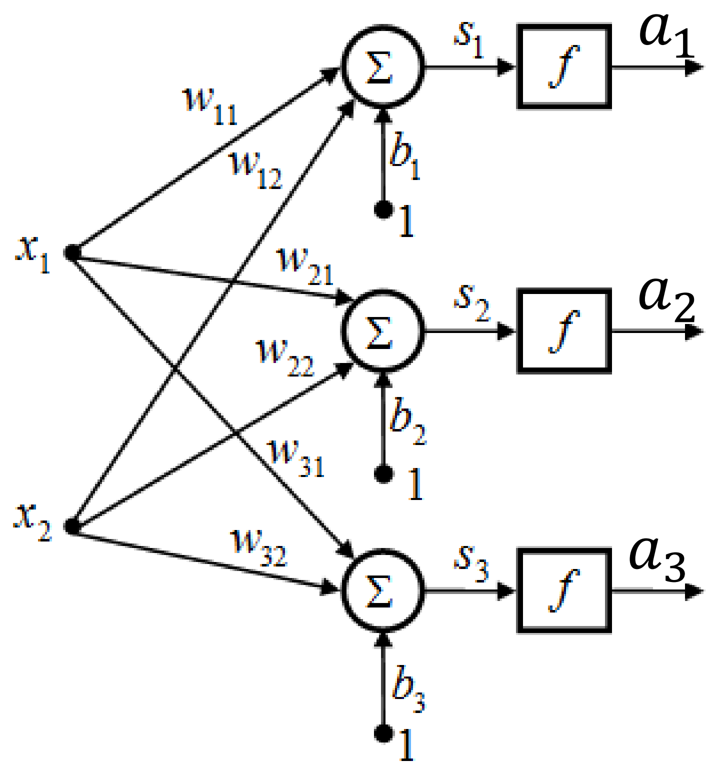
Shallow Neural Network

(Small-size, standard ANN)

1 layer ANN
2 inputs, 3 outputs



$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}$$

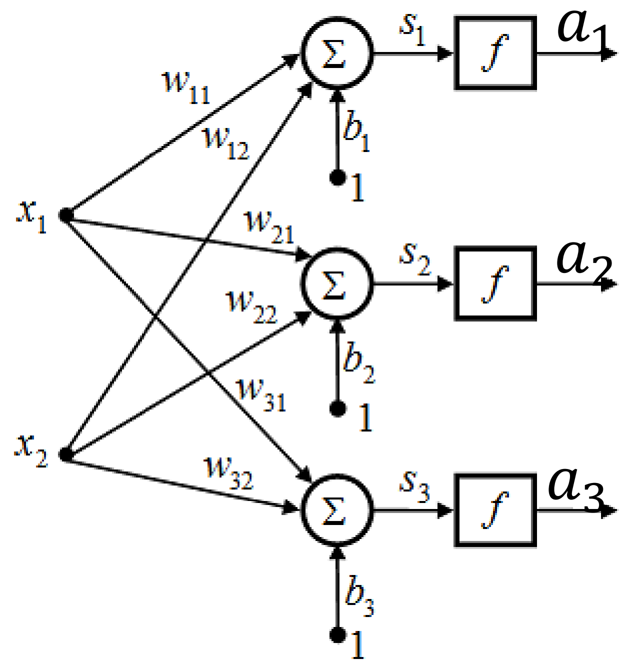
$$b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \qquad s = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} \qquad a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

$$\boxed{s = Wx + b} \qquad \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \qquad \boxed{a = f(s)}$$

Exercise

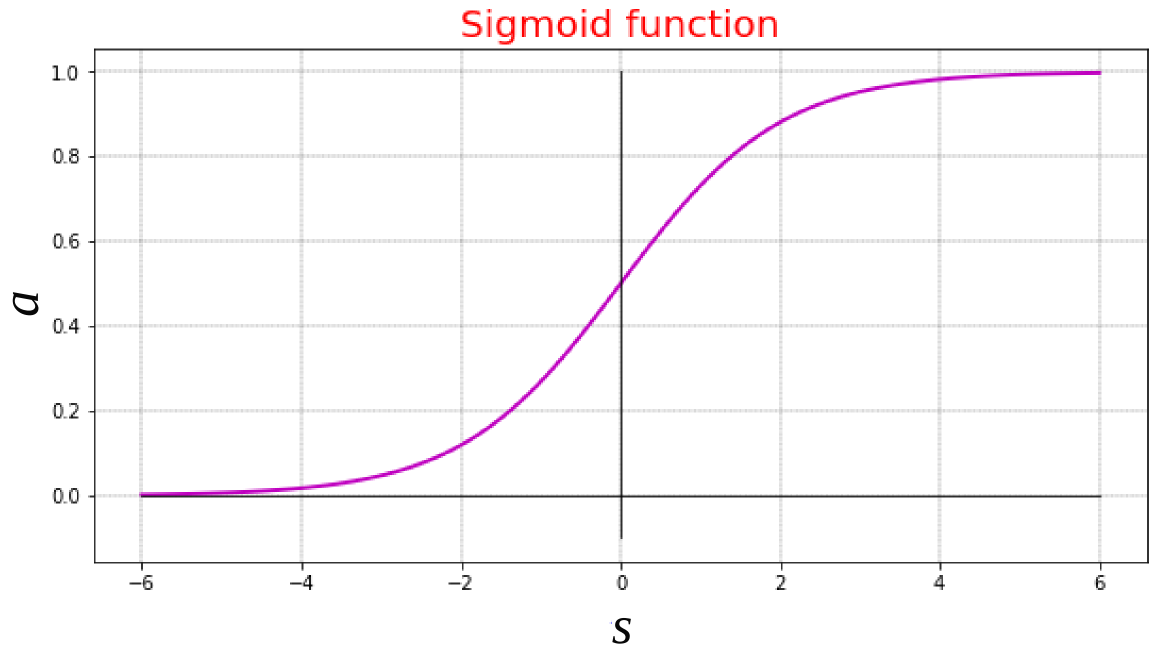
$$x^{(1)} = \begin{bmatrix} -1 \\ -4 \end{bmatrix} \qquad x^{(2)} = \begin{bmatrix} 2 \\ 4 \end{bmatrix} \qquad x^{(3)} = \begin{bmatrix} 10 \\ 0 \end{bmatrix}$$

$$W = \begin{bmatrix} -0.5 & -1 \\ -0.1 & 1 \\ 0.75 & -0.5 \end{bmatrix} b = \begin{bmatrix} 0.5 \\ -0.5 \\ -3 \end{bmatrix}$$

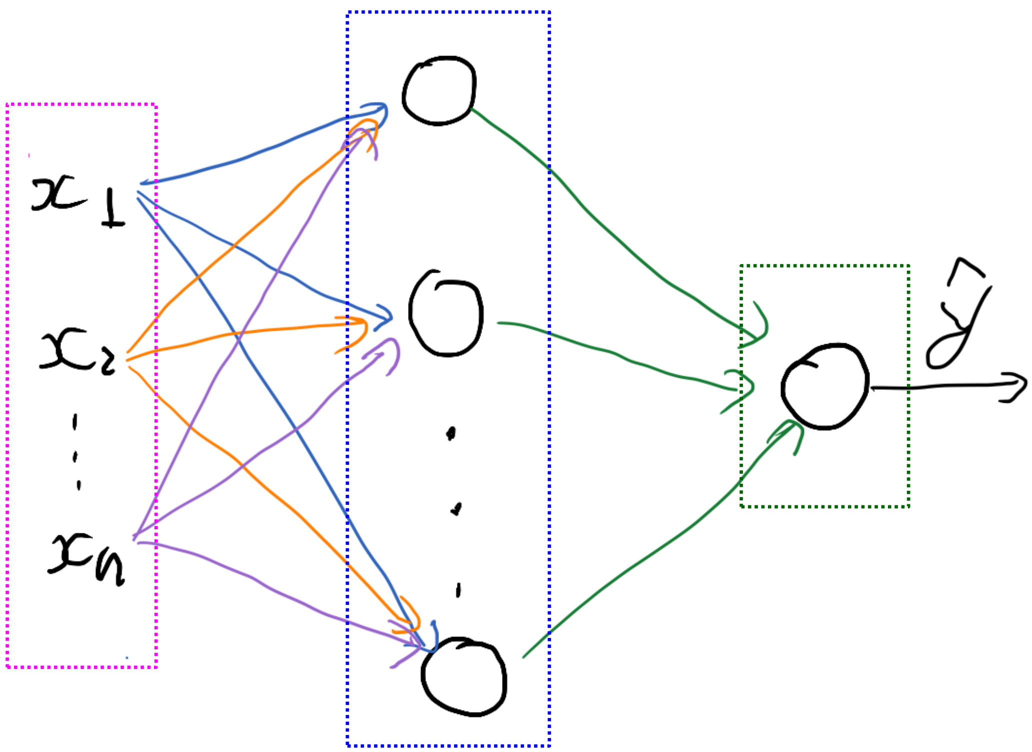


$$f(s) = \frac{1}{1+e^{-s}}$$

Compute the output vector a, for each input vector $x^{(i)}$



Neural network representation
2- layer NN

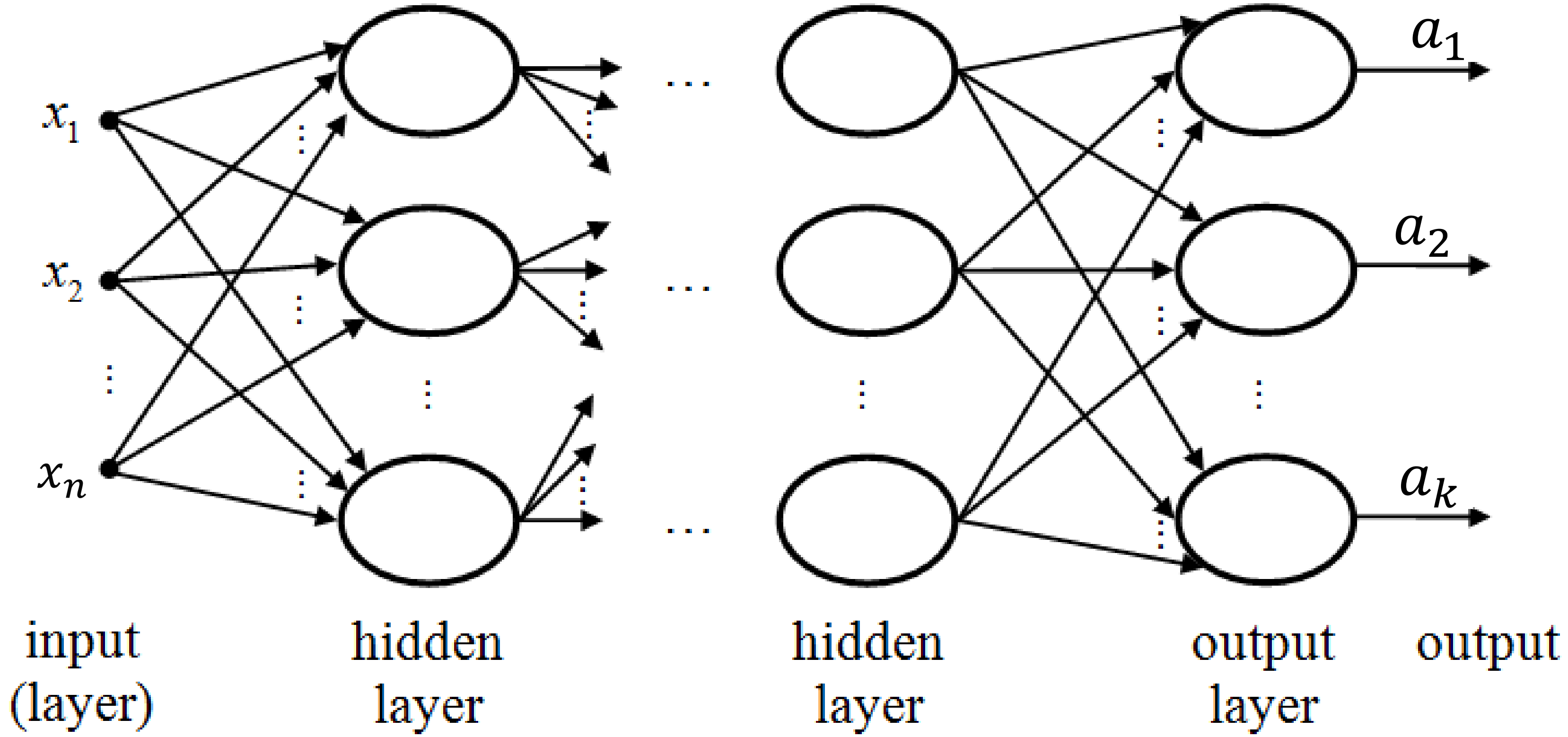


Input

Hidden layer
1st layer

Output layer
2nd layer

Feedforward Neural Network



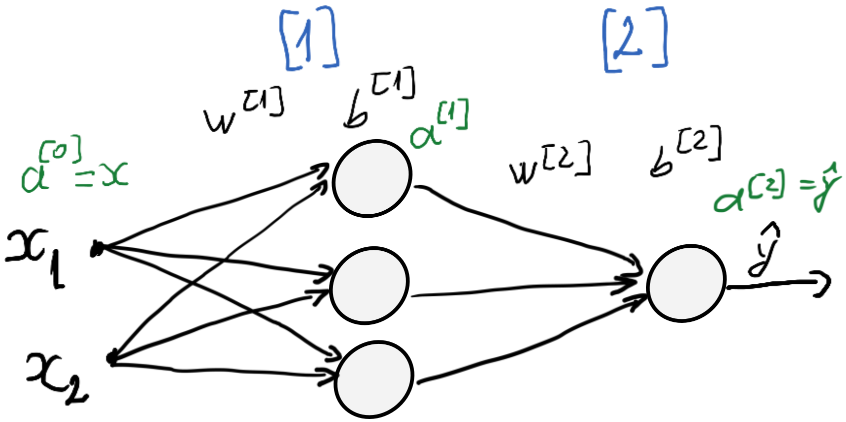
The arhitecture of a multilayer feedforward neural network

It is common for different layers to have different numbers of neurons.

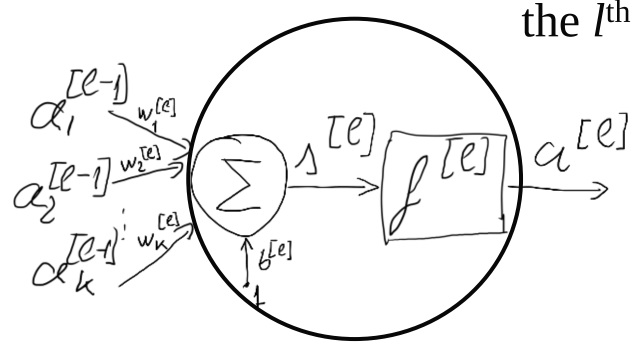
Multiple-layer networks are quite powerful.

An ANN of two layers, where the first layer is sigmoid and the second layer is linear, can be trained **to approximate any function** (with a finite number of discontinuities) **arbitrarily well** **- universal aproximator**

2-layer NN. Illustration for 2 inputs 1 output; 1 example



A neuron in the l^{th} layer



$$a^{[0]} = \begin{bmatrix} a_1^{[0]} \\ a_2^{[0]} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad s^{[1]} = \begin{bmatrix} s_1^{[1]} \\ s_2^{[1]} \\ s_3^{[1]} \end{bmatrix} \quad a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \end{bmatrix} \quad s^{[2]} = \begin{bmatrix} s_1^{[2]} \end{bmatrix} \quad a^{[2]} = \begin{bmatrix} a_1^{[2]} \end{bmatrix} = \hat{y}$$

$$W^{[1]} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} \\ w_{31}^{[1]} & w_{32}^{[1]} \end{bmatrix} \quad b^{[1]} = \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \end{bmatrix} \quad W^{[2]} = \begin{bmatrix} w_{11}^{[2]} & w_{12}^{[2]} & w_{13}^{[2]} \end{bmatrix} \quad b^{[2]} = \begin{bmatrix} b_1^{[2]} \end{bmatrix}$$

$$s^{[1]} = W^{[1]} a^{[0]} + b^{[1]}; \quad a^{[1]} = f^{[1]}(s^{[1]}); \quad s^{[2]} = W^{[2]} a^{[1]} + b^{[2]}; \quad a^{[2]} = \hat{y} = f^{[2]}(s^{[2]})$$

$$s^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}; \quad a^{[l]} = f^{[l]}(s^{[l]})$$

Vectorized implementation for one input example to compute the NN's output.

Exercise

a) Draw the schematic diagram for an ANN with:

- 3 inputs
- 1 hidden layer with 4 neurons
- output layer with 2 neurons (2 outputs).

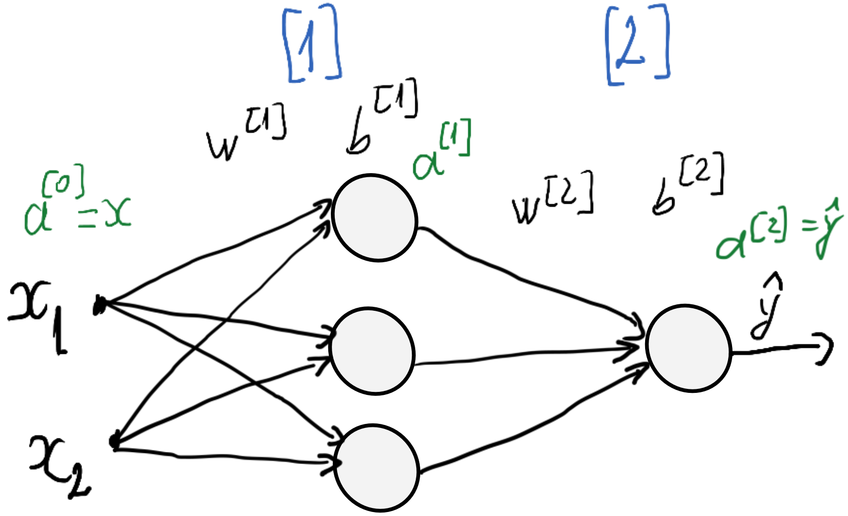
b) Represents the vectors or matrices for the input, all intermediate variables, and the output.

c) Represents the vectors / matrices for the training parameters in all layers.

d) Write all the equations to compute the ANN's output.

e) Compute the total number of training parameters

2-layer NN. Computing NN's output for one example.



for a single x input example

$$z^{[1]} = W^{[1]} x + b^{[1]}$$

$$a^{[1]} = f^{[1]}(z^{[1]})$$

$$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$$

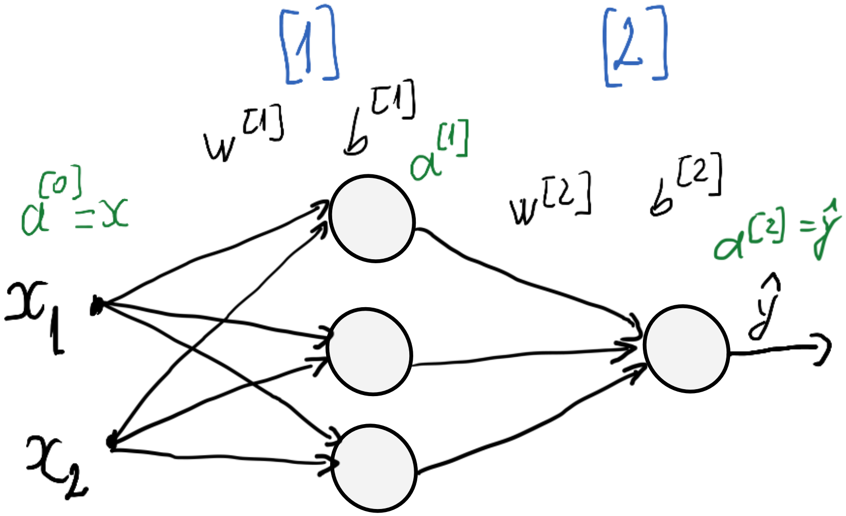
$$\hat{y} = a^{[2]} = f^{[2]}(z^{[2]})$$

one example

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \updownarrow \text{2 features}$$

$$\hat{y} = [\hat{y}]$$

2-layer NN. Computing NN's output for multiple examples (m).



$$X = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(m)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(m)} \end{bmatrix}$$

2 features

m examples

$$\hat{Y} = \begin{bmatrix} \hat{y}^{(1)} & \hat{y}^{(2)} & \dots & \hat{y}^{(m)} \end{bmatrix}$$

m input examples:

$x \longrightarrow a^{[2]} = \hat{y}$

$x^{(1)} \longrightarrow a^{[2](1)} = \hat{y}^{(1)}$

$x^{(2)} \longrightarrow a^{[2](2)} = \hat{y}^{(2)}$

$\vdots$

$x^{(m)} \longrightarrow a^{[2](m)} = \hat{y}^{(m)}$

$a^{[l](i)}$

layer l example i

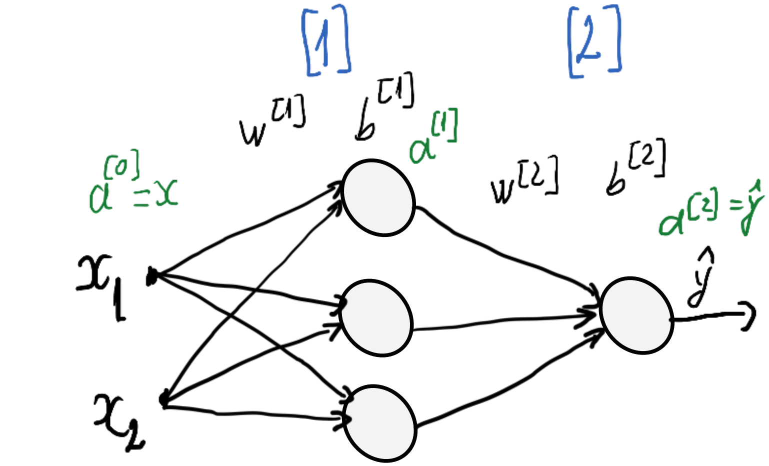
in case of multiple input examples
m examples

for $i = 1$ to m:

$$z^{[1](i)} = W^{[1]} x^{(i)} + b^{[1]}$$
$$a^{[1](i)} = f^{[1]}(z^{[1](i)})$$
$$z^{[2](i)} = W^{[2]} a^{[1](i)} + b^{[2]}$$
$$a^{[2](i)} = f^{[2]}(z^{[2](i)})$$

For loop

Not recommended for implementation!



2-layer NN. Computing NN's output for multiple examples.

Vectorising across multiple examples

input

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ | & | & & | \end{bmatrix} \quad n^{[0]} \text{ features}$$

m examples

$$S^{[l]} = \begin{bmatrix} | & | & & | \\ s^{[l](1)} & s^{[l](2)} & \dots & s^{[l](m)} \\ | & | & & | \end{bmatrix} \quad \text{hidden neurons in layer } [l]$$

m examples

$$A^{[l]} = \begin{bmatrix} | & | & & | \\ a^{[l](1)} & a^{[l](2)} & \dots & a^{[l](m)} \\ | & | & & | \end{bmatrix} \quad \text{hidden neurons in layer } [l]$$

m examples

$$S^{[1]} = W^{[1]} X + b^{[1]}$$

$$A^{[1]} = f^{[1]}(S^{[1]})$$

$$S^{[2]} = W^{[2]} A^{[1]} + b^{[2]}$$

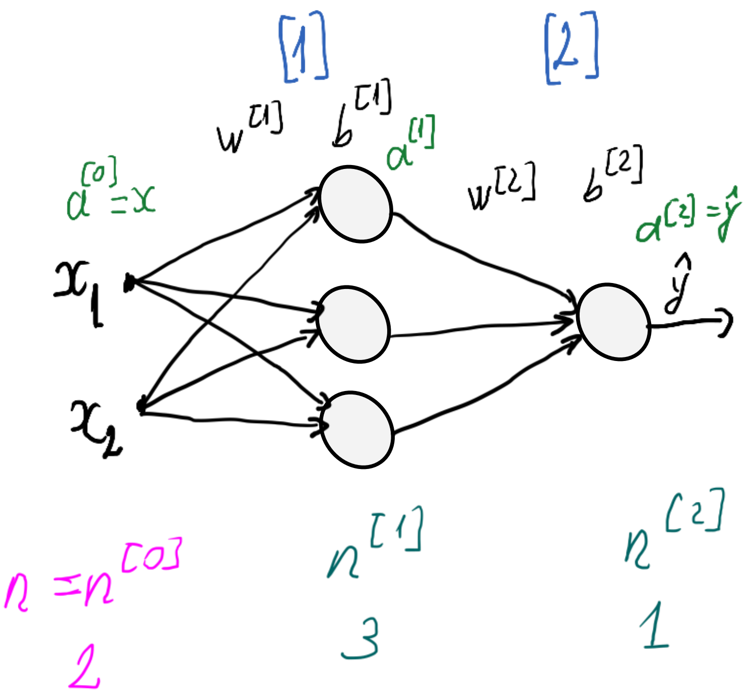
$$\hat{Y} = A^{[2]} = f^{[2]}(S^{[2]})$$

2-layer NN. Computing NN’s output for multiple examples.
Vectorising across multiple examples

$n = n^{[0]}$ – number of inputs (features)

m – number of examples

$n^{[1]}$ – number of neurons in the (1st) hidden layer



matrix $(n^{[1]}, m)$

matrix $(n^{[1]}, n^{[0]})$

matrix $(n^{[0]}, m)$

column vector $(n^{[1]}, 1)$

$$S^{[1]} = W^{[1]} X + b^{[1]}$$

$$A^{[1]} = f^{[1]}(S^{[1]})$$

$$S^{[2]} = W^{[2]} A^{[1]} + b^{[2]}$$

$$\hat{Y} = A^{[2]} = f^{[2]}(S^{[2]})$$

row vector $(1, m)$

matrix + *column* operates correctly due to broadcasting

2-layer NN.

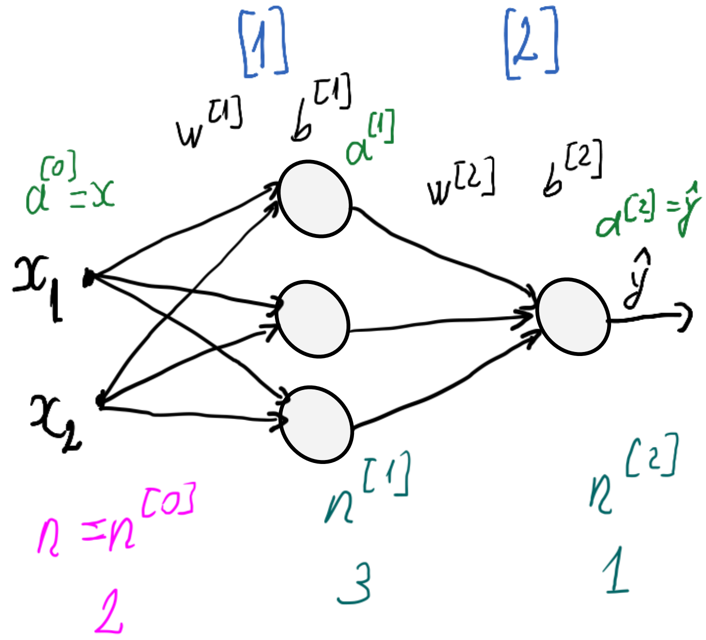
Computing NN's output for multiple example.

Vectorising across multiple examples

$n = n^{[0]}$ – number of inputs (features)

m – number of examples

$n^{[1]}$ – number of neurons in the hidden layer



$$\underset{n^{[1]} \times m}{\left[\, S^{[1]} \,\right]} = \underset{n^{[1]} \times n^{[0]}}{\left[\, W^{[1]} \,\right]} \cdot \underset{n^{[0]} \times m}{\left[\, X \,\right]} + \underset{n^{[1]} \times 1}{\left[\, b^{[1]} \,\right]} \qquad \underset{n^{[1]} \times m}{\left[\, A^{[1]} \,\right]} = f^{[1]}(S^{[1]})$$

$$\underset{n^{[2]} \times m}{\left[\, S^{[2]} \,\right]} = \underset{n^{[2]} \times n^{[1]}}{\left[\, W^{[2]} \,\right]} \cdot \underset{n^{[1]} \times m}{\left[\, A^{[1]} \,\right]} + \underset{n^{[2]} \times 1}{\left[\, b^{[2]} \,\right]} \qquad \underset{n^{[2]} \times m}{\left[\, A^{[2]} \,\right]} = f^{[2]}(S^{[2]})$$

2 - layer NN (2 inputs, 1 output). Forward and backward propagation

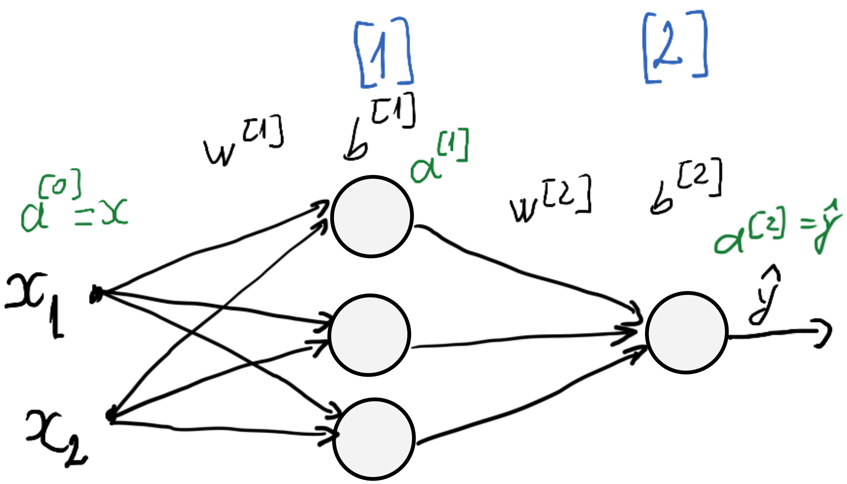
1. Compute the output - Forward propagation
2. Compute the gradients – backward propagation
3. Update the parameters

All 3 phases are necessary for **ANN training**:

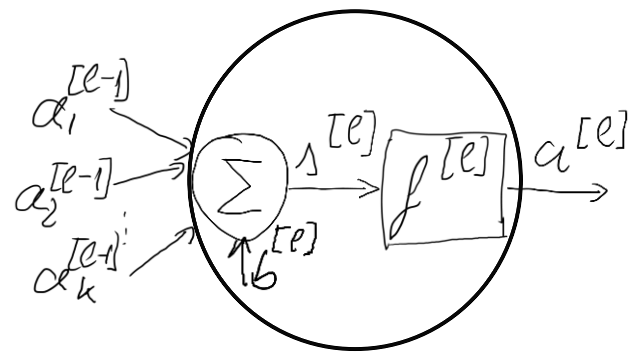
- needs multiple training examples
- multiple training epochs

For **ANN simulation (inference, utilization, testing)**:

- only forward propagation is involved



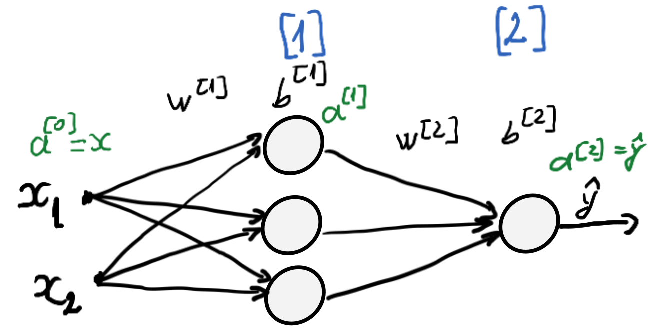
Schematic diagram



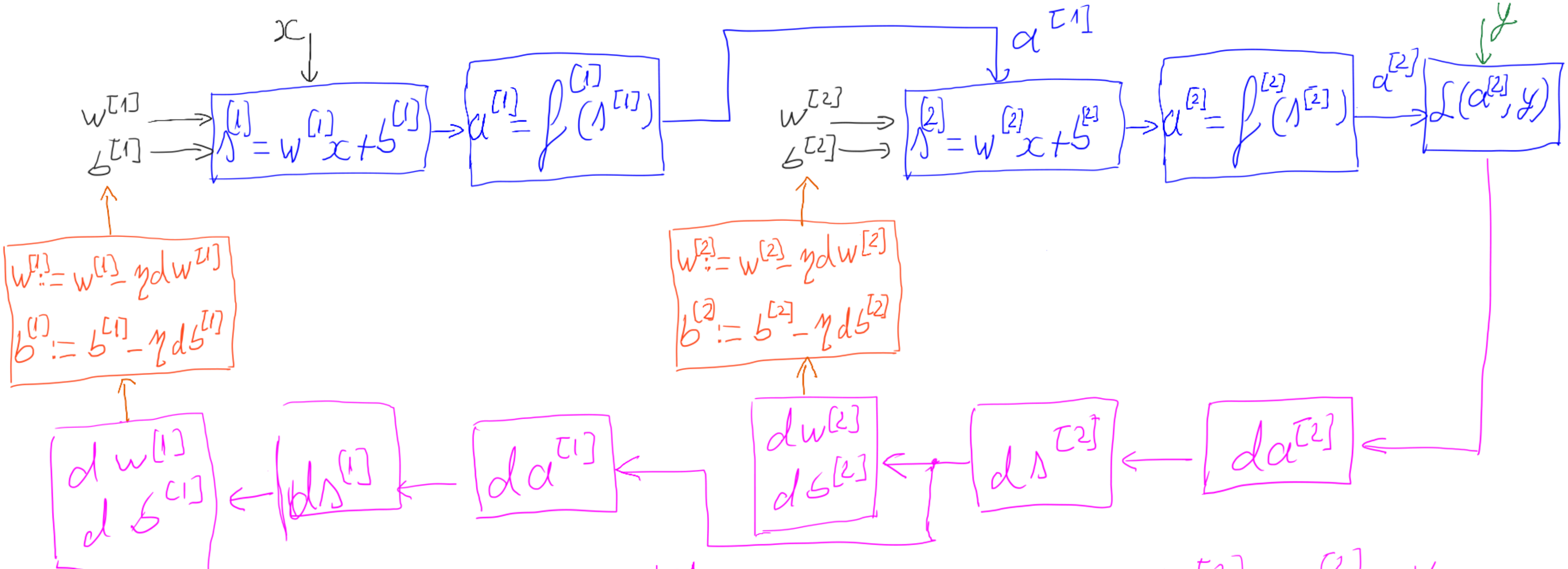
A neuron in the l^{th} layer

2-layer NN (2 inputs, 1 output).

1. Compute the output - Forward propagation
2. Compute the gradients – backward propagation
3. Update the parameters



Training for 1 example



$$ds^{[1]} = w^{[2]T} \cdot ds^{[2]} * f^{[1]'}(s^{[1]})$$

elementwise multiplication

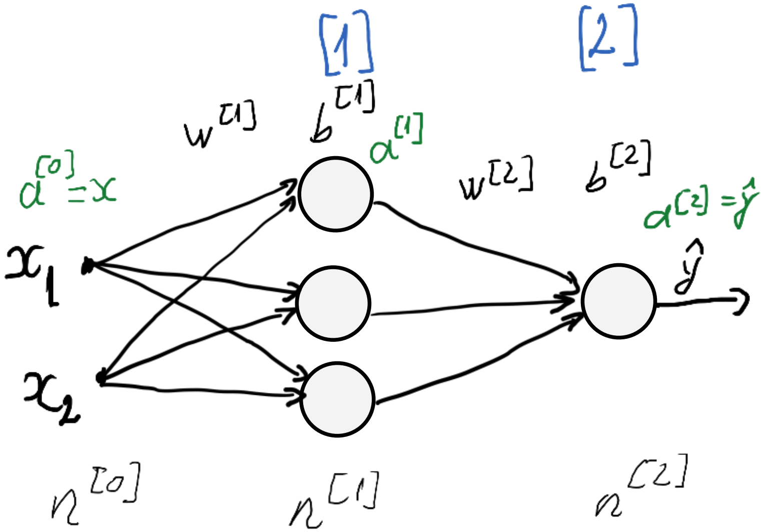
$$ds^{[2]} = a^{[2]} - y$$

$$\begin{cases} dw^{[1]} = ds^{[1]} x^T \\ db^{[1]} = ds^{[1]} \end{cases}$$

$$\begin{cases} dw^{[2]} = ds^{[2]} a^{[1]T} \\ db^{[2]} = ds^{[2]} \end{cases}$$

2-layer NN **(2 inputs, 1 output).**

1. Compute the output - Forward propagation
2. Compute the gradients – backward propagation
3. Update the parameters



Checking dimensions

$w^{[2]}, \; dw^{[2]} \quad (n^{[2]}, n^{[1]}); \; (1, 3)$

$z^{[2]}, \; dz^{[2]} \quad (n^{[2]}, 1); \; (1, 1)$

$w^{[1]}, \; dw^{[1]} \quad (n^{[1]}, n^{[0]}); \; (3, 2)$

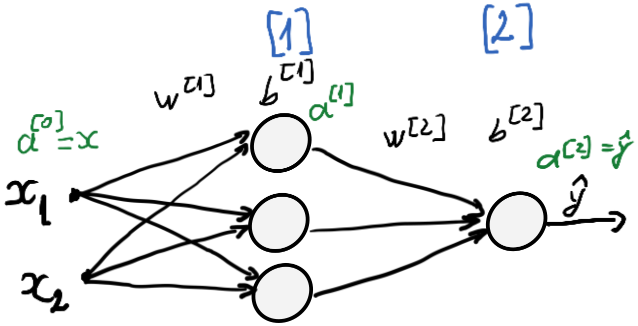
$z^{[1]}, \; dz^{[1]} \quad (n^{[1]}, 1); \; (3, 1)$

$$\underset{(n^{[1]},1)}{dz^{[1]}} = \underbrace{\underset{(n^{[1]},n^{[2]})}{w^{[2]T}} \cdot \underset{(n^{[2]},1)}{dz^{[2]}}}_{(n^{[1]},1)} * \underset{(n^{[1]},1)}{f^{[1]'}(z^{[1]})}$$

2-layer NN (2 inputs, 1 output).

Backward propagation (Compute the gradients)

Summary



$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(a^{(i)}, y^{(i)})$$

1 example	m examples
$ds^{[2]} = a^{[2]} - y$	$dS^{[2]} = A^{[2]} - Y$
$dw^{[2]} = ds^{[2]} a^{[1]^T}$	$dW^{[2]} = \frac{1}{m} dS^{[2]} A^{[1]^T}$
$db^{[2]} = ds^{[2]}$	$db^{[2]} = \frac{1}{m}$ np.sum($dS^{[2]}$, axis = 1, keepdims = True)
$ds^{[1]} = W^{[2]^T} \cdot ds^{[2]} * f^{[1]'}(s^{[1]})$	$dS^{[1]} = W^{[2]^T} dS^{[2]} * f^{[1]'}(S^{[1]})$
$dw^{[1]} = ds^{[1]} x^T$	$dW^{[1]} = \frac{1}{m} dS^{[1]} X^T$
$db^{[1]} = ds^{[1]}$	$db^{[1]} = \frac{1}{m}$ np.sum($dS^{[1]}$, axis = 1, keepdims = True)

2 layer NN (2 inputs, 1 output).

Training for *m* examples

X

$W^{[1]}$, $b^{[1]}$, $W^{[2]}$, $b^{[2]}$

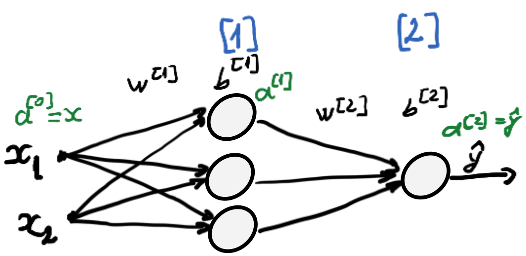
$$S^{[1]} = W^{[1]} X + b^{[1]}$$
$$A^{[1]} = f^{[1]}(S^{[1]})$$
$$S^{[2]} = W^{[2]} A^{[1]} + b^{[2]}$$
$$\hat{Y} = A^{[2]} = f^{[2]}(S^{[2]})$$

$\hat{Y} = A^{[2]}$

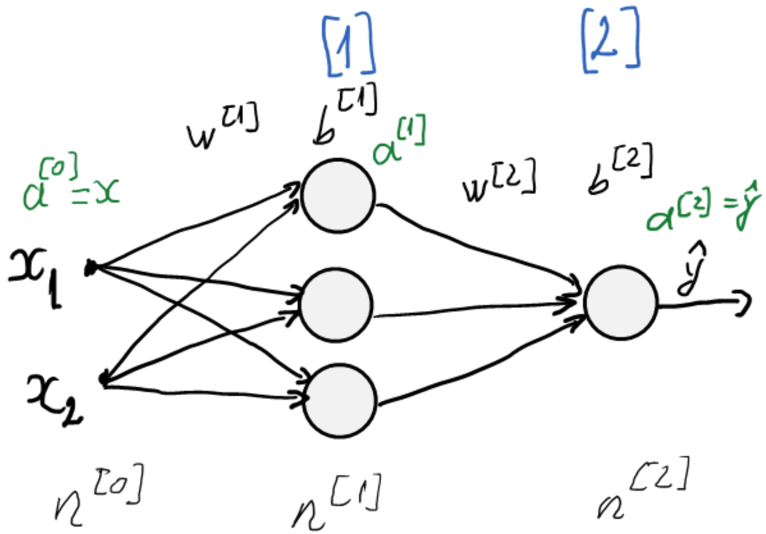
Y

$$dS^{[2]} = A^{[2]} - Y$$
$$dW^{[2]} = \frac{1}{m} dS^{[2]} A^{[1]T}$$
$$db^{[2]} = \frac{1}{m} np.sum(dS^{[2]}, axis=1, keepdims=True)$$
$$dS^{[1]} = W^{[2]T} dS^{[2]} * f^{[1]\prime}(S^{[1]})$$
$$dW^{[1]} = \frac{1}{m} dS^{[1]} X^{T}$$
$$db^{[1]} = \frac{1}{m} np.sum(dS^{[1]}, axis=1, keepdims=True)$$

$$W^{[1]} := W^{[1]} - \eta\, dW^{[1]}$$
$$b^{[1]} := b^{[1]} - \eta\, db^{[1]}$$
$$W^{[2]} := W^{[2]} - \eta\, dW^{[2]}$$
$$b^{[2]} := b^{[2]} - \eta\, db^{[2]}$$



Initialization of NN (*W*, *b*)



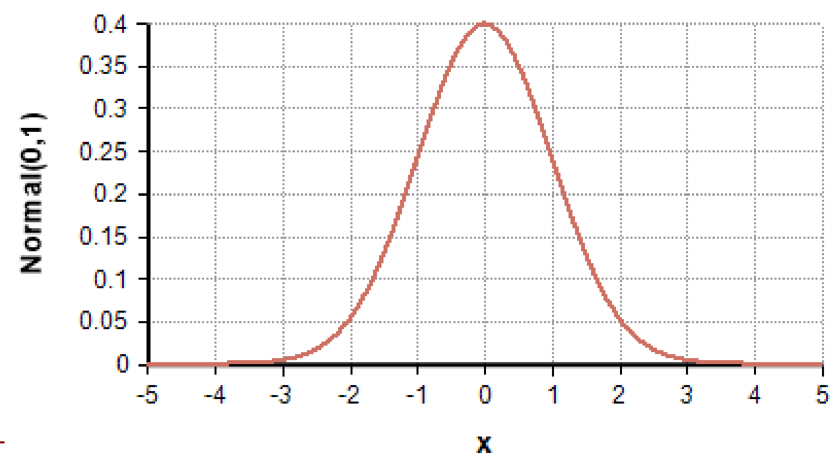
- **It is recommended to initialize the weights matrices (*W*) with small random values (not zero!)**
- For large values, the activation functions (sigmoid type) will be saturated – passive regions
- If we are using zero for *W*, the gradient descent cannot evolve at all!

- **The bias vectors can be initialized by 0.**

$$W^{[1]} = \text{np.random.randn}(n^{[1]}, n^{[0]}) * 0.01$$
$$b^{[1]} = \text{np.zeros}((n^{[1]}, 1))$$

$$W^{[2]} = \text{np.random.randn}(n^{[2]}, n^{[1]}) * 0.01$$
$$b^{[2]} = \text{np.zeros}((n^{[2]}, 1))$$

randn generates an array, filled with random floats sampled from a univariate “normal” (Gaussian) distribution of mean 0 and variance 1.



Methodology to build and train an ANN model