ &= 0.51130\end{aligned}$$

$$\begin{aligned}w_8^+ &= w_8 + \eta \times \delta_{o_2} \times \text{out}_{h_2} \\ &= 0.55 + 0.5 \times 0.03810 \times 0.59689 \\ &= 0.56137\end{aligned}$$

$$\begin{aligned}b_4^+ &= b_4 + \eta \times \delta_{o_2} \times 1 \\ &= 0.60 + 0.5 \times 0.03810 \times 1 \\ &= 0.61905\end{aligned}$$

Adjust the weights coming to h1

$$\begin{aligned}\delta_{h_1} &= (\delta_{o_1} \times w_5 + \delta_{o_2} \times w_7) \times \text{out}_{h_1} \times (1 - \text{out}_{h_1}) \\ &= (-0.13850 \times 0.40 + 0.03810 \times 0.50) \times 0.59327 \times (1 - 0.59327) \\ &= -0.00877\end{aligned}$$



$$\begin{aligned}w_1^+ &= w_1 + \eta \times \delta_{h_1} \times i_1 \\ &= 0.15 + 0.5 \times (-0.00877) \times 0.05 \\ &= 0.14978\end{aligned}$$

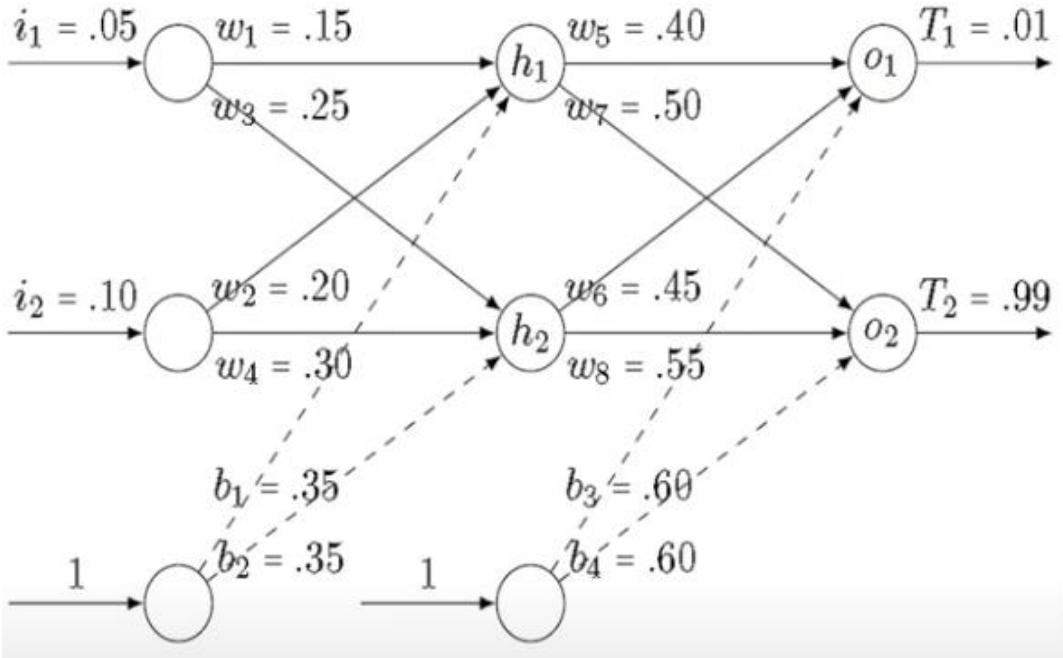
$$\begin{aligned}w_2^+ &= w_2 + \eta \times \delta_{h_1} \times i_2 \\ &= 0.20 + 0.5 \times (-0.00877) \times 0.10 \\ &= 0.19956\end{aligned}$$

$$\begin{aligned}b_1^+ &= b_1 + \eta \times \delta_{h_1} \times 1 \\ &= 0.35 + 0.5 \times (-0.00877) \times 1 \\ &= 0.34562\end{aligned}$$

Adjust the weights coming to h2

$$\delta_{h_2} = (\delta_{o_1} \times w_6 + \delta_{o_2} \times w_8) \times \text{out}_{h_2} \times (1 - \text{out}_{h_2})$$
$$= ((-0.13850) \times 0.45 + 0.03810 \times 0.55) \times 0.59689 \times (1 - 0.59689)$$
$$= -0.00995$$

$$w_3^+ = w_3 + \eta \times \delta_{h_2} \times i_1$$
$$= 0.25 + 0.5 \times (-0.00995) \times 0.05$$
$$= 0.24975$$



$$w_4^+ = w_4 + \eta \times \delta_{h_2} \times i_2$$
$$= 0.30 + 0.5 \times (-0.00995) \times 0.10$$
$$= 0.29950$$

$$b_2^+ = b_2 + \eta \times \delta_{h_2} \times 1$$
$$= 0.35 + 0.5 \times (-0.00995) \times 1$$
$$= 0.34503$$