1.

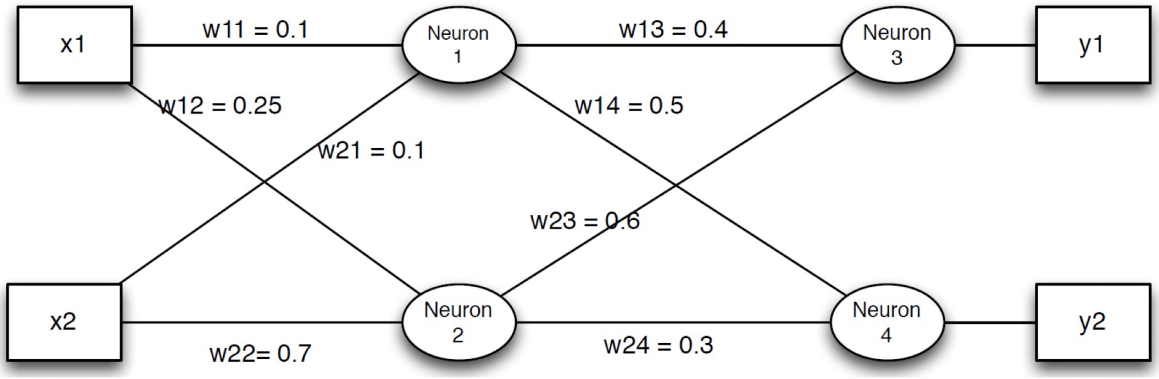


Figure 1:

Given: $J(w) = \frac{1}{2} \sum_{i=1}^{N} e_i^2$; $e = d - y$.

$d = [1\ 0]^T = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$; $x = [1, 1]^T = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

$\phi(x) = \frac{1}{1+e^{-x}}$; $\phi'(x) = \phi(x)(1 - \phi(x))$.

$\eta = 0.1$

Forward Propagation

$L^{[0]}: X = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$; $L^{[1]}: W^{[1]} = \begin{bmatrix} 0.1 & 0.25 \\ 0.1 & 0.7 \end{bmatrix}$

$$Z^{[1]} = W^{[1]} X = \begin{bmatrix} 0.1 & 0.25 \\ 0.1 & 0.7 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.35 \\ 0.80 \end{bmatrix}$$

$$A^{[1]} = \sigma(Z^{[1]}) = \begin{bmatrix} 0.587 \\ 0.690 \end{bmatrix}$$

$$L^{[2]} : W^{[2]} = \begin{bmatrix} 0.4 & 0.5 \\ 0.6 & 0.3 \end{bmatrix}$$

$$Z^{[2]} = W^{[2]} A^{[1]} = \begin{bmatrix} 0.4 & 0.5 \\ 0.6 & 0.3 \end{bmatrix} \begin{bmatrix} 0.587 \\ 0.690 \end{bmatrix}$$

$$A^{[2]} = \sigma\left(Z^{[2]}\right) = \begin{bmatrix} 0.641 \\ 0.636 \end{bmatrix}$$

$$= \begin{bmatrix} 0.235 + 0.345 \\ 0.352 + 0.207 \end{bmatrix}$$

$$= \begin{bmatrix} 0.580 \\ 0.560 \end{bmatrix}$$

Now, $\hat{Y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 0.641 \\ 0.636 \end{bmatrix}$

The cost function $J(W) = \frac{1}{2} \sum_{i=1}^{N} e_i^2 = \frac{1}{2} \sum_{i=1}^{N} (d - \hat{Y})^2$

So, Loss Function : $L(W) = \frac{1}{2} (d - \hat{Y})^2$

- $\frac{dL}{dA^{[2]}}_{(2\times1)} = \frac{1}{2} \cdot 2 \cdot (d - \hat{Y})(-1) = (\hat{Y} - d)_{(2\times1)}$

- $\frac{dL}{dZ^{[2]}}_{(2\times1)} = \frac{dL}{dA^{[2]}} \cdot \frac{dA^{[2]}}{dZ^{[2]}}$

Now, $A^{[2]} = \sigma(Z^{[2]})$

So, $\frac{dA^{[2]}}{dZ^{[2]}} = \sigma(Z^{[2]}) \cdot (1 - \sigma(Z^{[2]})) = A^{[2]} \cdot (1 - A^{[2]})$

Thus,

$$\underset{(2\times1)}{\frac{dL}{dZ^{[2]}}} = \underset{(2\times1)}{(\hat{y} - y)} \cdot \underset{(2\times1)}{A^{[2]}} \cdot \underset{(2\times1)}{(1 - A^{[2]})}$$

$$\cdot \underset{(2\times2)}{\frac{dL}{dW^{[2]}}} = \underset{(2\times1)}{\frac{dL}{dZ^{[2]}}} * \underset{(1\times2)}{\frac{dZ^{[2]}}{dW^{[2]}}}$$

Now, $Z^{[2]} = W^{[2]} A^{[1]}$; $\frac{dZ^{[2]}}{dW^{[2]}} = A^{[1]T}$

So,

$$\underset{(2\times2)}{\frac{dL}{dW^{[2]}}} = \underset{(2\times1)}{(\hat{y} - y)} \cdot \underset{(2\times1)}{A^{[2]}} \cdot \underset{(2\times1)}{(1 - A^{[2]})} * \underset{(1\times2)}{A^{[1]T}}$$

$$= \left(\begin{bmatrix} 0.641 \\ 0.636 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) \cdot \left(\begin{bmatrix} 0.641 \\ 0.636 \end{bmatrix} \right) \left(1 - \begin{bmatrix} 0.641 \\ 0.636 \end{bmatrix} \right)$$

$$* \begin{bmatrix} 0.587 & 0.690 \end{bmatrix}$$

$$= \begin{bmatrix} -0.359 \\ 0.636 \end{bmatrix} \cdot \begin{bmatrix} 0.641 \\ 0.636 \end{bmatrix} \cdot \begin{bmatrix} 0.359 \\ 0.364 \end{bmatrix} * \begin{bmatrix} 0.587 & 0.690 \end{bmatrix}$$

$$= \begin{bmatrix} -0.230 \\ 0.404 \end{bmatrix} \cdot \begin{bmatrix} 0.359 \\ 0.364 \end{bmatrix} * \begin{bmatrix} 0.587 & 0.690 \end{bmatrix}$$

$$= \begin{bmatrix} -0.083 \\ 0.147 \end{bmatrix} * \begin{bmatrix} 0.587 & 0.690 \end{bmatrix}$$

$\frac{dL}{dZ^{[1]}}$

$$= \begin{bmatrix} -0.049 & 0.057 \\ 0.086 & 0.101 \end{bmatrix}$$

Again: $Z^{[2]} = W^{[2]} A^{[1]}$

So, $\frac{dz^{[2]}}{dA^{[1]}} = W^{[2]T}$

- $\frac{dL}{dA^{[1]}} = \frac{dL}{dZ^{[2]}} \cdot \frac{dZ^{[2]}}{dA^{[1]}}$

  (2×1)

$$= W^{[2]T} * \frac{dL}{dZ^{[2]}}$$

(2×2) (2×1)

Already,

$$\frac{dL}{dZ^{[2]}} = (\hat{Y} - d) \cdot A^{[2]} \cdot (1 - A^{[2]})$$

(2×1) (2×1) (2×1) (2×1)

So, $\frac{dL}{dA^{[1]}} = W^{[2]T} * \left\{ (\hat{Y} - d) \cdot A^{[2]} \cdot (1 - A^{[2]}) \right\}$

- $\frac{dL}{dz^{[1]}} = \frac{dL}{dA^{[1]}} \cdot \frac{dA^{[1]}}{dz^{[1]}}$

(2×1) (2×1) (2×1)

$A^{[1]} = \sigma(z^{[1]})$

$\frac{dA^{[1]}}{dz^{[1]}} = \sigma(z^{[1]})(1-\sigma(z^{[1]}))$

$$= \left(W^{[2]T} * \left\{ (\hat{y} - a) \cdot A^{[2]} \cdot (1 - A^{[2]}) \right\} \right) \cdot A^{[1]} \cdot (1 - A^{[1]})$$

So,

$$\frac{dL}{dW^{[1]}} = \frac{dL}{dz^{[1]}} * \frac{dz^{[1]}}{dW^{[1]}}$$

(2×2) (2×1) (1×2)

$$z^{[1]} = W^{[1]} X^{[0]} \quad ; \quad \frac{dz^{[1]}}{dW^{[1]}} = X^{[0]T}$$

- $\frac{dL}{dW^{[1]}} = \left(W^{[2]T} * \underbrace{\left\{ (\hat{y} - a) \cdot A^{[2]} \cdot (1 - A^{[2]}) \right\}}_{\frac{dL}{dz^{[2]}}} \right) \cdot A^{[1]} \cdot (1 - A^{[1]}) * X^{[0]T}$

$$= \left(\begin{bmatrix} 0.4 & 0.6 \\ 0.5 & 0.3 \end{bmatrix} * \begin{bmatrix} -0.083 \\ 0.147 \end{bmatrix} \right) \cdot \begin{bmatrix} 0.587 \\ 0.690 \end{bmatrix} \cdot \left\{ 1 - \begin{bmatrix} 0.587 \\ 0.690 \end{bmatrix} \right\} * X^{[0]T}$$

$$= \begin{bmatrix} -0.033 + 0.088 \\ -0.042 + 0.044 \end{bmatrix} \cdot \begin{bmatrix} 0.587 \\ 0.690 \end{bmatrix} \cdot \begin{bmatrix} 0.413 \\ 0.310 \end{bmatrix} * X^{[0]T}$$

$$= \begin{bmatrix} 0.055 \\ 0.003 \end{bmatrix} \cdot \begin{bmatrix} 0.587 \\ 0.690 \end{bmatrix} \cdot \begin{bmatrix} 0.413 \\ 0.310 \end{bmatrix}^T * X^{[0]^T}$$

$$= \begin{bmatrix} 0.032 \\ 0.002 \end{bmatrix} \cdot \begin{bmatrix} 0.413 \\ 0.310 \end{bmatrix} * X^{[0]^T}$$

$$= \begin{bmatrix} 0.013 \\ 0.0006 \end{bmatrix} * X^{[0]^T} = \begin{bmatrix} 0.013 \\ 0.0006 \end{bmatrix} * \begin{bmatrix} 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.013 & 0.013 \\ 0.0006 & 0.0006 \end{bmatrix} = \frac{dL}{dW^{[1]}}$$

Now,

$$W^{[1]} := W^{[1]} - \eta \frac{dL}{dW^{[1]}}$$

$$= \begin{bmatrix} 0.1 & 0.25 \\ 0.1 & 0.7 \end{bmatrix} - 0.1 \begin{bmatrix} 0.013 & 0.013 \\ 0.0006 & 0.0006 \end{bmatrix}$$

$$= \begin{bmatrix} 0.1 & 0.5 \\ 0.1 & 0.7 \end{bmatrix} - \begin{bmatrix} 0.0013 & 0.0013 \\ 0.00006 & 0.00006 \end{bmatrix}$$

$$= \begin{bmatrix} 0.0987 & 0.2487 \\ 0.09994 & 0.69994 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

2. $W^{[2]} := W^{[2]} - \eta \frac{dL}{dW^{[2]}}$

The value of $\frac{dL}{dW^{[2]}}$ is already computed in Ans 1

which is: $\begin{bmatrix} -0.049 & 0.057 \\ 0.086 & 0.101 \end{bmatrix}$

So, $W^{[2]} = \begin{bmatrix} 0.4 & 0.5 \\ 0.6 & 0.3 \end{bmatrix} - 0.1 \begin{bmatrix} -0.049 & 0.057 \\ 0.086 & 0.101 \end{bmatrix}$

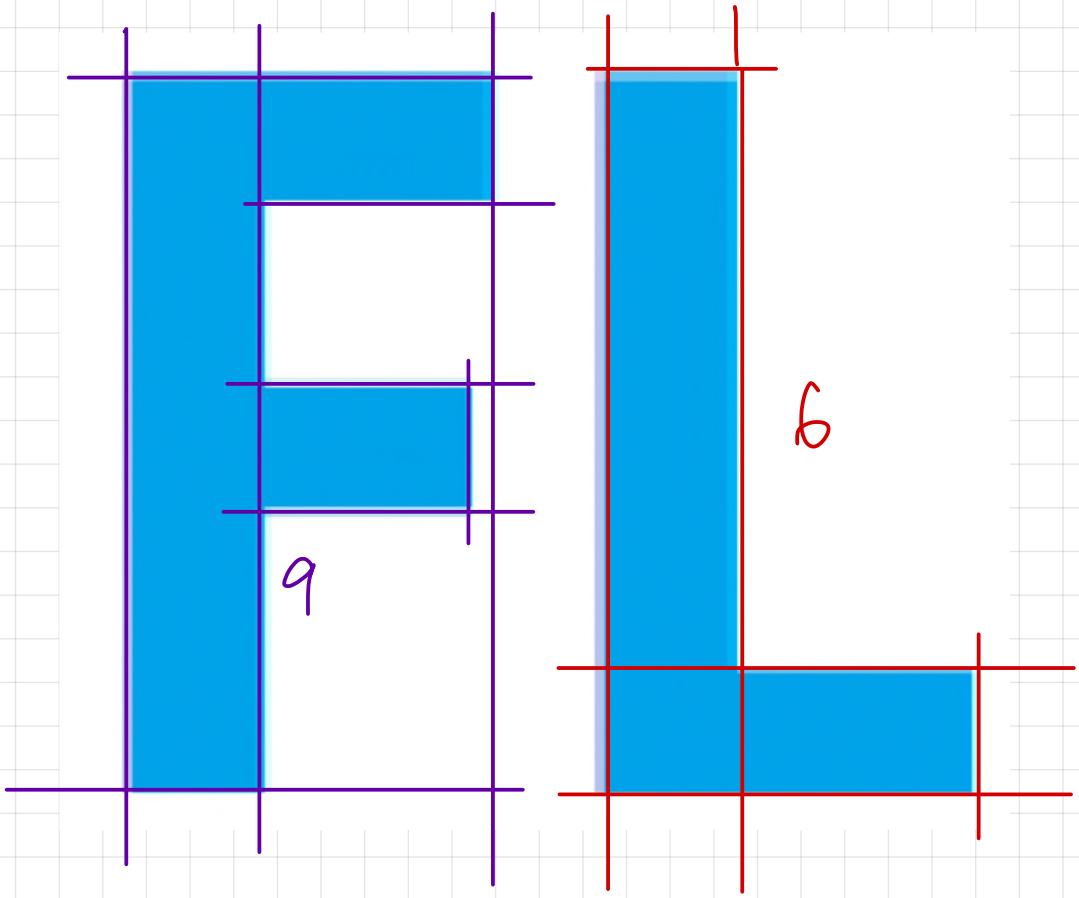
$$= \begin{bmatrix} 0.4 & 0.5 \\ 0.6 & 0.3 \end{bmatrix} - \begin{bmatrix} -0.0049 & 0.0057 \\ 0.0086 & 0.0101 \end{bmatrix}$$

$$= \begin{bmatrix} 0.4049 & 0.4943 \\ 0.5914 & 0.2899 \end{bmatrix} = \begin{bmatrix} w_{13} & w_{14} \\ w_{23} & w_{24} \end{bmatrix}$$

3. Create a 2-hidden layer:

How many units will you need in the first and second hidden layers? Why? Justify your answer by providing an explanation of each hidden unit role in creating this network. (Just write your answer. Not necessary to code a real model)
Can you achieve the same goal with a single hidden layer network? Why or why not?

The linear separable boundaries:



As we can see: F: 9 linearly separable boundaries

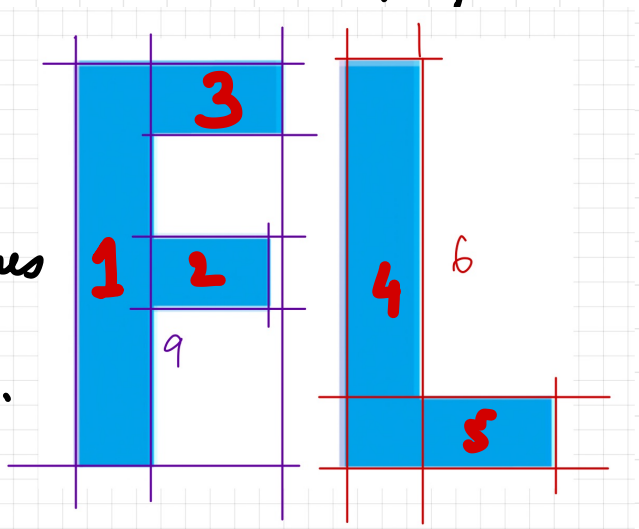
L: 6 linearly separable boundaries

So 15 boundaries are needed to identify the two letters

from the background.

This demands 15 perceptrons (neurons) in the 1st hidden layer to depict these 15 hyperplanes.

Now, for the 2nd layer, instead of computing hyperplanes This layer will identify regions.



In total, there are 5 regions.

So, We will have 5 perceptions.

∴ 15 neurons in the 1st Layer and 5 neurons in the 2nd layer.

Now, if asked to do the same problem in a single hidden layer, we can, since MLP's can compute any Boolean functions.

"MLPs are universal Boolean functions".

4. The regularization term

$$E_2 = \sum_{i=1}^{M} |w_i|$$

will be a better choice to bring in sparsity and drive unnecessary neurons to zero.

In L_1 regularization,

The cost function becomes: $\tilde{J}(w; X, y) = \|w\|_1 + J(w; X, y)$

The corresponding gradient is:

$$\tilde{\nabla} J(w; X, y) = \text{sign}(w) + \nabla_w J(w; X, y)$$

(actually subgradient)

↳ sign of w element-wise.

So, The regularization term no longer scales linearly with each w_i, instead it is a constant factor with sign equal to $\text{sign}(w_i)$.

The quadratic approximation of L1 regularized objective function decomposes into a sum over the parameter:

$$\hat{J}(w; X, y) = J(w^*; X, y) + \sum_i \left[\frac{1}{2} H_{i,i} (w_i - w_i^*)^2 + |w_i| \right]$$

Here, H is the Hessian matrix of J w.r.t w^* evaluated at w^*.

The analytical solution of minimizing the cost function is:

$$w_i = \text{sign}(w_i^*) \max \left\{ |w_i^*| - \frac{1}{H_{i,i}}, 0 \right\}$$

Assuming H is diagonal with each $H_{i,i} > 0$.

Now,

- if $w_i^* \leq 1/H_{i,i}$: Here $w_i = 0$.

This is because, contribution of $J(w; X, y)$ to the regularized objective $\tilde{J}(w; X, y)$ is overwhelmed in the direction of 'i' by the L^1 regularization which pushes the value of w_i to zero.

- $w_i^* > \frac{1}{H_{i,i}}$: Here, regularization does not move the optimal value of w_i to zero but shifts it in that direction by $1/H_{i,i}$.

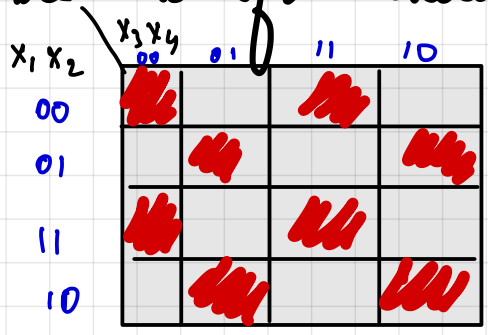
6.

6. Alex has built a Multi-layer Perceptron which consists of a single hidden layer of neurons. The model evaluates a boolean function f on four input variables X_1, X_2, X_3, X_4. Also it is known that each neuron in the network is simple enough and acts as a boolean gate. Alex observed that working with this simple network, he ends up using maximum number of neurons that is necessary to evaluate any boolean function in four variables. What is the number of neurons that Alex uses? If Alex is allowed to use any number of layers (i.e. he could increase the depth of the network) what would be minimum number of neurons required to model the same function? Can you say something about the importance of depth in an MLP from this example?

Multi-Layered Perceptron are universal Boolean functions.

By expressing a boolean function as a Truth Table,
A one hidden layer MLP is an Universal Boolean Function.

For a 4 variable input,
We can reduce the function using Karnaugh Maps.
The largest irreducible DNF (Disjunctive Normal Form) for 4 variable is of a checkered board



This can be generalized and seen that

The No. of neurons required is 2^{N-1} in the hidden layer.

Along with the o/p neuron,

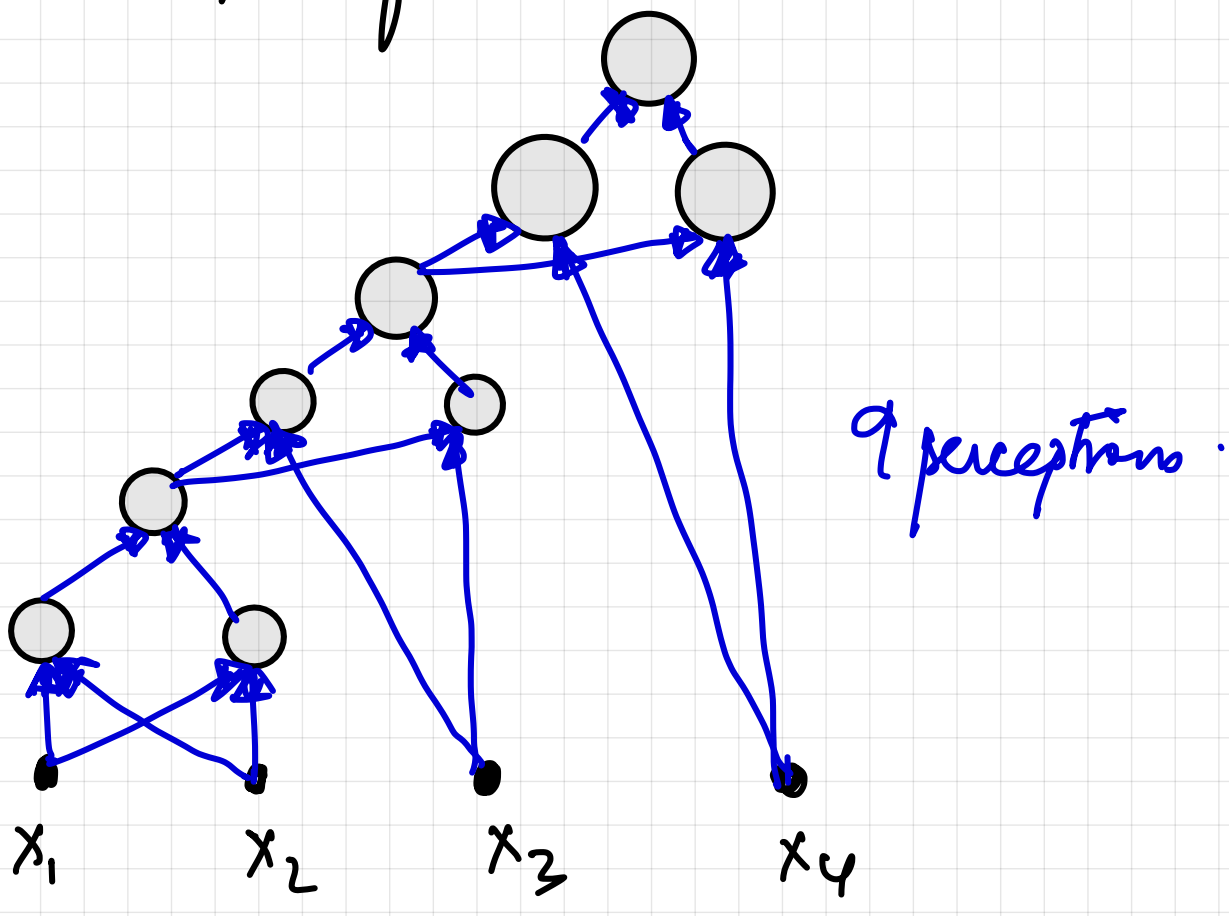
$$\text{No. of Neurons} = 2^{N-1} + 1 = 2^{4-1} + 1 = 8 + 1 = 9$$

!Ans!

If width is allowed,

An XOR MLP takes 3 neurons.

So, The 4 input function will need atmost



9 perceptrons.

Generally, a XOR of N variables will require 3(N-1) perceptrons.

So, 9 perceptrons. Ans'

5. In a presumably unfair dice the unknown probabilities of appearance of the individual faces are respectively $p_1, ..., p_6$ (none of them are zeros). Through repeated experiments it has been noted that the probability of rolling a streak of 6 consecutive numbers (i.e. rolling a sequence 1,2,..., 6) is greater than or equal to: $(\Pi_i p_i^{p_i})^{n/\Sigma_i p_i}$. What are the possible values of $p_1, ..., p_6$?

No. of faces of a dice = 6.

So, $n = 6$.

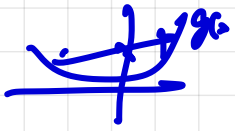
So, $\left(\prod_i p_i^{p_i}\right)^{6/\sum_i p_i} = \prod_i p_i$

Taking log on both sides,

$$\frac{6}{\sum_i p_i} \sum_i p_i \log p_i = \sum_i \log p_i$$

Let $f(x) = \log(x)$

Now $\log(x)$ is concave.



$E[g(x)] \geq g(E(x))$

Applying Jensen's Inequality now:

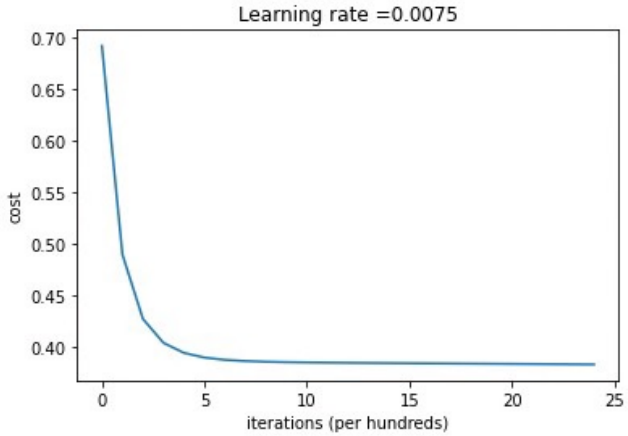
$$\sum_i f(p_i \log p_i) \geq f\left(\sum_i p_i \log p_i\right)$$

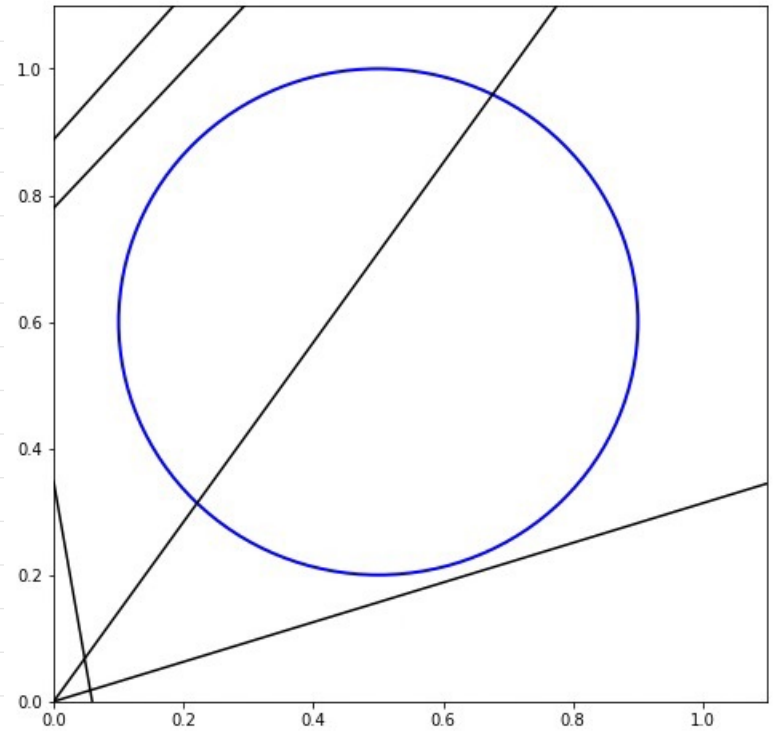
$$= 1 \sum_i \log(p_i \log p_i) \geq \log\left(\sum_i p_i \log p_i\right)$$

7. Code and test a two layer feed-forward net of sigmoidal nodes with two input units, ten hidden units and one output unit that learns the concept of a circle in $2D$ space. The concept is: $\langle x, y \rangle$ is labeled $+$ if $(x-a)^2+(y-b)^2 < r^2$ and is labeled $-$ otherwise. Draw all data from the unit square $[0,1]^2$. Set $a = 0.5, b = 0.6, r = 0.4$. Generate 100 random samples uniformly distributed on $[0,1]^2$ to train the network using error back-propagation and 100 random samples to test it. Repeat the procedure multiple epochs and with multiple initial weights. Report the changing accuracy and the hyperplanes corresponding to the hidden nodes (when the sigmoid is turned into a step function).

The data and plot for the sigmoid function is:

```
######### SIGMOID #############
Cost after iteration 0: 0.6919357043320539
Cost after iteration 100: 0.4897097419857324
Cost after iteration 200: 0.4272085087043201
Cost after iteration 300: 0.4040420723022598
Cost after iteration 400: 0.39426348082519336
Cost after iteration 500: 0.3897521065508461
Cost after iteration 600: 0.38752515322414904
Cost after iteration 700: 0.3863560680023228
Cost after iteration 800: 0.38569932185157724
Cost after iteration 900: 0.3852985726546028
Cost after iteration 1000: 0.3850285880447368
Cost after iteration 1100: 0.3848264651803757
Cost after iteration 1200: 0.38465997196680124
Cost after iteration 1300: 0.38451235190903055
Cost after iteration 1400: 0.38437480077532193
Cost after iteration 1500: 0.38424265509782013
Cost after iteration 1600: 0.3841134282405416
Cost after iteration 1700: 0.38398578585130394
Cost after iteration 1800: 0.3838590065437755
Cost after iteration 1900: 0.38373269615647226
Cost after iteration 2000: 0.3836066356605828
Cost after iteration 2100: 0.3834806999421674
Cost after iteration 2200: 0.3833548143343209
Cost after iteration 2300: 0.38322893131803143
Cost after iteration 2400: 0.383103018021253
```

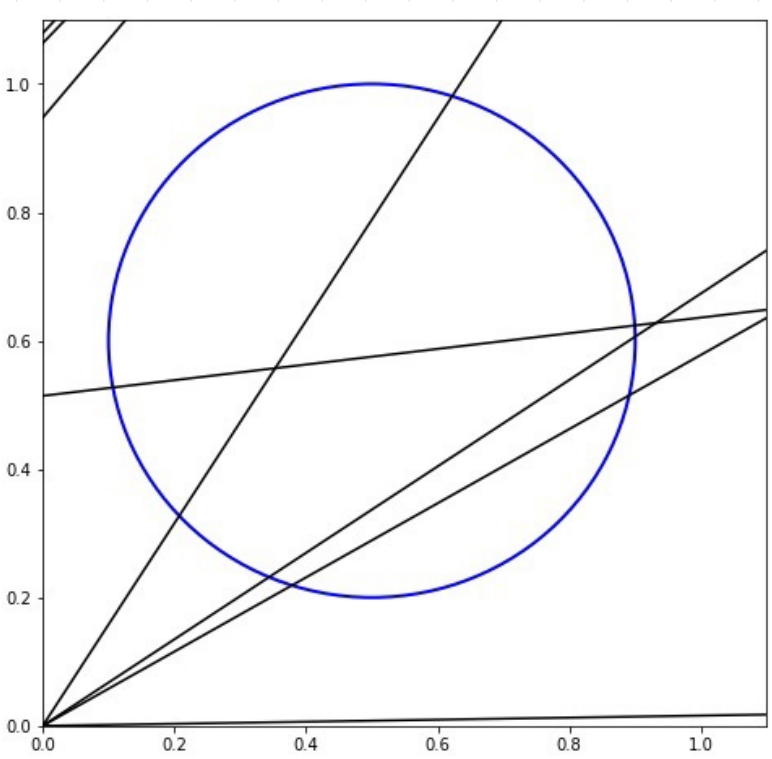


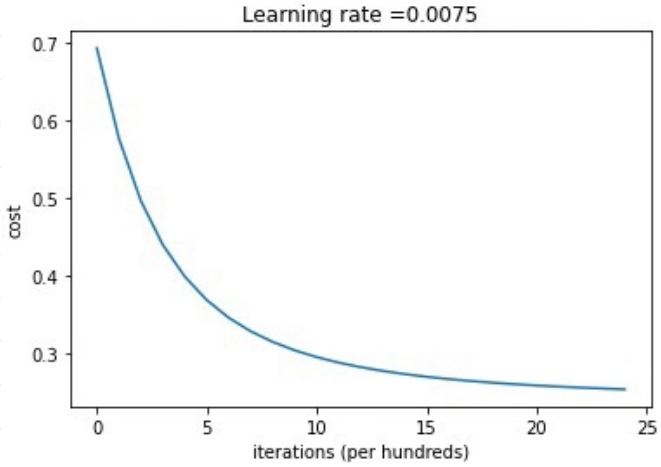


```
Accuracy: 0.8599999999999999
```

For the step function :

```
######### STEP #############
Cost after iteration 0: 0.6930324511318732
Cost after iteration 100: 0.5771267268390233
Cost after iteration 200: 0.4967734523214116
Cost after iteration 300: 0.44006910217263345
Cost after iteration 400: 0.39916982512451266
Cost after iteration 500: 0.3690053739546356
Cost after iteration 600: 0.34628445313838574
Cost after iteration 700: 0.32883816461366766
Cost after iteration 800: 0.31520890836984305
Cost after iteration 900: 0.3043963940625773
Cost after iteration 1000: 0.2956999645627685
Cost after iteration 1100: 0.2886191763187431
Cost after iteration 1200: 0.2827899777080524
Cost after iteration 1300: 0.277943105198174
Cost after iteration 1400: 0.2738763277119077
Cost after iteration 1500: 0.270435610271429
Cost after iteration 1600: 0.2675021049804851
Cost after iteration 1700: 0.26498301852868683
Cost after iteration 1800: 0.2628051024772523
Cost after iteration 1900: 0.26090994717861204
Cost after iteration 2000: 0.2592505348801733
Cost after iteration 2100: 0.2577886841119214
Cost after iteration 2200: 0.256493134027846
Cost after iteration 2300: 0.2553380930925627
Cost after iteration 2400: 0.2543021291295199
###### W1 #########
(10, 2)
###### W2 #########
(1, 10)
###### b1 #########
(10, 1)
###### b2 #########
(1, 1)
```






```
Accuracy: 0.83
```