- To develop some feeling for feed forward neural sets (or multi-layer perceptrons) we will talk about some simple cases with just a few neurons

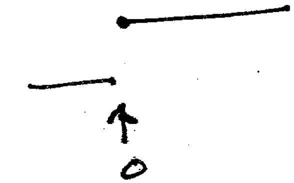
- The focus here is on what kinda of classifications the nets are capable of rather than trainging and generalization

- So we just see how nets can be "fit" to the training data

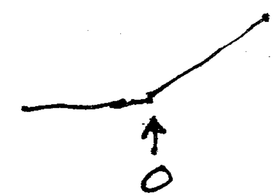
- First some notation

we will consider two activation functions σ

(·) The step function $\sigma_s(z) = \begin{cases} 0 & z \leq 0 \\ 1 & z > 0 \end{cases}$

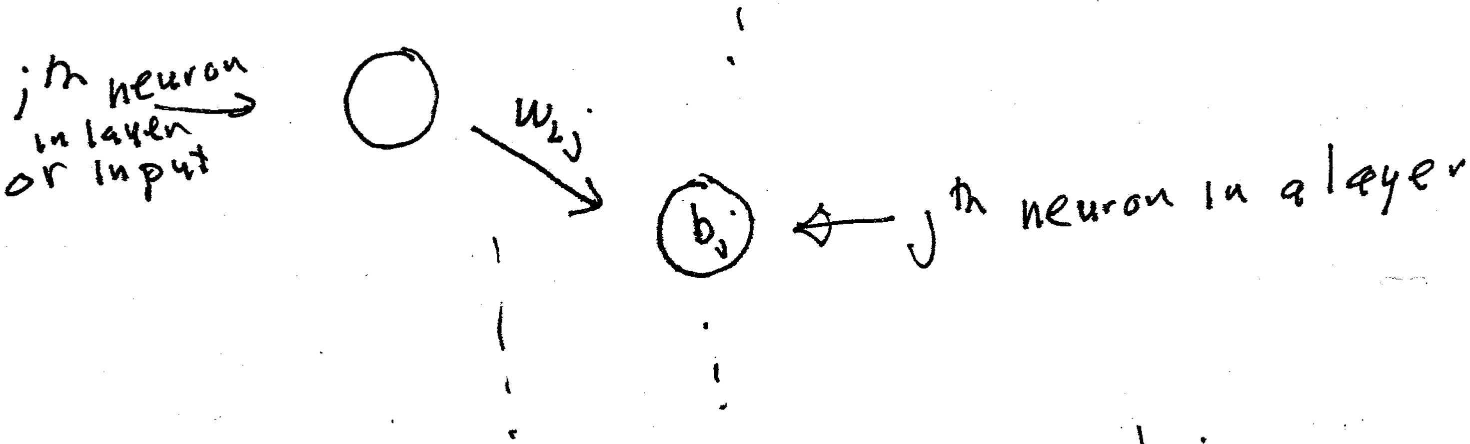


(·) The ramp or linear rectified function $\sigma_r(z) = \max(0, z)$



(·) Sometimes the output neuron will have no activation or $\sigma(z) = z$

The nets are represented by diagrams



means this ~~a~~ describes $w_{ij} x_i + b_j$

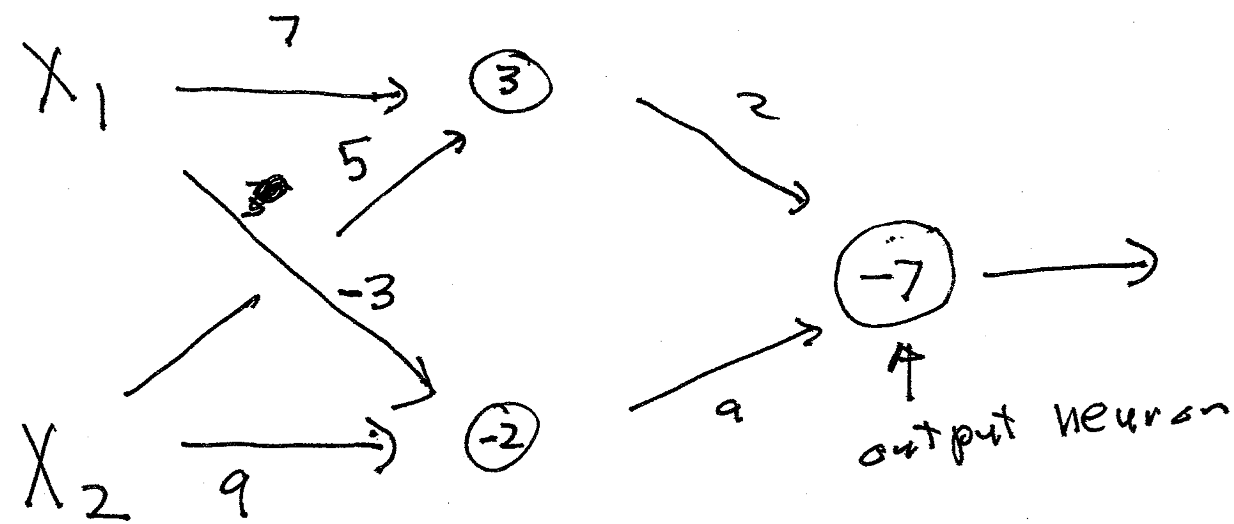
so if $W = (w_{ij})$ and $\vec{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$

This layer yields

$$\sigma (W^T \vec{x} + \vec{b})$$

note the transpose

Let's do an example



yields with activation σ_R

$$F(x) = \sigma_R\left(\begin{bmatrix} 2 \\ 9 \end{bmatrix}^T \sigma_R\left(\begin{bmatrix} 7 & -3 \\ 5 & 9 \end{bmatrix}^T \vec{x} + \begin{bmatrix} 3 \\ -2 \end{bmatrix} \right) - 7 \right)$$

if the output neuron has no activation we drop this σ_R.

Let's compute $F\left(\begin{bmatrix}1\\0\end{bmatrix}\right)$ by working from the inside out with an arrow indicating each step

$$\begin{bmatrix}7 & -3\\5 & 9\end{bmatrix}^T \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}7 & 5\\-3 & 9\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}7\\-3\end{bmatrix}$$

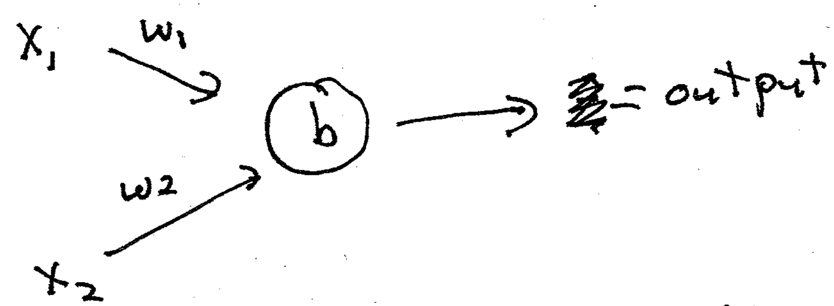
$$\rightarrow \left(\begin{bmatrix}7\\3\end{bmatrix} + \begin{bmatrix}3\\-2\end{bmatrix}\right) = \begin{matrix}10\\-5\end{matrix} \rightarrow \sigma_R\begin{pmatrix}10\\-5\end{pmatrix} = \begin{pmatrix}10\\0\end{pmatrix}$$

$$\boxed{\sigma_R(-5) = 0}$$

$$\rightarrow \begin{bmatrix}2\\9\end{bmatrix}^T \begin{bmatrix}10\\0\end{bmatrix} = 20 \rightarrow 20 - 7 = \cancel{22}\ 13$$

$$\rightarrow \sigma_R(13) = 13$$

we now see what a single neuron with the step activation does

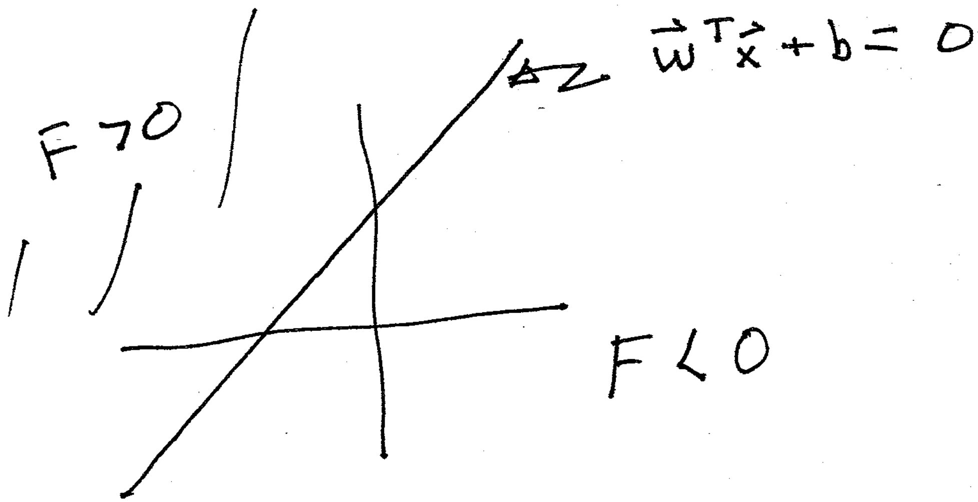


So $F(\vec{x}) = \sigma_S(w^T \vec{x} + b) = \sigma_S(w_1 x_1 + w_2 x_2 + b)$

$$= \begin{cases} 0 & \text{when } w_1 x_1 + w_2 x_2 + b \leq 0 \\ 1 & w_1 x_1 + w_2 x_2 + b > 0 \end{cases}$$

Now $0 = w_1 x_1 + w_2 x_2 + b$ is a line in the (x_1, x_2)-plane

It divides the plane into two halves one when $F(\vec{x}) = 0$ and one where $F(\vec{x}) = 1$

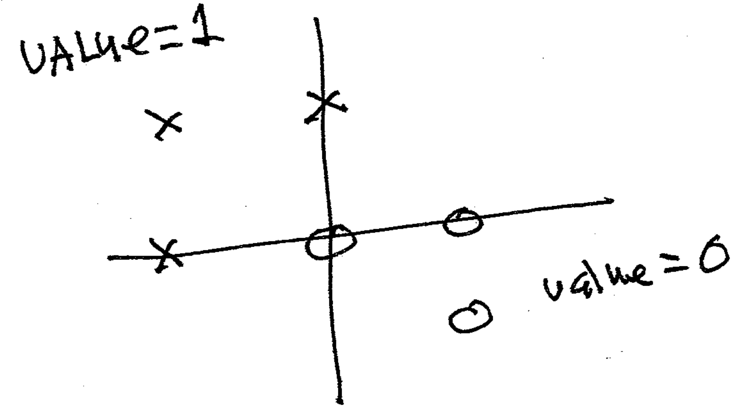


<u>Example</u>: Find the weights and bias of a single neuron with step-activation that classifies the points

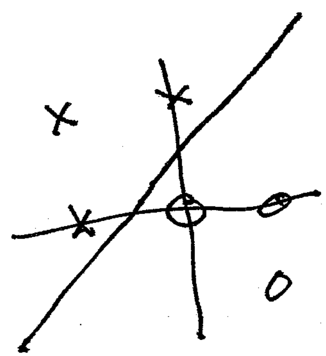
$(-1,-1), (-1,0), (0,1) \longrightarrow$ have value 1 (yes)

$(0,0), (1,0), (1,-1) \longrightarrow$ have value 0 (no)

<u>Soln</u> Plot them in the plane



We find a "decision line" that divides them



one such line is

$x_1 - x_2 + 1/2 = 0$

but $F(x_1, x_2) = \sigma(x_1 - x_2 + 1/2)$

yields $F(0,0) = \sigma(1/2) = 1$

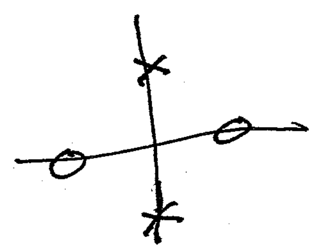
the wrong value so

we use the description $-x_1 + x_2 - 1/2 = 0$

So the soln is $w_1 = -1, w_2 = 1, b = -1/2$

- Now there are many decision lines - I just chose a simple one - If we wanted to use the line to generalize and decide about new data, we would want the best line - This is done with Support VECTOR Machines, which we cover later.

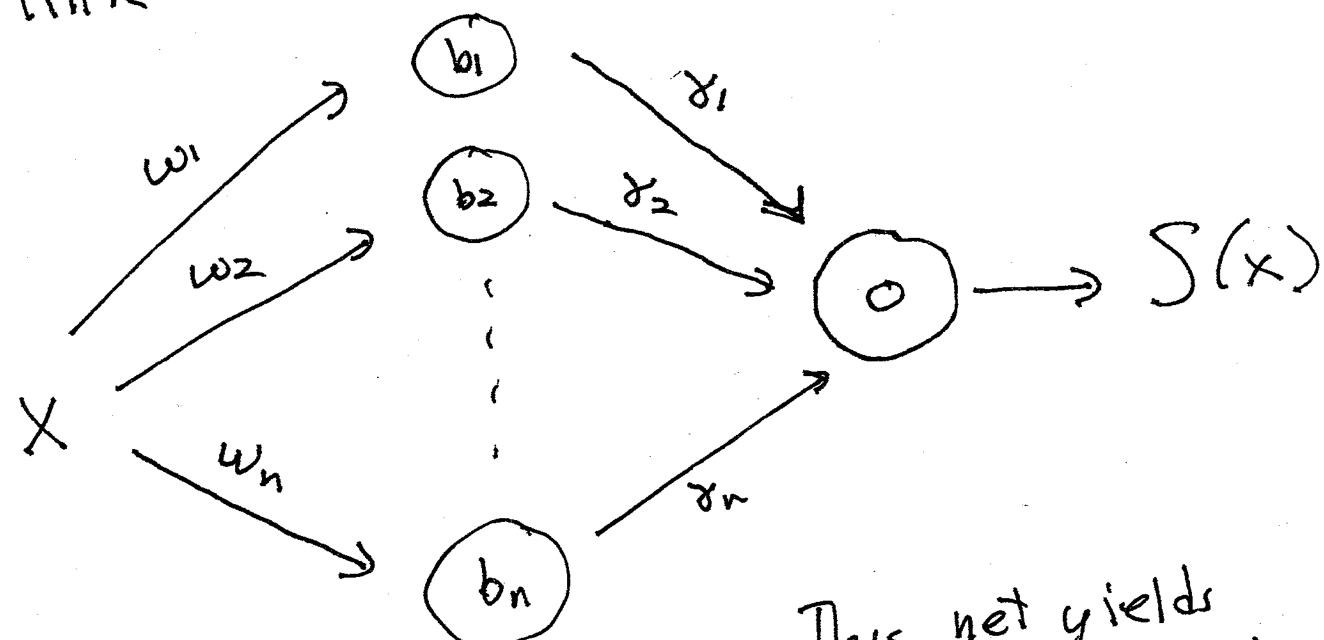
- It should be clear that one neuron can decide only for special data

 doesn't work.

- So we need more neurons or more layers or both.

- There is a theorem which indicates how powerful even simple n.n. are.

- Consider this net where I simplify notation a little using ramp activation on the middle layer and none on the output



This net yields with <u>no</u> output activation

$$S(x) = \sum_{i=1}^{n} \gamma_i \, \sigma_R(w_i x + b_i)$$

<u>Theorem</u>: Given a continuous function $f: [0,1] \to \mathbb{R}$ and an $\varepsilon > 0$, there is an n and $\gamma_1, .., \gamma_n$ and weights $w_1, .., w_n$ and bias' $b_1, .., b_n$ so that $\max_{x \in [0,1]} |f(x) - S(x)| < \varepsilon$ where

$$S(x) = \sum_{i=1}^{n} \gamma_i \, \sigma_R(w_i x + b_i)$$