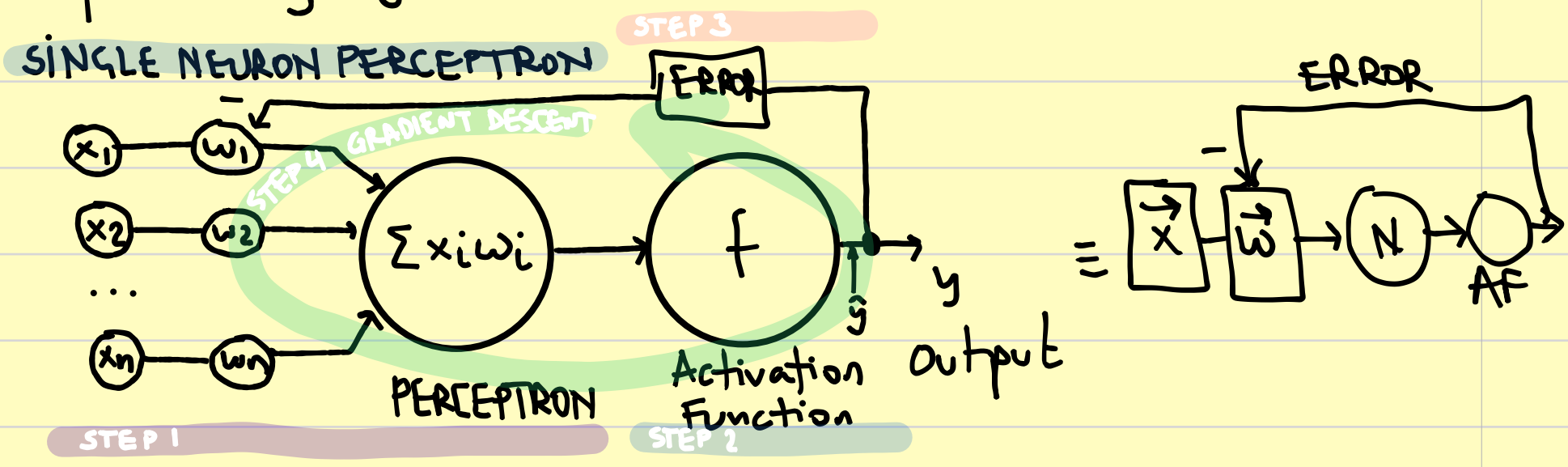
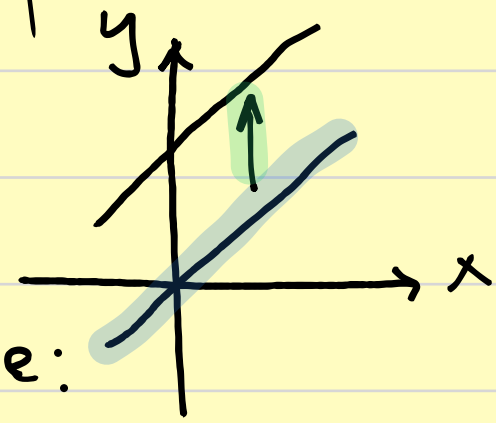
Deep Learning.. by hand..



Step 1. Multiply all input values x_i with their corresponding weights w_i and then calculate the weighted sum.

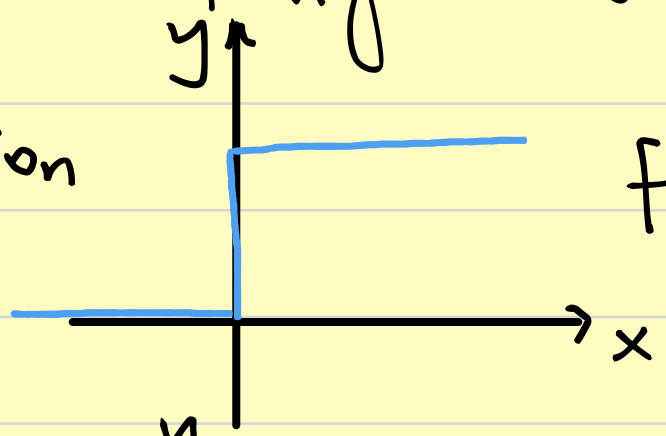
Step 2. Apply an activation function is applied to the above giving an output of the form:

$$\hat{y} = f\left(\sum x_i w_i + \text{bias}\right)$$



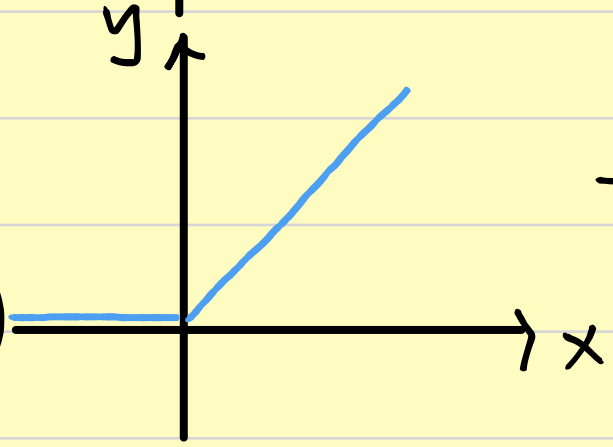
Examples of activation functions are:

- Step function



$$f(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$$

- Rectified Linear Unit (ReLU)



$$f(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases}$$

Step 3. We need to calculate the weight values which make the equation $\hat{y} = f\left(\sum w_i x_i + b\right)$ as similar as possible to a prediction.

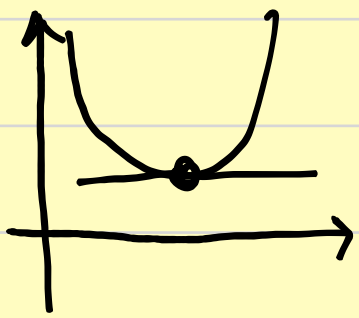
For this we use the quadratic of the difference between the two values $(y - \hat{y}) \equiv$ ERROR. This quadratic difference is called the quadratic error:

$(y - \hat{y})$: y ← PREDICTION, $\hat{y}$ ← CALCULATED VALUE.

$$\text{COST FUNCTION} = \frac{1}{2}\left(y - \hat{y}\right)^2.$$

Explanation: the error $(y - \hat{y})$ is squared so that all values are positive (they do not cancel each other).

we divide it by two because we want to find the minimum of the function C and for this we will find the first derivative of the cost function.



$$C = \frac{1}{2}\left(y - \hat{y}\right)^2 = \frac{1}{2}\left(y - \sum w_i x_i - b\right)^2$$

We call this function error or cost and we want to find the values of $\vec{w}$ $[w_1, w_2, \ldots, w_n]$ that minimize the cost.

$$\frac{dC}{dw_1} = \frac{1}{2} \cdot 2 \cdot \left[y - w_1 x_1 - w_2 x_2 - \cdots - w_n x_n \right] \cdot (-x_1)$$

$$\frac{dC}{dw_2} = \frac{1}{2} \cdot 2 \cdot \left[y - w_1 x_1 - w_2 x_2 - \cdots - w_n x_n \right] \cdot (-x_2)$$

$$\cdots$$

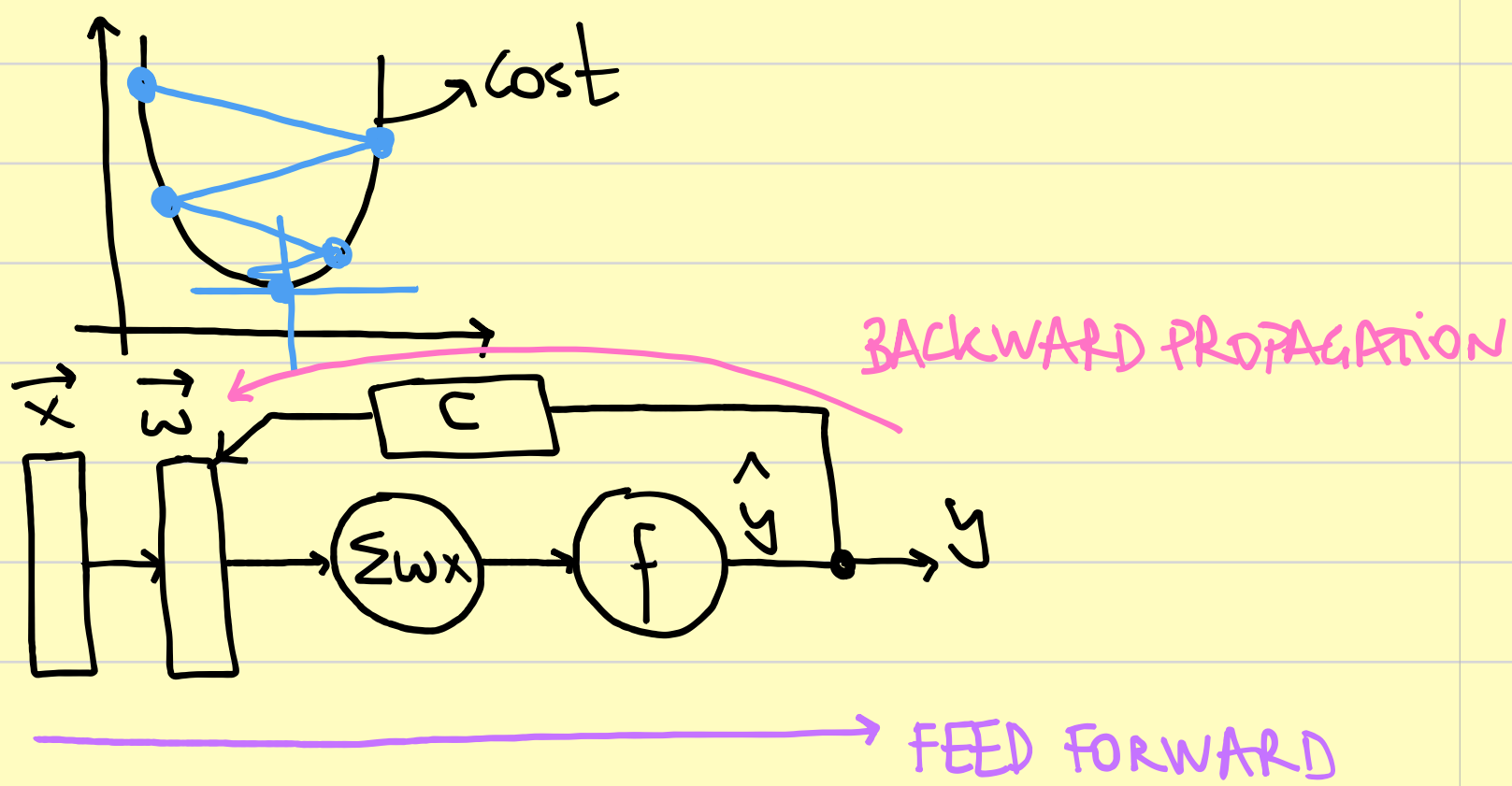
$$\frac{dC}{dw_n} = \frac{1}{2} \cdot 2 \cdot \left[y - w_1 x_1 - w_2 x_2 - \cdots - w_n x_n \right] \cdot (-x_n)$$

These derivatives are called gradients.

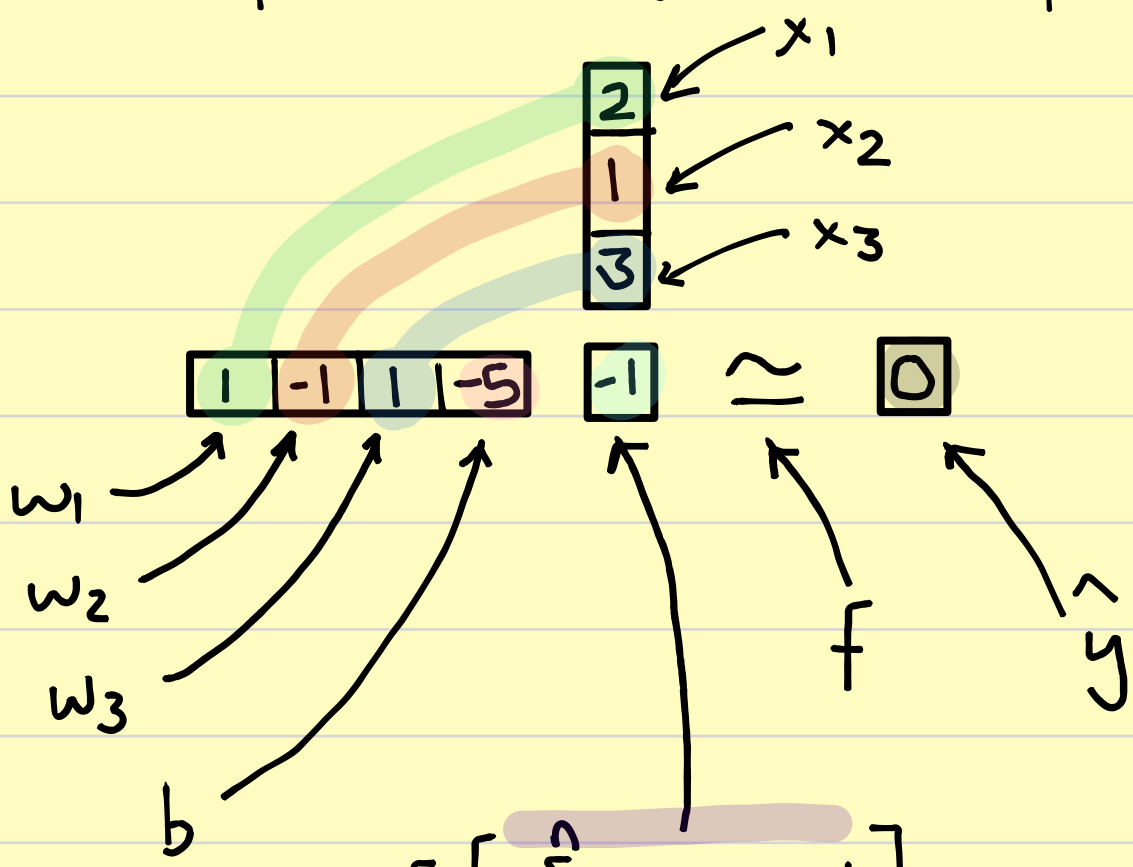
Step 4. Gradient descent

In order to reach a minima in the cost function we will start taking small steps in the direction of the minima (opposite direction of the gradient).

We do so by updating the weights by a small amount. This amount is called "learning rate".



Example 1. Single neuron perception forward pass



$$\vec{x} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix} \begin{matrix} x_1 \\ x_2 \\ x_3 \end{matrix} \qquad \vec{w} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \begin{matrix} w_1 \\ w_2 \\ w_3 \end{matrix} \qquad b = -5$$

$$\hat{y} = f\left[\sum_{i=1} w_i x_i + b\right]$$

$$f \equiv \text{ReLU}: \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases}$$

$$\hat{y} = f\left[\left(w_1 \cdot x_1 + w_2 x_2 + w_3 x_3 + b\right)\right] =$$

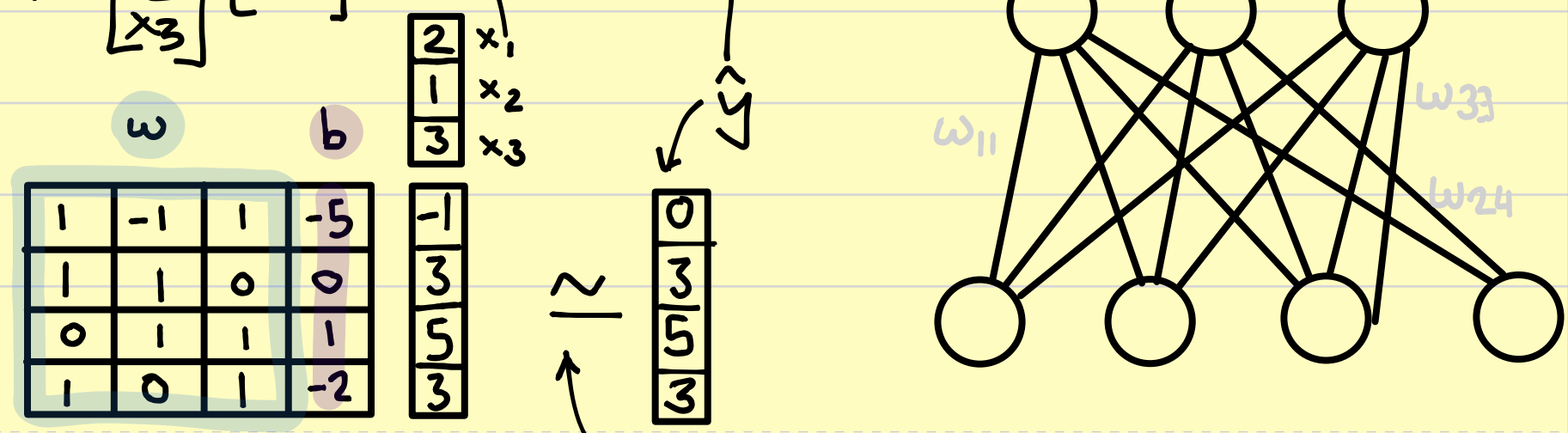
$$= f\left[\left(1 \cdot 2 + -1 \cdot 1 + 1 \cdot 3 + (-5)\right)\right] =$$

$$= f\left[2 - 1 + 3 - 5\right] = f\left[-1\right] = 0$$

Example 2. Calculate the output of two layers of perceptrons with 3 and 4 neurons respectively.

FORWARD PASS

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} [1 \times 3] \qquad x \; [x_1 \quad x_2 \quad x_3]$$



$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \quad f_{ReLU}$$

$$\hat{y}\,[\,\hat{y}_1 \quad \hat{y}_2 \quad \hat{y}_3 \quad \hat{y}_4\,]$$

Fully connected multi. layer perceptron.

$[3\times4]$ $\qquad$ $[1\times4]$

$$f\left(\sum w_i x_i + b\right) = f\left(\begin{bmatrix}2 & 1 & 3\end{bmatrix}\begin{bmatrix}1 & -1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1\end{bmatrix} + \begin{bmatrix}-5 \\ 0 \\ 1 \\ -2\end{bmatrix}\right) = f\left(\begin{bmatrix}-1 \\ 3 \\ 5 \\ 3\end{bmatrix}\right) =$$

$$= \begin{bmatrix}0 \\ 3 \\ 5 \\ 3\end{bmatrix} = \hat{y}$$

$$2\cdot1 + 1\cdot(-1) + 3\cdot1 - 5 = -1$$
$$2\cdot1 + 1\cdot1 + 3\cdot0 + 0 = 3$$
$$2\cdot0 + 1\cdot1 + 3\cdot1 + 1 = 5$$
$$2\cdot1 + 1\cdot0 + 3\cdot1 - 2 = 3$$

Example 3. 3 Layer perceptron (forward pass) $[3, 4, 2]$

HIDDEN LAYER

$\vec{x}\,[x_1 \quad x_2 \quad x_3]$

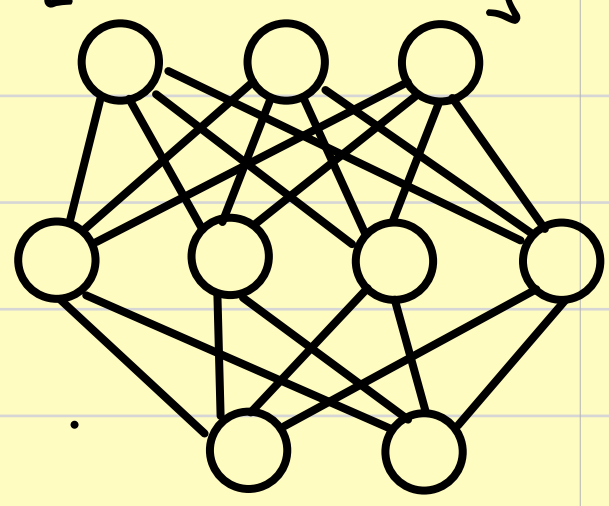
$[3\times4]$ with $\qquad$ $[1\times4]$ b_1

1	−1	1	−5
1	1	0	0
0	1	1	1
1	0	1	−2

x_1 2, x_2 1, x_3 3

$$\begin{bmatrix}-1 \\ 3 \\ 5 \\ 3\end{bmatrix} \simeq \text{ReLU} \quad \hat{h}: \begin{bmatrix}0 \\ 3 \\ 5 \\ 3\end{bmatrix}$$

h



$[4\times2]$ $\qquad$ $[1\times2]$

w_{h_2} $\qquad$ b_2

1	1	−1	0	0
0	0	1	−1	1

$$\begin{bmatrix}-2 \\ 3\end{bmatrix} \simeq \text{ReLU} \quad \hat{y}: \begin{bmatrix}0 \\ 3\end{bmatrix}$$

$[1\times2]$

$\hat{y}\,[\hat{y}_1 \quad \hat{y}_2]$

$$f(\sum w_{h_2} \cdot \hat{h} + b_2) = f\left(\begin{bmatrix} 1 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}\begin{bmatrix} 0 \\ 3 \\ 5 \\ 3 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right) = f\left(\begin{bmatrix} -2 \\ 3 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$$

$1.0+1.3+(-1)\cdot 5+0.3+0 = -2$

$0.0+0.3+1.5+(-1)\cdot 3+1 = 3$

Example 4. Perception with hidden layer [3,4,2,3] with feed forward & Back propagation.

SOFT MAX $\vec{x} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix} \rightarrow Sf+Max(\vec{x}) = \begin{bmatrix} \frac{2}{2+1+3} \\ \frac{1}{2+1+3} \\ \frac{3}{2+1+3} \end{bmatrix} = \begin{bmatrix} 2/6 \\ 1/6 \\ 3/6 \end{bmatrix} = \begin{bmatrix} 0'33 \\ 0'16 \\ 0'5 \end{bmatrix}$

$\sum = 1$

$\vec{x}[x_1\ x_2\ x_3]$

h_1

h_2

y

$\hat{y}\ [\hat{y}_1\ \hat{y}_2\ \hat{y}_2]$

- Decission layer.

$\vec{x}$

2	2	-4	4	-2
1	1	-2	2	-1
3	3	-6	6	-3
	1	-2	2	-1

dot product

$w_1 h_1^*$

b_1^*

$w_1 h_1$ b_1

1	-1	1	-5	-1
1	1	0	0	3
0	1	1	1	5
1	0	1	-2	3

$\simeq$ ReLU

0	0	0		
3	3	-3		
5	5	-5		
3	3	-3		

$w_{h_1} h_2^*$ dot product

1	-2	2	-1	
	1	-1		

b_2^* 2.0'23 2.(-0'77) 2.054 dot product!

$w_{h_1 h_2}$ b_2

1	-1	1	0	0
0	1	-1	1	3

error$_1$

2	0'46	-1'54	1'08	②
4	0'92	-3'08	2'16	

$w_{h_2,3}^*$

$\simeq$ ReLU

1	-1	

b_3^* ①

0'23	-0'77	0'54

ERROR$_2$

③

$0'23 \cdot 2 + 0 \cdot (-0'77) + 1 \cdot 0'54$

$0'23 \cdot 0 + 2 \cdot (-0'77) + 1 \cdot 0'54$

2	0	-1
0	2	-5
1	1	1

$w_{h_2 3}$ b_3

3
3
7

$\simeq$ Soft Max (*)

0'23
0'23
0'54

$\hat{y}$

$-$

0
1
0

y

$=$

0'23
-0'77
0'54

ERROR$_3$

(*) $\text{SoftMax}\begin{bmatrix}3\\3\\7\end{bmatrix} = \begin{bmatrix}\frac{3}{3+3+7}\\ \frac{3}{3+3+7}\\ \frac{7}{3+3+7}\end{bmatrix} = \begin{bmatrix}0'23\\0'23\\0'54\end{bmatrix}$

PREDICTION TARGET

Back Propagation:

1. Transpose the Error vector = $\hat{y} - y$
2. Dot. Product of the transposed error_3 and the output of previous layer
3. Multiply the transposed error_3 with weight matrix of current layer. $w_{h2,3} \cdot \text{error}_3^t$
4. Repeat until we are above.

Now we repeat our feed forward and backward propagation until the weights deliver a minimum in the cost function.

$\begin{bmatrix}2\\1\\3\end{bmatrix}$

w_{1,h_1}^* b_1^*

2	1	3	1
-4	-2	-6	-2
4	2	6	2
2	-1	3	-1

$\begin{bmatrix}15\\-30\\30\\11\end{bmatrix} \simeq_{\text{ReLU}} \begin{bmatrix}15\\0\\30\\11\end{bmatrix}$

w_{h_1,h_2}^* b_2^*

0	3	5	3	1
0	-3	-5	-3	-1

$\begin{bmatrix}184\\-184\end{bmatrix} \simeq_{\text{ReLU}} \begin{bmatrix}184\\0\end{bmatrix}$

0'46	0'92	0'23
-1'54	-3'08	-0'77
1'08	2'16	0'54

84'87
-284'13
199'26

$\simeq$ Soft Max

0'15
0'5
0'35

PRED

$\sim$

0
1
0

TARGET

$=$

$$\text{SoftMax}\begin{bmatrix}84'87\\-284'13\\199'26\end{bmatrix} = \begin{bmatrix}84'87\\284'13\\199'26\end{bmatrix} \cdot \frac{1}{568'26} = \begin{bmatrix}0'15\\0'5\\0'35\end{bmatrix} = \begin{bmatrix}-0'15\\-0'5\\-0'35\end{bmatrix} \text{error}_3^*$$

$84'87 + 284'13 + 199'26 = 568'26$