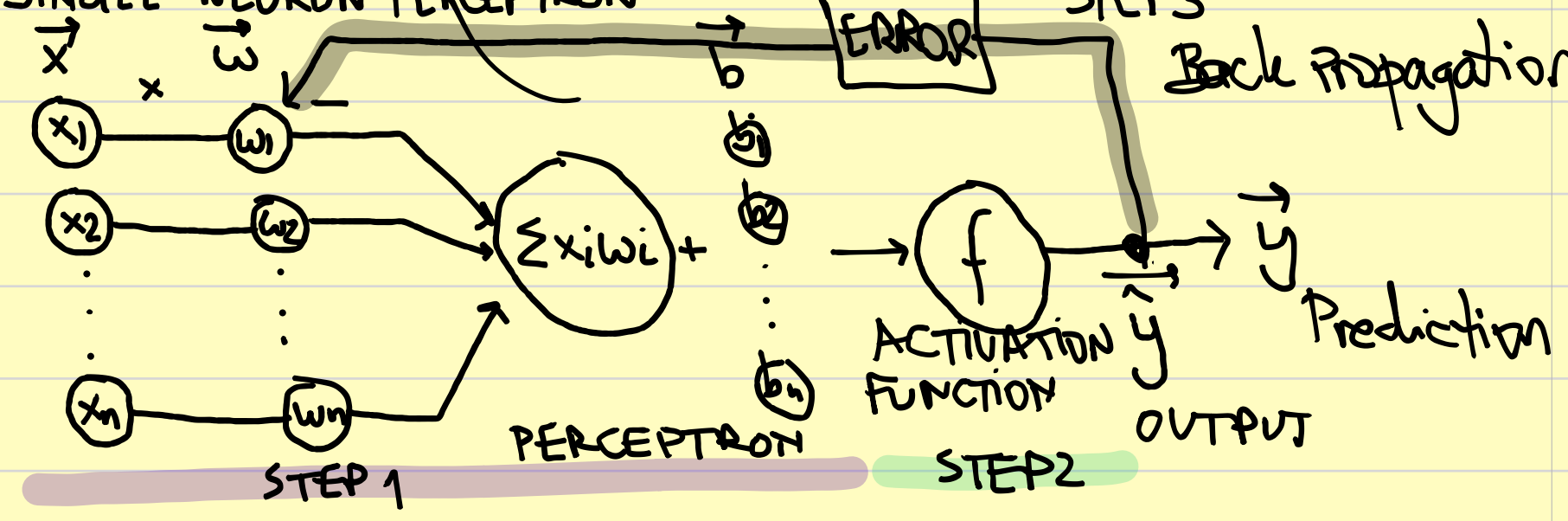
Deeplearning ..by hand

$(\hat{y}-y)^2 \equiv$ error

- SINGLE NEURON PERCEPTRON



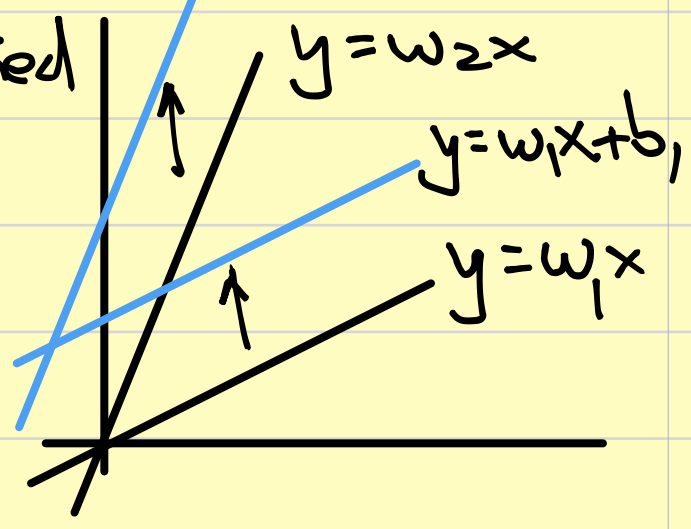
Step 1. Multiply all input values x_i with their corresponding weights w_i and calculate the weighted sum. Then we add a bias.

$$\sum_{i=1}^{n} w_i x_i$$

$y = w_2 x + b_2$

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

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



Examples of Activation Functions:

- Step function:

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

- Rectified Linear Unit: ReLU

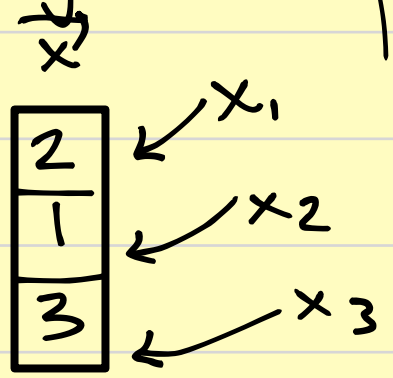
$$f(x) = \begin{cases} 0 & x<0 \\ x & x \geq 0 \end{cases}$$

Step 3. we need to calculate the weight values which make the equation $\hat{y} = f(\sum x_i w_i + b_i)$ as similar as possible to a prediction. For this, we use the quadratic of the difference btw the two values $(y - \hat{y})$ ≡ ERROR and make this squared so that ≥ 0.

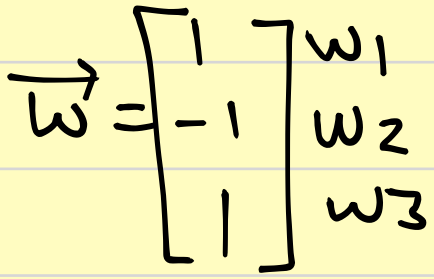
$$\text{Cost} = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}\left(y - \sum x_i w_i + b_i\right)^2$$

Prediction — Output

Example 1. single neuron perceptron. FORWARD PASS.

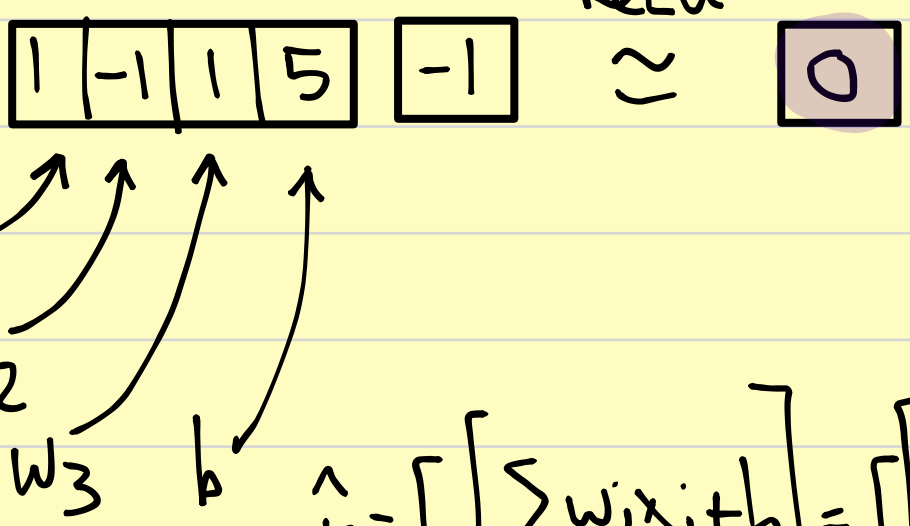


$$\vec{x} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix} \begin{matrix} x_1 \\ x_2 \\ x_3 \end{matrix}$$



$$\vec{w} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \begin{matrix} w_1 \\ w_2 \\ w_3 \end{matrix}$$

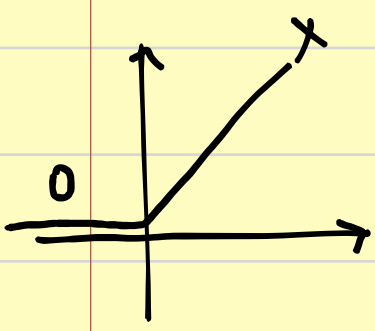
$b = 5$



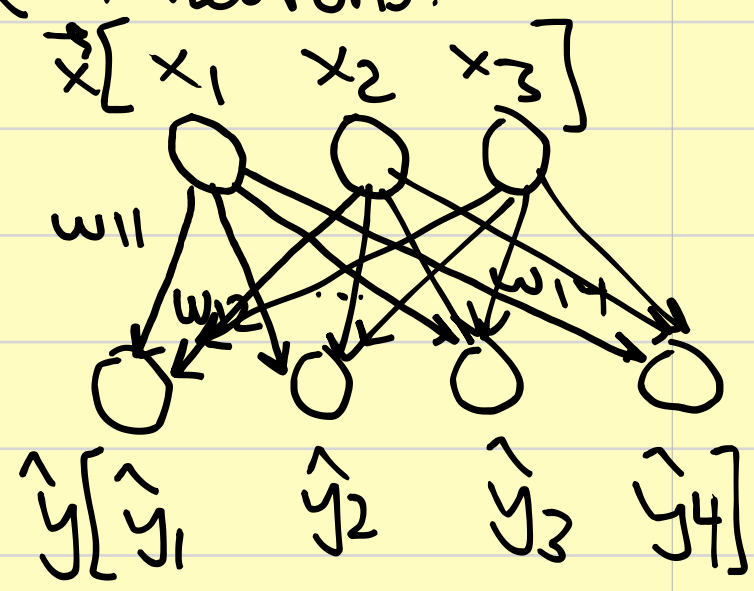
$$\hat{y} = f\left[\sum w_i x_i + b\right] = f\left[\left(w_1 x_1 + w_2 x_2 + w_3 x_3 + b\right)\right] =$$

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

$$= \text{ReLU}(-1)$$



Example 2. Calculate the output of two layers of Perceptrons with 3 & 4 neurons.



(·) $1 \cdot 2 + (-1) \cdot 1 + 1 \cdot 3 - 5 = -1$

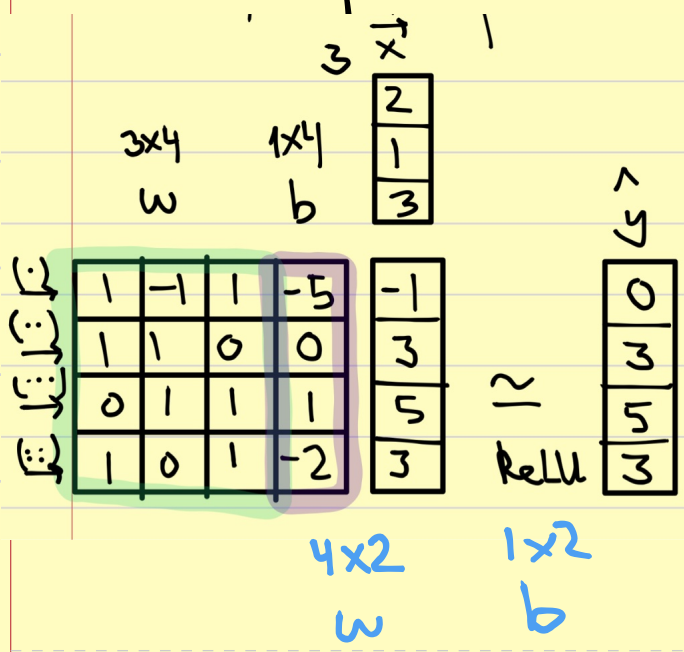
(··) $1 \cdot 2 + 1 \cdot 1 + 0 \cdot 3 + 0 = 3$

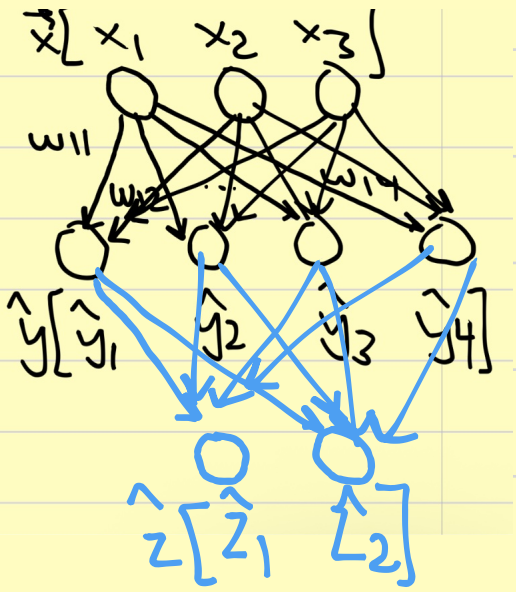
(···) $0 \cdot 2 + 1 \cdot 1 + 1 \cdot 3 + 1 = 5$

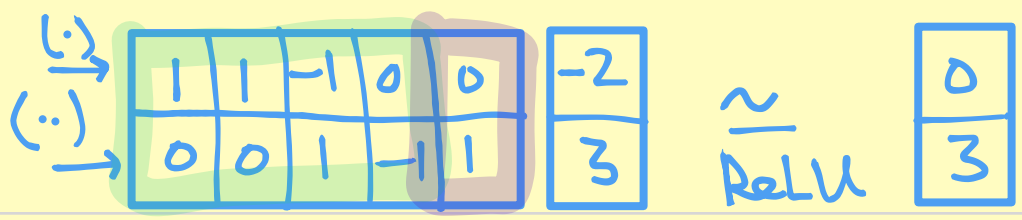
(::) $1 \cdot 2 + 0 \cdot 1 + 1 \cdot 3 - 2 = 3$

Example 3. Add a 2 neuron layer after the 2nd layer of the previous example.

FEED FORWARD



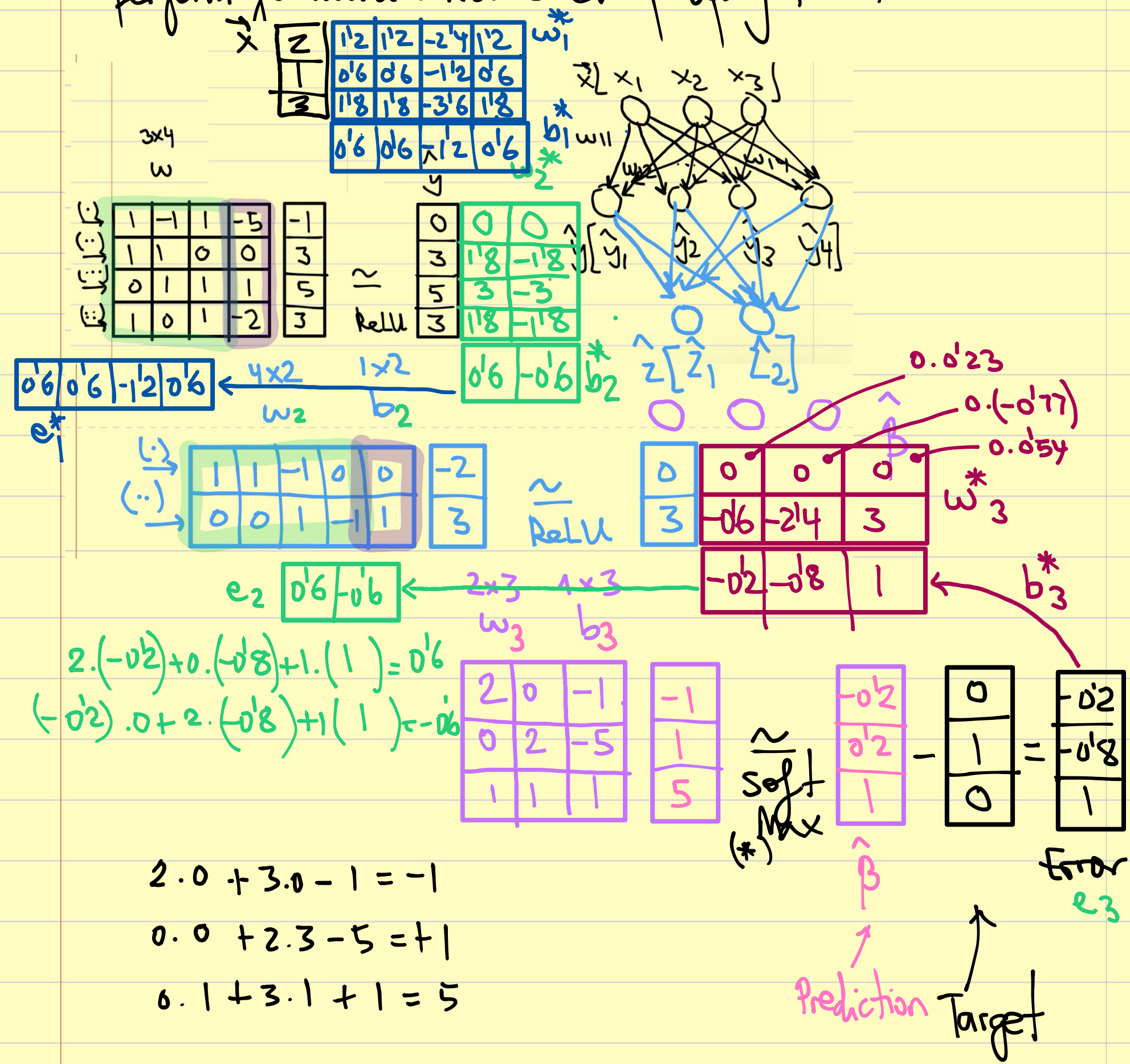




$(\cdot)\ 0.1+3.1+5.(-1)+3.0+0=-2$

$(\cdot\cdot)\ 0.0+3.0+5.1+3(-1)+1=3$

Example 4. Add another layer with 3 neurons, perform forward and back propagation.

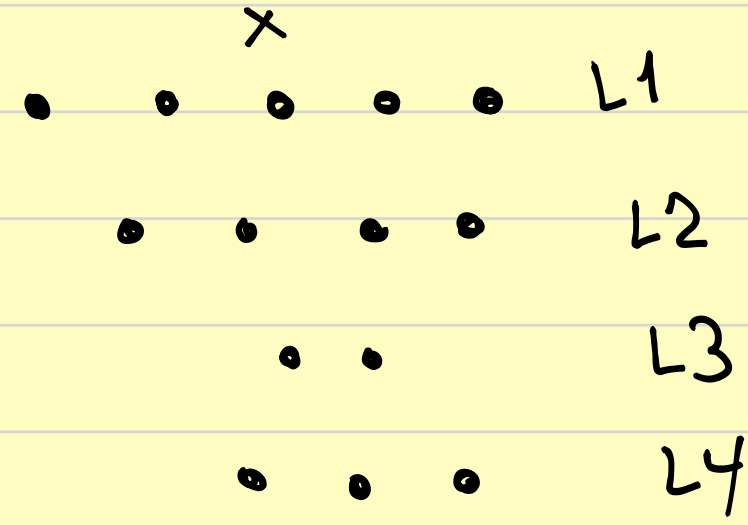


(*) In the last layer we do not implement ReLU, rather we generate a probability distribution in the vector. We do so with a SOFT-MAX function. This function makes sure that the sum of all values in the vector is ONE.

$$\text{Softmax}\begin{bmatrix}3\\3\\7\end{bmatrix} = \begin{bmatrix}\frac{3}{3+3+7}\\ \frac{3}{3+3+7}\\ \frac{7}{3+3+7}\end{bmatrix} = \begin{bmatrix}0'23\\0'23\\0'54\end{bmatrix}$$

Exercise to test your skills:

$$\vec{x} = \begin{bmatrix}1\\2\\3\\4\\5\end{bmatrix}$$



$$\vec{w}_1 = \left[\begin{array}{cccc|c} 1 & 1 & -1 & 0 & 1 \\ 1 & -1 & 0 & 1 & -1 \\ 1 & 0 & 1 & -1 & 1 \\ 1 & 1 & -1 & 0 & -1 \end{array}\right] \quad b_1 = \begin{bmatrix}2\\-2\\0\\0\end{bmatrix}$$

$$\vec{w}_2 = \left[\begin{array}{cccc|c} 1 & 0 & 1 & -1 & 1 \\ 0 & -1 & -1 & 0 & -1 \end{array}\right] \quad b_2$$

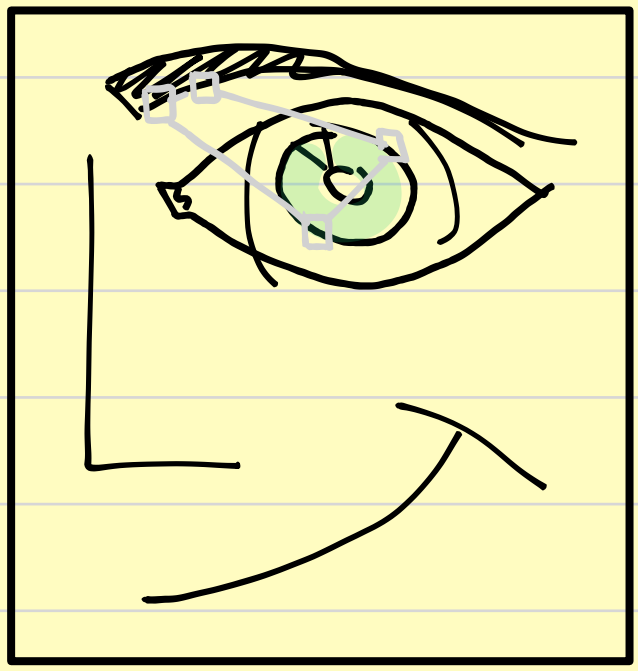
$$\vec{w}_3 = \left[\begin{array}{cc|c} 1 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & -1 & 2 \end{array}\right] \quad b_3$$

$$\hat{y} \equiv \text{TARGET} \equiv \begin{bmatrix}0\\0\\1\end{bmatrix}$$

⊙ Perform a feed forward (5p) and a back prop. (15p) in this perception layers.

Remarks.

- Classic deep learning works only on EUCLIDEAN Datasets!



$$d = \sqrt{(x-x_1)^2 + (y-y_1)^2}$$

. Example of non-Euclidean domains: NETWORK!

Book: Deeplearning (Adaptive Computation & ML Series) 2016 [Goodfellow, Bengio, Courville]