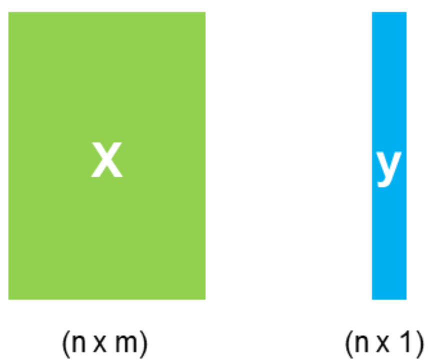
DAT300 – Applied Deep Learning

Math from ANN

A multi-layer neural network architecture

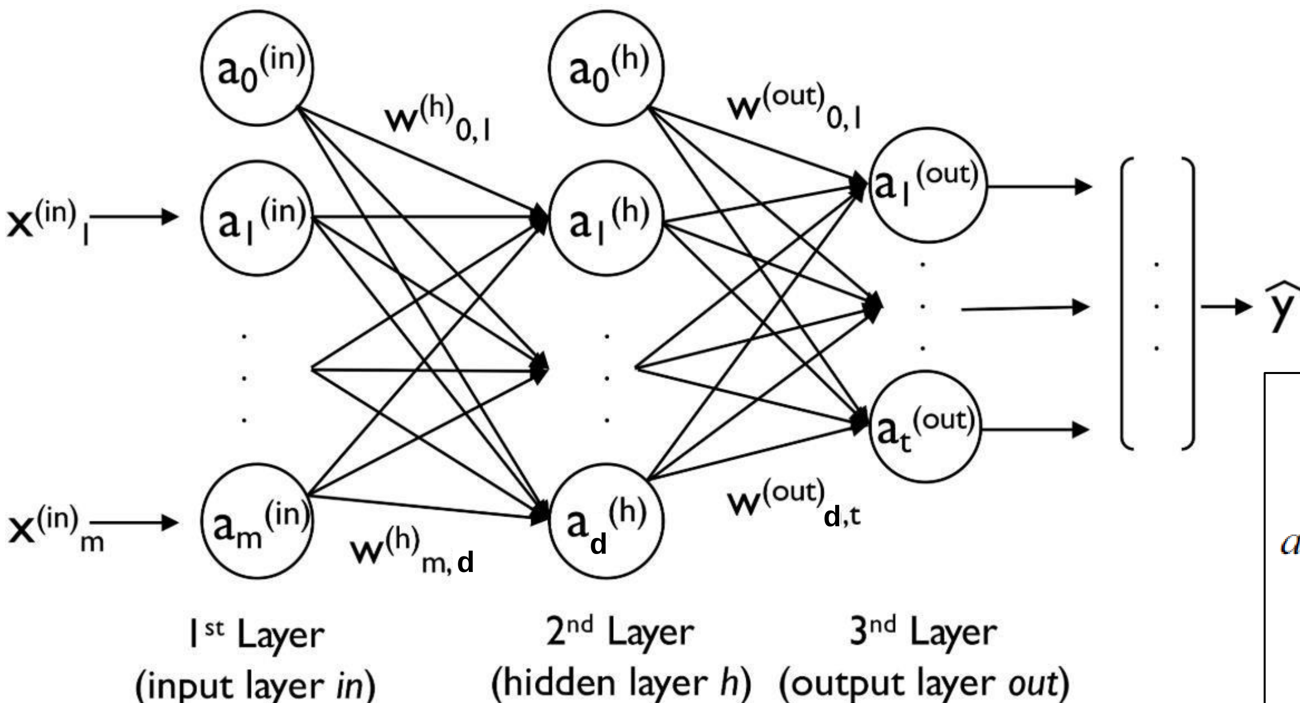
- Data arrays



- Indexing

$i = 1, \ldots, n$	sample / instance index
$j = 0, \ldots, m$	feature / instance index

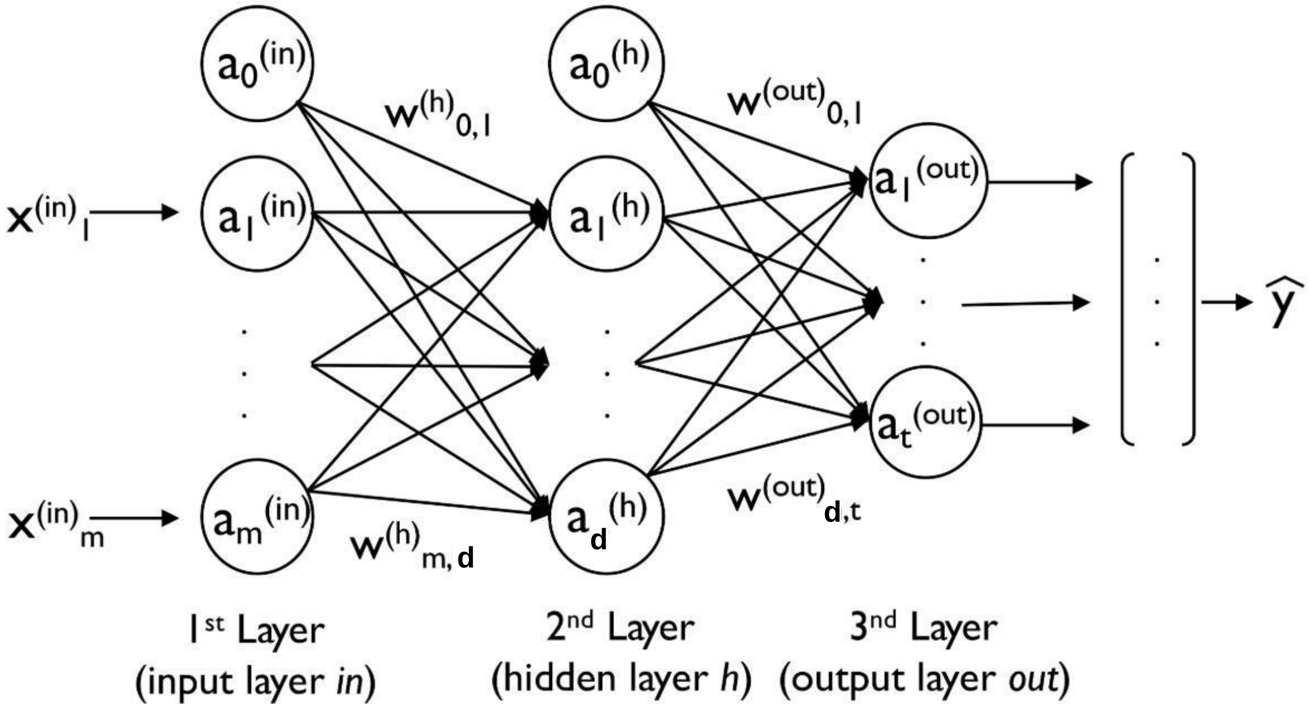
A multi-layer neural network architecture



$$a^{(in)} = \begin{bmatrix} a_0^{(in)} \\ a_1^{(in)} \\ \vdots \\ a_m^{(in)} \end{bmatrix} = \begin{bmatrix} 1 \\ x_1^{(in)} \\ \vdots \\ x_m^{(in)} \end{bmatrix}$$

(m x 1) + bias   (m x 1) + bias

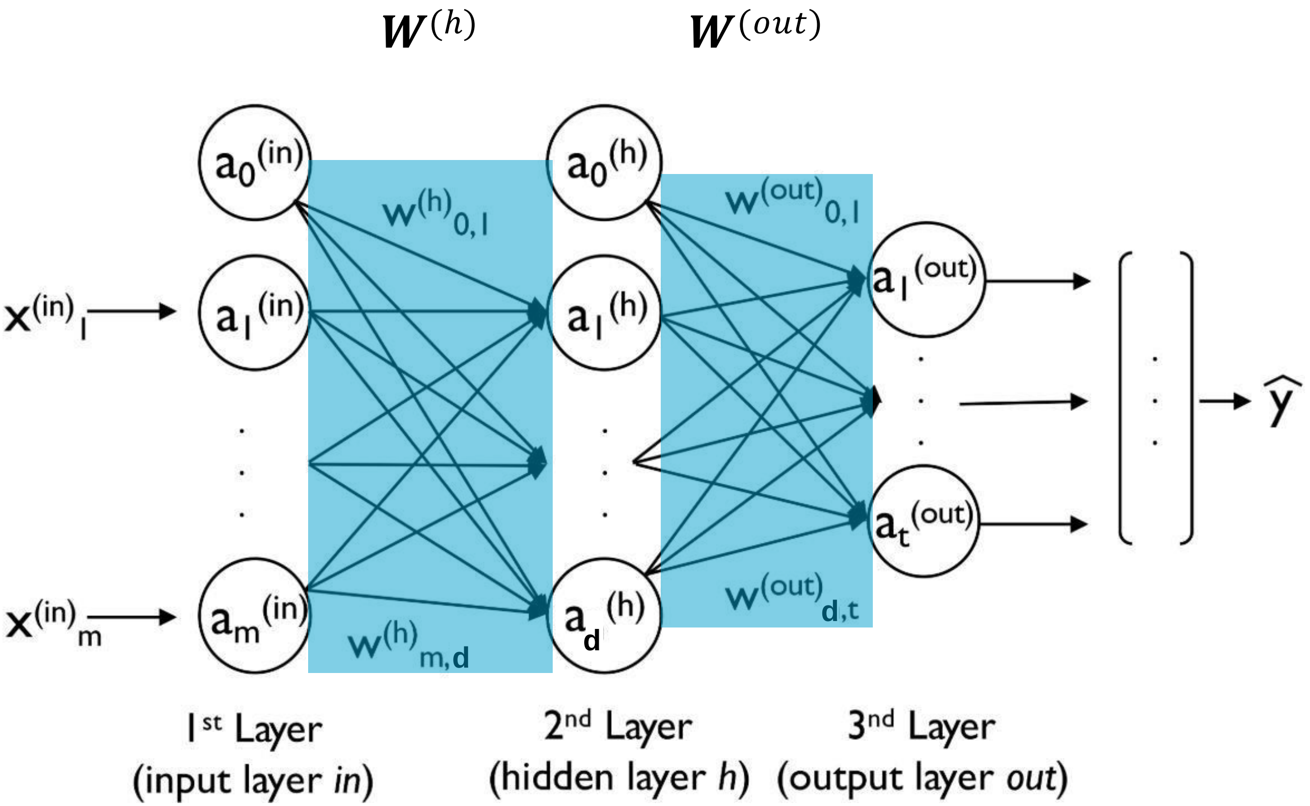
A multi-layer neural network architecture

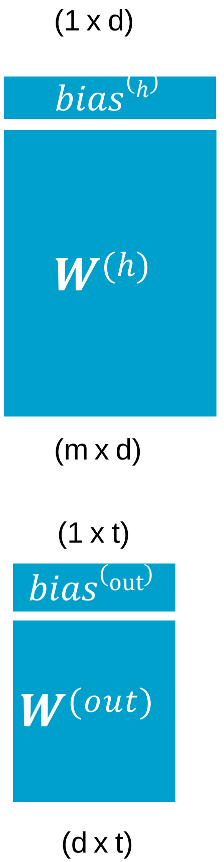


Example of model for three classes with t=3

$$0 = \begin{bmatrix}1\\0\\0\end{bmatrix}, 1 = \begin{bmatrix}0\\1\\0\end{bmatrix}, 2 = \begin{bmatrix}0\\0\\1\end{bmatrix}$$

A multi-layer neural network architecture



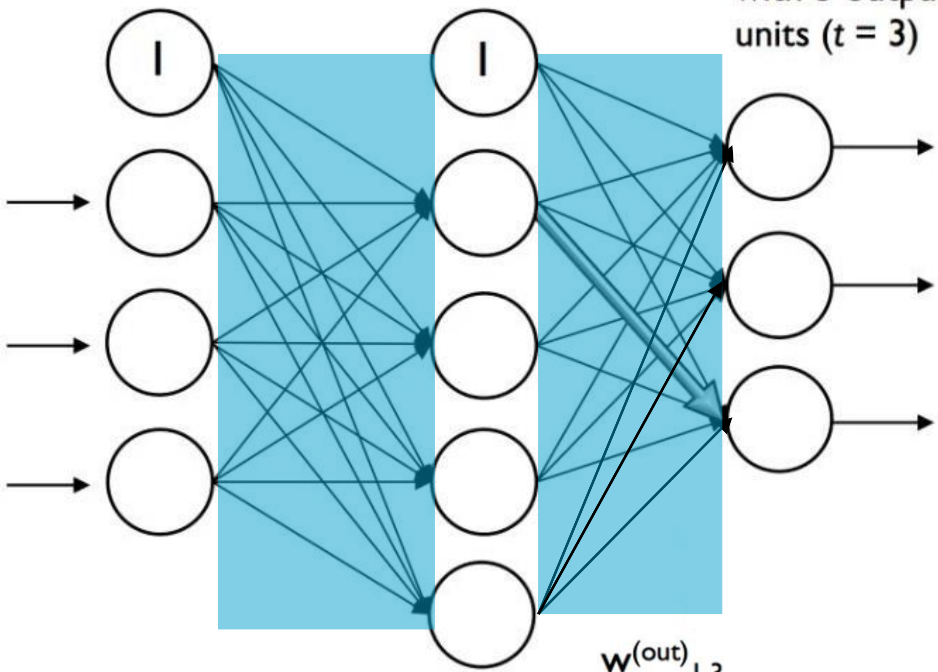


A 3-4-3 multi-layer perceptron

Input layer with 3 input units plus bias unit (m = 3+1)

Hidden layer with 4 hidden units plus bias unit (d = 4+1)

Output layer with 3 output units (t = 3)



Number of layers: L = 3

$w^{(out)}_{1,3}$
connects 1st non-bias neuron in the 2nd layer (hidden layer h) to the 3rd unit in the 3rd layer (output layer out)

(1 x 4)

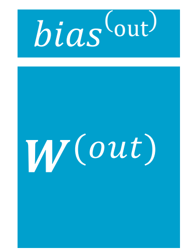
4 weights

12 weights

(3 x 4)

Total:
16 weights

(1 x 3)



3 weights

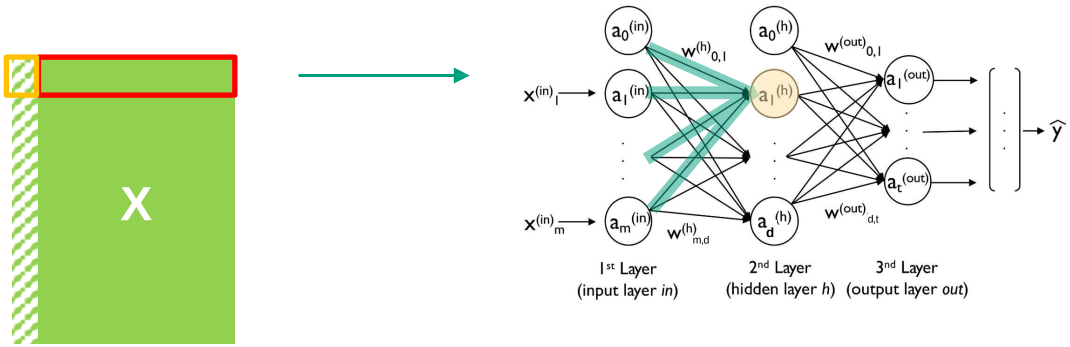
12 weights

(4 x 3)

Total:
15 weights

Need to train 16 + 15 = 31 weights in NN model

A 3-4-3 multi-layer perceptron



(n x m)

$$z_1^{(h)} = a_0^{(in)} w_{0,1}^{(h)} + a_1^{(in)} w_{1,1}^{(h)} + \cdots + a_m^{(in)} w_{m,1}^{(h)}$$

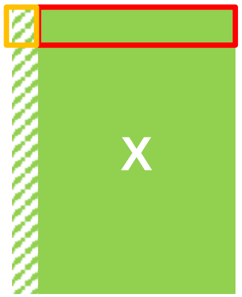
(1 x 1) (1 x 1) (1 x 1) (1 x 1) (1 x 1) (1 x 1) (1 x 1)

Computations for one sample (row) x_i in $\boldsymbol{X}$ for one neuron in hidden layer

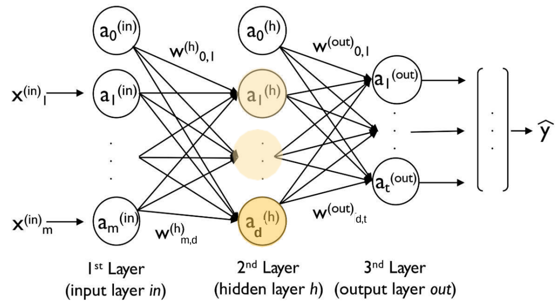
$$a_1^{(h)} = \phi\left(z_1^{(h)}\right)$$

(1 x 1) (1 x 1)

A 3-4-3 multi-layer perceptron



(n x m)



m: features
d: neurons
\# Samples: 1

$$\boldsymbol{z}^{(h)} = \boldsymbol{a}^{(in)}\boldsymbol{W}^{(h)}$$

(1 x d) (1 x m) (m x d)

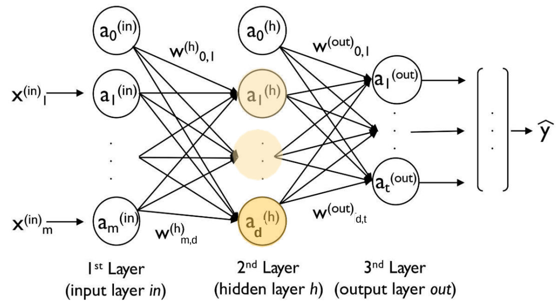
$$\boldsymbol{a}^{(h)} = \phi\left(\boldsymbol{z}^{(h)}\right)$$

(1 x d) (1 x d)

Computations
for one sample
(row) x_i in $\boldsymbol{X}$
for all neurons
in hidden layer

A 3-4-3 multi-layer perceptron

(n x m)



m: features
d: neurons
Samples: n

$$\boldsymbol{Z}^{(h)} = \boldsymbol{A}^{(in)}\boldsymbol{W}^{(h)}$$

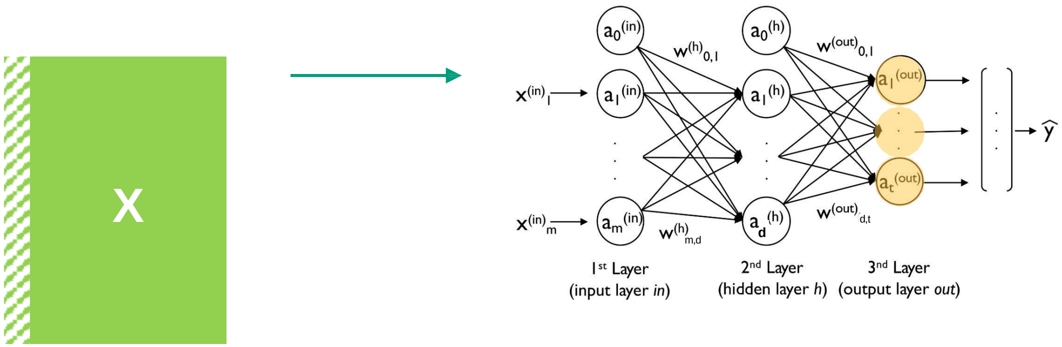
(n x d) (n x m) (m x d)

$$\boldsymbol{A}^{(h)} = \phi\left(\boldsymbol{Z}^{(h)}\right)$$

(n x d) (n x d)

Computations
for all n samples
(rows) x_i in $\boldsymbol{X}$
for all neurons in
hidden layer

A 3-4-3 multi-layer perceptron



(n x m)

t: labels
d: neurons
Samples: n

Computations
for all n samples
(rows) x_i in $\boldsymbol{X}$
for all neurons in
output layer

$$\boldsymbol{Z}^{(out)} = \boldsymbol{A}^{(h)}\boldsymbol{W}^{(out)}$$

(n x t) (n x d) (d x t)

$$\boldsymbol{A}^{(out)} = \phi\left(\boldsymbol{Z}^{(out)}\right)$$

(n x t) (n x t)

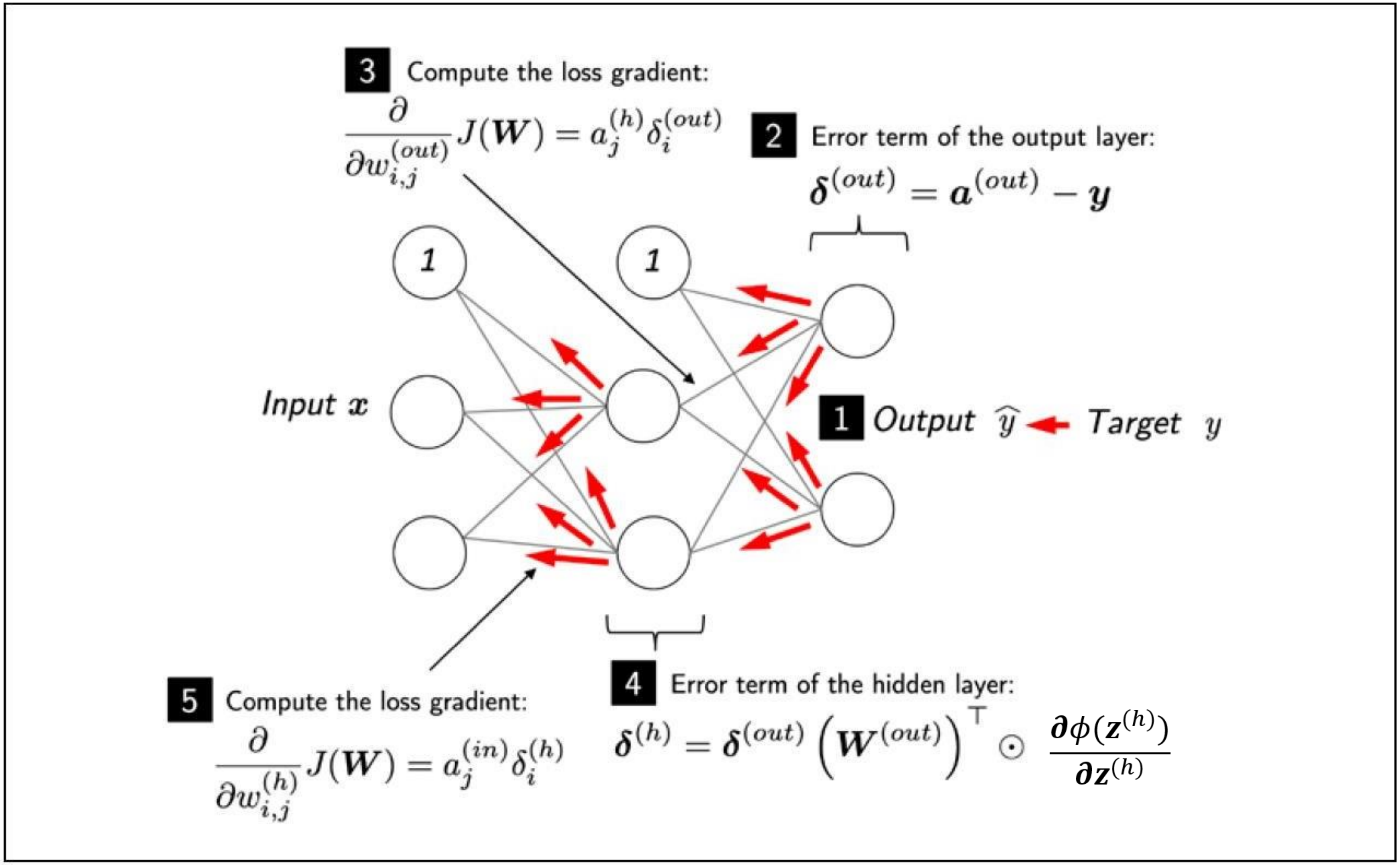
Backpropagation

One sample

$$\boldsymbol{\delta}^{(out)} = \boldsymbol{a}^{(out)} - \boldsymbol{y}$$

(1 x t) (1 x t) (1 x t)

Backpropagation



Backpropagation

One sample

t: labels
d: neurons
Samples: 1

$$\boldsymbol{\delta}^{(out)} = \boldsymbol{a}^{(out)} - \boldsymbol{y}$$

(1 x t) (1 x t) (1 x t)

$$\boldsymbol{\delta}^{(h)} = \boldsymbol{\delta}^{(out)} \left(\boldsymbol{W}^{(out)}\right)^{T} \odot \underbrace{\frac{\partial \phi\left(z^{(h)}\right)}{\partial z^{(h)}}}_{\text{Derivative of activation function}}$$

(1 x d) (1 x t) (t x d) (1 x d)

$$\frac{\partial \phi(z)}{\partial z} = \underbrace{\left(a^{(h)} \odot \left(1 - a^{(h)}\right)\right)}_{\text{Derivative of sigmoid activation function}}$$

(1 x d) (1 x d) (1 x d)

Backpropagation

n samples

t: labels
d: neurons
Samples: n

$$\boldsymbol{\Delta}^{(out)} = \boldsymbol{A}^{(out)} - \boldsymbol{Y}$$

(n x t) (n x t) (n x t)

$$\boldsymbol{\Delta}^{(h)} = \boldsymbol{\Delta}^{(out)}\left(\boldsymbol{W}^{(out)}\right)^T \odot \frac{\partial \phi\left(\boldsymbol{Z}^{(h)}\right)}{\partial \boldsymbol{Z}^{(h)}}$$

(n x d) (n x t) (t x d) (n x d)

Derivative of activation function

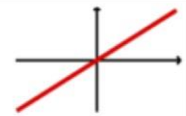
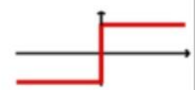
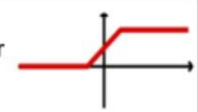
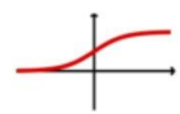
$$\frac{\partial \phi\left(\boldsymbol{Z}^{(h)}\right)}{\partial \boldsymbol{Z}^{(h)}} = \left(\boldsymbol{A}^{(h)} \odot \left(\boldsymbol{1} - \boldsymbol{A}^{(h)}\right)\right)$$

(n x d) (n x d) (n x d)

Derivative of ***sigmoid*** activation function

Vanishing gradients problem

Vanishing gradients problem

Activation Function	Equation	Example	1D Graph
Linear	$\phi(z) = z$	Adaline, linear regression	
Unit Step (Heaviside Function)	$\phi(z) = \begin{cases} 0 & z < 0 \\ 0.5 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Sign (signum)	$\phi(z)= \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Piece-wise Linear	$\phi(z)= \begin{cases} 0 & z \le -\frac{1}{2} \\ z + \frac{1}{2} & -\frac{1}{2} \le z \le \frac{1}{2} \\ 1 & z \ge \frac{1}{2} \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z)= \dfrac{1}{1 + e^{-z}}$	Logistic regression, Multilayer NN	
Hyperbolic Tangent (tanh)	$\phi(z)= \dfrac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multilayer NN, RNNs	
ReLU	$\phi(z)= \begin{cases} 0 & z < 0 \\ z & z > 0 \end{cases}$	Multilayer NN, CNNs	

Vanishing gradients problem

$$\boldsymbol{\delta}^{(h)} = \boldsymbol{\delta}^{(out)} \left(\boldsymbol{W}^{(out)}\right)^{\top} \odot \underbrace{\frac{\partial \phi(\mathbf{z}^{(h)})}{\partial \mathbf{z}^{(h)}}}_{\text{Derivative of activation function}}$$

Page 419, 3rd edition

Derivative of *sigmoid* activation function

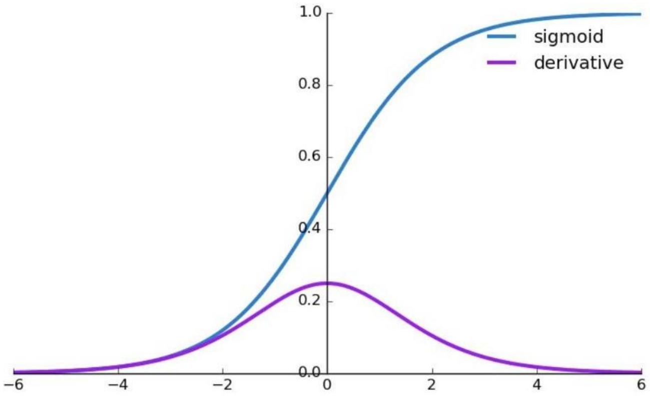
$$\frac{\partial \phi(z)}{\partial z} = (a \odot (1 - a))$$

$$a = \phi(z) = \frac{1}{1 + e^{-z}}$$

Vanishing gradients problem

Derivative of *sigmoid* activation function

$$\frac{\partial \phi(z)}{\partial z} = (a \odot (1 - a))$$



a	$1 - a$	$a \odot (1 - a)$
0.1	0.9	0.09
0.2	0.8	0.16
0.3	0.7	0.21
0.4	0.6	0.24
0.5	0.5	0.25
0.6	0.4	0.24
0.7	0.3	0.21
0.8	0.2	0.16
0.9	0.1	0.09

$$\boldsymbol{\delta}^{(h)} = \boldsymbol{\delta}^{(out)} \left(\boldsymbol{W}^{(out)}\right)^{\top} \odot \frac{\partial \phi(\boldsymbol{z}^{(h)})}{\partial \boldsymbol{z}^{(h)}}$$

$$\boldsymbol{\delta}^{(h)} = \boldsymbol{\delta}^{(out)} \left(\boldsymbol{W}^{(out)}\right)^{\top} \odot \left(a^{(h)} \odot \left(1 - a^{(h)}\right)\right)$$

Vanishing gradients problem

$$\boldsymbol{\delta}^{(h)} = \boldsymbol{\delta}^{(out)} \left(\boldsymbol{W}^{(out)}\right)^{\top} \odot \underbrace{\frac{\partial \phi(\mathbf{z}^{(h)})}{\partial \mathbf{z}^{(h)}}}_{\text{Derivative of activation function}}$$

Derivative of *sigmoid* activation function

$$\frac{\partial \phi(z)}{\partial z} = (a \odot (1 - a))$$

$$a = \phi(z) = \frac{1}{1 + e^{-z}}$$

Derivative of *tanh* activation function

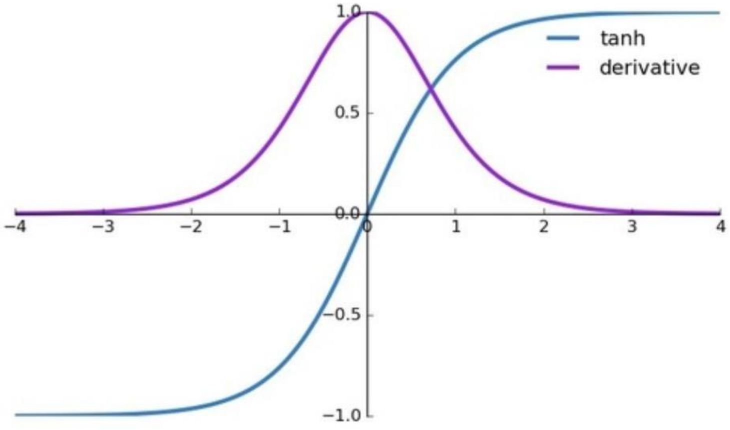
$$\frac{\partial \phi(z)}{\partial z} = (1 - a^2)$$

$$a = \phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

Vanishing gradients problem

Derivative of *tanh* activation function

$$\frac{\partial \phi(z)}{\partial z} = 1 - a^2$$

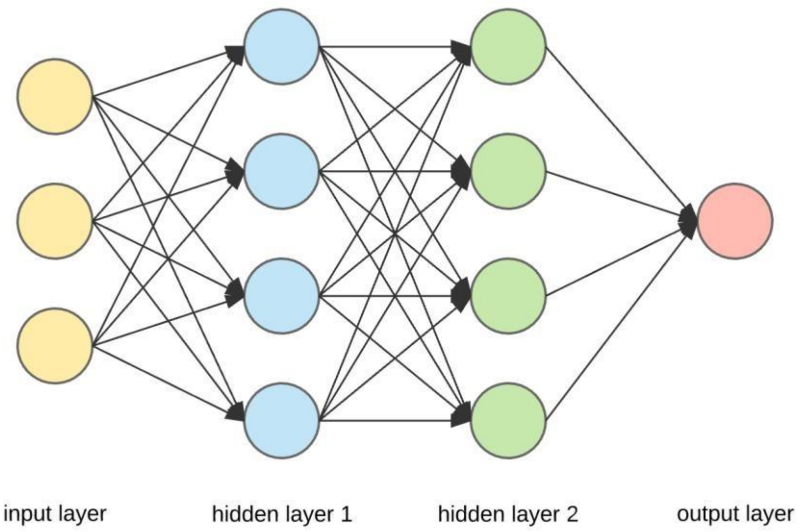


$$\boldsymbol{\delta}^{(h)} = \boldsymbol{\delta}^{(out)} \left(\boldsymbol{W}^{(out)}\right)^{\top} \odot \frac{\partial \phi(\boldsymbol{z}^{(h)})}{\partial \boldsymbol{z}^{(h)}}$$

$$\boldsymbol{\delta}^{(h)} = \boldsymbol{\delta}^{(out)} \left(\boldsymbol{W}^{(out)}\right)^{\top} \odot \left(1 - a^{(h)^2}\right)$$

a	$1 - a^2$
- 0.9	0.19
- 0.5	0.75
- 0.2	0.96
- 0.1	0.99
0.0	1.00
0.1	0.99
0.2	0.96
0.5	0.75
0.9	0.19

Backpropagation



$$\boldsymbol{\delta}^{(out)} = \boldsymbol{a}^{(out)} - \boldsymbol{y}$$

$$\boldsymbol{\delta}^{(h_2)} = \boldsymbol{\delta}^{(out)} (\boldsymbol{W}^{(out)})^T \odot \frac{\partial \phi(\boldsymbol{z}^{(h_2)})}{\partial \boldsymbol{z}^{(h_2)}}$$

$$\boldsymbol{\delta}^{(h_1)} = \boldsymbol{\delta}^{(h_2)} (\boldsymbol{W}^{(h_2)})^T \odot \frac{\partial \phi(\boldsymbol{z}^{(h_1)})}{\partial \boldsymbol{z}^{(h_1)}}$$

$$\boldsymbol{\delta}^{(h_1)} = \boldsymbol{\delta}^{(out)} (\boldsymbol{W}^{(out)})^T \odot \frac{\partial \phi(\boldsymbol{z}^{(h_2)})}{\partial \boldsymbol{z}^{(h_2)}} (\boldsymbol{W}^{(h_2)})^T \odot \frac{\partial \phi(\boldsymbol{z}^{(h_1)})}{\partial \boldsymbol{z}^{(h_1)}}$$