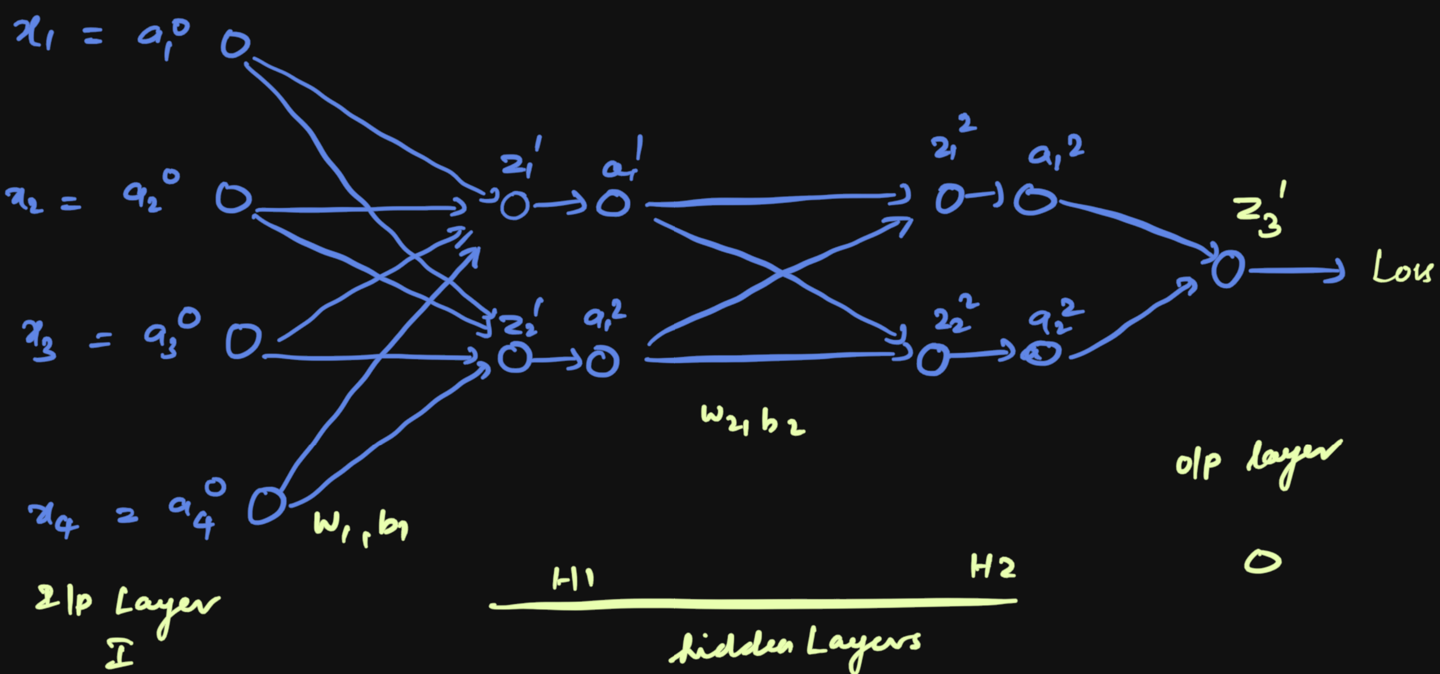
MLP & RBF

- LOKESH . N

(Thanks to prof. Annie)

Because MLP and RBF are already covered in class. Let us go over Numericals to get a complete flavour of it

Disclaimer – There may be mistakes in derivation.
Pls cross check once.



By convention, $W_1 = 2 \times 4$ matrix

$\underset{\sim}{x} = 4 \times 1$ column vector

and If mini Batched – $[4 \times B]$

Fwd propagation

$$\begin{bmatrix} z_1^1 \\ z_1^2 \end{bmatrix} = \begin{bmatrix} W_{2\times4} \end{bmatrix} \begin{bmatrix} x \\ 4\times1 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$$\boxed{Z_1 = W_1 x + b_1}$$

$a_1 = \sigma(z_1) \rightarrow$ elementwise operation

$\sigma =$ Relu / sigmoid

Relu = $\max(0, x)$

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

III'ly $\quad z_2 = \overset{[2\times 2]}{W_2}\, a_1 + b_2$

$a_2 = \sigma(z_2)$

Finally $\quad o = z_3' = \overset{[1\times 2]}{W_3}\, a_2 + b_3$

Loss = mean squared Error

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} (y - o)^2$$ For Batched case

$$\mathcal{L} = (y-o)^2$$ for single example

X ent

$$\mathcal{L} = y \ln o + (1-y) \ln (1-o)$$

Now let us derive Back prop rules.

Goal

find $\frac{\partial L}{\partial W_3}, \frac{\partial L}{\partial b_3}, \frac{\partial L}{\partial W_2}, \frac{\partial L}{\partial b_2}, \frac{\partial L}{\partial W_1}, \frac{\partial L}{\partial b_1}$

and update rule is

parameters of the NN = $\Theta = \{W_i, b_i\}_{i=1}^{\# \text{Layers}}$

$$\Theta_{new} = \Theta_{old} - \eta \nabla \Theta_{old}$$

what does chain rule say?

assume f and g are continuous & differentiable

(need not be, we may still use sub gradients.) — out of scope.

$$h(\theta) = f(g(\theta))$$

$$h'(\theta) = f'(g(\theta)) * g'(\theta) * 1$$

sneak peek at fwd prop

$$x \longrightarrow \underbrace{z_1 = w_1 x + b_1}_{f_1} \rightarrow \underbrace{a_1 = \sigma(z_1)}_{f_2} \rightarrow \underbrace{z_2 = w_2 a_1 + b_2}_{f_3} \downarrow$$

$$\mathcal{L} = (y - o)^2 \longleftarrow \underbrace{o = w_3 a_2 + b_3}_{f_5} \longleftarrow \underbrace{a_2 = \sigma(z_2)}_{f_4}$$

If we Interpret $\mathcal{L} = (NN(x) - o)^2$

$$NN(x) = f_5(f_4(f_3(f_2(f_1(x)))))$$

$w_3\ b_3$ $\quad$ $w_2\ b_2$ $\quad$ $w_1\ b_1$

f_5 f_3 and f_1 are linear

$$\Rightarrow \frac{\partial f_5}{\partial w_3} = \frac{\partial o}{\partial w_3} = \vec{a_2}$$

$$\frac{\partial f_5}{\partial b_3} = \frac{\partial o}{\partial b_3} = \vec{1}$$

111^{4}

$$\frac{\partial f_3}{\partial w_2} = a_1 \ ?$$ wrong

$$f_3 = \begin{bmatrix} w^2_{11} & w^2_{12} \\ w^2_{21} & w^2_{22} \end{bmatrix} \begin{bmatrix} a_1^1 \\ a_1^2 \end{bmatrix}$$

$$= \begin{bmatrix} w_{11}^2 a_1^1 & w_{12}^2 a_1^2 \\ w_{21}^2 a_1^1 & w_{21}^2 a_1^2 \end{bmatrix}$$

$$\frac{\partial f}{\partial w_2} = \begin{bmatrix} a_1^1 & a_1^2 \\ a_1^1 & a_1^2 \end{bmatrix}$$

derivative of activation function

If $\sigma =$ Relu $\quad \sigma' = \begin{cases} 0 & \text{If } Ip \le 0 \\ 1 & \text{If } Ip > 0 \end{cases}$

If $\sigma =$ Sigmoid $\quad \sigma' = \sigma(1-\sigma)$

derivative with loss

$$\mathcal{L} = y \ln o + (1-y) \ln (1-o)$$

$$\mathcal{L}' = y * \frac{1}{o} + (1-y) \frac{1}{1-o}$$

In this derivation Let us use MSE Error

Recall

$$\mathcal{L} = \frac{1}{2} (y-o)^2$$

$$\mathcal{L}' = (y-o)(-1)$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial o} * \frac{\partial o}{\partial w_3} = -(y-o) * a_2$$

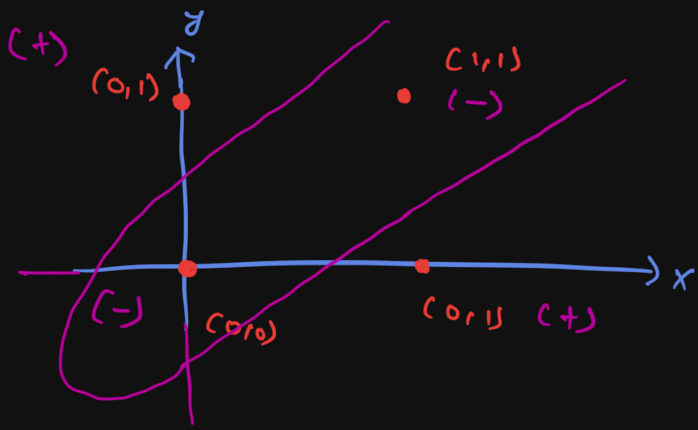
$$\frac{\partial L}{\partial b_3} = \frac{\partial L}{\partial o} * \frac{\partial o}{\partial b_3} = -(y-o) * \vec{I}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial o} * \frac{\partial o}{\partial a_2} * \frac{\partial a_2}{\partial z_2} * \frac{\partial z_2}{\partial w_2}$$

$$= -(y-o) * w_3^T * \sigma(z_2)(1-\sigma(z_2)) * \begin{bmatrix} \vec{a_1} \\ \vec{a_1} \end{bmatrix}$$

you can keep unrolling this and Immediately see an opportunity For "Dynamic programming"

H.W - Complete the derivation using 'DP' and Implement the DP formula you get classical XOR problem



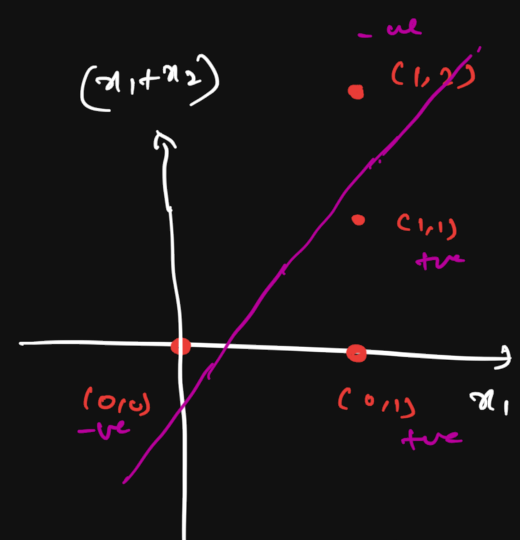
Any linear plane cannot separate the above simple 4 points dataset !

Logistic Regression / Perceptron will Fail

Solution - Basis Expansion

x_1	x_2	x_1+x_2	y
0	0	0	-ve
0	1	1	+ve
1	0	1	+ve
1	1	2	-ve

$\longrightarrow$



In new Basis Linear plane can seperate

Now Log Reg / Perceptron can win in Basis (x_1, x_1+x_2)

Can a Human Hand craft such Basis / feature function

Beauty of Neural Networks

Visualize NN(x) as such very complicated feature functions

NN learn them automatically through Back propagation

Universal Approximation Theorem

(vaguely) under certain conditions, a 3-Layer NN

with p hidden nodes can approximate any function

→ But no one knows how to find 'p'

can $p \to \infty$?

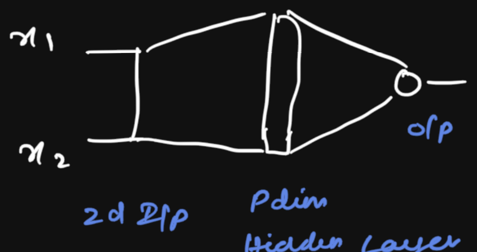
H.W

Solve XOR pblm using 2-Layer Neural N/w, Logistic Regression

and 3 layer Neural N/w and plot the Loss curves

Data = XOR table Label = XOR output

For 3 Layered NN



We can interpret the activations of Hidden Layer as Basis functions

For Each of the 4 data points take activations and train Logistic Regression model on activations.

Radial Basis Networks

① There are just 3 Layered N/Ws

assume m I/P nodes

p Hidden nodes

1 o/p node.

② But we need the activations to be Radially Symmetric.

↳ Each Neuron should account 'only' locally.

Earlier we had $\sigma(w^T a)$

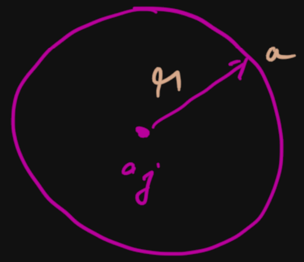
and activation fired for a long Range on 'a'

Now Each Neuron fixes a 'a_j' and accounts

for only certain distance around 'a_j'

In 1-d

In 2-d



accounts for a radius 'η' around a_j

Hence the name radially symmetric

The accountability of a neuron with center 'a_j' decreases as distance of a from it increase

Let us call such an activation as ϕ

Then possibilities of ϕ are

① $\phi(a) = \exp\left(-\frac{1}{2}\frac{\|a - a_j\|^2}{\sigma^2}\right)$

If $\sigma = 1$

$a = a_j \Rightarrow \phi(a) = 1$

$a = a_j + \xi \Rightarrow \phi(a+\xi) = \exp\left(-\frac{1}{2}\|\xi\|^2\right) < 1$

$a = a_j - \xi \Rightarrow \phi(a-\xi) = \exp\left(-\frac{1}{2}\|\xi\|^2\right) < 1$

Because $\phi(a_j + \xi) = \phi(a_j - \xi) \quad \forall \xi$

ϕ = radially symmetric

H.W → check relation b/w

$\phi(a_j + \xi_1)$ and $\phi(a_j + \xi_2)$

If $\xi_1 > \xi_2$

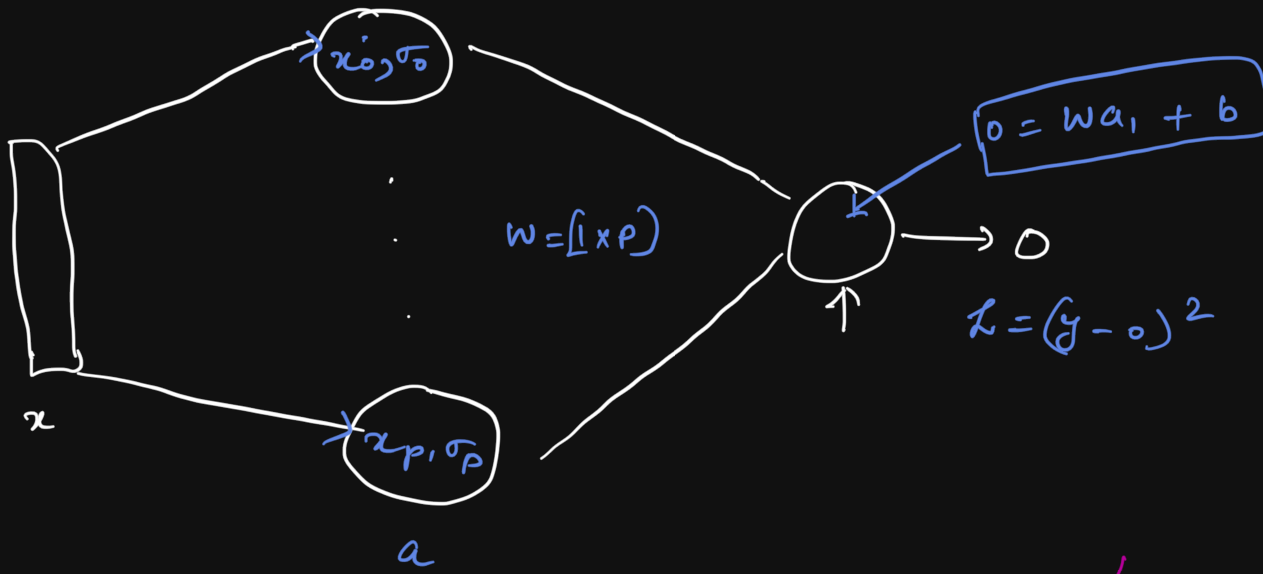
H.W2

Repeat the above process for the following two choices of ϕ

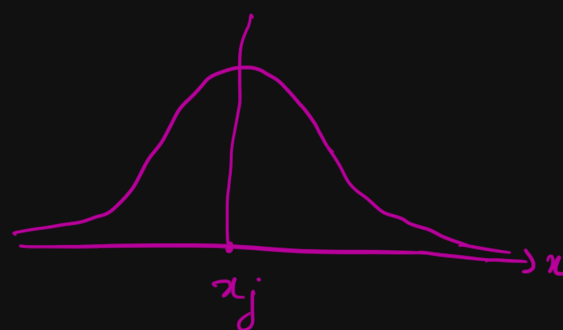
① $\phi(a) = \|a_j - a\|^2$

② $\phi(a) = \frac{1}{\|a_j - a\|^2} \quad \forall a_j \neq a$

RBF Network



$$a_j = \exp\left(-\frac{1}{2}\frac{\|x - x_j\|^2}{\sigma_j^2}\right)$$



Perfect Interpolation

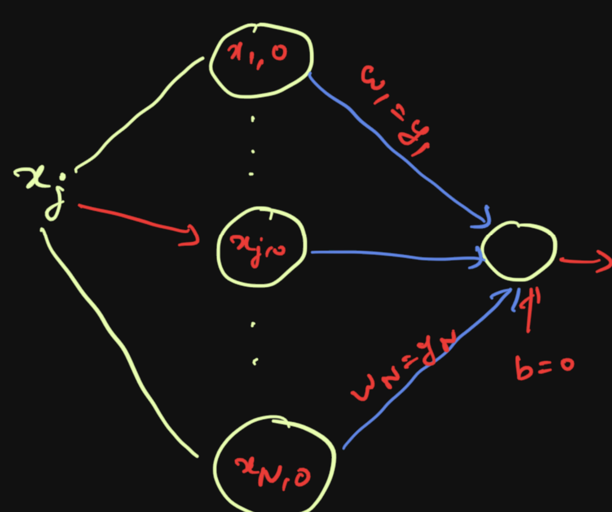
given $D = \{(x_i, y_i)\}_{i=1}^{N}$ $\exists$ RBF with Loss $= 0$

take $p = N$

$x_j = x_i$

$\sigma_j = 0$

and assume that $\frac{0}{0} = 0$



H/W → check the above N/W produces Loss $= 0$.

You have already seen a solution for

How RBF N/ws train for XOR data

Caveat

↳ ML is not about getting '0' error on D ☹

we aspire to get '0' error on unseen test data

Final comments

① If x_j and σ_j are fixed then

RBF's are Linear classifiers

② How do we fix x_j and σ_j

one naive approach

* fix $\sigma_j = 1$ (Blindly)

* Fix x_j = means of p means algorithm run on D

How to fix P?

This is a question where many practitioners suffer to answer

Heuristic - Kmeans Elbow technique (out of Scope)

Training RBF

Because it is only a Linear Network

$\frac{\partial L}{\partial w}$ and $\frac{\partial L}{\partial b}$ can be obtained at Ease ☺

Code walk Through!

Side Note:

For RBF (or any Linear classifiers)

$\exists$ many cute optimization algorithms that work elegantly in practice other than gradient descent (SVM, LMS algo)

↳ Out of Scope