University of Amsterdam
Informatics Institute

AMLAB Amsterdam Machine Learning Lab

Machine Learning 1

Lecture 8.1 - Supervised Learning
Neural Networks

Erik Bekkers

(Bishop 5.1)

Slide credits: Patrick Forré and Rianne van den Berg

Fixed Basis Functions

Dataset: inputs $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N)^T$ and targets $\mathbf{t} = (t_1, ..., t_N)^T$

$\underline{x}_n \in \mathbb{R}^D$

Previously:

- Fixed features: $\boldsymbol{\phi}(\mathbf{x}) = (\phi_0(\mathbf{x}), ..., \phi_M(\mathbf{x}))^T$, $\phi_0(\mathbf{x}) = 1$
- Linear regression: $y(\mathbf{x}, \mathbf{w}) =$ $\underline{w}^T \underline{\phi}(\underline{x})$ $t_n \in \mathbb{R}$
- Classification: $y(\mathbf{x}, \mathbf{w}) = f(\underline{\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})})$ $t_n \in \{0, 1\}$

  f : nonlinear activation function

  e.g. (logistic sigmoid)

Neural Networks: Adaptive Basis Functions

Dataset: inputs $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N)^T$ and targets $\mathbf{t} = (t_1, ..., t_N)^T$

$\underline{x}_n = (1, x_{n1}, \ldots, x_{nD})^T \in \mathbb{R}^{D+1}$

Neural networks:

- Create flexible non-linear features and learn them!

$$\phi_m(\mathbf{x}, \mathbf{w}_m^{(1)}) = h(\underbrace{(\mathbf{w}_m^{(1)})^T\mathbf{x}}_{\text{linear}}) = h(\sum_{d=0}^{D} w_{md}^{(1)}\, x_d)$$

non-linear activation fn

$$W^{(1)} = \begin{pmatrix} | & | & & | \\ \underline{w}_1^{(1)} & \underline{w}_2^{(1)} & \cdots & \underline{w}_M^{(1)} \\ | & | & & | \end{pmatrix}$$

2-layer NN

- Regression:

$$y(\mathbf{x}, \mathbf{W}^{(1)}, \mathbf{w}^{(2)}) = \sum_{m=0}^{M} w_m^{(2)}\, \underbrace{h(\sum_{d=0}^{D} w_{md}^{(1)}\, x_d)}_{\phi(\underline{x})} = \underline{w}^{(2)T} \underbrace{h(W^{(1)T}\underline{x})}_{\underline{\phi}(\underline{x})}$$

- Classification:

σ, softmax

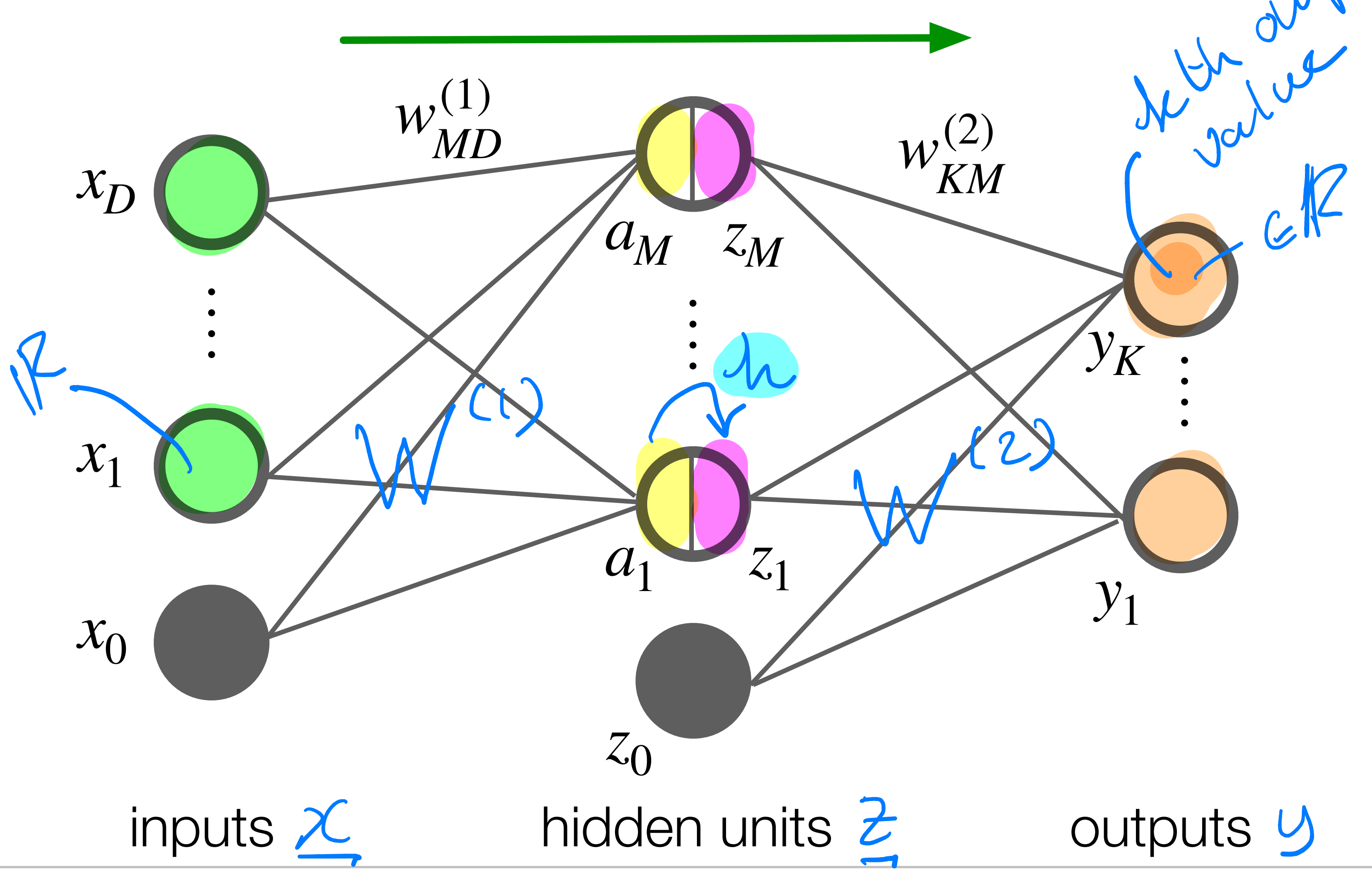
$$y(\mathbf{x}, \mathbf{W}^{(1)}, \mathbf{w}^{(2)}) = f(\underline{w}^{(2)T} h(W^{(1)T}\underline{x}))$$

Multilayer Perceptron (MLP): 2 layers

Model:

$$y_k(\mathbf{x}, \mathbf{W}^{(1)}, \mathbf{W}^{(2)}) = h^{(2)}\left(\sum_{m=0}^{M} w^{(2)}_{km}\, h^{(1)}\left(\underbrace{\sum_{d=0}^{D} w^{(1)}_{md}\, x_d}_{a_m}\right)\right)$$

Network diagram:



Input units x_d

$$\underline{x} = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_D \end{pmatrix} \in \mathbb{R}^{D+1}$$

Activations a_m

$$\underline{a} = \begin{pmatrix} a_1 \\ \vdots \\ a_M \end{pmatrix} = W^{(1)} \underline{x} \in \mathbb{R}^M$$

Hidden units z_m

$$z_m = h(a_m)$$

Output units y_k

$$\underline{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_K \end{pmatrix} = W^{(2)} \underline{z} \in \mathbb{R}^K$$

Activation functions

$$h^{(1)}, h^{(2)}$$

Activation functions

We need activation functions! Otherwise the NN is just a linear model...

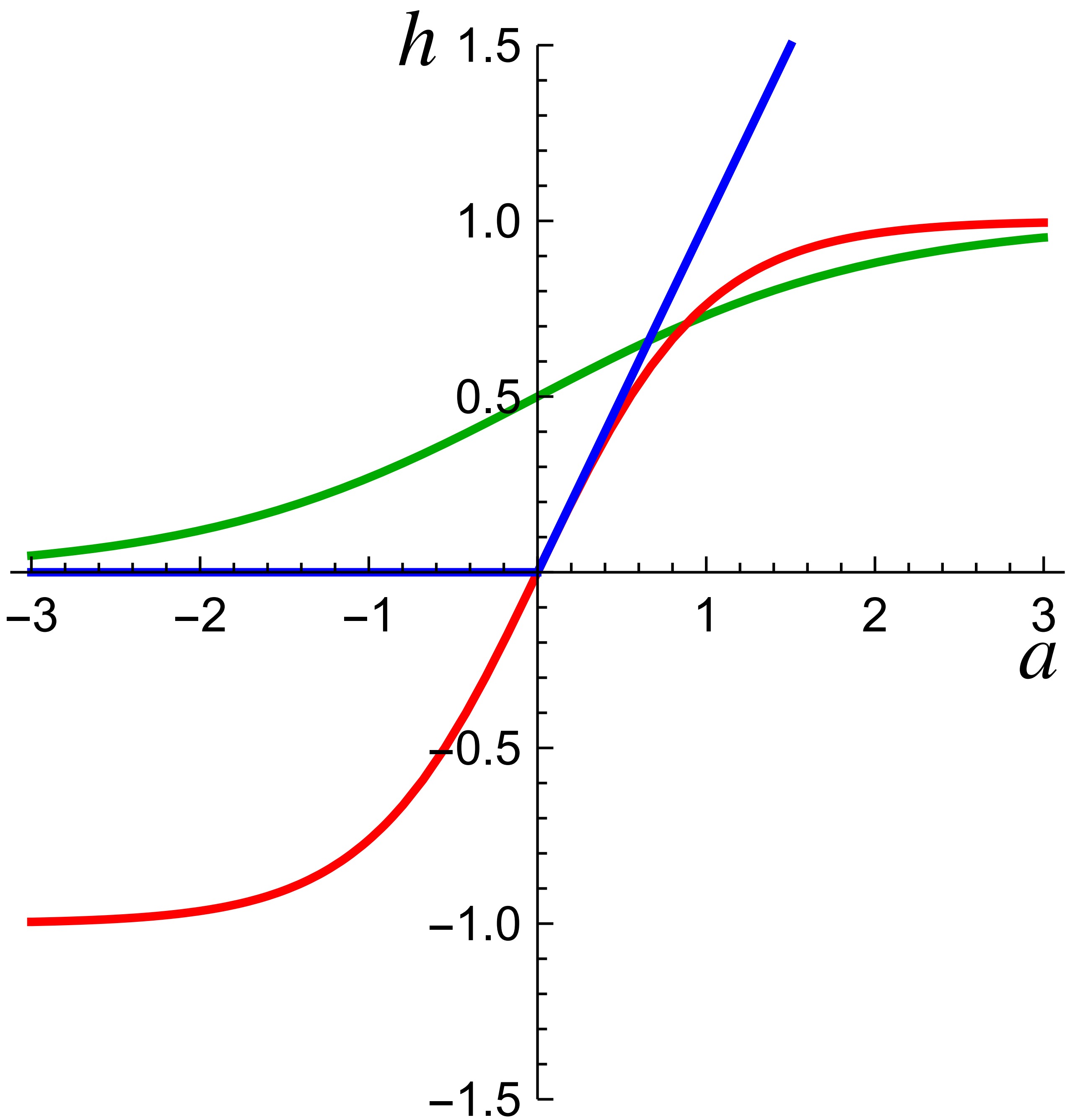


Figure: Popular activation functions.

Green: Logistic sigmoid

$$h(a) = \sigma(a) = \frac{1}{1+e^{-a}}$$

Red: Hyperbolic tan

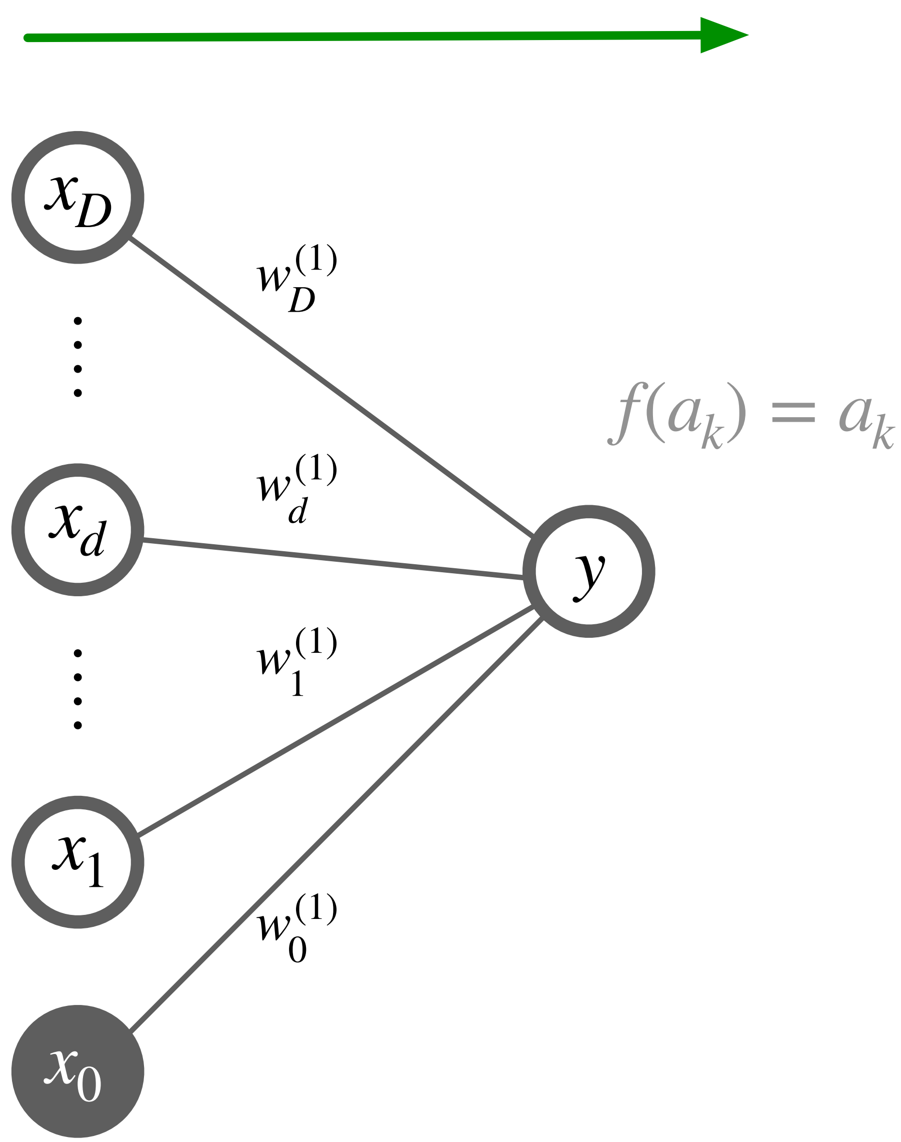
$$h(a) = \tanh(a) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Blue:

$$h(a) = \mathrm{ReLU}(a) = \max(0, a)$$

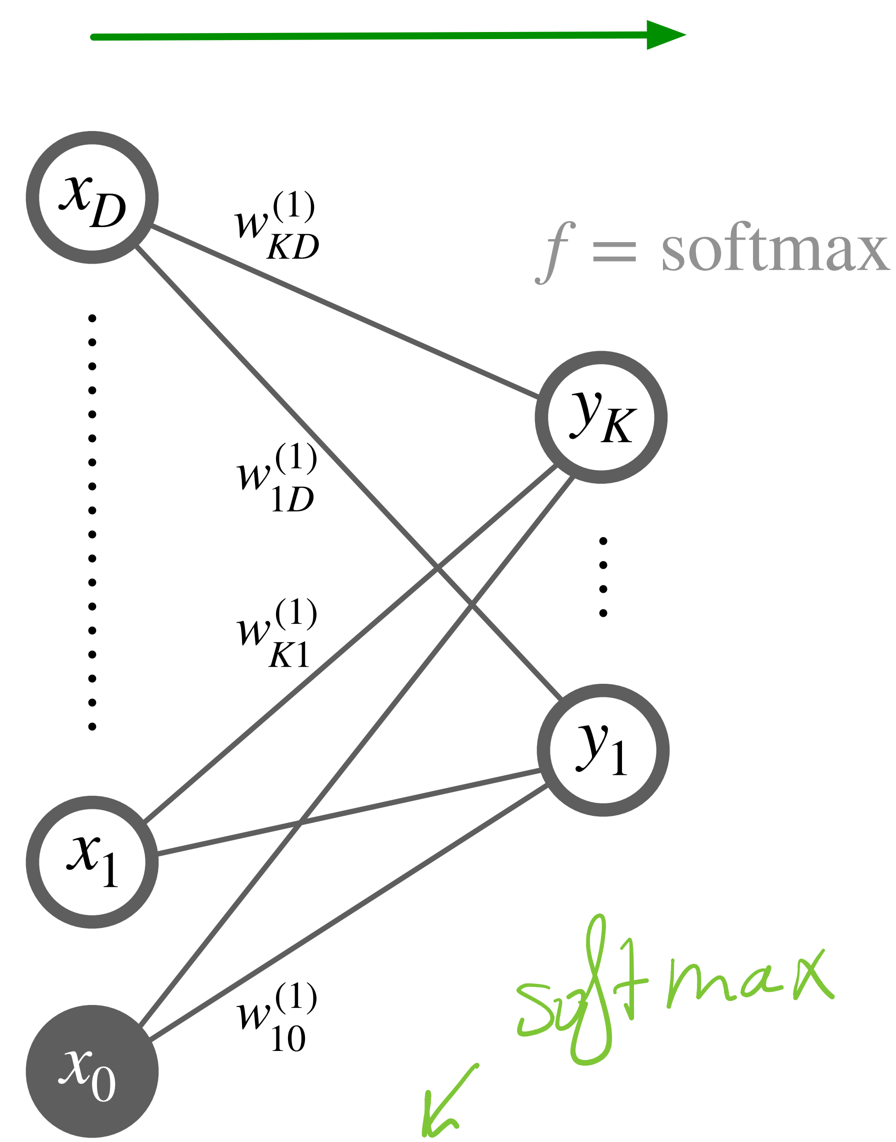
(Rectified Linear Unit)

Linear regression & classification as NN



$y(\underline{x}, \underline{w}) = \underline{w}^T x$

Figure: Linear regression as 1-layer NN

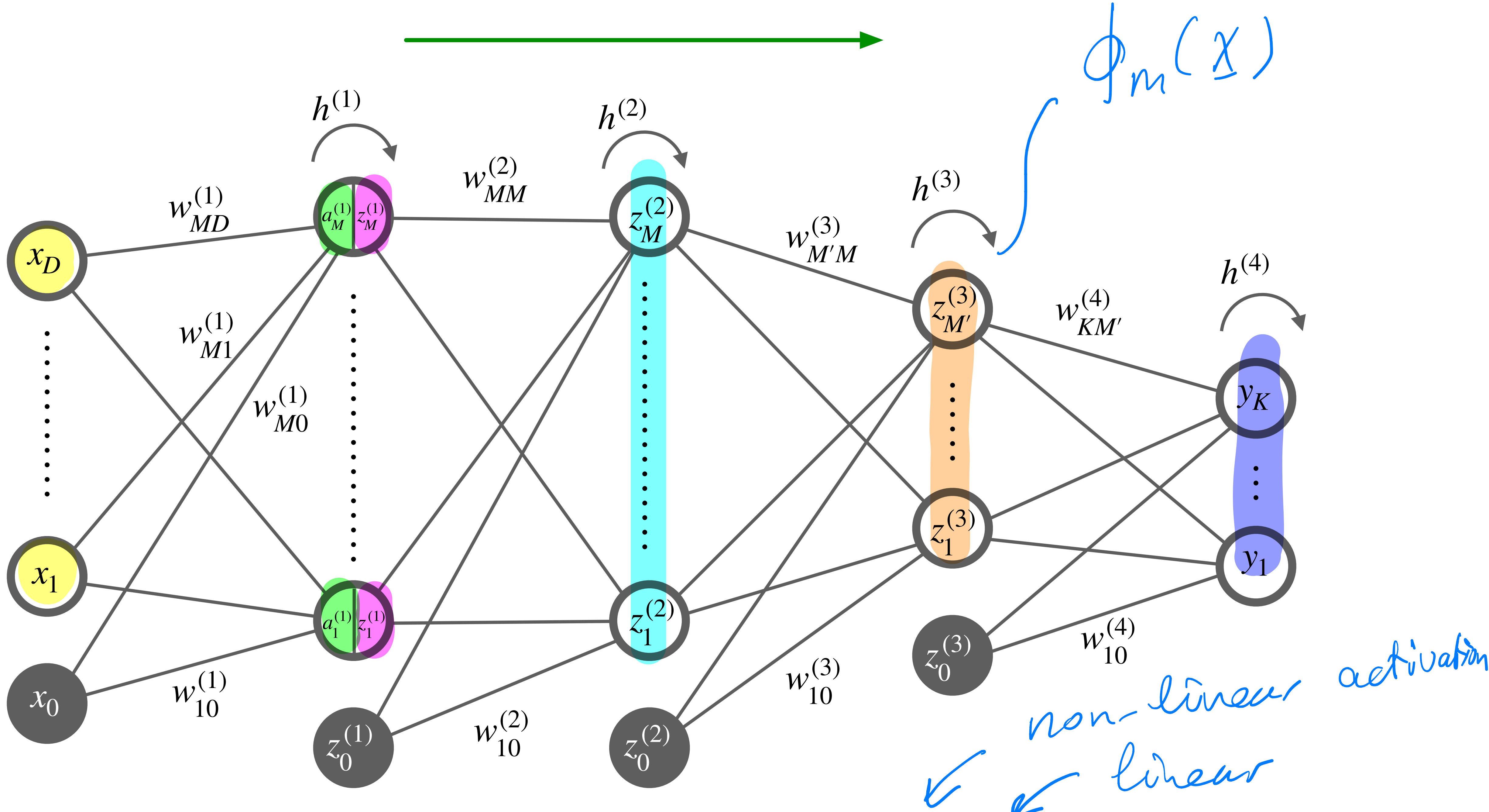


softmax

$y_k(\underline{x}, \underline{w}) = f(\underline{w}_k^T \underline{x}) = p(C_k, \underline{x})$

Figure: Linear Classification with K classes as 1-layer NN

Feed-Forward Networks: Multiple layers



$$y_k(\mathbf{x}, \mathbf{w}) = h^{(4)}(\mathbf{a}^{(4)}(h^{(3)}(\mathbf{a}^{(3)}(h^{(2)}(\mathbf{a}^{(2)}(h^{(1)}(\mathbf{a}^{(1)}(\mathbf{x}))))))))$$
$$= h^{(4)} \circ \mathbf{a}^{(4)} \circ h^{(3)} \circ \mathbf{a}^{(3)} \circ h^{(2)} \circ \mathbf{a}^{(2)} \circ h^{(1)} \circ \mathbf{a}^{(1)}(\mathbf{x})$$

Figure: 4 layer network. Number of layers = number of layers of adaptive weights.

Feed-Forward Networks: Skip Connections

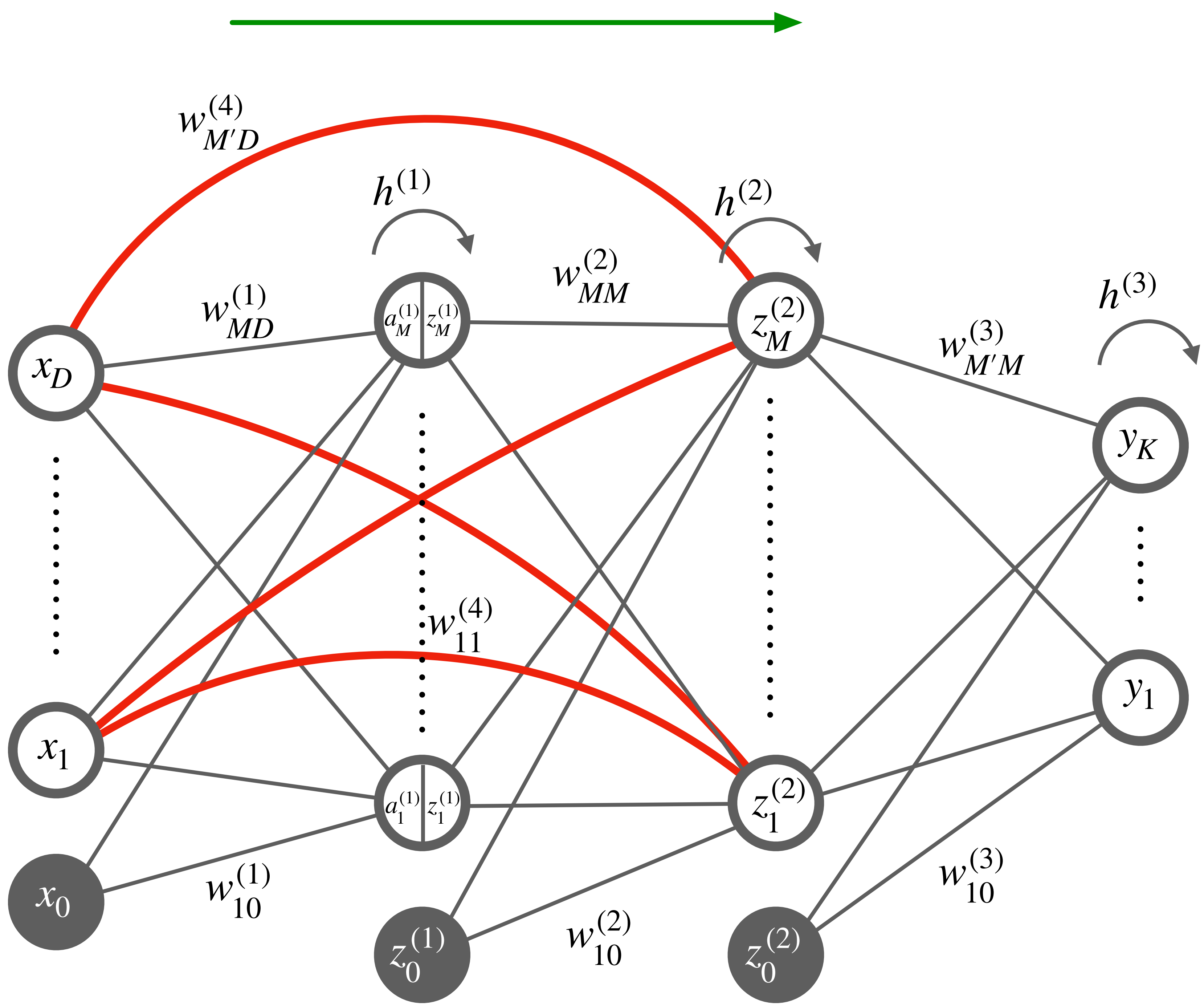


Figure: 3 layer feed-forward net with skip connections

Feed-Forward Networks: Sparse Connections

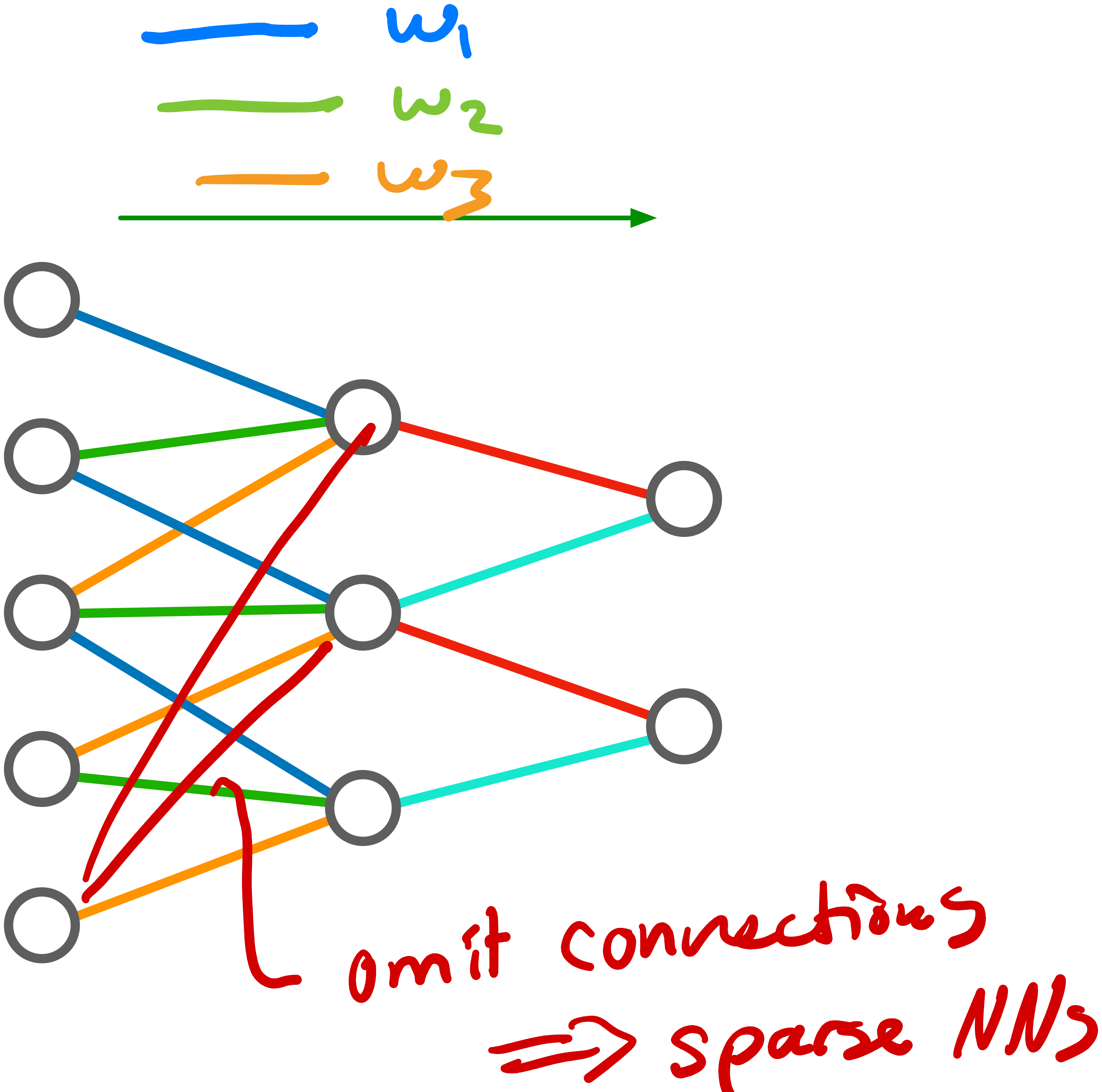
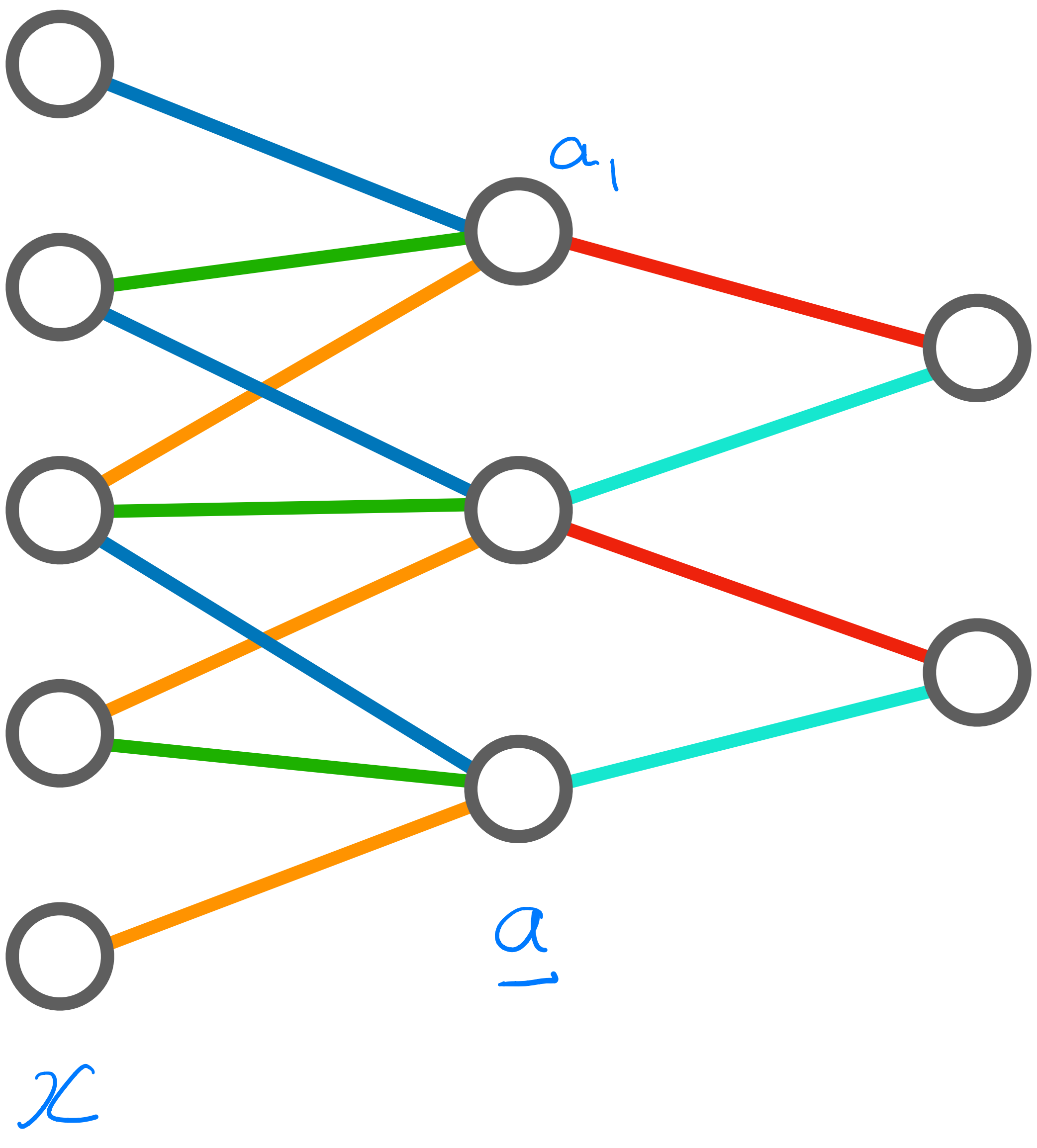


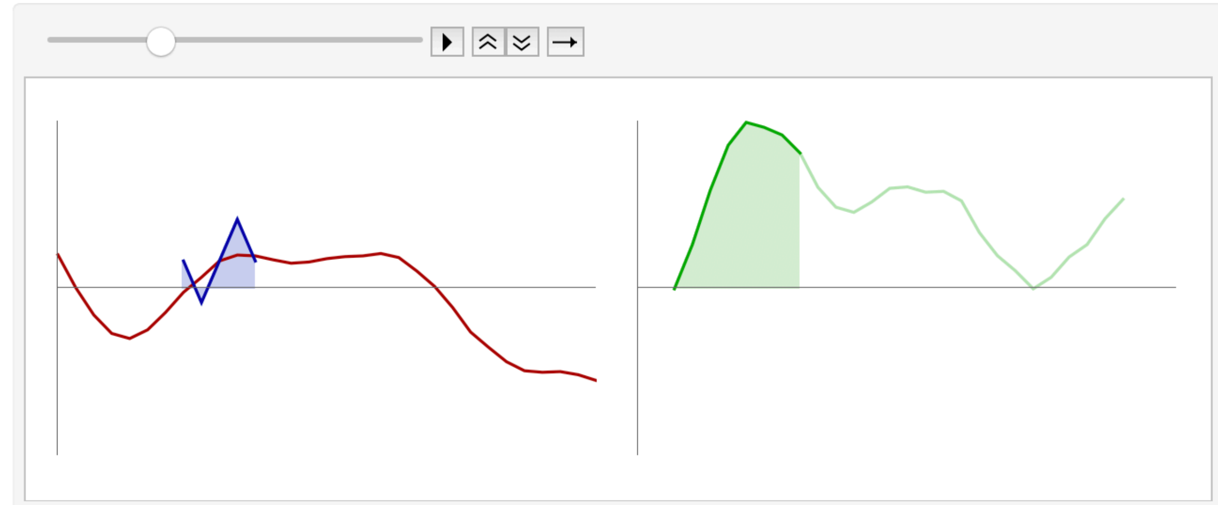
Figure: Feed-forward architecture with sparse connections. With special weight sharing --> Convolutional Neural Nets (Le Cun et al 1989)

Feed-Forward Networks: Sparse Connections

$\underline{a} = \underline{x} \star \underline{w}$

$a_j = \sum_{|i-j|<K} x_i w_{i-j}$





$\mathbf{x} \star \mathbf{w} \quad = \quad \mathbf{a}$

Example: 1D convolution = sparse + weightsharing:

$$\begin{pmatrix} a_1 \\ a_2 \\ a_2 \\ \vdots \\ a_M \end{pmatrix} = \cancel{\begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} & \cdots \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} & \cdots \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} & \cdots \\ w_{41} & w_{42} & w_{43} & w_{44} & w_{45} & \cdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \end{pmatrix}} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_D \end{pmatrix}$$

sparse + weight sharing

$$= \begin{pmatrix} w_1 & w_2 & w_3 & 0 & 0 & \cdots \\ 0 & w_1 & w_2 & w_3 & 0 & \cdots \\ 0 & 0 & w_1 & w_2 & w_3 & \cdots \\ 0 & 0 & 0 & w_1 & w_2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_D \end{pmatrix}$$

Figure: Feed-forward architecture with sparse connections. With special weight sharing --> Convolutional Neural Nets (Le Cun et al 1989)

General Feed-Forward Architectures

✦ Each unit (hidden & output) in feed-forward architectures computes a function of the form

$$z_m = h\left(\sum_j w_{mj} z_j\right)$$

any hidden unit from lower layer

✦ No closed directed cycles!

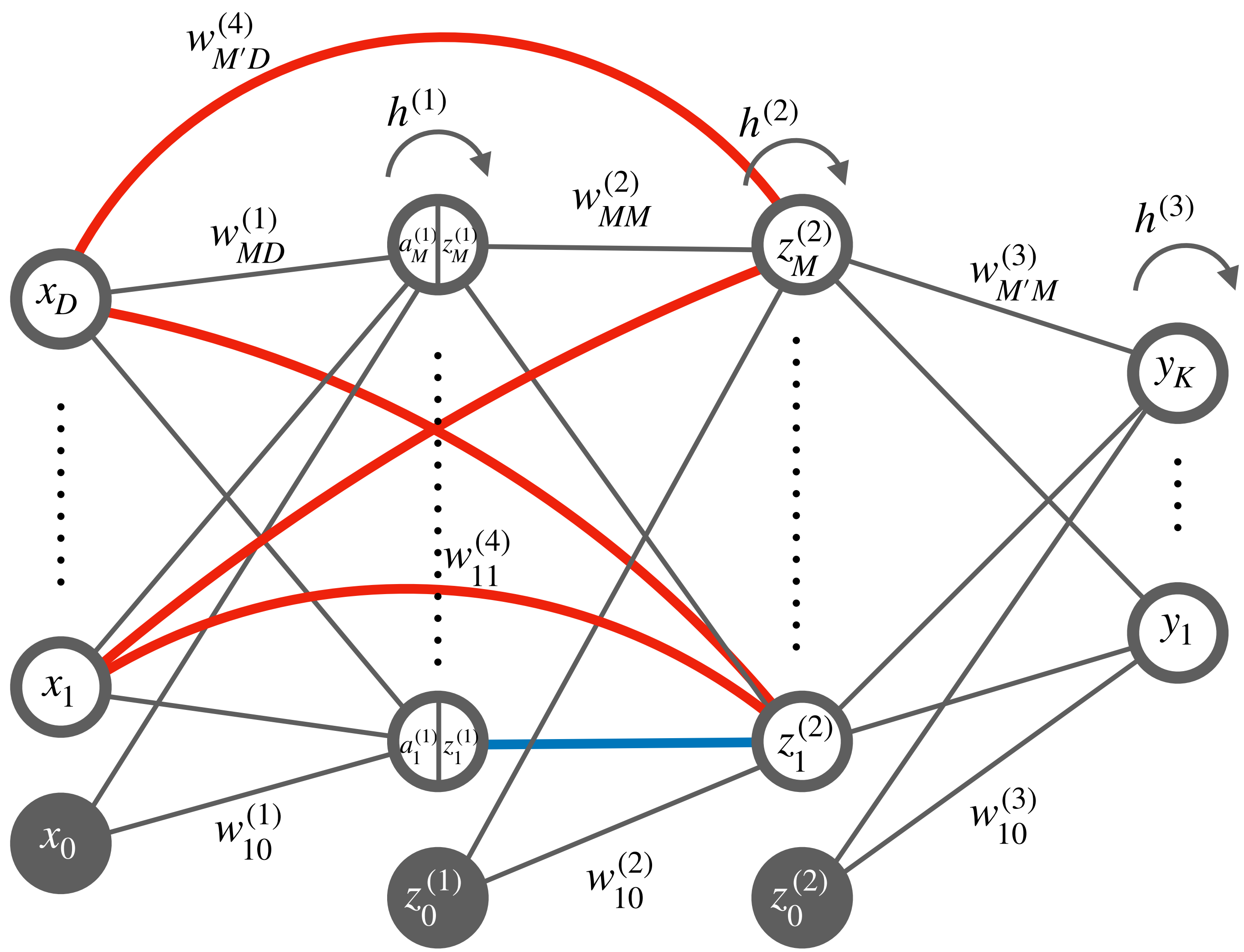


Figure: example of general feed-forward architecture