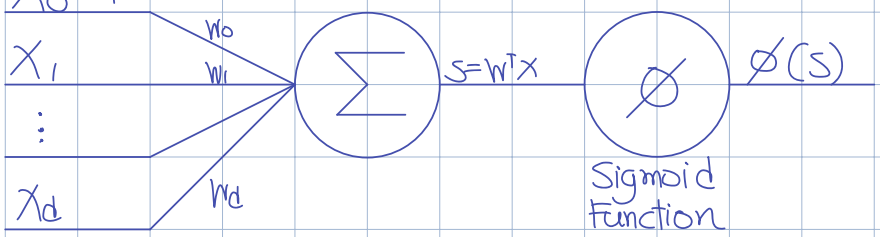
Introduction to Neural Network

Recap, Binary Logistic Regression

$\underline{x} = (x_0, x_1, \ldots, x_d)$

$x_0 = 1$

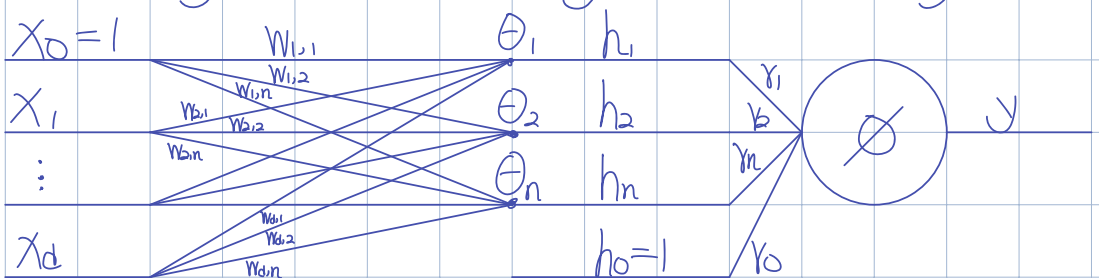


$$\phi(s) = \frac{e^s}{1+e^s}$$

2 Layer Neural Network

Non Linear Transformation

Input Layer | Hidden Layer | Output Layer



Let θ and ϕ be some non linear function

$w_{i,j}$ = weight on the edge connecting input node i and hidden unit node j

$W_{i,j}$ → Hidden Node

↓

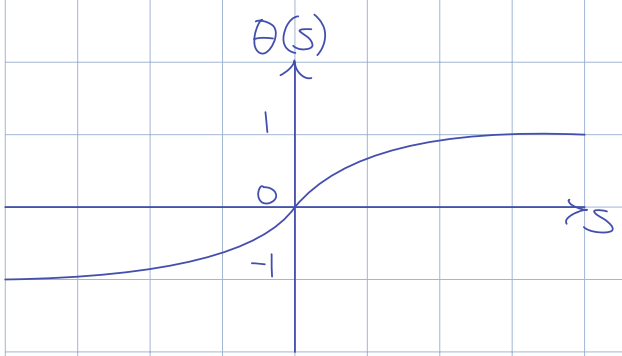
Input Node

$v_i \equiv$ weight on the edge between hidden node j and output node

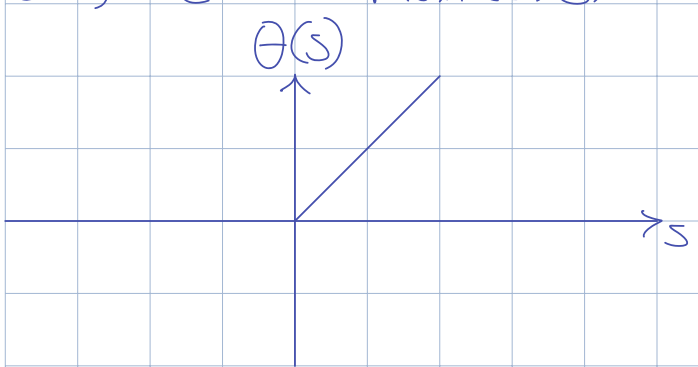
$h_j \equiv$ output of node j in the hidden layer

$$h_j = \theta\left(w_{0,j} + \sum_{i=1}^{d} w_{i,j} x_i\right)$$

$$\theta(s) = \tanh(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}$$



$\theta(s) = \text{Relu}(s) = \max(0, s)$ Commonly Used



$$y = \phi\left(v_0 + \sum_{j=1}^{n} v_j h_j\right)$$

For Classification, ϕ → Sigmoid

For Regression, ϕ → Identity Function

Summary

Given $\underline{x} \in \mathbb{R}^d$, use a 2 step process to compute output y

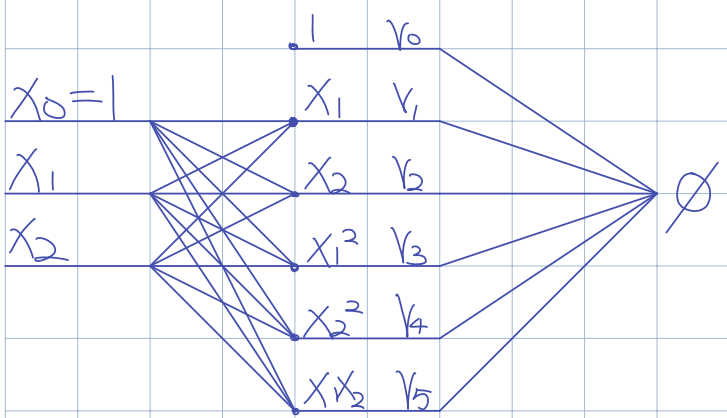
1. Apply a "non linear transform"

$$h_j = \theta(w_j^T x) \quad j = 1, 2, \ldots, n$$

→ Learning the weights (not fixed)

2. Apply Linear Model (eg. logistic regression) On $(h_1, h_2, \ldots, h_n)$

Recall, **Non Linear Transformations (eg 2nd Order)**



Note:

Unlike the case with non-linear transforms, the functions $h_i = \theta(w_i^T x_i)$ are learned during training. This makes neural networks much more powerful

Note:

We should not select $\theta(s) = s$

If $\theta(s) = s$

$$y = \phi\left(v_0 + \sum_{j=1}^{n} v_j h_j\right)$$

$$= \phi\left(v_0 + \sum_{j=1}^{n} v_j \theta(w_j^T x)\right)$$

$$= \phi\left(v_0 + \left(\sum_{j=1}^{n} v_j w_j^T\right) x\right)$$

$$\hat{w} = \sum_{j=1}^{n} v_j w_j^T$$

$$y = \phi(v_0 + \hat{w}^T x)$$ Just A Linear Model

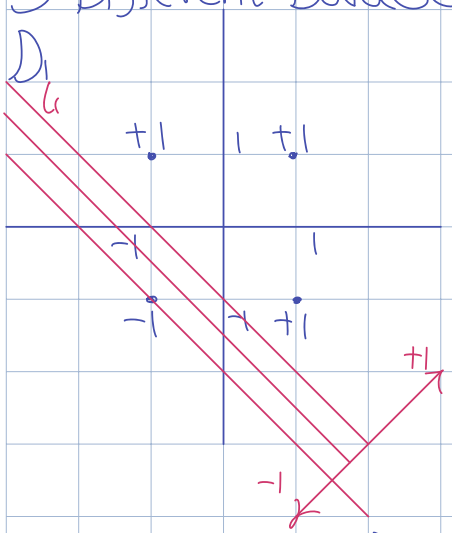
Example,

Training Set $N=4$ Points

Input Data Vector $\underline{h} = (h_1, h_2)$, $h_1, h_2 \in \{-1, +1\}$

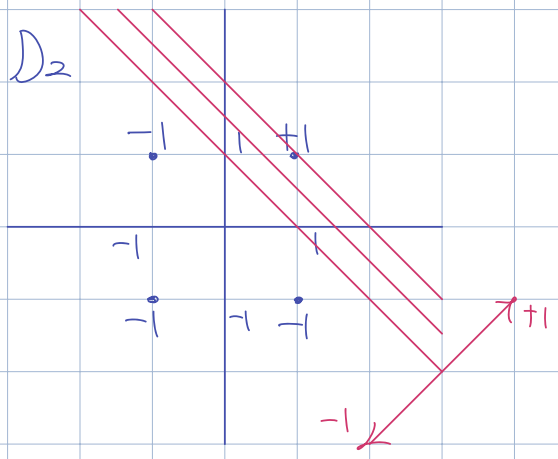
3 Different Data Sets

D_1



$y_1 = OR(h_1, h_2)$

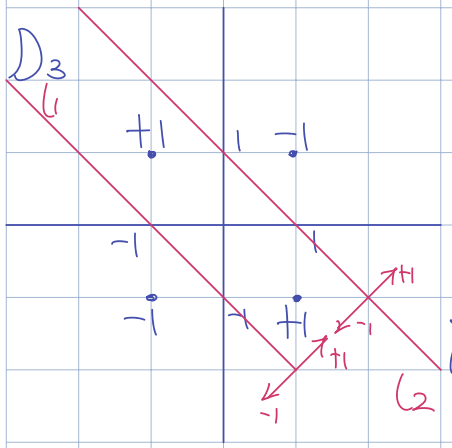
D_2



$y_2 = AND(h_1, h_2)$

D_1, D_2 Linearly Separable

D_3



$h_1 = h_2, \hat{y_3} = -1$

$h_1 \neq h_2, \hat{y_3} = +1$

$y_3 = XOR(h_1, h_2)$

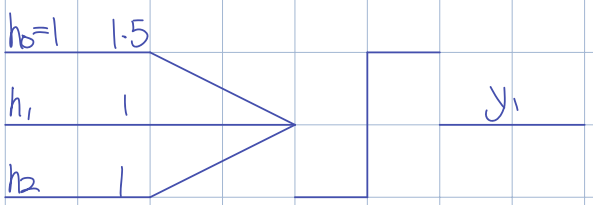
D_1

$\hat{y}_1 = \text{sign}(h_1 + h_2 + c)$, $c > 0$

$c < 0$

$0 < c < 2$

Fix $c = 1.5$



D_2

$\hat{y}_2 = \text{sign}(h_1 + h_2 + c)$, $-2 < c < 0$

$c = 1.5$

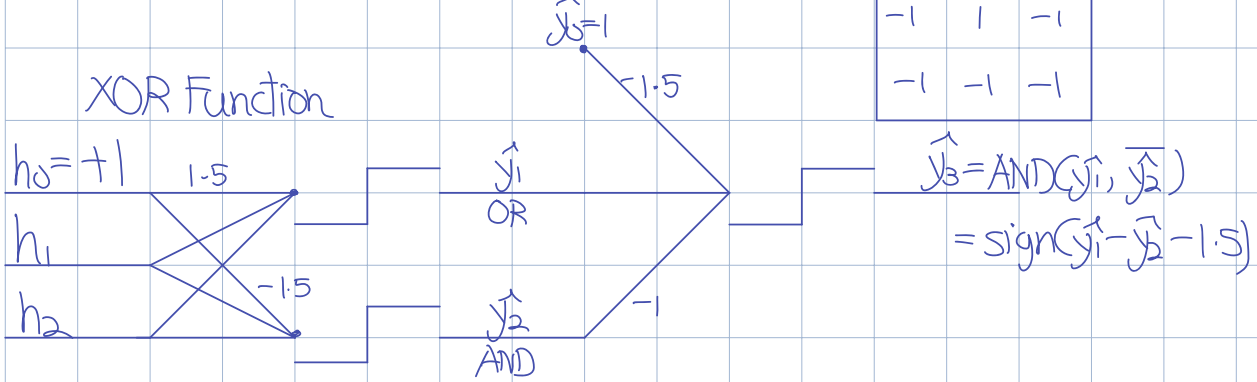
D_3

$\hat{y}_3 = +1 \longleftrightarrow \hat{y}_1 = +1 \;\&\; \hat{y}_2 = -1$

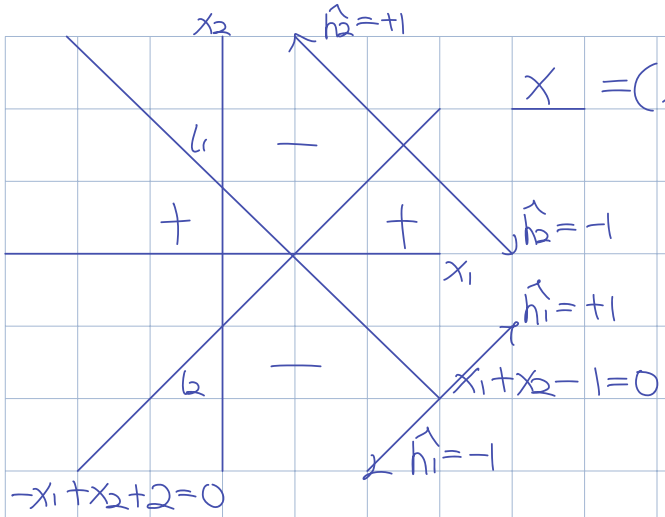
$\hat{y}_3 = \text{AND}(\hat{y}_1, \overline{\hat{y}_2})$

$\hat{y}_1$	$\hat{y}_2$	$\hat{y}_3$
1	1	-1
1	-1	1
-1	1	-1
-1	-1	-1

XOR Function



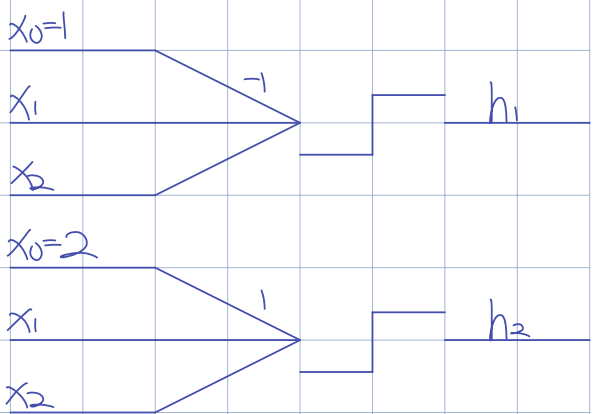
Two Layer Neural Network



$\underline{x} = (x_1, x_2)$

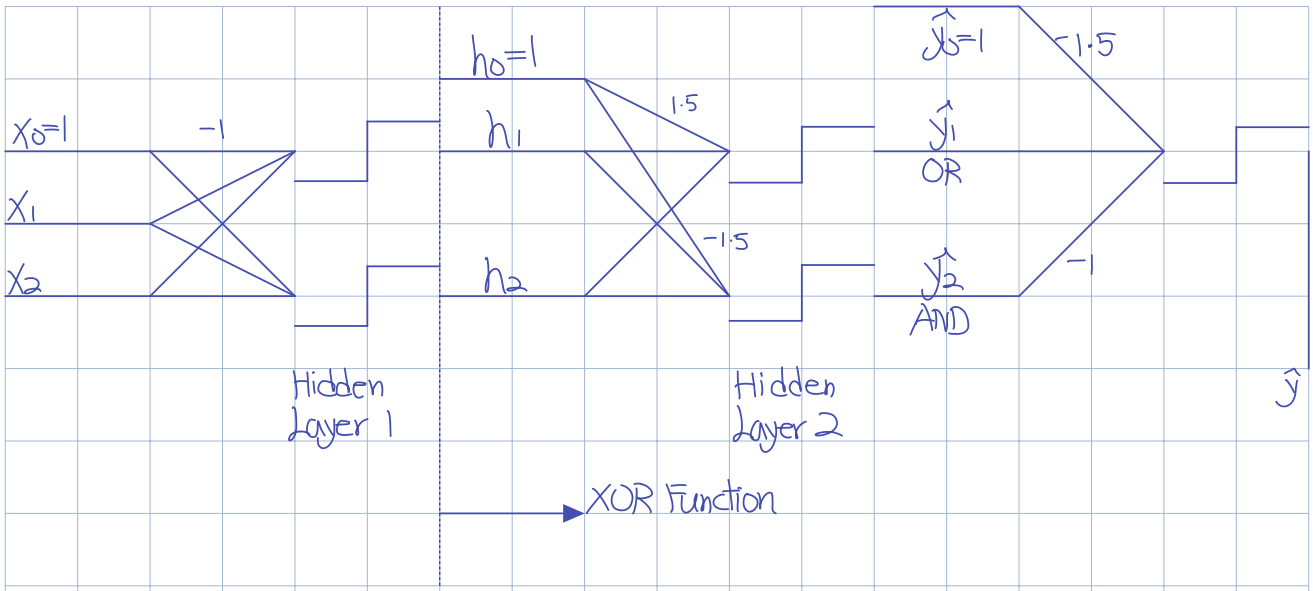
Define $h_1 = \text{sign}(x_1 + x_2 - 1)$

$h_2 = \text{sign}(-x_1 + x_2 + 2)$



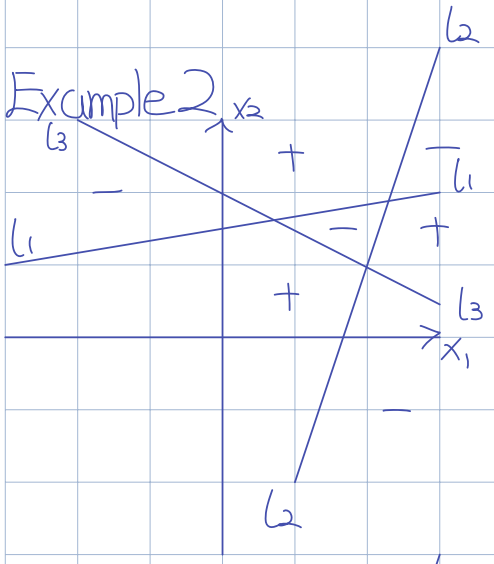
h_1	h_2	y
1	1	-1
1	-1	1
-1	1	1
-1	-1	-1

$y = \text{XOR}(h_1, h_2)$

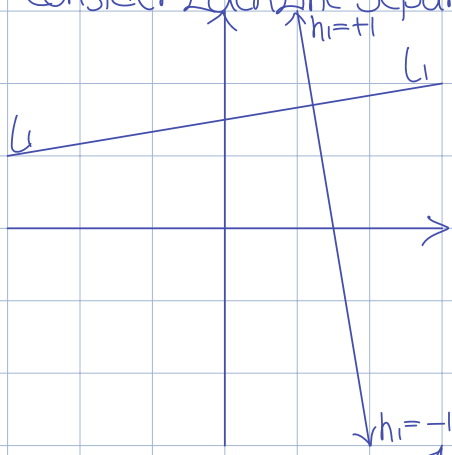


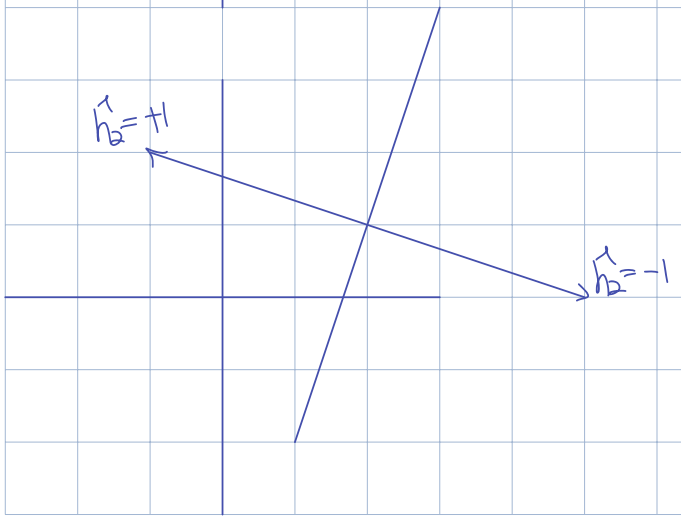
MLP (Multi Layer Perceptron)

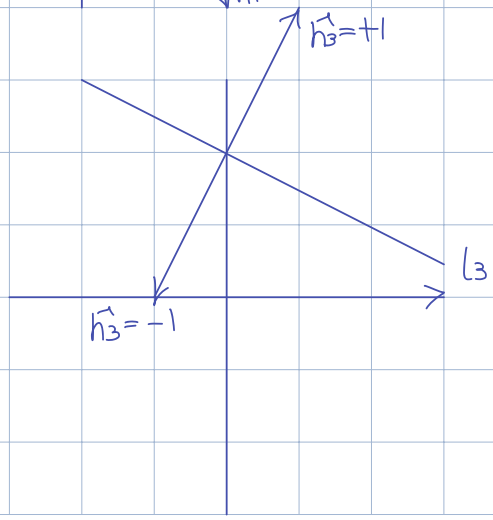
Example 2

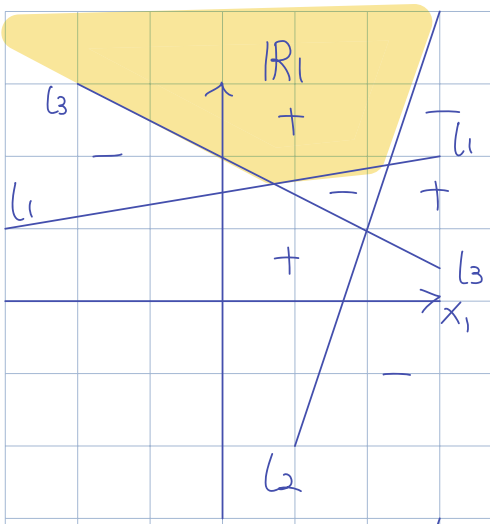


Consider Each Line Separately





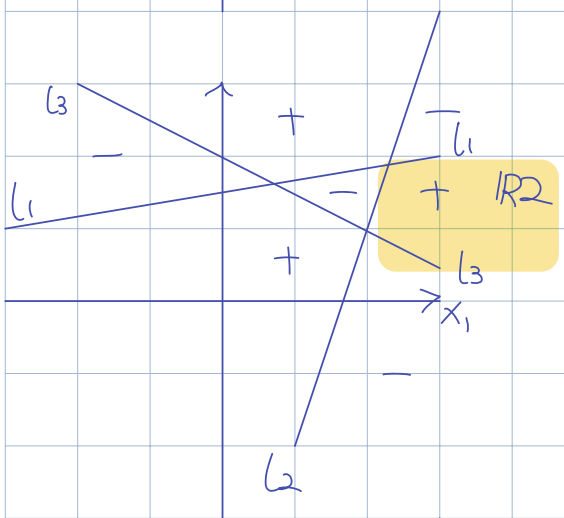




$y_1 = 1 \longleftrightarrow x \in IR_1$

$y_1 = 1 \longleftrightarrow h_1 = +1, h_2 = +1, h_3 = +1$

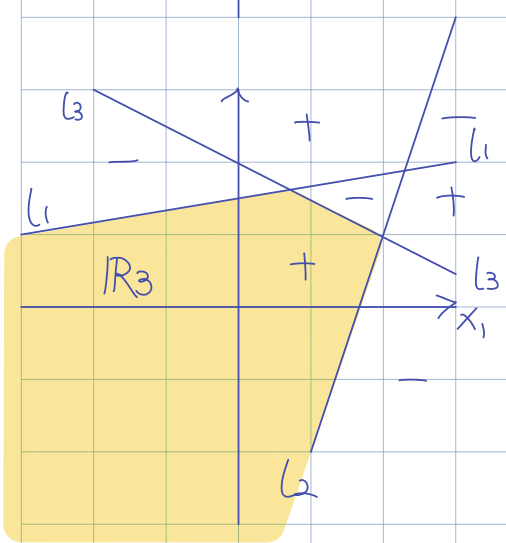
$y_1 = AND(h_1, h_2, h_3)$



$y_2 = 1 \longleftrightarrow x \in IR_2$

$h_1 = -1, h_2 = -1, h_3 = +1$

$y_2 = AND(\overline{h_1}, \overline{h_2}, h_3)$



$y_3 = 1 \longleftrightarrow x \in IR_3$

$h_1 = -1, h_2 = +1, h_3 = -1$

$y_3 = AND(\overline{h_1}, h_2, \overline{h_3})$

$y = OR(y_1, y_2, y_3)$

Summary 3 Layer MLP (NN)

Layer 1, Implement L_1, L_2, L_3 to output h_1, h_2, h_3

Layer 2, Implement y_1, y_2, y_3

$y_1 = AND(h_1, h_2, h_3)$

$y_2 = AND(\overline{h_1}, h_2, h_3)$

$y_3 = AND(\overline{h_1}, h_2, \overline{h_3})$

Layer 3, $y = OR(y_1, y_2, y_3)$