1.
$$\frac{\partial J(w)}{\partial w_{11}} = \left(\frac{\partial J(w)}{\partial y_1} \times \frac{\partial y_1}{\partial n_3} \times \frac{\partial n_3}{\partial n_{1out}} + \frac{\partial J(w)}{\partial y_2} \times \frac{\partial y_2}{\partial n_4} \times \frac{\partial n_4}{\partial n_{1out}} \right) \times \frac{\partial n_{1out}}{\partial n_{1in}} \times \frac{\partial n_{1in}}{\partial w_{11}}$$

$$\frac{\partial J(w)}{\partial y_1} = \frac{\partial}{\partial y_1}\left(\frac{1}{2}\left((d_1-y_1)^2+(d_2-y_2)^2\right)\right)$$
$$= \frac{\partial}{\partial y_1}\left(\frac{1}{2}d_1^2 - d_1y_1 + \frac{1}{2}y_1^2\right)$$
$$= (-d_1 + y_1)$$

$$\frac{\partial J(w)}{\partial y_2} = \frac{\partial}{\partial y_2}\left(\frac{1}{2}\left((d_1-y_1)^2+(d_2-y_2)^2\right)\right)$$
$$= \frac{\partial}{\partial y_2}\left(\frac{1}{2}d_2^2 - d_2y_2 + \frac{1}{2}y_2^2\right)$$
$$= (-d_2 + \frac{1}{2}y_2)$$

$$\frac{\partial y_1}{\partial n_3} = \frac{\partial \phi(n_3)}{\partial n_3} = \phi(n_3)(1-\phi(n_3))$$

$$\frac{\partial y_2}{\partial n_4} = \frac{\partial \phi(n_4)}{\partial n_4} = \phi(n_4)(1-\phi(n_4)).$$

$$\frac{\partial n_3}{\partial n_{1out}} = \frac{\partial}{\partial n_{1out}}\left(w_{13}n_{1out} + w_{14}n_{2out}\right)$$
$$= w_{13}$$

$$\frac{\partial n_4}{\partial n_{1out}} = \frac{\partial}{\partial n_{1out}}\left(w_{14}n_{1out} + w_{24}n_{2out}\right)$$
$$= w_{14}.$$

$$\frac{\partial n_{1out}}{\partial n_{1in}} = \frac{\partial}{\partial n_{1in}}\phi(n_{1in}) = \phi(n_{1in})(1-\phi(n_{1in}))$$

$$\frac{\partial n_{1in}}{\partial w_{11}} = \frac{\partial}{\partial w_{11}}\left(w_{11}x_1 + w_{21}x_2\right) = x_1$$

$$\Rightarrow \frac{\partial J(w)}{\partial w_{11}} = \left((-d_1+y_1)\phi(n_3)(1-\phi(n_3))w_{13} + (-d_2+y_2)\phi(n_4)(1-\phi(n_4))w_{14}\right)\phi(n_{1in}) \times (1-\phi(n_{1in}))\,x_1$$

$$w_{11}' = w_{11} - y\frac{\partial J(w)}{\partial w_{11}}$$

Similarly, $w_{12}' = w_{12} - \eta \frac{\partial J(w)}{\partial w_{12}}$, $w_{21}' = w_{21} - \eta \frac{\partial J(w)}{\partial w_{21}}$. $w_{22}' = w_{22} - \eta \frac{\partial J(w)}{\partial w_{22}}$.

where.

$$\frac{\partial J(w)}{\partial w_{12}} = ((-d_1 + y_1)\phi(n_3)(1-\phi(n_3))w_{23} + (-d_2 + y_2)\phi(n_4)(1-\phi(n_4))w_{24})\phi(n_{2in})(1-\phi(n_{2in}))x_1$$

$$\frac{\partial J(w)}{\partial w_{21}} = ((-d_1 + y_1)\phi(n_3)(1-\phi(n_3))w_{13} + (-d_2 + y_2)\phi(n_4)(1-\phi(n_4))w_{14})\phi(n_{1in})(1-\phi(n_{1in}))x_2$$

$$\frac{\partial J(w)}{\partial w_{22}} = ((-d_1 + y_1)\phi(n_3)(1-\phi(n_3))w_{23} + (-d_2 + y_2)\phi(n_4)(1-\phi(n_4))w_{24})\phi(n_{2in})(1-\phi(n_{2in}))x_2$$

After computation. $w_{11}' = 0.0971$
$w_{12}' = 0.2500$
$w_{21}' = 0.0971$
$w_{22}' = 0.7000$

2. $$\frac{\partial J(w)}{\partial w_{13}} = \frac{\partial J(w)}{\partial y_1} \times \frac{\partial y_1}{\partial n_3} \times \frac{\partial n_3}{\partial w_{13}}$$

$$\frac{\partial J(w)}{\partial y_1} = (-d_1 + y_1)$$

$$\frac{\partial y_1}{\partial n_3} = \phi(n_3)(1-\phi(n_3))$$

$$\frac{\partial n_3}{\partial w_{13}} = \frac{\partial}{\partial w_{13}}(w_{13} n_{1out} + w_{14} n_{2out})$$

$$= n_{1out}$$

$$\Rightarrow \frac{\partial J(w)}{\partial w_{13}} = (-d_1 + y_1)\phi(n_3)(1-\phi(n_3))n_{1out}. \quad w_{13}' = w_{13} - \eta \frac{\partial J(w)}{\partial w_{13}}$$

Similarly. $w_{14}' = w_{14} - \eta \frac{\partial J(w)}{\partial w_{14}}$, $w_{23}' = w_{23} - \eta \frac{\partial J(w)}{\partial w_{23}}$, $w_{24}' = w_{24} - \eta \frac{\partial J(w)}{\partial w_{24}}$.

where $\frac{\partial J(w)}{\partial w_{14}} = (-d_2 + y_2)\,\phi(n_4)(1-\phi(n_4))\,n_{1out}$

$$\frac{\partial J(w)}{\partial w_{23}} = (-d_1 + y_1)\,\phi(n_3)(1-\phi(n_3))\,n_{2out}$$

$$\frac{\partial J(w)}{\partial w_{24}} = (-d_2 + y_2)\,\phi(n_4)(1-\phi(n_4))\,n_{2out}.$$

After computation. $w_{13}' = 0.4042$

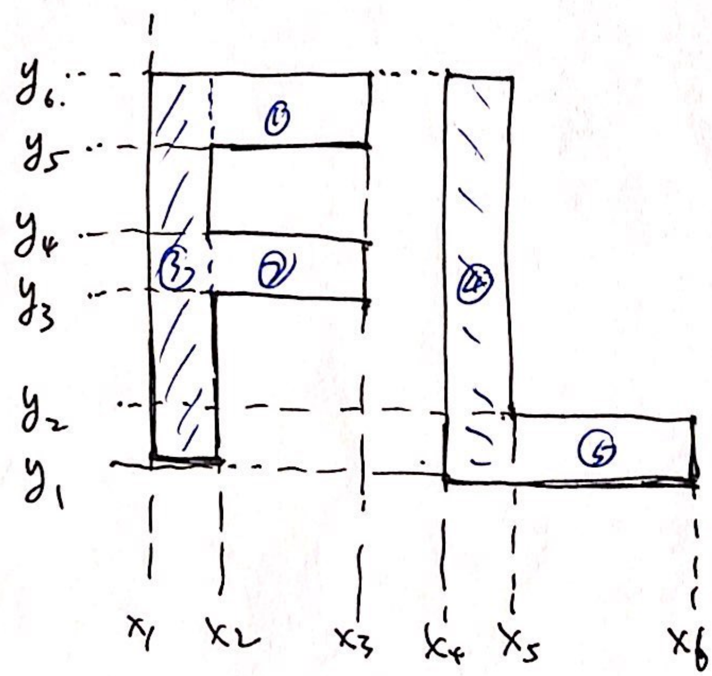
$w_{14}' = 0.9918$

$w_{23}' = 0.6056$

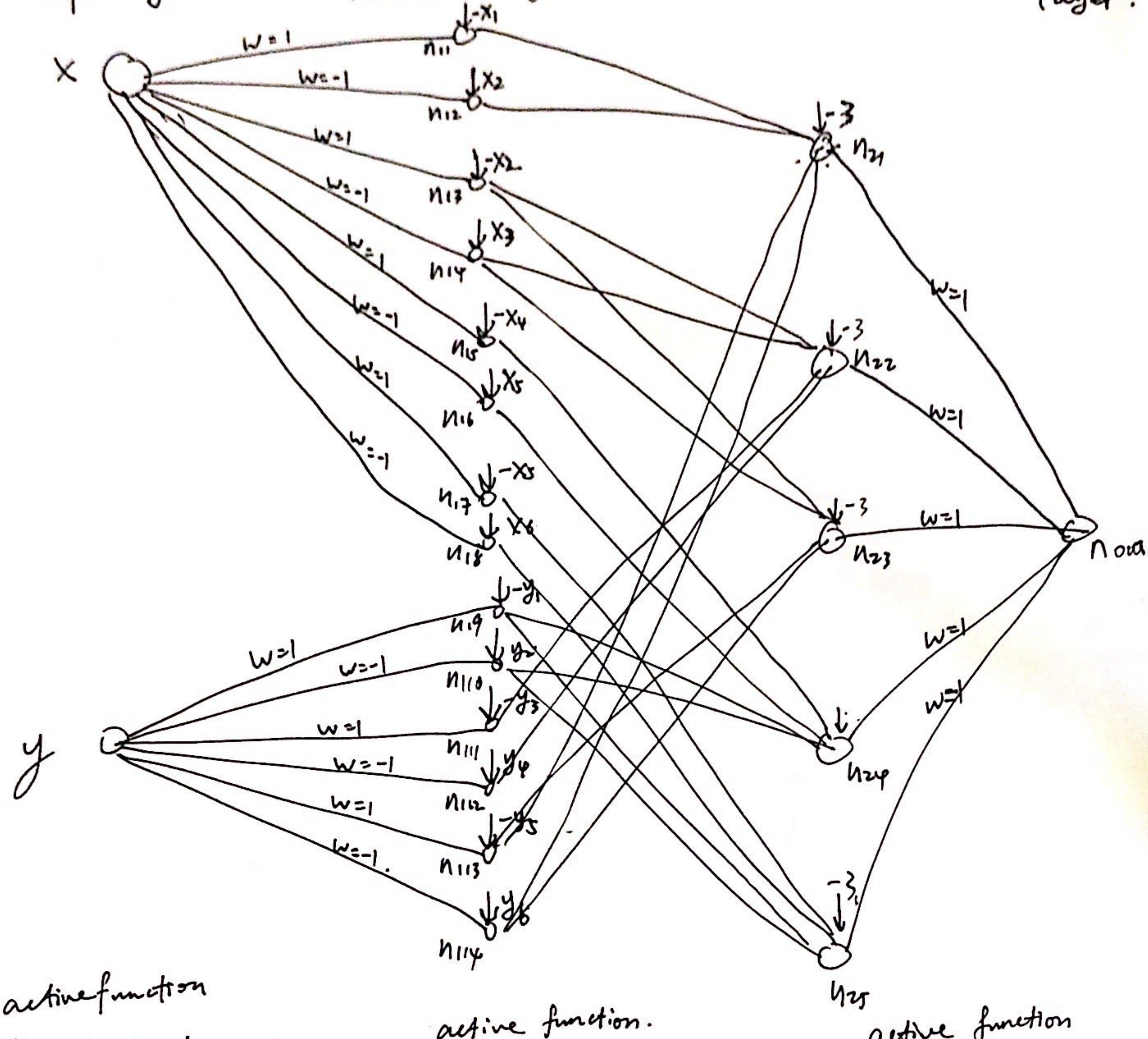
$w_{24}' = 0.2893$

3.

(a)



I need 14 units in the first hidden layer and 5 units in the second layer. Say giving the coordinates of a point as input, this network should tell whether this point belongs to "UF" or not. If the output is 1, it means this point belongs to the letters. otherwise, it belongs to the background.



active function for hidden layer 1.

$$f_1 = \begin{cases} 1 & \sum w_i x_i + \sum w_i y_i \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

active function. for hidden layer 2.

$$f_2 = \begin{cases} x & (x \geq 0) \\ 0 & \text{otherwise} \end{cases} \quad \text{(ReLU).}$$

active function for output layer.

$$f_3(x) = x$$

weights of edges between hidden layer 1 and hidden layer 2 are all 1. $(w=1)$

Take unit n_{11} as an example. For any input $(x, y)^T$,

$n_{11}\,in = x - x_1$, $n_{11}\,out = f_1(n_{11}\,in) = f(x-x_1) = \begin{cases} 1 & x \geq x_1 \\ 0 & x < x_1. \end{cases}$

which means unit n_{11} can tell if $x \geq x_1$.

Similarly. n_{12} tells whether $x \leq x_2$. ... n_{113} tells whether $y \geq y_5$. n_{114} tells whether $y \leq y_6$. All the output of hidden layer 1 is either 1 or 0.

For hidden layer 2, take n_{21} as an example. It takes the outputs of n_{11}, n_{12}, n_{113} and n_{114}. If for an input $(x, y)^T$ where $x_1 \leq x \leq x_2$ and $y_5 \leq y \leq y_6$, all these four outputs should be 1s. So

$n_{21}\,in = 1+1+1+1-3 = 1$. $f_2(n_{21}\,in) = f(1) = 1$. For all nodes in hidden layer 2, each of them tells whether $(x,y)^T \in$ block k $(k = ①, ②, ③, ④, ⑤)$

Which means at most one of nodes $n_{21}, n_{22}, n_{23}, n_{24}, n_{25}$ has output $=1$.

So the input of output layer can be either 1 or 0. If $n_{out} = 1$, $f(n_{out}) = 1$, which means the input point $(x,y)^T$ belongs to letter "U" or "F". else, $(x,y)^T$ belongs to the background.

(b) I can't achieve the same goal with a single hidden layer because a single hidden layer is able to tell whether an input point belongs to a rectangle, while neither the background nor the letters is a rectangle. So at least 2 hidden layers are required for this goal.

4. E_2 is better.

Say $J_{E_1(w)} = J(w) + E_1(w)$, $J_{E_2(w)} = J(w) + E_2(w)$

$$\frac{\partial J_{E_1(w)}}{\partial w_i} = \frac{\partial J(w)}{\partial w_i} + 2w_i \quad , \quad \frac{\partial J_{E_2(w)}}{\partial w_i} = \frac{\partial J(w)}{\partial w_i} + \operatorname{sgn}(w_i)$$

$$\operatorname{sgn}(w_i) = \begin{cases} 1 & w_i > 0 \\ -1 & w_i < 0 \\ 0 & w_i = 0 \end{cases}$$

$$\left.\frac{\partial J_{E_1(w)}}{\partial w_i}\right|_{w_i=0} = \frac{\partial J(w)}{\partial w_i}$$

$$\left.\frac{\partial J_{E_2(w)}}{\partial w_i}\right|_{w_i=0^+} = \frac{\partial J(w)}{\partial w_i} + 1 \quad , \quad \left.\frac{\partial J_{E_2(w)}}{\partial w_i}\right|_{0^-} = \frac{\partial J(w)}{\partial w_i} - 1 .$$

So after adding E_1, $\left.\frac{\partial J_{E_1(w)}}{\partial w_i}\right|_{w_i=0}$ doesn't change. which means w_i will be the same as before, if it was not 0, it remains a non-zero value. While after adding E_2, $\left.\frac{\partial J_{E_2(w)}}{\partial w_i}\right|_{w_i=0^+}$ changes rapidly from $\left.\frac{\partial J_{E_2(w)}}{\partial w_i}\right|_{w_i=0^-}$. Which means it's easilier for $J_{E_2(w)}$ to move to the local optimal where $w_i=0$, that means we dropped the ith feature.

5. From the problem we know that $\prod_{j=1}^{n} p_j \geq \left(\prod_i p_i^{p_i}\right)^{n/\sum_i p_i}$

$$p_1 \geq \left(\prod_i p_i^{p_i}\right)^{\frac{1}{\sum_i p_i}}$$

Let $p_i \geq \left(\prod_i p_i^{p_i}\right)^{\frac{1}{\sum_i p_i}}$

$$p_1 p_2 \geq \left(\prod_i p_i^{p_i}\right)^{\frac{2}{\sum_i p_i}}$$

Simply let $p_1 = p_2 = \cdots = p_6 = \frac{1}{6}$.

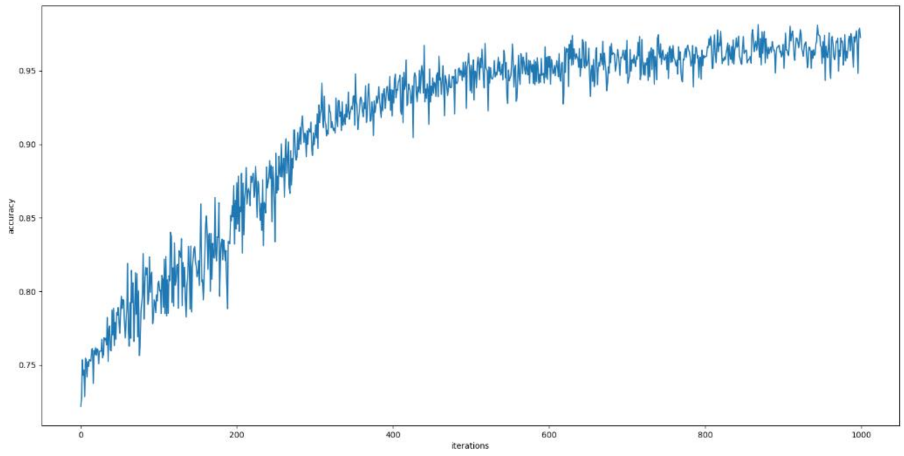
$$\vdots$$

$$p_1 p_2 \cdots p_6 \geq \left(\prod_i p_i^{p_i}\right)^{\frac{6}{\sum_i p_i}}$$

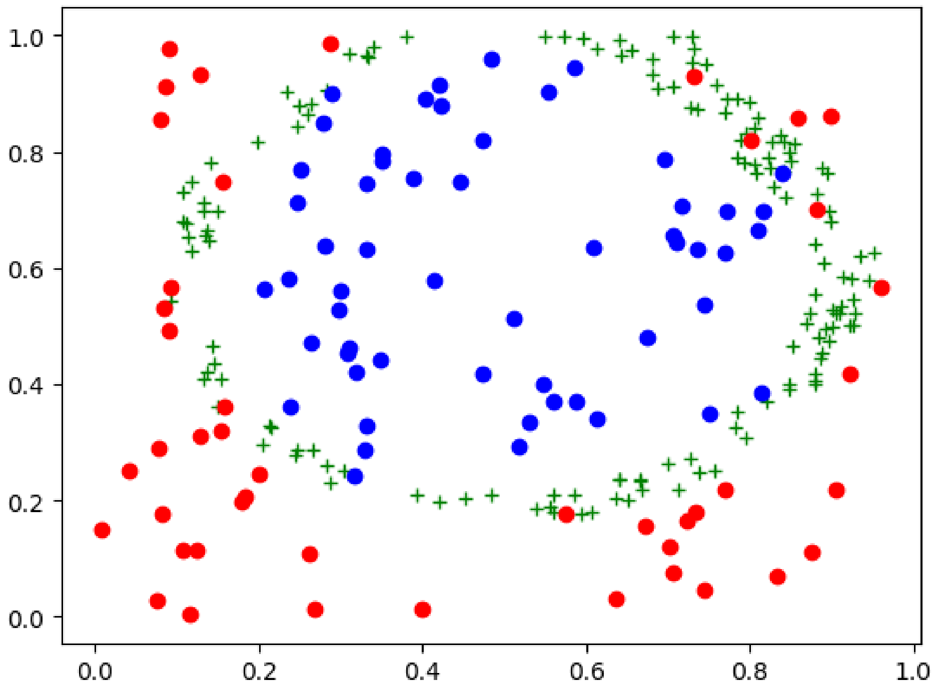
6. Alex uses 8 neurons for a single hidden layer MLP. If allowed to use any number of layers, he just need to use 6 neurons for hidden layers.

The deeper MLP is, the less neurons might be used for hidden layers.

7. Below is the iterations-accuracy figure:



Below is the figure of hyperplane(green ‘+’s):



Below is the figure of weights and bias of the hyperplane:

```
2.weight tensor([[-7.2539, -0.7160,  5.8378,  2.8152, -4.9444, -7.5179, -7.1440, -4.3030,
         -7.1165, -1.2807]])
2.bias tensor([-0.3289])
```