Some Interesting Interpretations (MLP)

(*) These are Feedforward Networks. If we include feedback, these become Recurrent Networks.

(*) To remove the restriction of linear models, we consider restricted non-linear models

One possible approach (Kernels): instead of $\underline{x}$, consider $\underline{\phi}(\underline{x})$

$\underline{\phi}(\cdot) \rightarrow$ possible non-linear mapping

Three possible methods

(1) General RBF kernel: SVM-based 'black box' methods use a generic RBF kernel

(2) Manually engineer $\underline{\phi}(x)$: difficult! does not generalise across domains e.g., speech & computer vision domains

(3) Deep learning: typically learn $\underline{\phi}(\underline{x})$

$$\underline{y}() = \underline{f}(\underline{x}) = \underline{f}(\underline{x}, \underline{W})$$

Earlier linear approach

$$\underline{w}^T \underline{x} \longrightarrow \underline{\tilde{w}}^T \underline{\phi}(\underline{x})$$

linear in $\underline{\phi}(\cdot)$

(homogeneous representation)

itself is possibly non-linear

$$\underline{\phi}(\underline{x}) = \underline{\phi}(\underline{x}, \underline{\theta})$$

parameters

We can learn $\underline{\phi}(\cdot)$ from a broad class of functions

This generalises the 1st & 2nd methods. How?

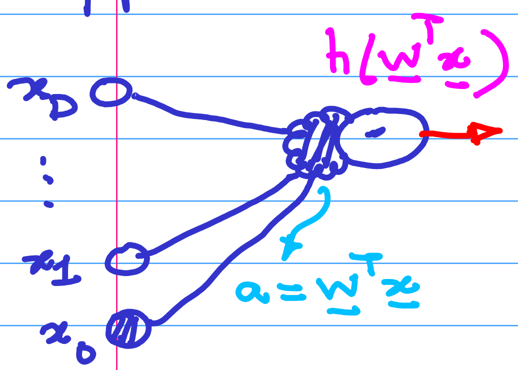
1st $\rightarrow$ generic, very broad family

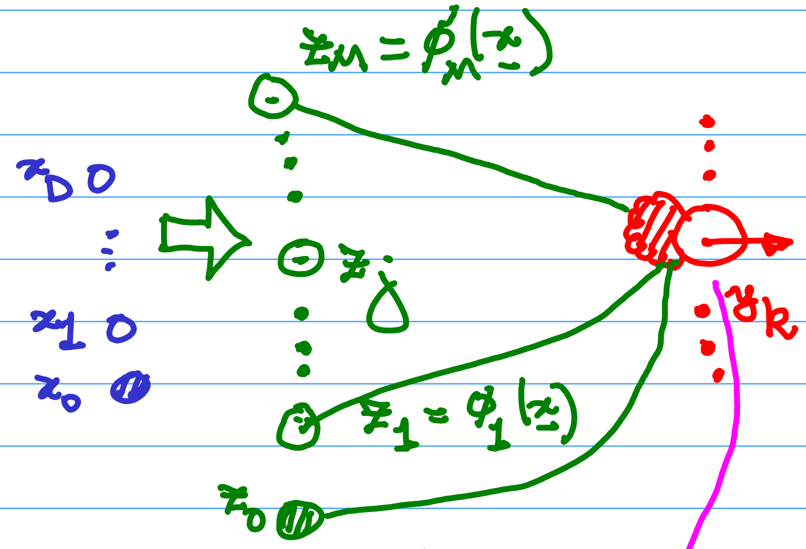
2nd $\rightarrow$ human knowledge not to find the exact right function $\longrightarrow$

but to find the right family of functions for $\underline{\phi}(\underline{x})$

Main Idea " $\underline{\phi}(\underline{x})$ defines a hidden layer "

Instead of the earlier approach





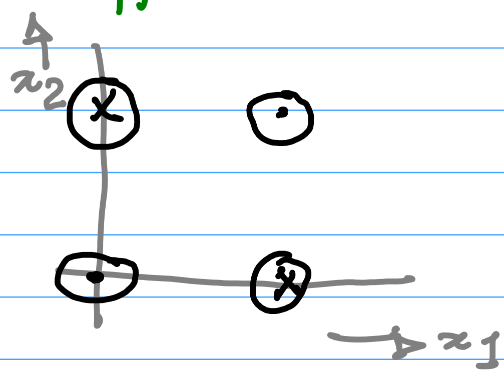
We transform the inputs (possibly in a non-linear manner) to get the hidden layer, from where we take a linear combination of $\underline{\phi}(\underline{x})$ in the output layer

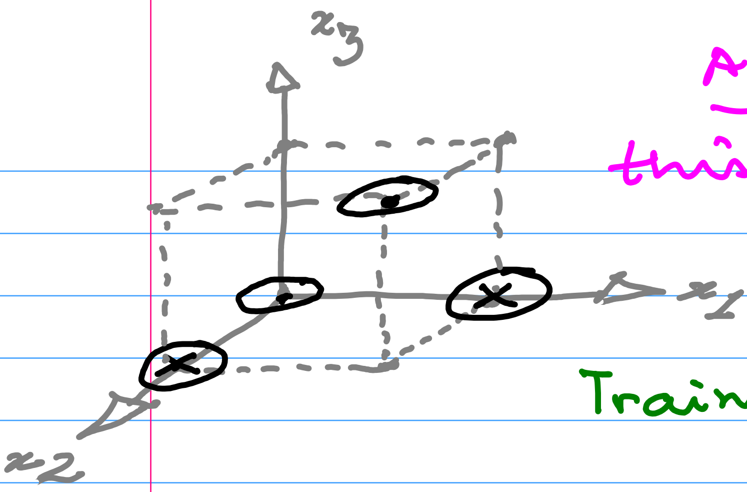
$$a_k^{(2)} = \underline{w}^{(2)^T}\underline{\phi}(\underline{x})$$

$$y_k = \sigma(a_k^{(2)})$$

The X-OR problem (again!) What was our previous approach to solve it? Handcrafted kernel / feature transformation approach

x_2	x_1	new feature $x_3 = x_2 x_1$	$y = x_2 \oplus x_1$
0	0	0	0 $\rightarrow$ o
0	1	0	1 $\rightarrow$ ×
1	0	0	1 $\rightarrow$ ×
1	1	1	0 $\rightarrow$ o





Approach here: To formulate this as a regression problem and use mean square error loss.

Training set $\underline{x} = \begin{bmatrix} x_2 \\ x_1 \end{bmatrix}$

$$X = \left[\underline{x}_{(1)} \; \underline{x}_{(2)} \; \underline{x}_{(3)} \; \underline{x}_{(4)} \right] = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

$$\boxed{\underline{t}} = \left[t_{(1)} \; t_{(2)} \; t_{(3)} \; t_{(4)} \right] = \left[0 \; 1 \; 1 \; 0 \right]$$

$$\text{MSE loss } J(\underline{w}) \triangleq \frac{1}{4} \sum_{n=1}^{4} \left[y(\underline{x}_{(n)}, \underline{w}) - t_{(n)} \right]^2$$

$\underline{w}$ → parameters

linear model

$$y(\underline{x}) = y(\underline{x}, \underline{w}) = \underbrace{\underline{w}^T \underline{x}}_{\text{homogeneous}} = \underbrace{\underline{w}^T \underline{x} + b}_{\text{non-homogeneous.}}$$

b or w_0