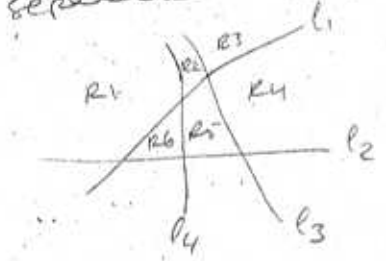
How Many Hidden Layers?

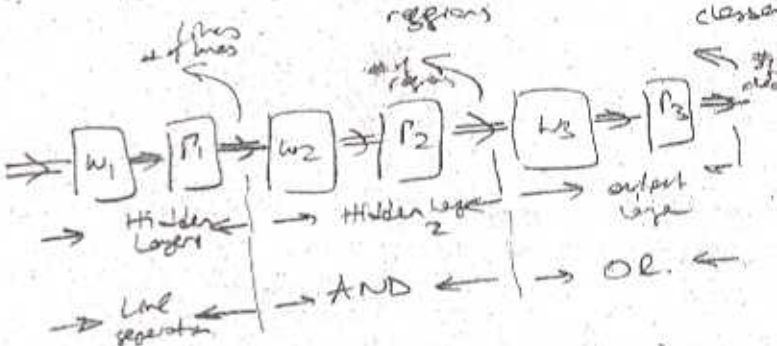
1) Too many layers slow down the convergence.

Too few may result in divergence

(CLASSIFICATION VIEWPOINT)

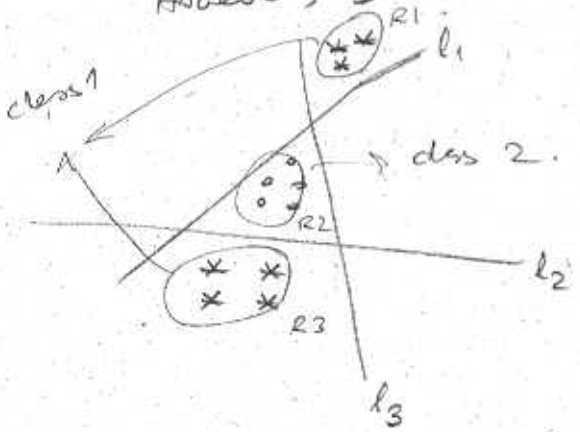
We know that at most 3 layers (2 hidden + one output) is sufficient to classify linear separable classes



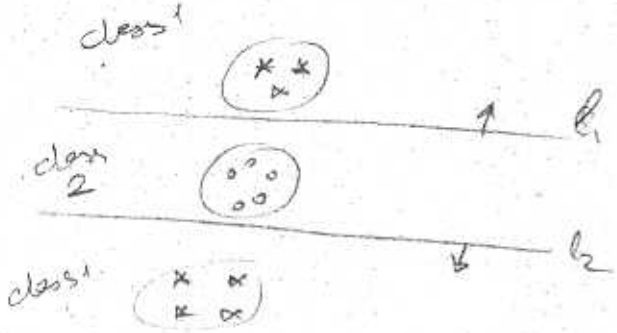


So in this approach 2 hidden layers (+1 output layer) is sufficient and safe for many applications.

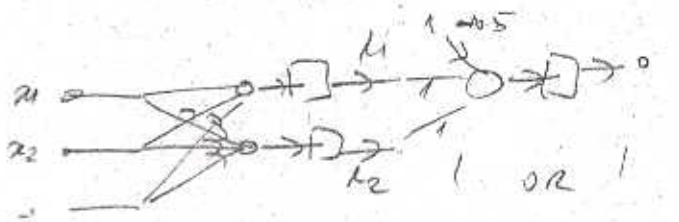
However, sometimes one of the layers may not be necessary.



⇒ In this case region separation (AND) and union (OR) blocks are necessary.



⇒ can problem viewed this way.



$l_1 = 1$ OR $l_2 = 1$ ⇒ class 1

(AND block is not necessary!)

⇒ FUNCTION APPROXIMATION VIEWPOINT

One hidden layer + one output layer is sufficient for approximating any continuous function.

Hence: 2 hidden layers should be used at most.

For approximation → 1 hidden layer is sufficient.

Advantage of one hidden layer: Simple structure, but may not be sufficient for some problems.

Disadvantage: Too many neurons may be required. That may slows down the convergence.

Advantage of two hidden layers: sufficient for many applications. total # of weights may be less than single layer one.

Disadvantage: Slightly more complex structure.

RECOMMENDATION: May start with single layer. If error cannot be reduced below a prescribed level, add another layer.

OR: For simple-looking problems (function approximation, classification with few classes, use single layer)
For seemingly complex problems (classification with large number of classes) use two hidden layers.

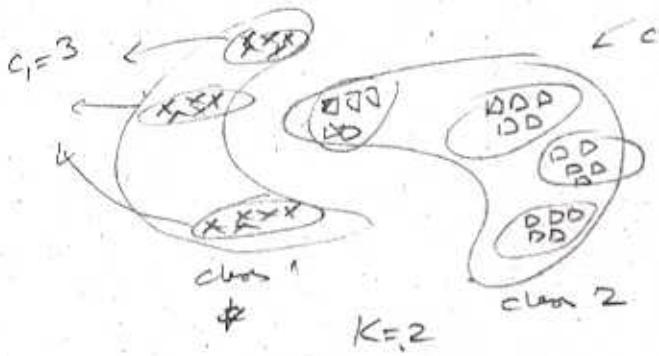
single hidden

How many neurons in single hidden layer

K: Number of distinct classes

Each class is assumed to be divided into linearly separable C_i subclasses.

total linearly separable subclasses: $C = C_1 + C_2 + \cdots + C_K$



#of classes

$K=2$
$C = C_1 + C_2 = 7$ → # of regions

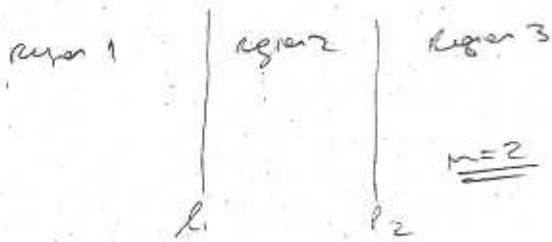
Each neuron output separates the input space into 2 regions.

→ M neurons : at most : 2^M regions

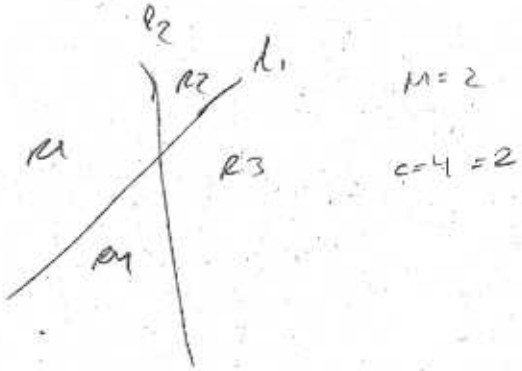
at least : $M+1$ regions.

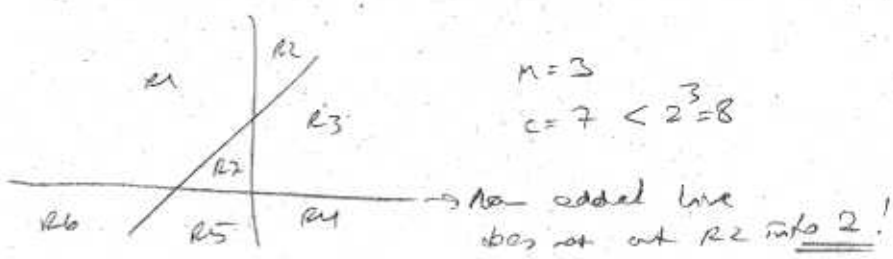
At most: Each line cuts the previous spaces into 2.

At least: Lines are parallel.



Total regions: $3 = 2+1$





In general: upper bound may not be achieved (It is almost impossible for a newly added line to cut all of the previous regions into 2

(pessimistic)
(least regions)

(optimistic)
most regions

$$M+1 \leq C \leq 2^M$$

$$\Rightarrow \quad \boxed{M < C - 1} \qquad \boxed{M > \frac{\ln C}{\ln 2}}$$

Take average!

$$\boxed{\text{Move} = \frac{1}{2}\left((C-1) \cdot \frac{\ln C}{\ln 2}\right)}$$

For large C $\quad C \gg \ln C \quad (C \gg 1) \quad \rightarrow \quad \boxed{\text{Move} \approx \frac{1}{2}C}$

(If we know C!)

In general, we know K, but we hardly know C. One assumption: On the average, each class K contains 2 subclasses that are linearly separable.

$\Rightarrow$ An estimate for C $\Rightarrow$ $\boxed{C = 2K}$

$\Rightarrow$ Use $\boxed{M = \frac{\ln 2K}{\ln 2}}$ neurons.

For highly (complex) nonlinear cases (?) we may use

$\boxed{C = pK}$ $\quad 2 \le p \le 4$

$\rightarrow \boxed{M = \frac{\ln pK}{\ln 2}}$ $\quad 2 \le p \le 4$

$\rightarrow$ A realistic estimate for regions:

The upper bound on the number of regions is $C \le 2^M$. This is highly unrealistic (Because we assume that a new neuron $\Rightarrow$ a new separating hyperplane $\Rightarrow$ this plane cuts all of the previous regions into two.

Assume $C = r^M$ $\quad 1 < r < 2$ $\Rightarrow$ use $r = 3/2$

$\rightarrow C = \left(\frac{3}{2}\right)^M$.

$\rightarrow \boxed{M = \frac{\ln C}{\ln 3/2}}$

For C, use $C = 2K$ as before

$$\rightarrow \boxed{Max = \frac{\ln 2K}{\ln 3/2}} \quad ! \text{ or } pK$$

For two hidden layers, the previous heuristics will not work. Instead use the following:

L: # of neurons in the first hidden layer

M: # of " " " second " "

$$L = \sqrt{2K} + P$$

$$M = \sqrt{2K} - P$$

$$\boxed{2 \le P \le \frac{\sqrt{2K}}{2}}$$

select P such that $L \approx 2M$. (eg. choose $\boxed{P \approx \frac{\sqrt{2K}}{3}}$)

FOR OUTPUT LAYER. J: # of neurons in output layer

For small K, the best choice is $\boxed{J = K}$ ($K \le 16$). For $K > 16$, use $\boxed{J = \log_2 K = \frac{\ln K}{\ln 2}}$

(assume binary code at the output)

The first choice represents the maximum selectivity

The second " " " binary coding.

Hypothetical Design:

Design a MLP to recognize a set of 100 symbols. Each pattern is represented with a code length 60. ($x \in \mathbb{R}^{60}$).

$K = 100$

Single layer case:

$C = 2K = 200$!

Size of the hidden layer: The permissitic case:

$M \leq 2K - 1 = 199$ ⇒

optimistic case: $M = \frac{\ln 2K}{\ln 3/2} \simeq 12$

take average ⇒ $M_{ave} = \frac{12+199}{2} = 105$

output layer: $K = 100 > 16$ ⇒ use binary code

$J = 7$ ($2^7 = 128 > 100$, $2^6 = 64$, not efficient)

Two layer case:

$\sqrt{2K} \simeq 14$ $\frac{\sqrt{2K}}{3} \simeq 4$ ⇒

$L = 14 + 4 = 18$ ⇒ # of neurons in the first layer

$M = 14 - 4 = 10$ ⇒ # of neurons in the second layer

$J = 7$

60.199

Total weights:

$60 \times 12 + 12 \times 7 = 804$ (the next optimist)

$60 \times 199 + 199 \times 7 = 13333$ (the next pessimist)

$60 \times 105 + 105 \times 7 = 7035$ → (average)

Two layer case:

Total weights:

$60 \times 18 + 18 \times 10 + 10 \times 7 = 1330$ (reasonable).