1

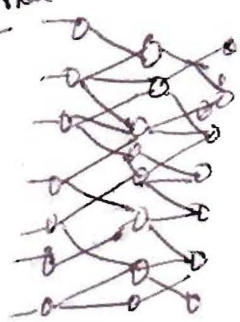
Monday Sep 16, 2013

Neural Networks:-

1. One of the first models for learning
2. Motivated by "neurons" in the brain.
3. Massively parallel Architecture

   ~10^{10} neurons
   ~10^{4-5} connections per neuron
   ~switching time ≃ .001 seconds
   ~scene recognition ~ .1 second

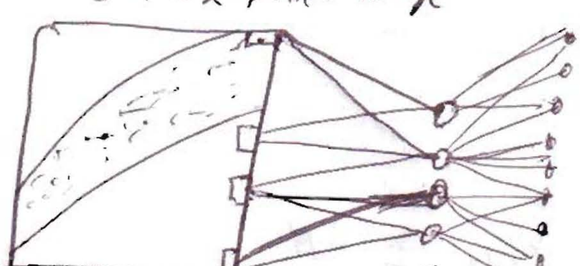
external input



→ switch to on/off state

Example: Autonomous Car Driving

30X32 pixel image



→ sharp left

30 different outputs

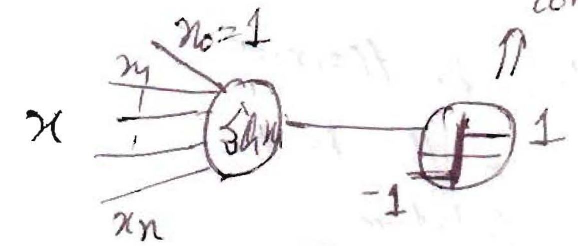
→ sharp right

⇓ 4 Hidden units

Learn the "weights" / functions.

Basic Unit:-

Perceptron:- (neuron)



computes $g(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$

$y^{(i)} \in \{-1, 1\}$

$h_\theta(x) = \begin{cases} 1 & \text{if } \theta^T x \geq 0 \\ -1 & \text{if } \theta^T x \underline{< 0} \end{cases}$

$\Downarrow$

$g(\theta^T x)$

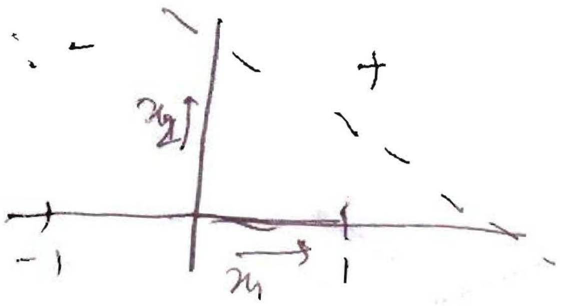
What kind of function is this?
(classifier)

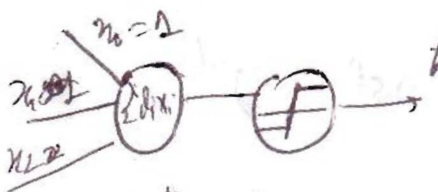
Linear

~~Can Represent linearly functions~~

Can classify linearly separable Data

Example:- AND function





$h_\theta(x) \equiv$ AND

θ_j's?

$\theta_2 x_2 + \theta_1 x_1 + \theta_0 > 0$ if $x_1 = 1$, $x_2 = 1$

$\theta_2 x_2 + \theta_1 x_1 + \theta_0 < 0$ o.w.

$\theta_2 x_2 + \theta_1 x_1 > -\theta_0$ $\Bigg\}$ $\boxed{\sum_{j=1} \theta_j x_j > \text{threshold}}$ $\to$ +ve / -ve

$\underline{\theta_2 = 1,\ \theta_1 = 1,\ -\theta_0 = 1.5}$ $\Downarrow$ Alternate formulation

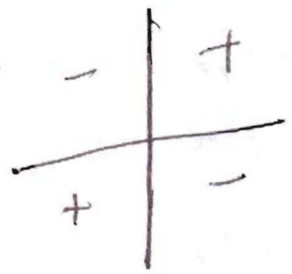
Represents AND

(2)

What about OR function?

$\theta_2 = 1$
$\theta_1 = 1$
$\theta_0 = -0.5$

XOR?



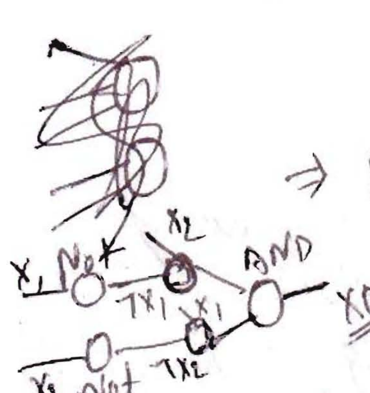
There is no line which separates + & −.

$\Rightarrow \nexists\ \theta_2, \theta_1, \theta_0$
s.t. $\theta_2 x_2 + \theta_1 x_1 + \theta_0 > 0$ for +ve
AND $\theta_2 x_2 + \theta_1 x_1 + \theta_0 < 0$ for −ve.

Example

$\Rightarrow$ Perceptron is a linear classifier.
$\rightarrow$ Need to combine individual perceptrons to get a non linear classifier.

$\boxed{x_1 \oplus x_2 = x_1 \neg x_2 + \neg x_1 x_2}$



How do we learn the weights (parameters)?

Perceptron training rule (Delta rule)

$$\Delta\theta_j = \eta\left[y - h_\theta(x^{(i)})\right] x_j^{(i)}$$ similar to gradient descent

where $x^{(i)}$ is a wrongly classified example.

Note $J(\theta) = \sum_{i=1}^{m}\left[y^{(i)} - h_\theta(x^{(i)})\right]^2$ is not differentiable

So, not really gradient descent

Algorithm-

init(θ)
do{ $x^{(i)} \leftarrow$ wrongly classified example based on
  for $j = 1$ to n
    $\Delta\theta_j = \eta\,(y^{(i)} - h_\theta(x^{(i)}))\, x_j^{(i)}$

    $\theta_j \leftarrow \theta_j + \Delta\theta_j$
} until (all examples correctly classified)

Properties:- Guaranteed to converge for linearly separable data

(Assuming η is sufficiently small).

Won't work non-linearly separable data

Issues:- 1. Sort of ad-hoc?

2. Non- differentiable error function

3. Can not handle non linear data

Context:- Much before then some modern algorithms came about.

Why studying:-

1. Historical context
2. Works well for certain class of problems (Digit recognition)
3. Similarity to working on brain
4. Deep, Networks:- Very powerfull Belief effective.

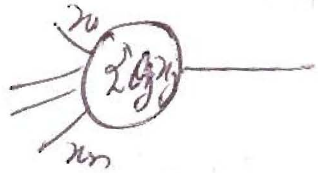
On going research

Rectify some of the problems:-

$h_\theta(x) = \theta^T x$.

Directly optimize the unthresholded Output.

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m} [y^{(i)} - h_\theta(x^{(i)})]^2$$



$\Downarrow$

$\theta^T x^{(i)}$

Similar to linear regression -

Difference:- $y^{(i)} \in \{-1, 1\}$

(3)

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{2}\sum_{i=1}^{m} 2(y^{(i)} - h_\theta(x^{(i)})) \cdot \frac{\partial}{\partial \theta_j} (-1)\, x_j^{(i)}$$

Aside:-
$\left[\frac{\partial}{\partial \theta_j} \theta^T x^{(i)} = x_j^{(i)} \right]$ We want to minimize $J(\theta)$

$$\Rightarrow \theta_j = \theta_j - \eta \frac{\partial}{\partial \theta_j} J(\theta)$$

$\Rightarrow$ update rule:-

$$\theta_j = \theta_j + \Delta\theta_j$$

$$\Delta\theta_j = \eta \sum_{i=1}^{m} [y^{(i)} - h_\theta(x^{(i)})]\, x_j^{(i)}$$

Batch Gradient Descent / Stochastic Gradient Descent

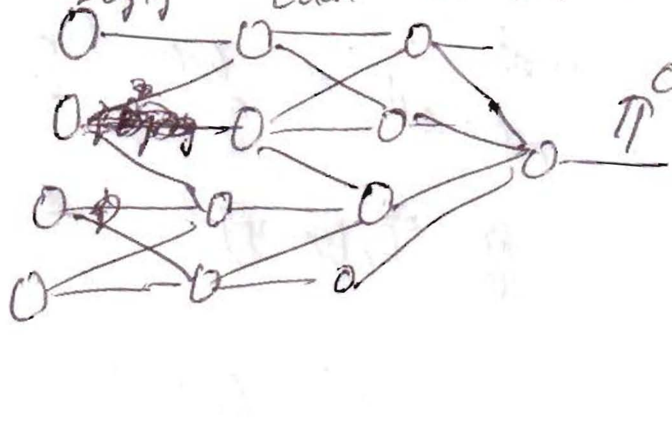
Issues:- (1) $y^{(i)} \in \{-1, 1\}$

$h_\theta(x^{(i)}) \in (-\infty, \infty)$

$\Rightarrow$ Bigger issue:- Can not handle non linear data!

[*Could use combination thresholded perceptrons → but non differentiable]

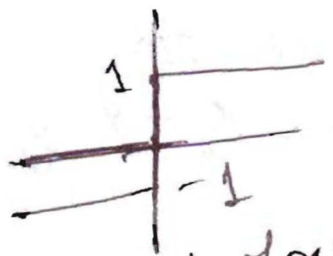
$\sum \theta_j x_j$ Each unit computes a linear fn

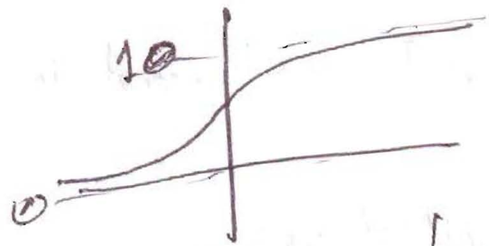


output is a linear function of inputs

How do we solve this?

Idea:- Use a sigmoid unit (instead of thresholding)





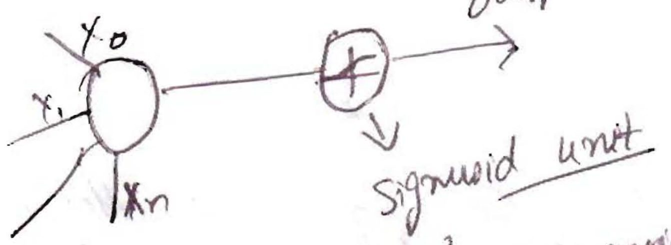
$$g(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

$$g(x) = \frac{1}{1+e^{-x}}$$

$\Downarrow$

$y^{(i)} \in \{0, 1\}$

$\Rightarrow \quad h_\theta(x) = \dfrac{1}{1+e^{-\theta^T x}}$



perceptron with sigmoid unit ~~at the~~ ~~output~~ at the output

Properties:-

1. Differentiable
2. can represent Non-linear functions.

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} [y^{(i)} - h_\theta(x^{(i)})]^2$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{2} \sum_{i=1}^{m} [2(y^{(i)} - h_\theta(x^{(i)}))] \cdot (-1) \frac{\partial}{\partial \theta_j} h_\theta(x^{(i)})$$

$$\frac{\partial}{\partial \theta_j} h_\theta(x^{(i)}) = \frac{\partial}{\partial [\theta^T x^{(i)}]} \underbrace{h_\theta(x^{(i)})}_{\text{sigmoid}} \cdot \frac{\partial [\theta^T x^{(i)}]}{\partial \theta_j}$$

(7)

$$\frac{d}{dx} g(x) = g(x)(1-g(x))$$

$$g(x) = \frac{1}{1+e^{-x}} \qquad h_\theta(x) = g(\theta^T x)$$

$$\Rightarrow \frac{d}{\partial \theta_j} h_\theta(x^{(i)}) = h_\theta(x^{(i)})(1-h_\theta(x^{(i)}))\, x_j^{(i)}$$

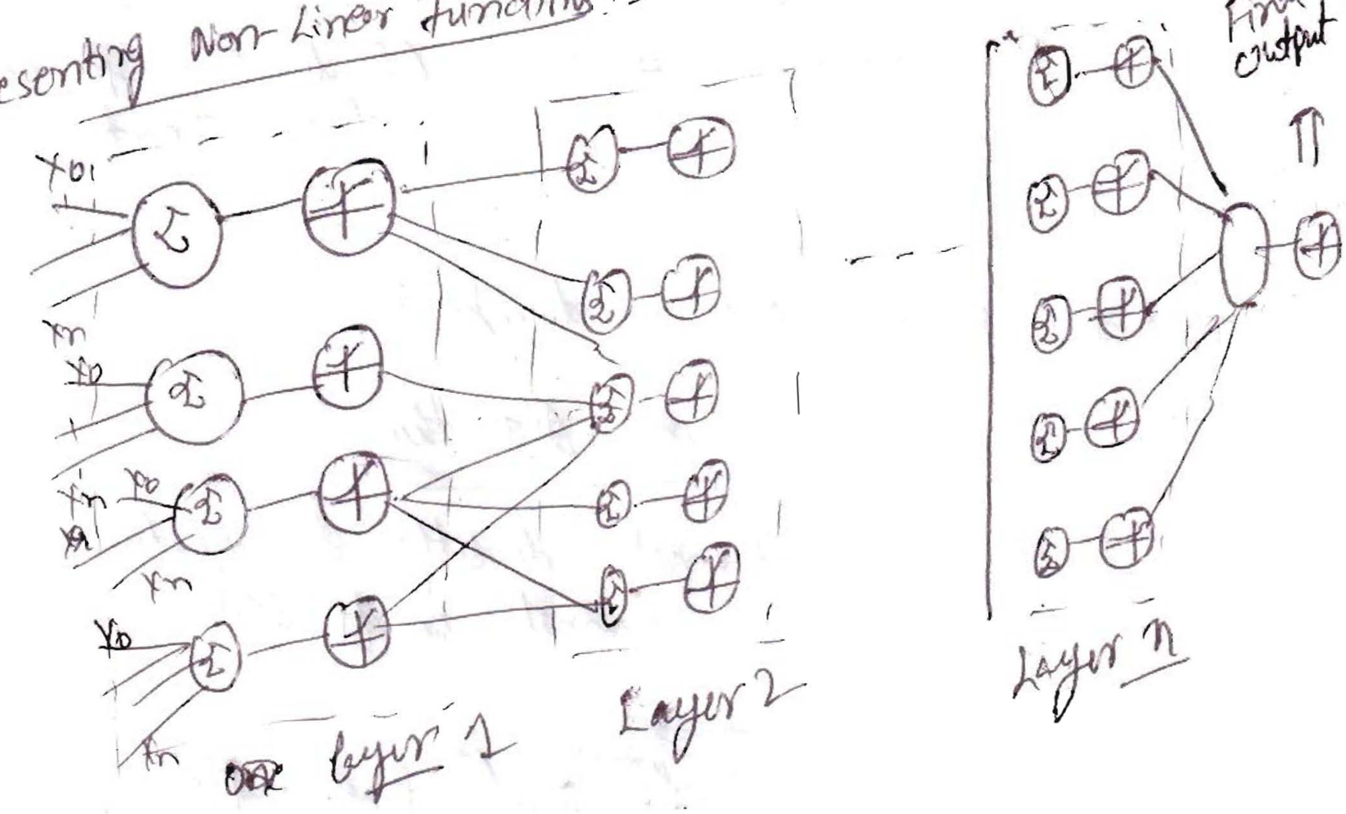
$$\boxed{\Rightarrow \frac{d}{\partial \theta_j} J(\theta) = \sum_{i=1}^{m} \left[y^{(i)} - h_\theta(x^{(i)})\right](-1)\, h_\theta(x^{(i)})(1-h_\theta(x^{(i)}))\, x_j^{(i)}}$$

$$\theta_j \leftarrow \theta_j + \eta \sum_{i=1}^{m} [y^{(i)} - h_\theta(x^{(i)})][h_\theta(x^{(i)})][1-h_\theta(x^{(i)})]\, x_j^{(i)}$$

(1) Different from logistic regression → what is difference? why?

(2) Non-convex optimization ⇒ Local minima ≠ Global minima

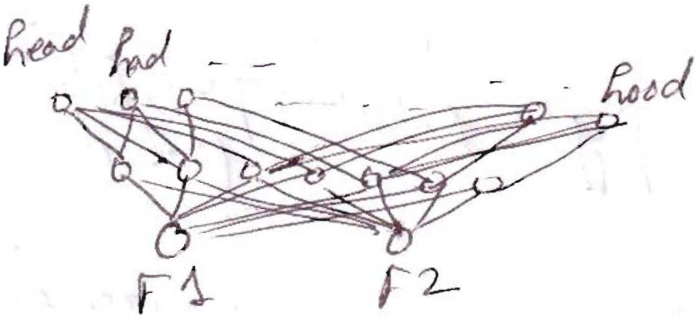
⇒ Representing Non-Linear Functions :-



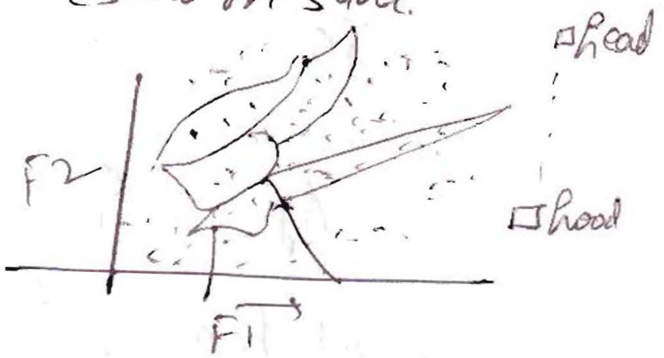
Examples:-

1. Non-linear bounderies

Recognition of different sounds given two different frequency inputs (characterizing sound)



⇒ Decision surface (show on slide.

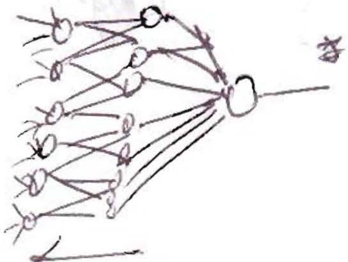


2. Video:- Andrew Ng Lecture 6
   1. 37:08 - 39:45 (Digit Recognition)
   2. 41:30 - 42:50 (Producing Speech from Text)

Training Multi-layered Networks :-

Backpropagation Algorithm

Main Issue:- $J(\theta)$ not directly a function of original inputs/weights



* final output

compare with $y^{(i)}$

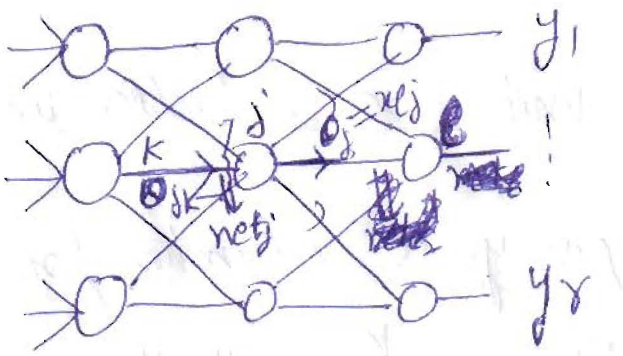
depends on inputs to this layer (& not original inputs/weights directly).

propagate the gradient back

Neural Networks

Sep 19, 2013

Deriving backpropagation :-



$$net_j = \sum_K \theta_{jK} \cdot x_{jK}$$

We ~~woild~~ will do it for one example

Let the output of units in last(output) layer be: O_1 .. or (implicitly a function of x).

$$J(\theta) = \frac{1}{2}\sum_{l=1}^{r} (y_l - O_l)^2$$

Note: θ_{jK} can influence $J(\theta)$ only through net_j

Now,

$$\boxed{\frac{\partial J(\theta)}{\partial \theta_{jK}} = \frac{\partial}{\partial net_j} J(\theta) \cdot \frac{\partial net_j}{\partial \theta_{jK}} = \frac{\partial J(\theta)}{\partial net_j} \cdot x_{jK}}$$

① Output units :-

Note: net_j can influence $J(\theta)$ only through O_j [~~since we are considering change in W_{jK} & W_{jK} are unique for each unit~~]

$$\frac{\partial J(\theta)}{\partial net_j} = \frac{1}{2}\frac{\partial}{net_j}\sum_{l=1}^{r}(y_l - O_l)^2$$

$$= \frac{1}{2}\frac{\partial}{net_j}(y_j - O_j)^2$$

$$= \frac{1}{2} \times 2 (y_j - O_j)(-1)\frac{\partial O_j}{\partial net_j} = -(y_j - O_j) \cdot (O_j)(1 - O_j)$$

Let

$$\boxed{\frac{\partial J(\theta)}{\partial net_j} = \delta_j} \qquad \boxed{\delta_j = (y_j - O_j)\, O_j (1 - O_j)}$$

$j \in$ output layer

Next, <u>hidden layers</u>:-

Let unit j be a hidden unit

<u>Aside</u>:-

$f(y_1 - y_k)$ suppose y_j is a function of x

Then $$\frac{\partial f(y_1 - y_k)}{\partial x} = \sum_{j=1}^{k} \frac{\partial f(y_1 - y_k)}{\partial y_j} \cdot \frac{\partial y_k}{\partial x}$$

Using this:-

$$\frac{\partial J(\theta)}{\partial net_j} = \sum_{\ell \in downstream(j)} \frac{\partial J(\theta)}{\partial net_\ell} \cdot \frac{\partial net_\ell}{\partial net_j}$$

$$\frac{\partial net_\ell}{\partial net_j} = \frac{\partial \sum_j \theta_{\ell j} x_{\ell j}}{\partial O_j} \quad \frac{\partial O_j}{\partial net_j}$$

$$= \theta_{\ell j} \cdot O_j (1 - O_j)$$

$$\Rightarrow \frac{\partial J(\theta)}{\partial net_j} = \sum_{\ell \in downstream(j)} - \delta_\ell \cdot \theta_{\ell j} \cdot O_j (1 - O_j)$$

$$\boxed{\Rightarrow \delta_j = O_j (1 - O_j) \sum_{\ell \in downstream(j)} \delta_\ell \cdot \theta_{\ell j}}$$

$j \in$ <u>hidden unit</u>