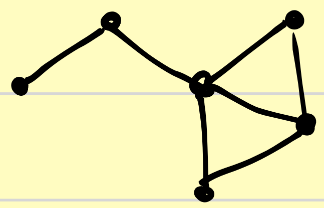
... NETWORK TOPOLOGY

NEW NODE



GRAPH $G = \{N, E\}$

What is the probability that this new node connects to an existing node that already has "k" neighbors?

(a) RANDOM NETWORK
(b) REAL NETWORK

(a) RANDOM NETWORK.

The probability that a node with "k" neighbours attaches to a new node in a random network is given by a Poisson distribution with frequency parameter "λ":

$$\boxed{P_R(X=k) = \frac{\lambda^k \cdot e^{-\lambda}}{k!}} \quad (1)$$

$$k! = k(k-1)(k-2)\ldots 3 . 2 . 1$$

(b) REAL NETWORK.

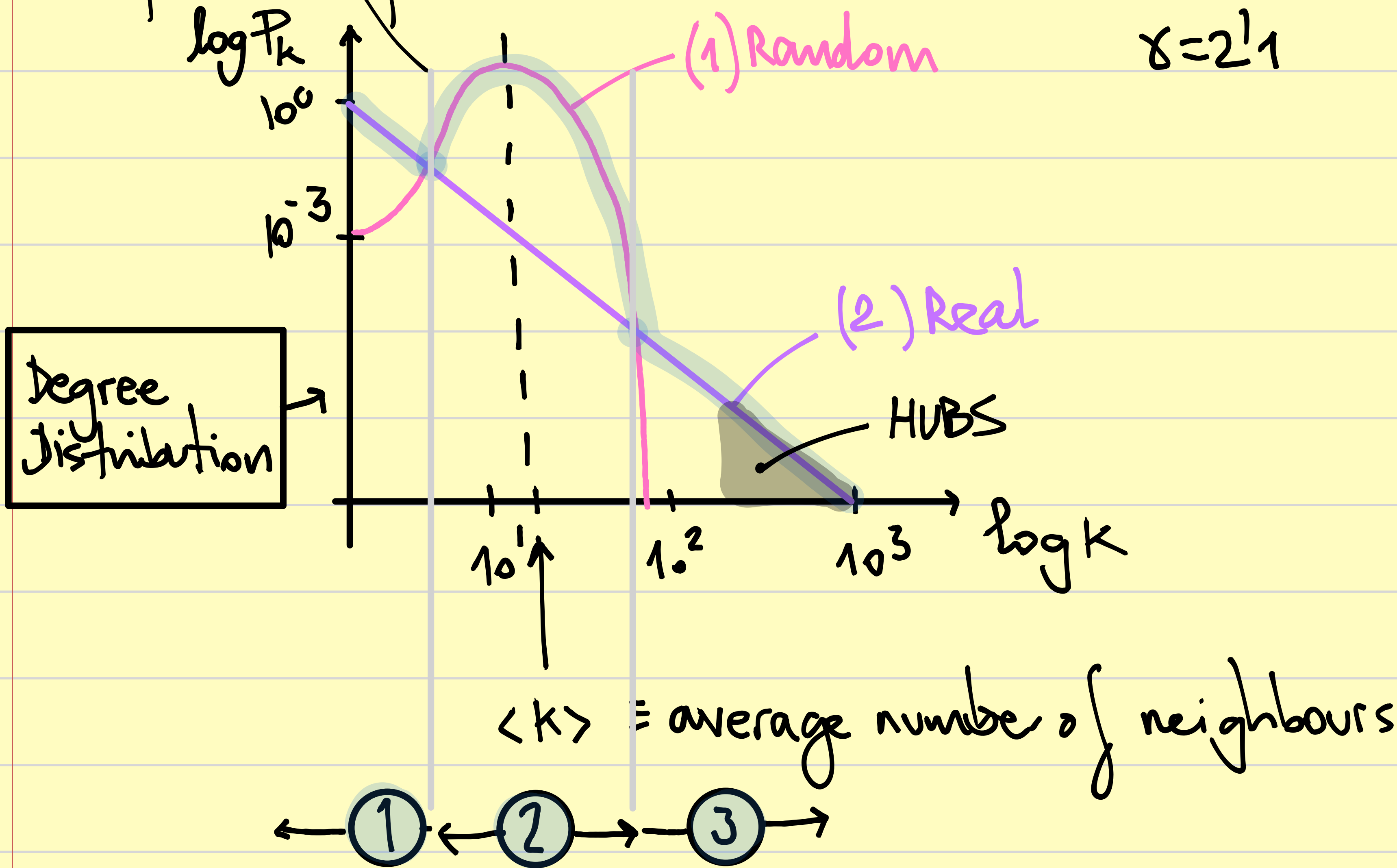
The probability that a node with "k" neighbours attaches to a new node in a real network is given by a POWERLAW distribution with power exponent "γ":

$$\boxed{P_{Real}(X=k) = k^{-\gamma}} \quad (2)$$

$$2 < \gamma < 3$$

[Barabasi, 2016] → Network Science

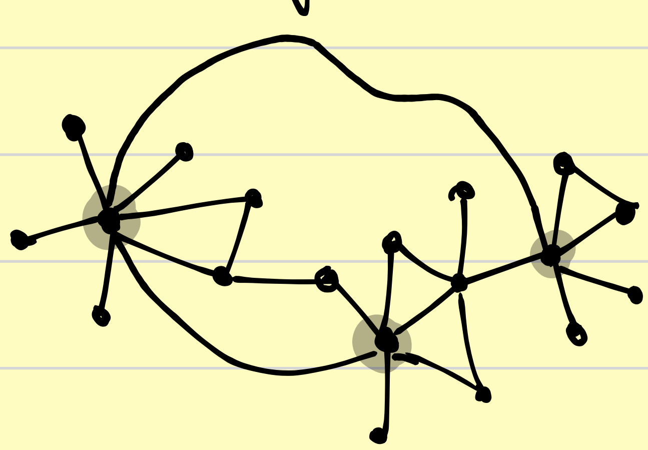
Graphical Representation:



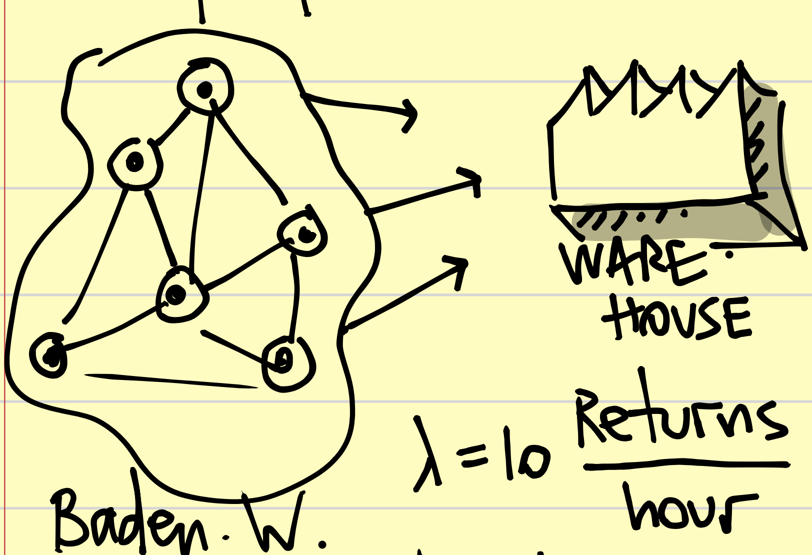
1 Random $\ll$ Real. For small k, the power law of the real network is above the poisson of the random network, indicating that the real world has more nodes with few neighbours.

2 Random > Real. For nodes around the average number of neighbours $\langle k \rangle$, the random network has excess of nodes.

3 Random $\ll$ Real. For large k, the real network presents nodes with high value of k. These nodes are called hubs.



Example for a random network:



What is the probability of getting exactly 3 products back in the next hour?

$\lambda = 10 \frac{\text{Returns}}{\text{hour}}$

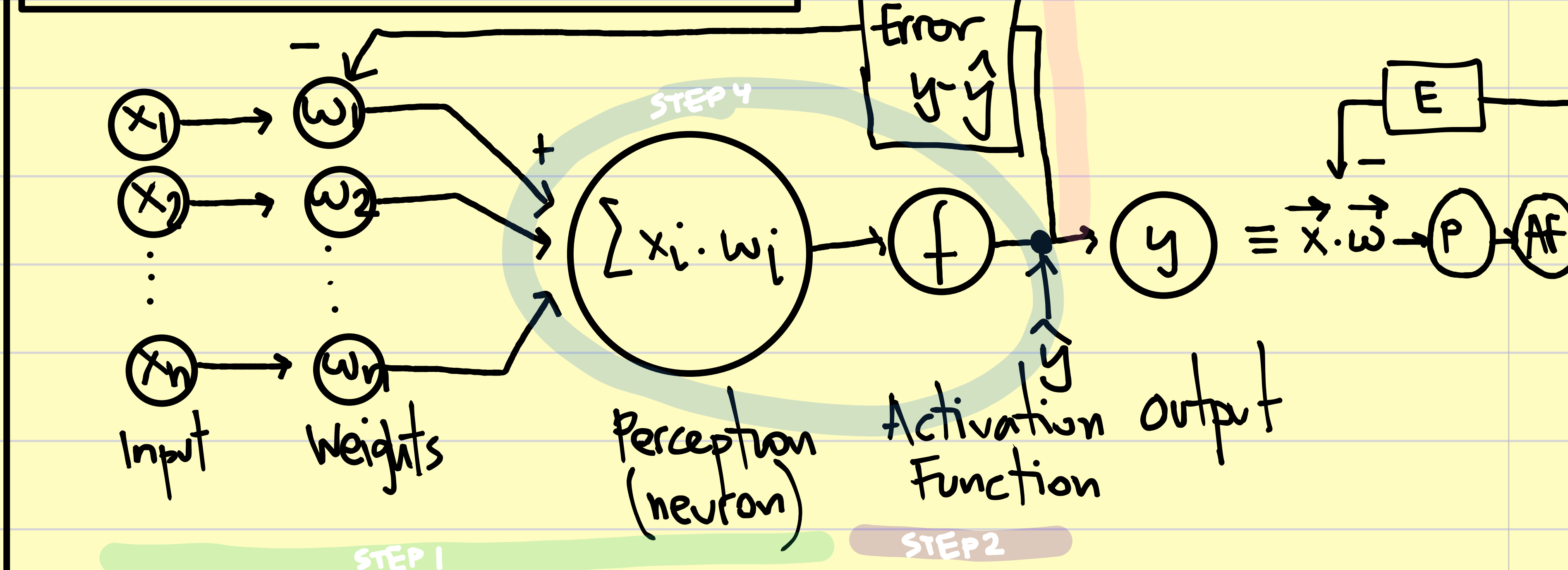
Random

$$P(X=3) = \frac{e^{-10} \cdot 10^{3}}{3!} = \frac{e^{-10} \cdot 10^{3}}{3 \cdot 2 \cdot 1} = 7{,}5 \cdot 10^{-3}$$

We have a random network of customers that are returning their products independently from each other.

DEEP LEARNING ..by hand..

SINGLE NEURON ..PERCEPTRON..

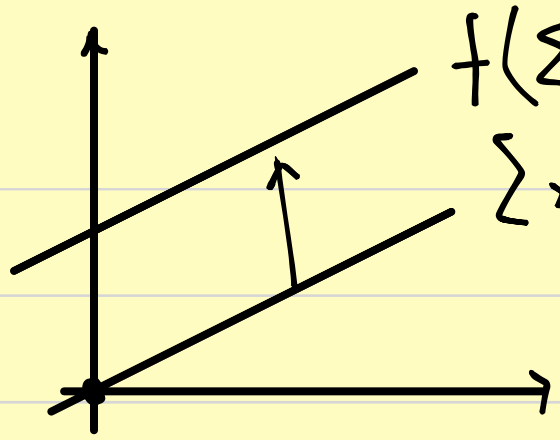


Step 1. Multiply all input values x_i with their corresponding weights w_i and then calculate the weighted sum.

Step 2. Apply an activation function to step 1.

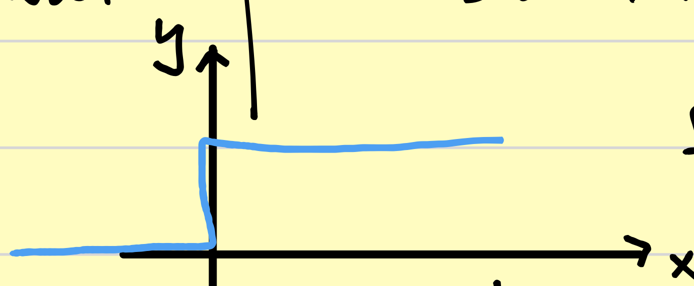
$$f\left(\sum x_i w_i\right) \simeq f\left(\sum x_i w_i + bias\right)$$

$$\hat{y} \quad \text{MEASURE}$$



Examples of activation functions used in deep learning...

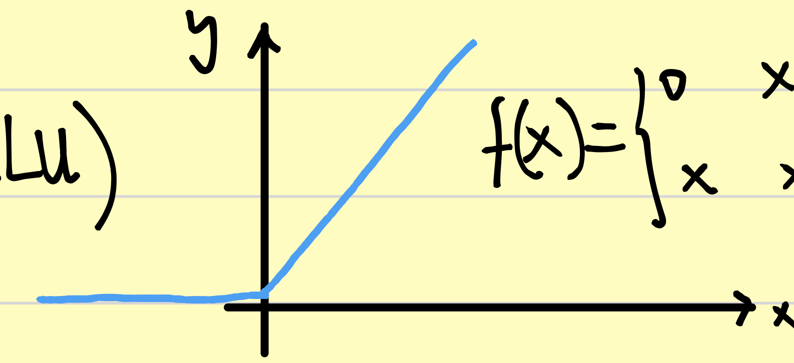
1) Step function



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

This eliminates all negative values.

2) Rectified Linear Unit (ReLU)



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

3) We want to calculate the weight values which make our error minimal. We take the squared of the difference (distance) between expected and measurement.

$$\left.\begin{aligned} y &= y \quad \text{expected value} \\ \hat{y} &= f\left(\sum x_i w_i + b\right) \end{aligned}\right\} \text{MINIMIZE } (y - \hat{y})^2 \quad \text{QUADRATIC ERROR}$$

We use the quadratic error in order to avoid cancelling of positive and negative values.

$$\boxed{\text{COST FUNCTION: } \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}\left(y - f\left(\sum x_i w_i + b\right)\right)^2}$$

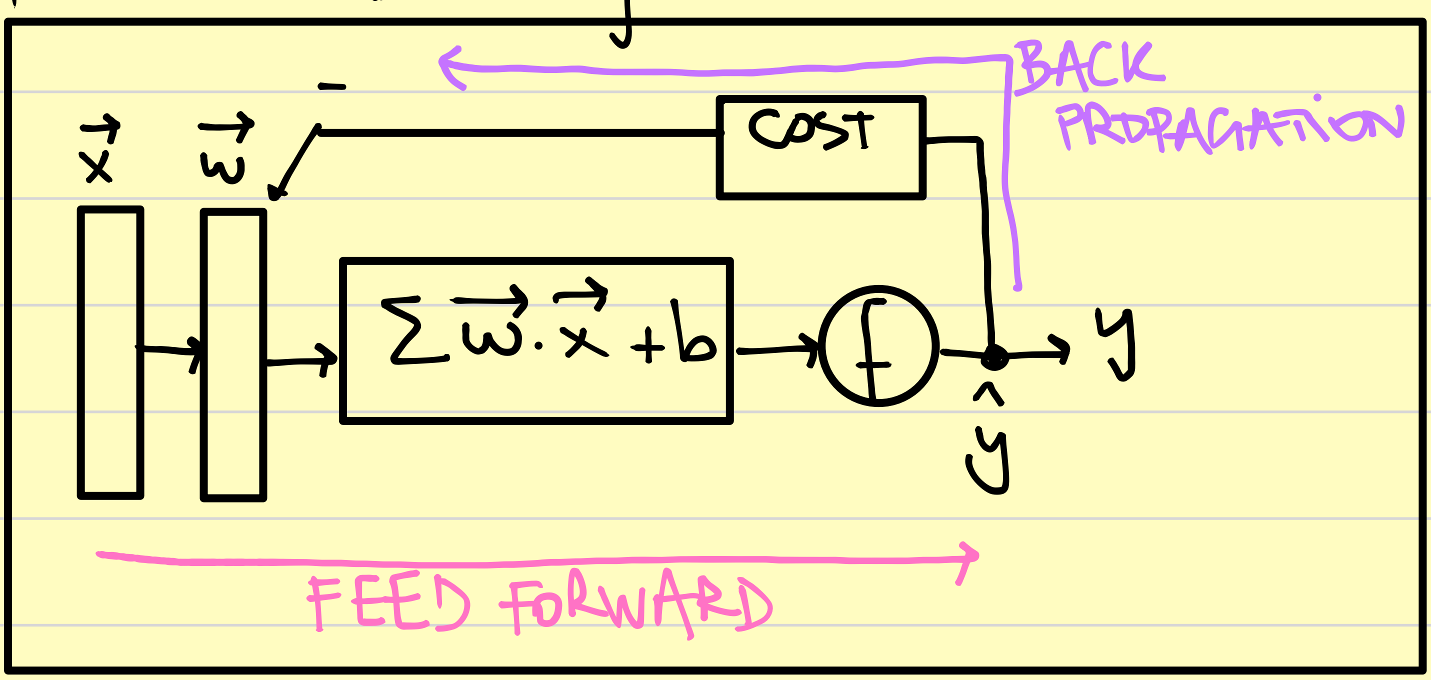
The perceptron adjusts the "weights" w_i so that this cost function / quadratic error is minimized.

The derivatives of this cost function $C(\vec{w})$ are called gradients

$$\frac{\partial C}{\partial w_i} = \frac{1}{2} \cdot 2 \cdot \left[y - w_1x_1 - w_2x_2 - \dots - w_nx_n\right]\left[-x_i\right]$$

STEP 4. In order to get the minimum of the cost function (learn) we want to find the values that make these gradients zero.

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



Example 1. SINGLE NEURON PERCEPTRON . FORWARD PASS

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

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

$$\begin{array}{cccc} w_1 & w_2 & w_3 & b \\ \hline 1 & -1 & 1 & -5 \end{array} \qquad \begin{bmatrix} -1 \end{bmatrix} \underset{f}{\simeq} \begin{bmatrix} 0 \end{bmatrix}$$

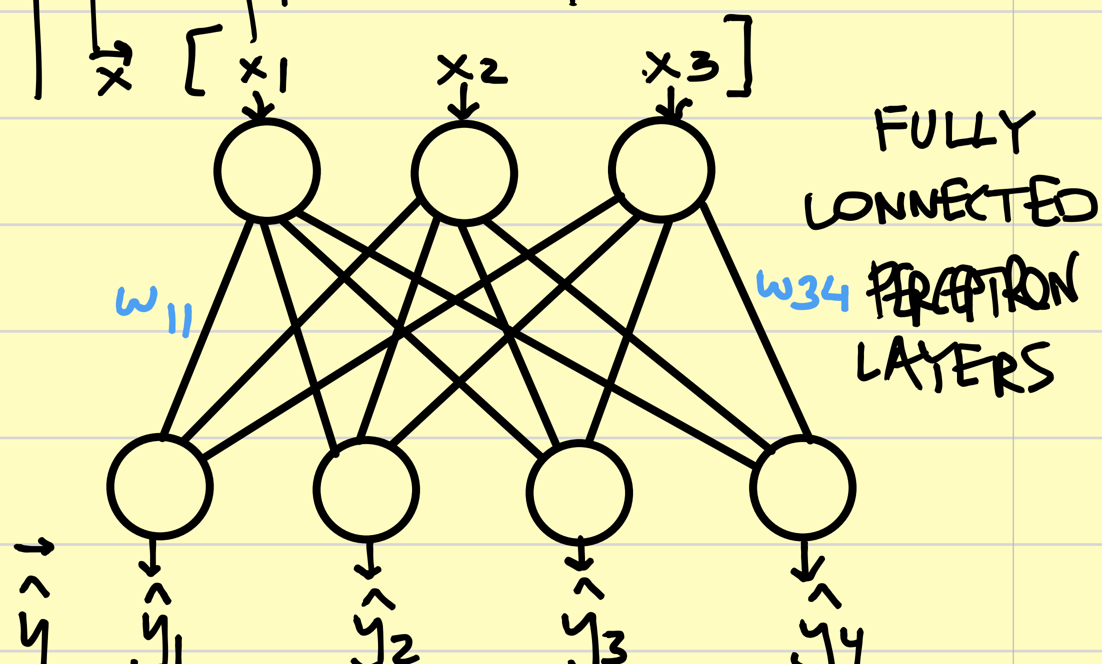
$$\hat{y} = f\left[\sum_{i=1}^{n} w_i x_i + b\right]$$

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

$$= f\left[\left[1 \cdot 2 + (-1) \cdot 1 + 1 \cdot 3\right] + (-5)\right] = f\left[-1\right] = 0 = \hat{y}$$

$$f \equiv \mathrm{ReLU} \equiv \begin{cases} 0 & x < 0 \\ x & x \geqslant 0 \end{cases}$$

EXAMPLE 2. Calculate the output of the forward pass of two layers of perceptrons with 3 and 4 neurons.



$$\begin{array}{c|c} w & \vec{b} \\ \begin{bmatrix} 1 & -1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} & \begin{bmatrix} -5 \\ 0 \\ 1 \\ -2 \end{bmatrix} \end{array} \quad \begin{array}{c} \vec{x} \\ \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix} \begin{matrix} x_1 \\ x_2 \\ x_3 \end{matrix} \end{array} \quad \begin{bmatrix} -1 \\ 3 \\ 5 \\ 3 \end{bmatrix} \underset{\sim}{\overset{f}{=}} \begin{bmatrix} 0 \\ 3 \\ 5 \\ 3 \end{bmatrix} \hat{y}$$

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

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

$2 \cdot 1 + 1 \cdot 1 + 3 \cdot 0 + 0 = 3$

$2 \cdot 0 + 1 \cdot 1 + 3 \cdot 1 + 1 = 5$

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

$f = \text{ReLU}$