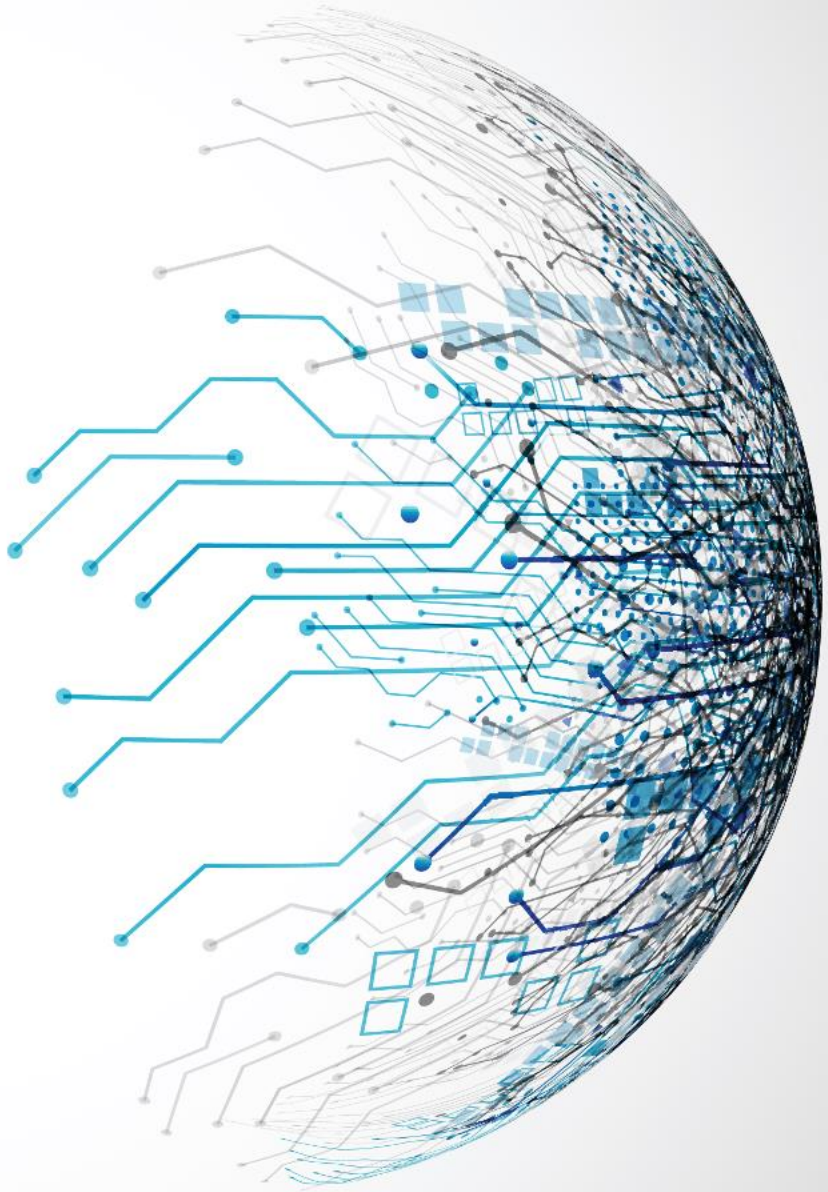
Deep Learning

Backpropagation



Dr. Mohammed Salah Al-Radhi

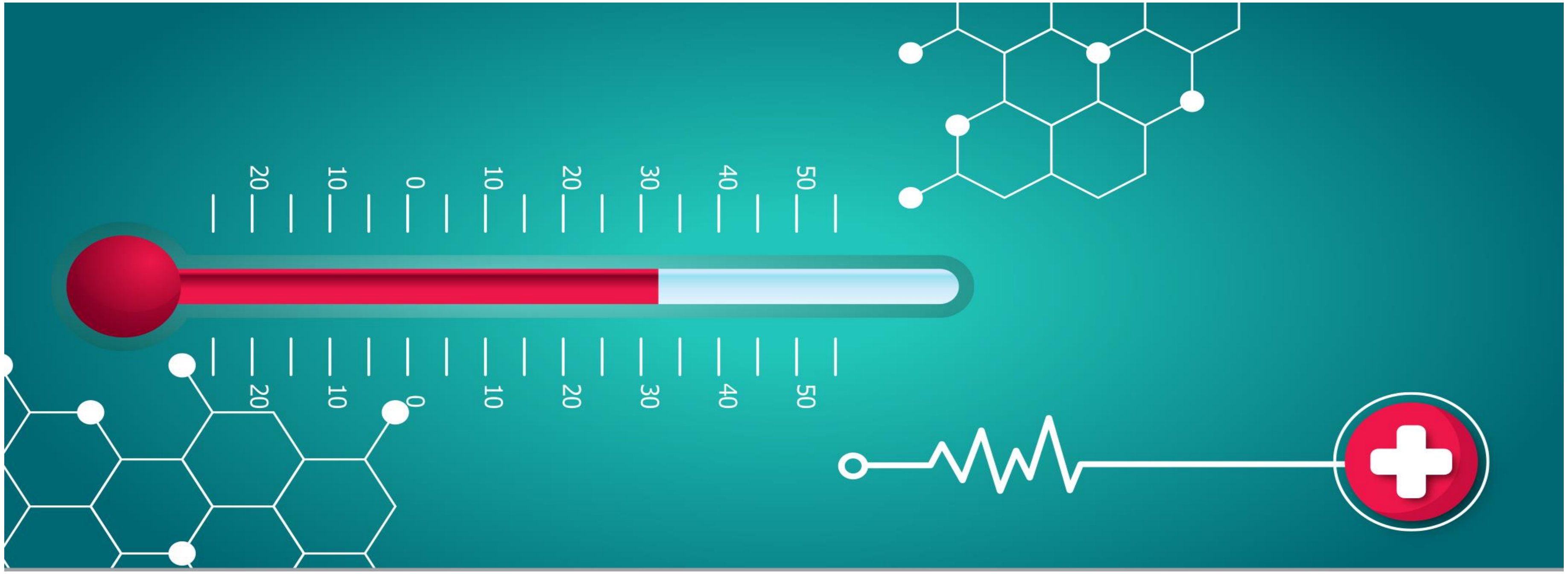
(slides by: Dr. Bálint Gyires-Tóth)

BME-VIK-TMIT

10 September 2024

Copyright

Task



- Input: `[temperature (°C), medicine (mg)]` $= x$
- Output:
  - Regression: `[temperature in 2 hours]` $= y$
  - Classification: `[fever/normal within 2 hours]`

Train and test datasets

$$X = \begin{bmatrix} 38.6 & 25 \\ 37.8 & 25 \\ 37.9 & 50 \\ 38.2 & 50 \end{bmatrix} \quad y = \begin{bmatrix} 37.2 \\ 37.3 \\ 36.6 \\ 36.9 \end{bmatrix} \quad X\_test = [38.3 \;\; 35]$$

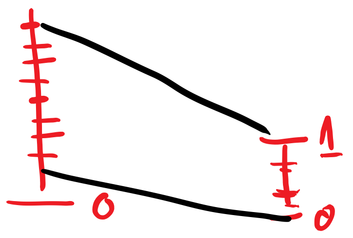
$\hat{y} = ?$

$x = (x - \text{mean}\, X) / \text{var}\, X$

var → variance

mean → mean

$y = \text{minmax}(y, 0, 1)$

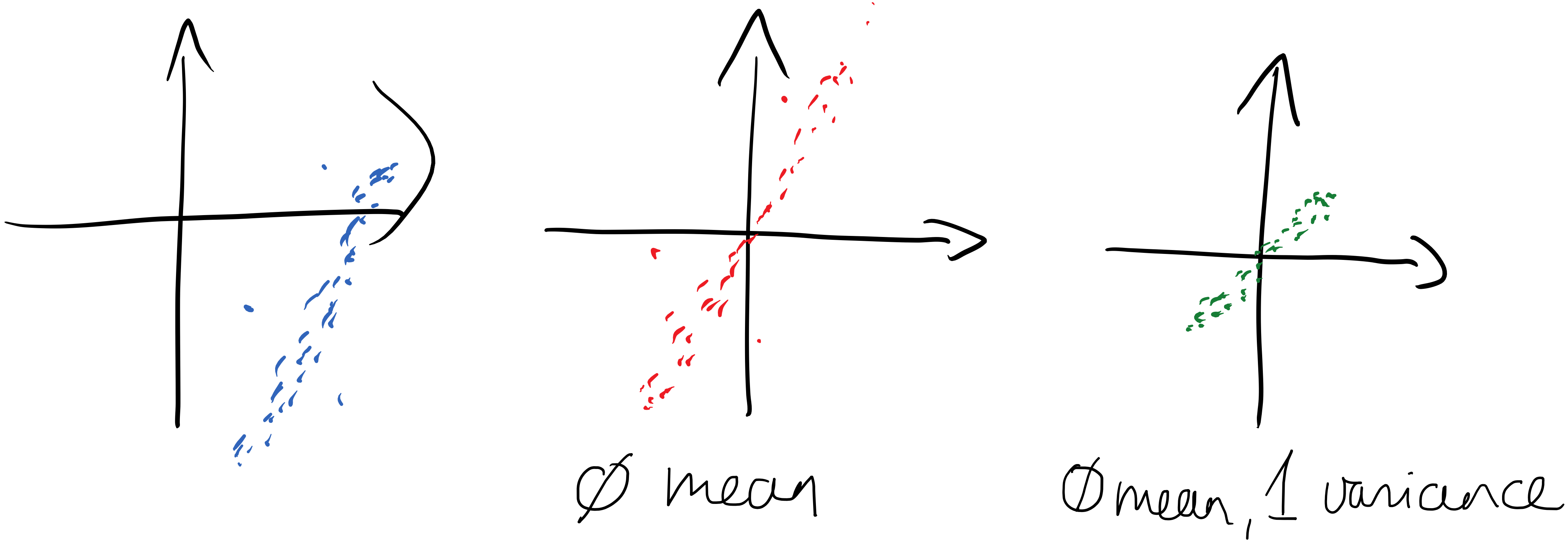


Standardization

To have 0 mean, 1 variance

In practice: subtracting the mean and dividing by the variance

$$X = (x - \text{mean}(x)) / \text{std}(x)$$



Standardization

Mean and variance should be calculated on training data only

The distribution of the data should be inspected first

Why we need it:

- To avoid large biases and slow convergence
- To have different features the same scale
- To match data to initial model (see random initialization methods)

Min-max scaler

The range is fixed.

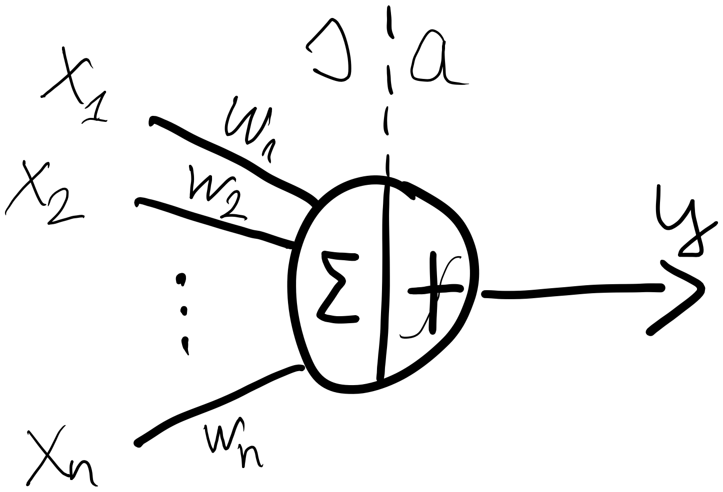
Values are not centered around 0.

$$y\_std = (y - \min y) / (\max y - \min y)$$

$$y\_scaled = y\_std \cdot (max - min) + min \Big|_{max=1,\ min=0}$$

Backpropagation algorithm

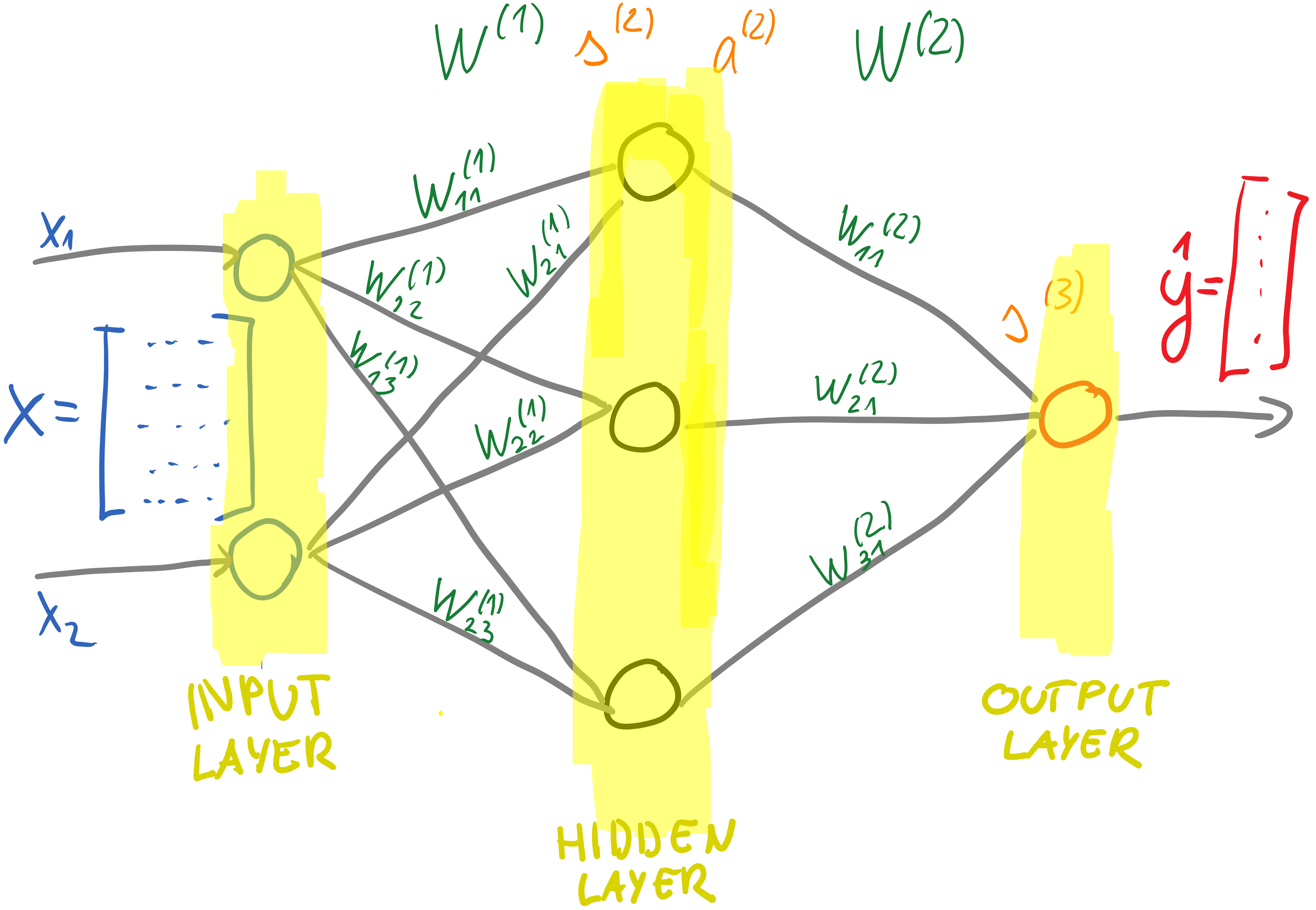
Single neuron with non-linear activation



$$s = \sum_{i=1}^{n} w_i x_i$$

$$y = a(s)$$

Fully connected feedforward neural network

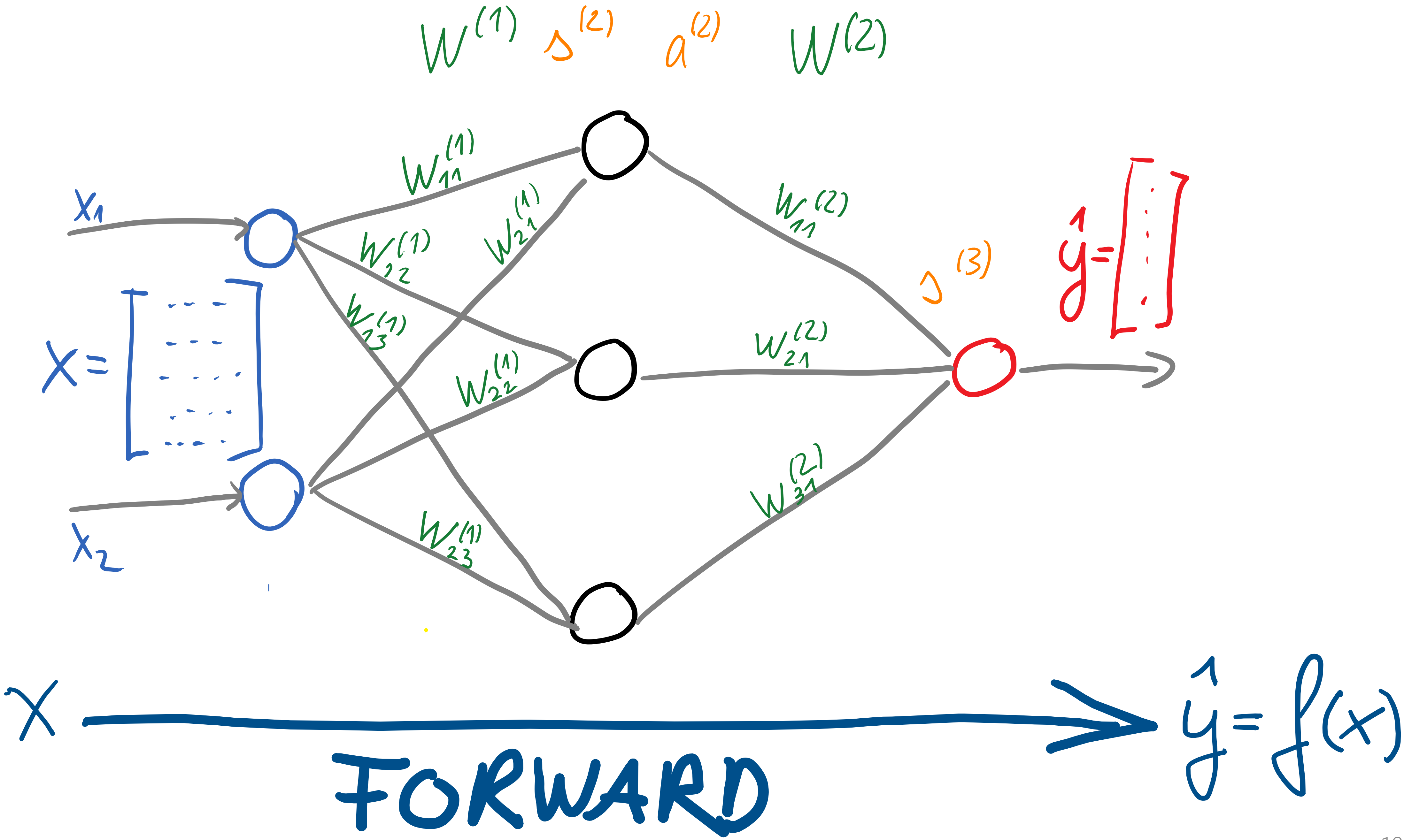


Matrix algebra

- Matrix multiplication
- Transpose
- Partial derivative

NumPy TensorFlow Chainer

MATLAB PYTORCH mxnet


W(1)
a(2)
W(2)
x1
x2
X =
ŷ =
X
FORWARD
ŷ = f(x)

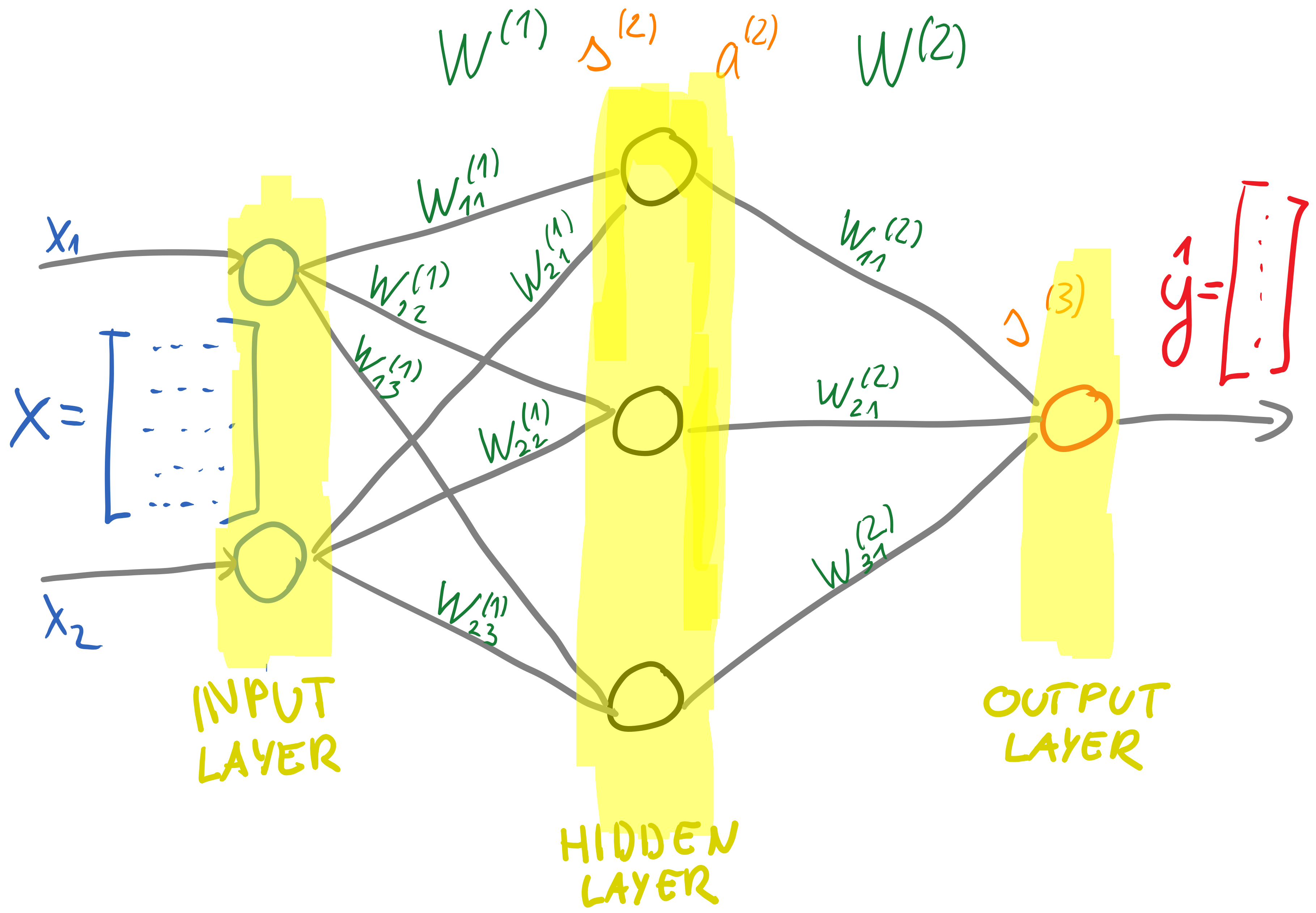
Forward propagation – step 1

① $\underset{(4\times 2)}{X}\ \underset{(2\times 3)}{W^{(1)}} = \underset{(4\times 3)}{z^{(2)}}$

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} \\ x_1^{(2)} & x_2^{(2)} \\ x_1^{(3)} & x_2^{(3)} \\ x_1^{(4)} & x_2^{(4)} \end{bmatrix}$$

minták száma

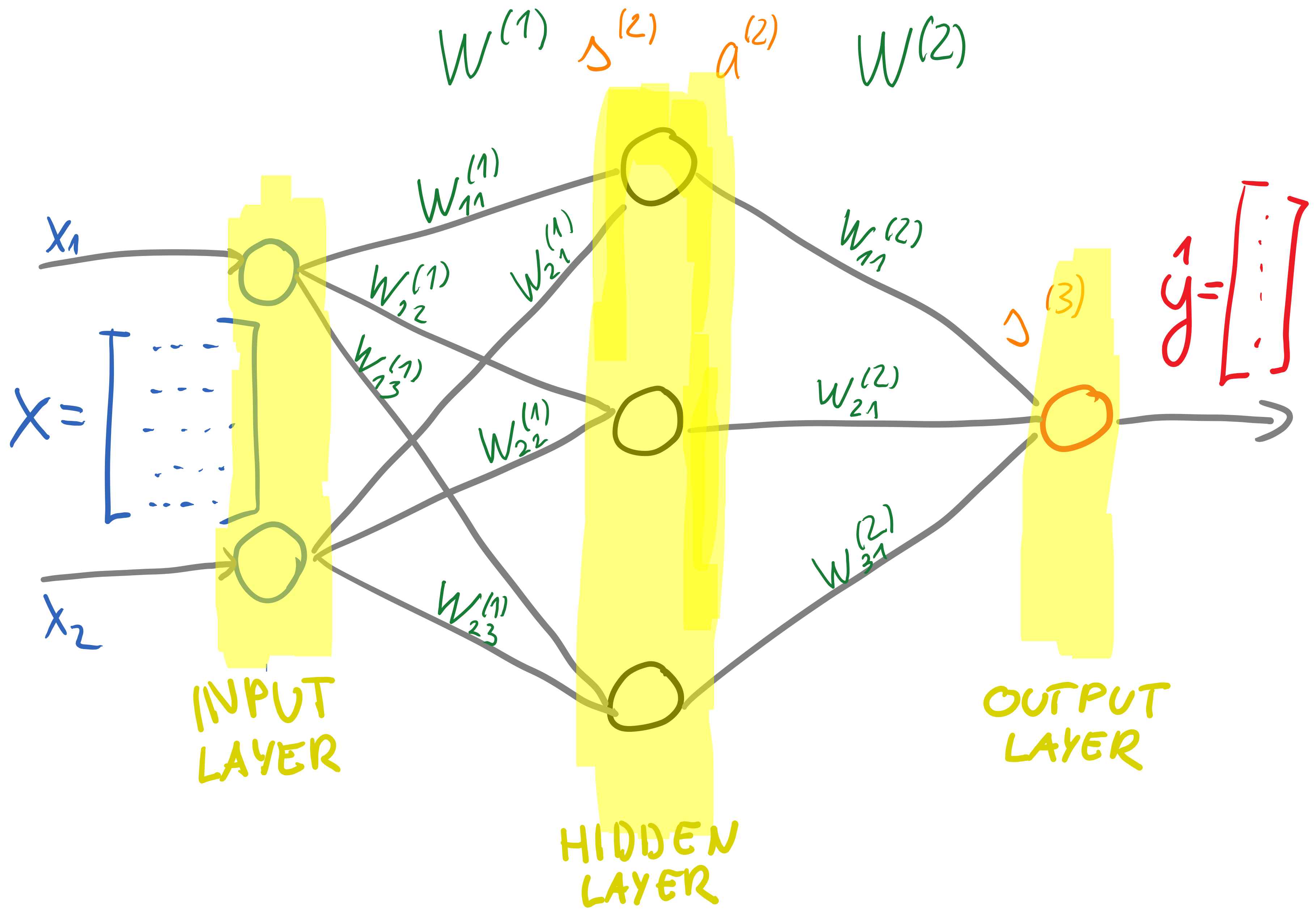
tulajdonságok (features)



W(1)
z(2)
a(2)
W(2)
x1
x2
X=
W11(1)
W12(1)
W21(1)
W13(1)
W22(1)
W23(1)
W11(2)
W21(2)
W31(2)
z(3)
ŷ=
INPUT LAYER
HIDDEN LAYER
OUTPUT LAYER

Forward propagation – step 2

② $a^{(2)} = f(s^{(2)}) = \text{Sigmoid}\{s^{(2)}\}$ (tanh/ReLU/PReLU...)



W(1)
z(2)
a(2)
W(2)
x1
x2
X=
W11(1)
W12(1)
W13(1)
W21(1)
W22(1)
W23(1)
W11(2)
W21(2)
W31(2)
z(3)
ŷ=
INPUT LAYER
HIDDEN LAYER
OUTPUT LAYER

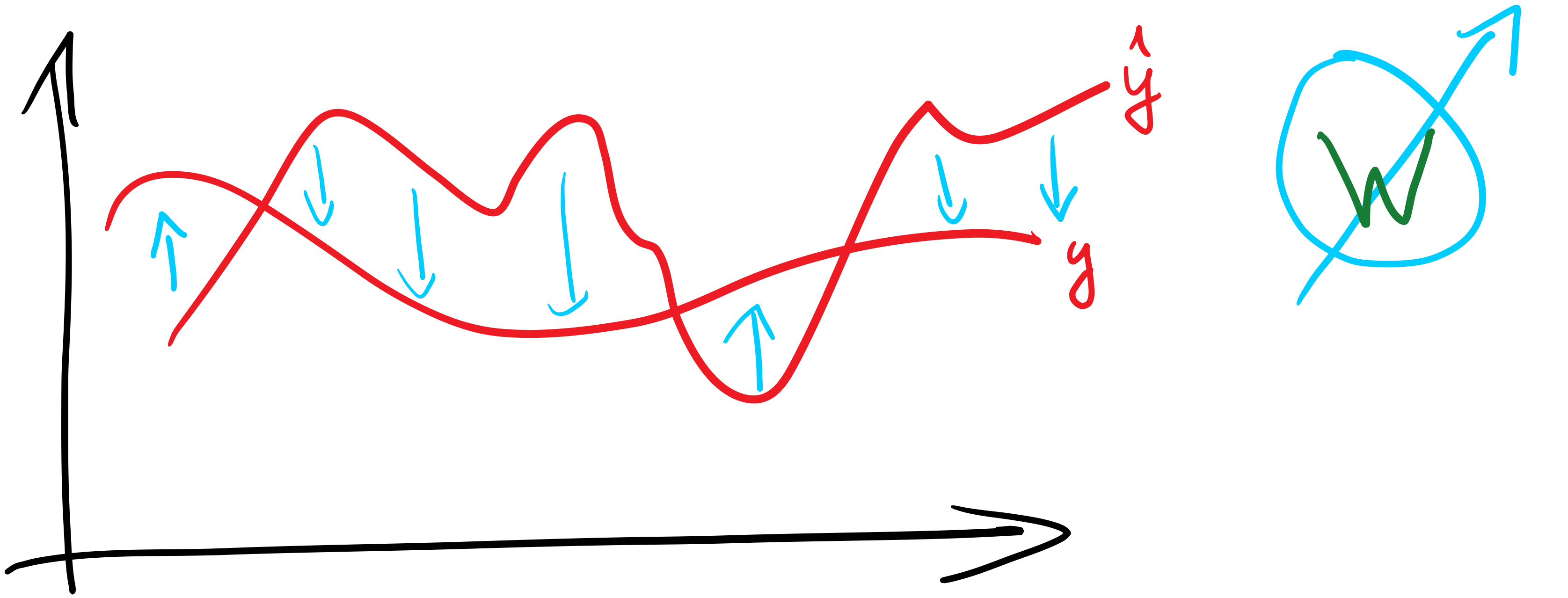
Forward – step 3 Loss/cost/error function

For example:

- Mean squarred error
- Cross entropy

⑤

$$C = \sum \frac{1}{2} (y - \hat{y})^2$$



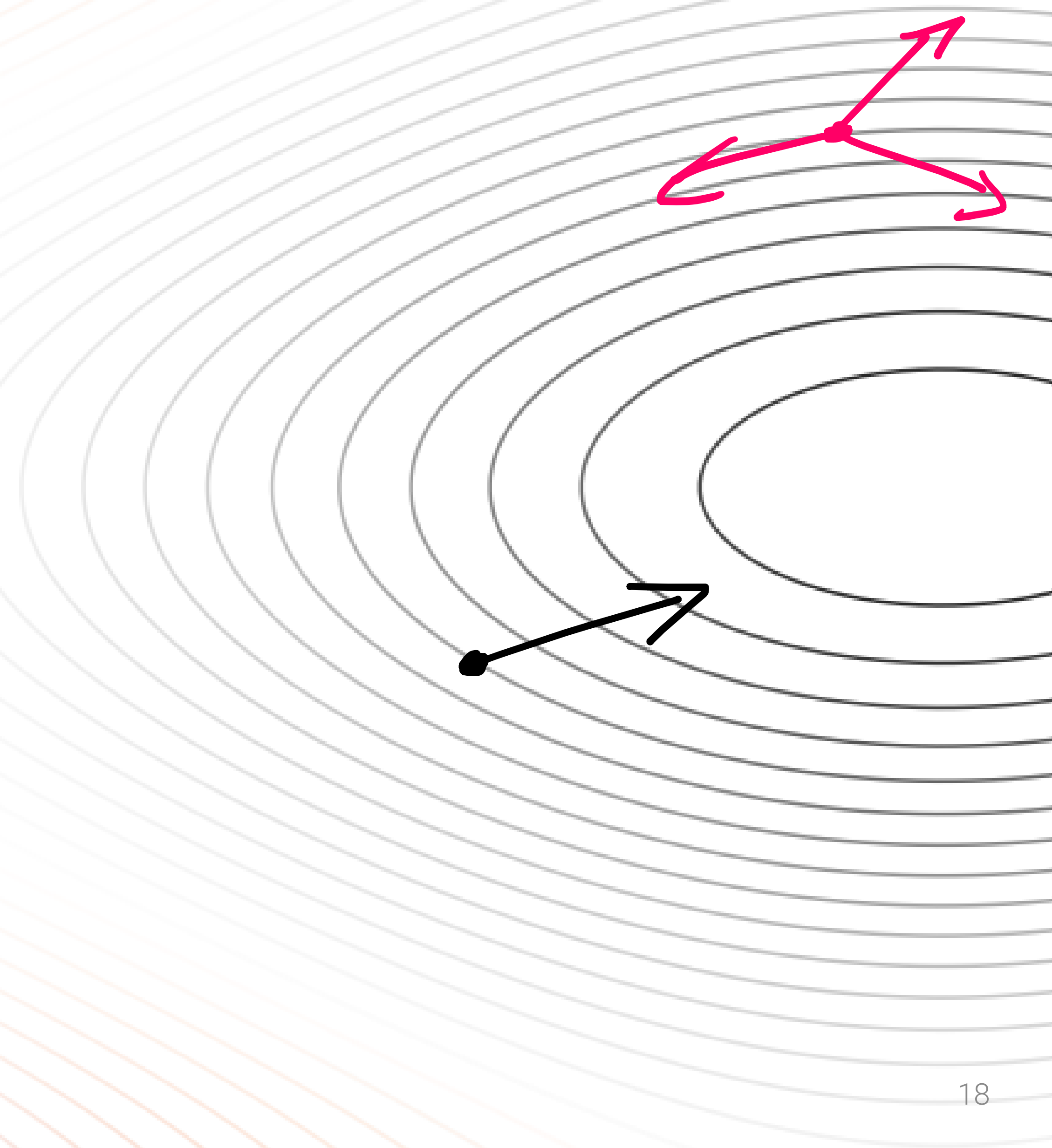
Loss function optimization

- Random search
- Numerical differentiation

$$g(\theta) \approx \frac{C(\theta+\varepsilon) - C(\theta-\varepsilon)}{2\varepsilon}$$

- Gradient descent

$$-\frac{\partial C}{\partial \theta}$$

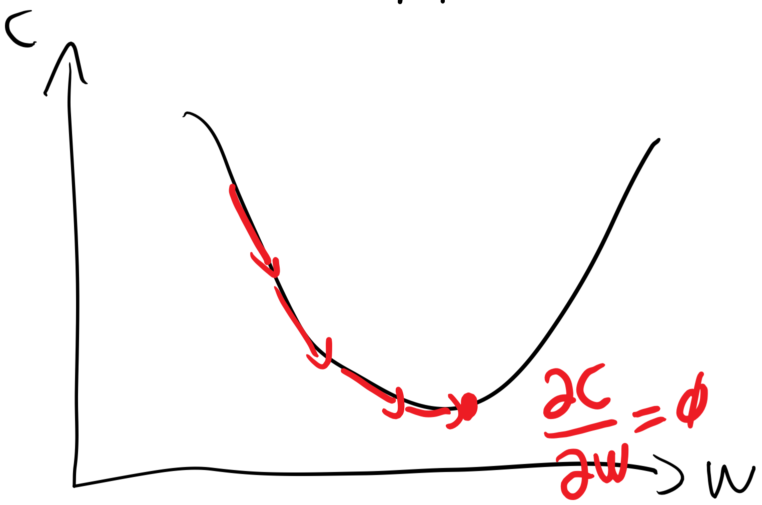


Gradient descent

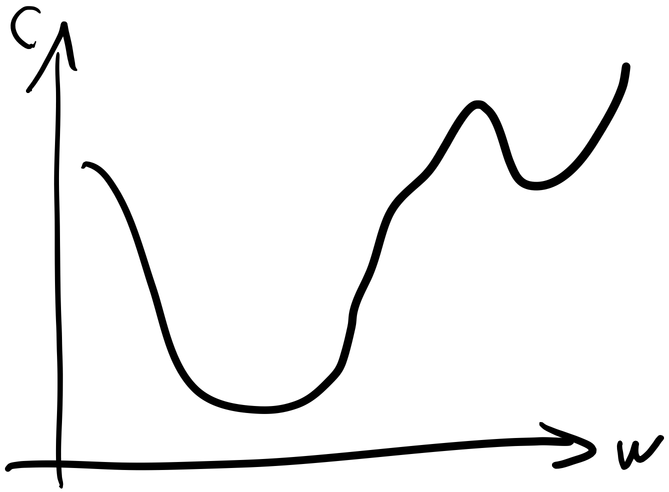
(6) $C = \sum \left\{ \frac{1}{2}\left(y - f\left(f\left(XW^{(1)}\right)W^{(2)}\right)\right)^2 \right\}$

$\frac{\partial C}{\partial W}$ $> \emptyset$ ↗ $< \emptyset$ ↘

GOAL: approach $\emptyset$



Non-convex



Matrix calculus – cheetsheet

y scalar, x vector

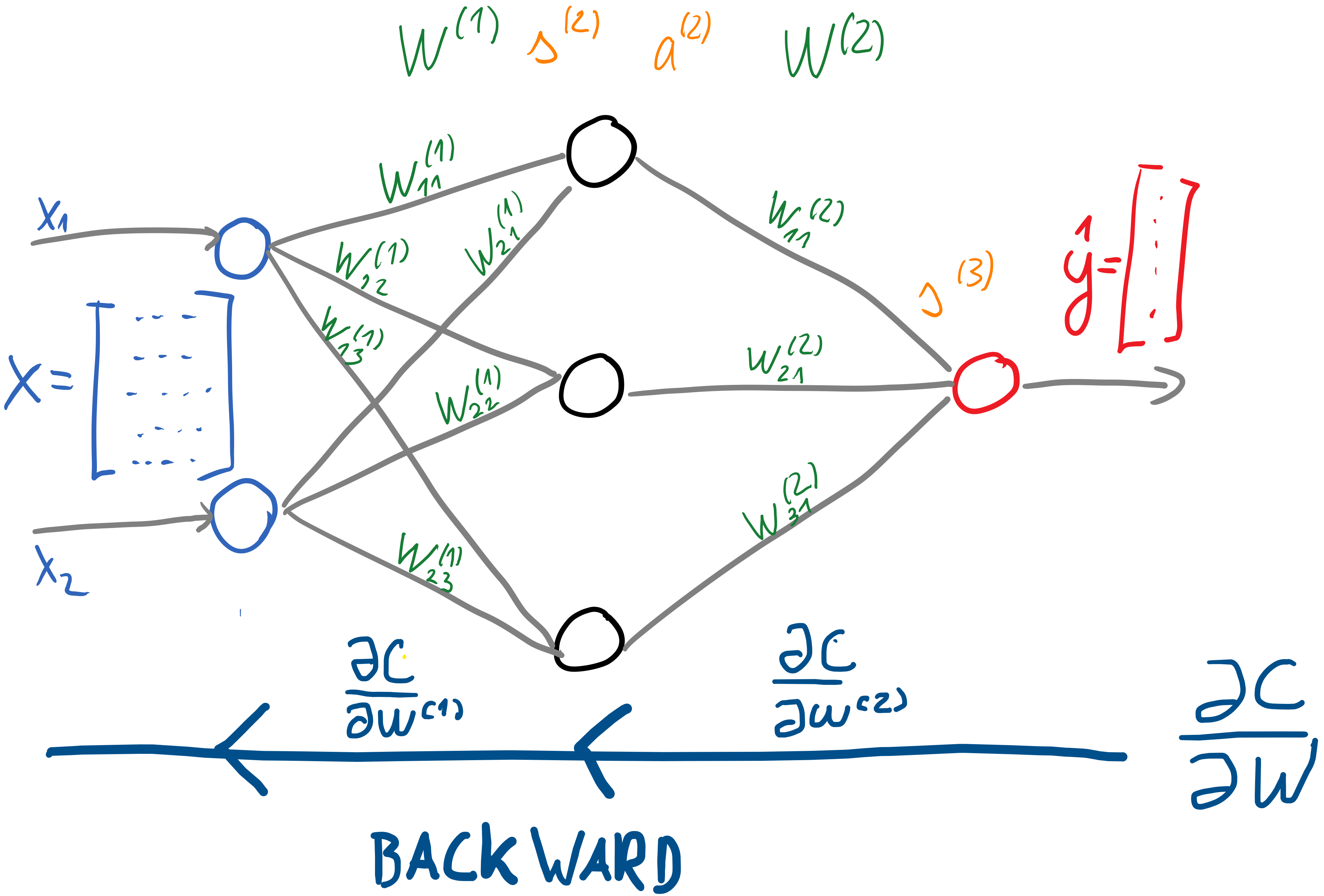
$$\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y}{\partial x_1} & \frac{\partial y}{\partial x_2} & \cdots & \frac{\partial y}{\partial x_n} \end{bmatrix}$$

y vector, x vector

$$\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

y scalar, W matrix

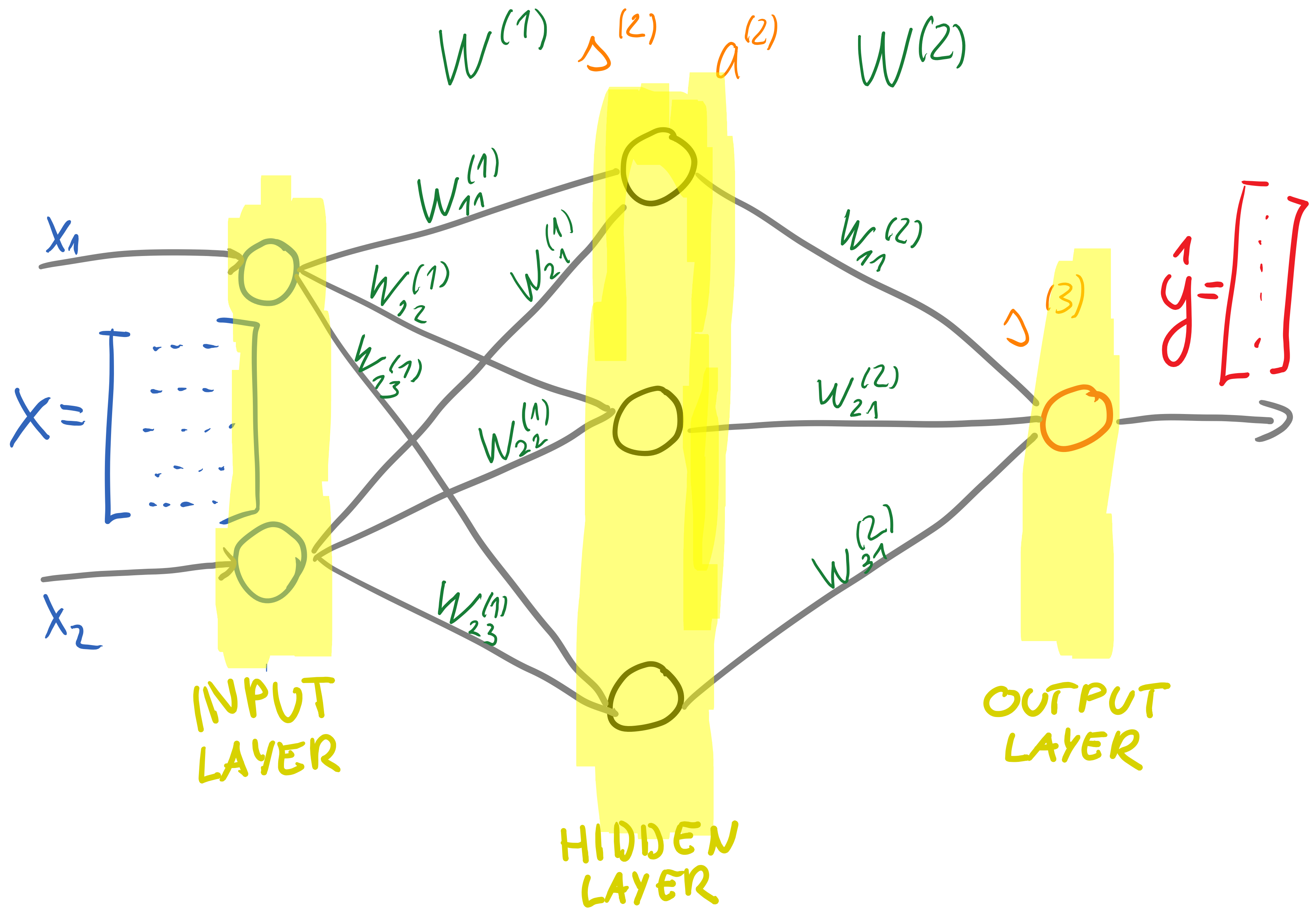
$$\frac{\partial y}{\partial W} = \begin{bmatrix} \frac{\partial y}{\partial W_{11}} & \cdots & \frac{\partial y}{\partial W_{1n}} \\ \vdots & & \vdots \\ \frac{\partial y}{\partial W_{m1}} & \cdots & \frac{\partial y}{\partial W_{mn}} \end{bmatrix}$$



W(1)
z(2)
a(2)
W(2)
x1
x2
X=
W11(1)
W12(1)
W13(1)
W21(1)
W22(1)
W23(1)
W11(2)
W21(2)
W31(2)
z(3)
y=
∂C/∂w(1)
∂C/∂w(2)
∂C/∂w
BACKWARD

Backward propagation – step 1

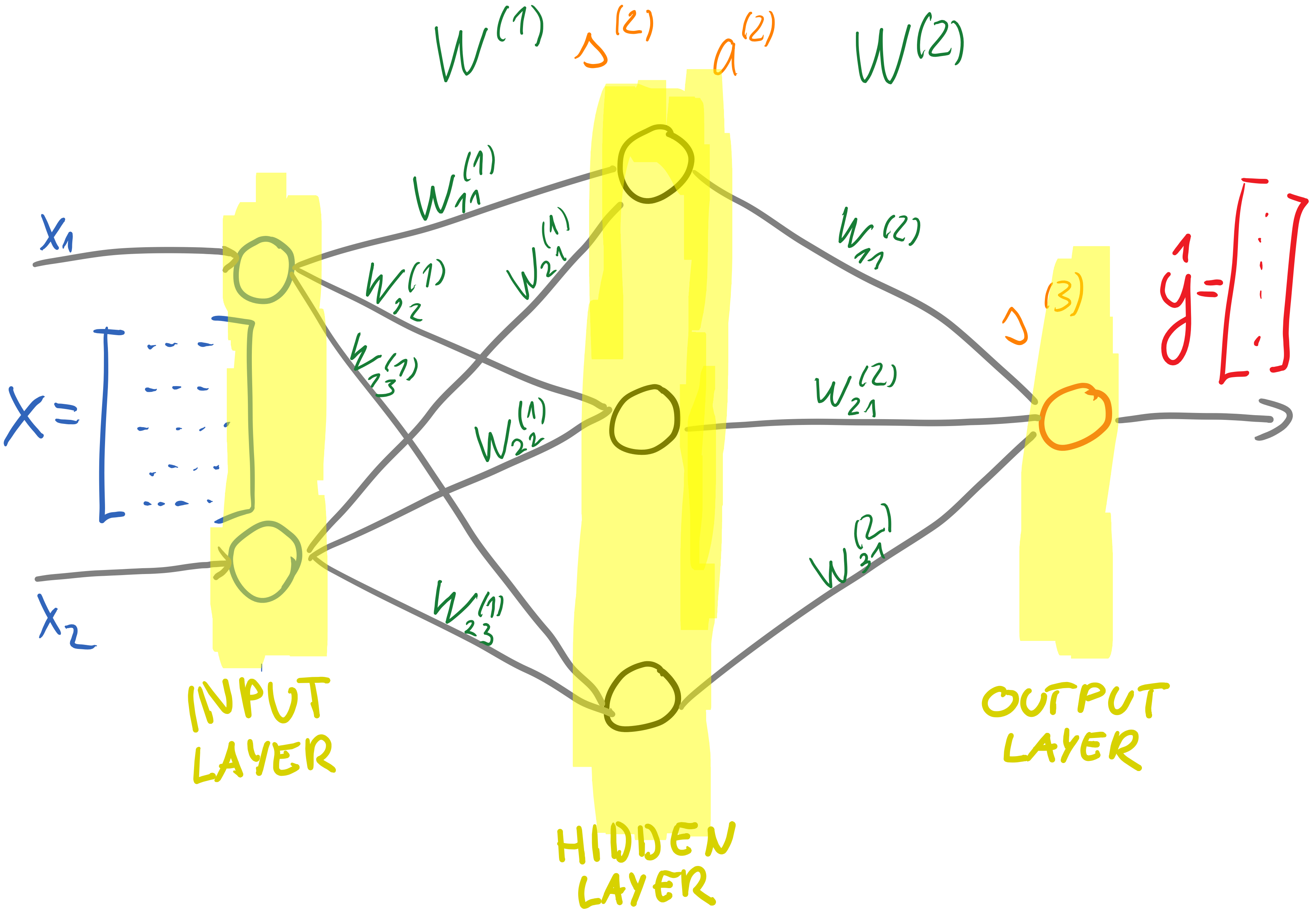
$$\frac{\partial C}{\partial W^{(2)}} = \frac{\partial \sum \frac{1}{2}(y-\hat{y})^2}{\partial W^{(2)}}$$



W(1)
z(2)
a(2)
W(2)
x1
x2
X=
ŷ=
z(3)
INPUT LAYER
HIDDEN LAYER
OUTPUT LAYER

Backward – step 2: batch gradient descent

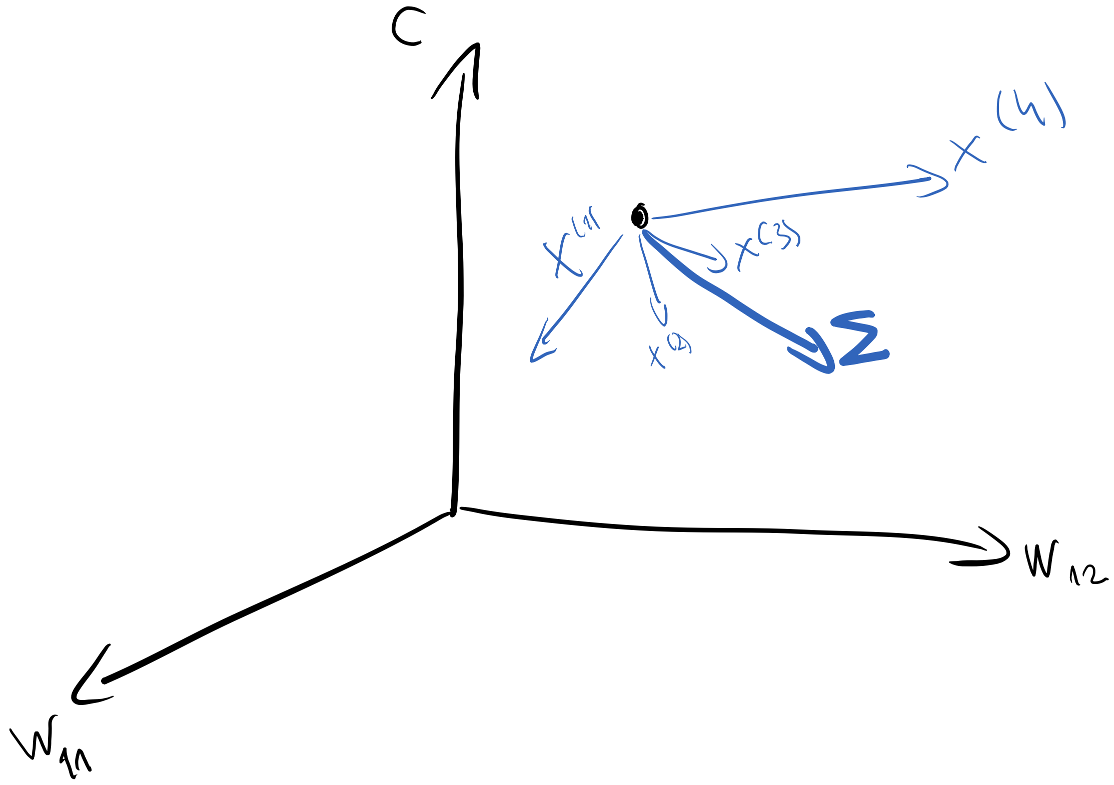
$$⑦ \quad \underset{(3\times 1)}{\frac{\partial C}{\partial W^{(2)}}} = \sum \frac{\partial \frac{1}{2}(y-\hat{y})^2}{\partial W^{(2)}} = \underset{(3\times 4)}{(a^{(2)})^T} \underset{(4\times 1)}{\delta^{(3)}} =$$



$W^{(1)}$
$z^{(2)}$
$a^{(2)}$
$W^{(2)}$
x_1
x_2
$X =$
$W_{11}^{(1)}$
$W_{12}^{(1)}$
$W_{13}^{(1)}$
$W_{21}^{(1)}$
$W_{22}^{(1)}$
$W_{23}^{(1)}$
$W_{11}^{(2)}$
$W_{21}^{(2)}$
$W_{31}^{(2)}$
$z^{(3)}$
$\hat{y} =$
INPUT LAYER
HIDDEN LAYER
OUTPUT LAYER

Backward – step 2: batch gradient descent

Calculate the gradient for all the training samples and calculate the sum or mean of it.



Backward – step 3

$$\frac{\partial C}{\partial W^{(1)}} = \frac{\partial \frac{1}{2}(y-\hat{y})^2}{\partial W^{(1)}}$$

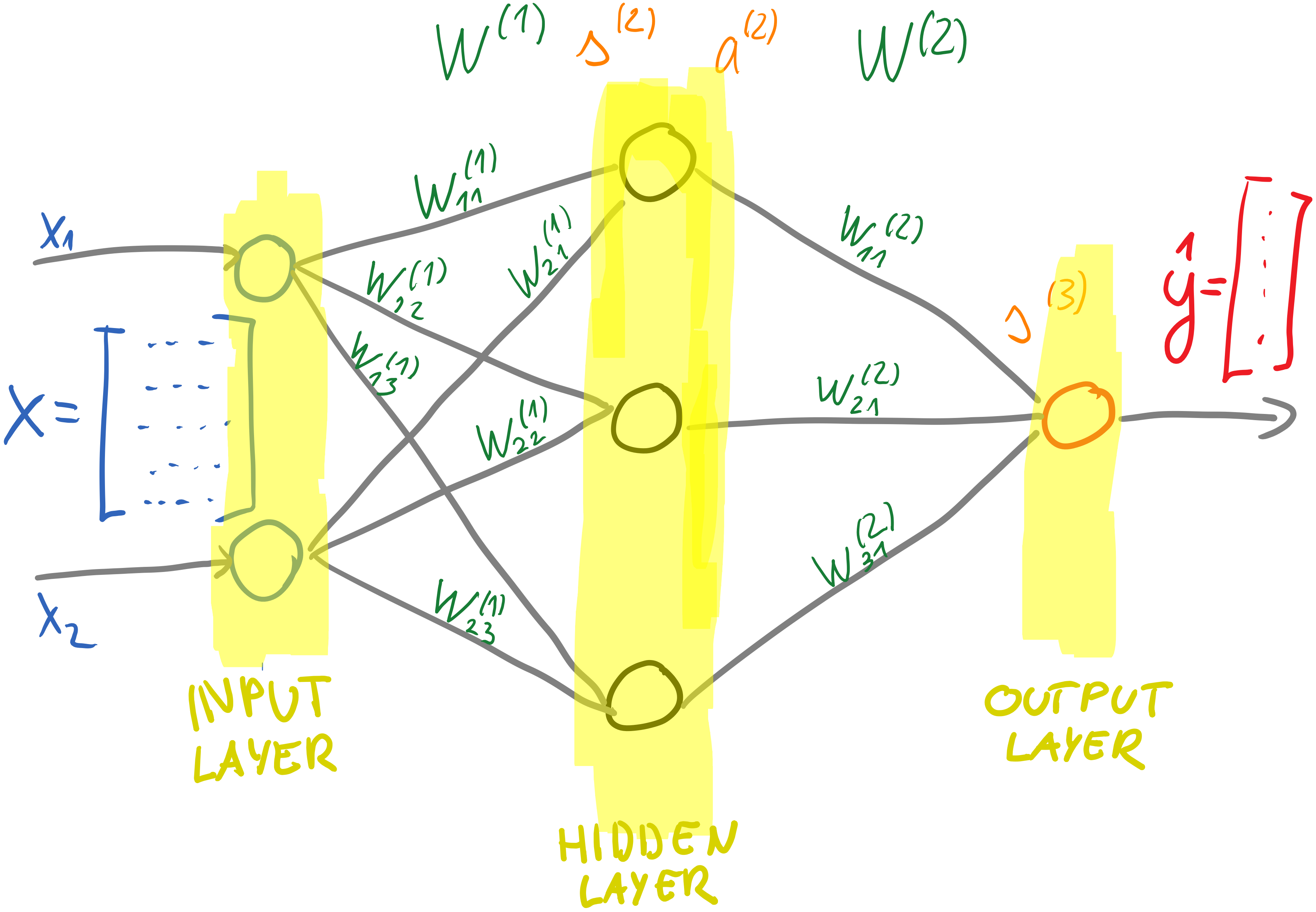
TRAINING

$$W^{(1)} = W^{(1)} - \mu \frac{\partial C}{\partial W^{(1)}}$$

$$W^{(2)} = W^{(2)} - \mu \frac{\partial C}{\partial W^{(2)}}$$

μ: learning rate

Fully connected feedforward neural network

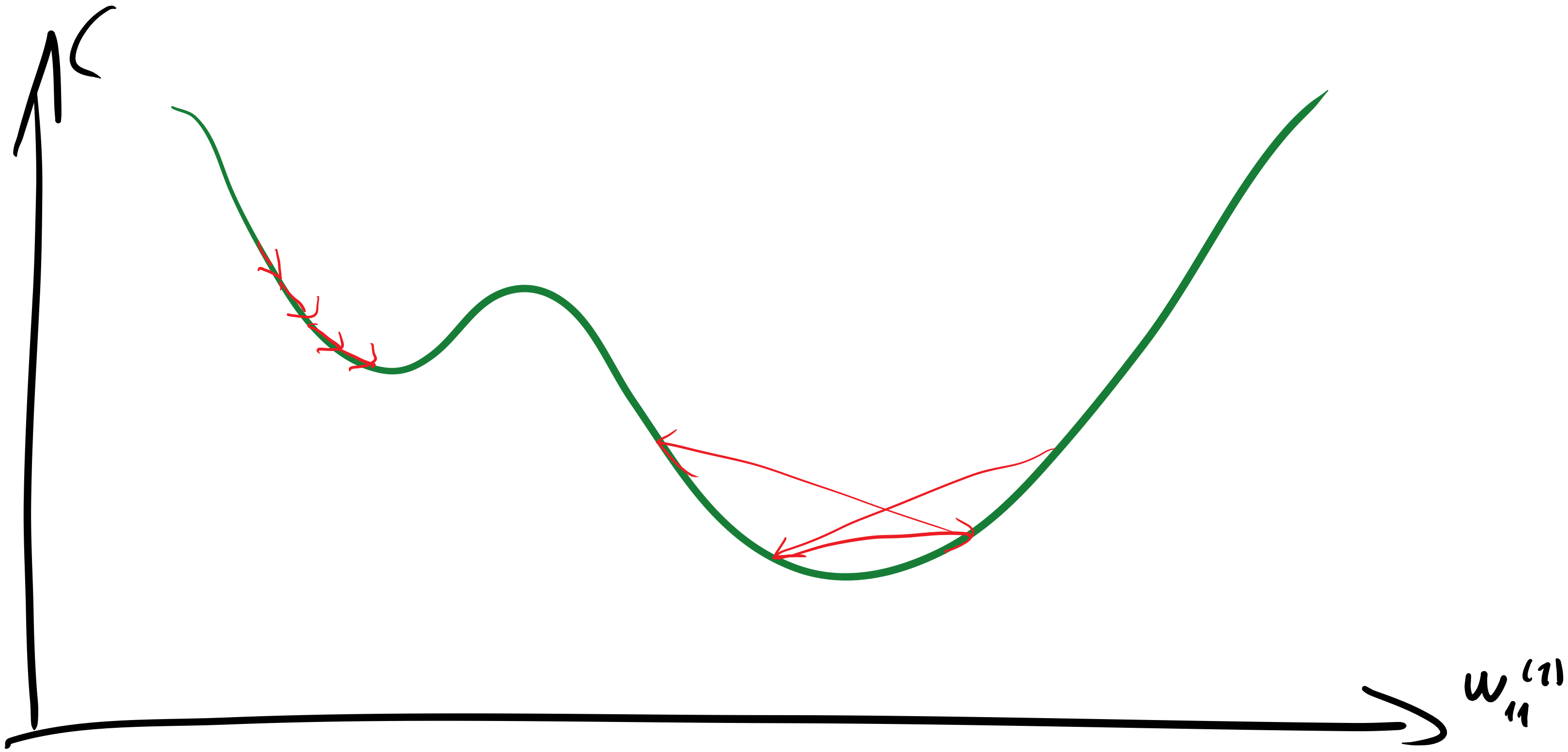


Gradient descent

- **Batch gradient descent:** using all the training examples
- **Stochastic Gradient Descent (SGD):** using one sample or a subset (called mini-batch)

Gradient descent

Local vs. global minimum



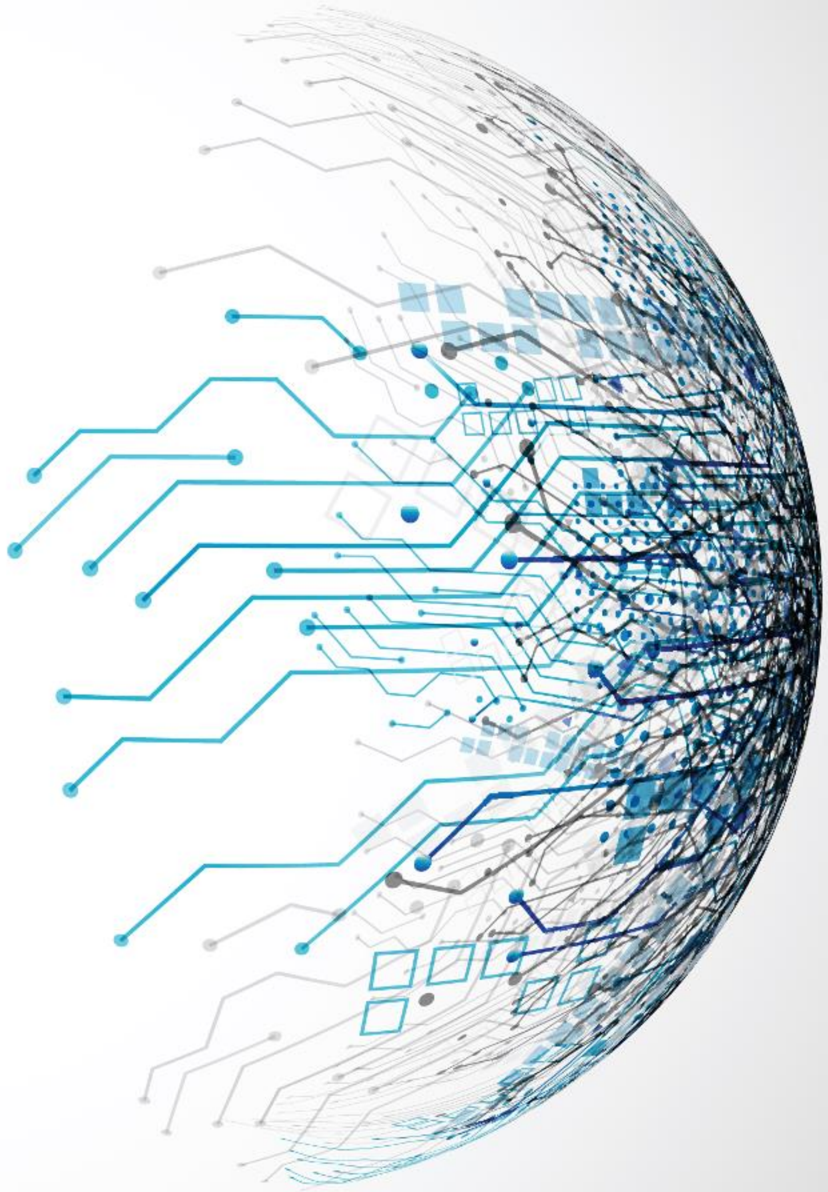
More information

- Yann LeCunn: Efficient Backprop (1998)
  http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf
- Description and optimization on artificial error surfaces:
  https://www.deeplearning.ai/ai-notes/optimization/
- Andrej Karpathy: Yes you should understand backprop
  https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b

Please, don’t forget to send feedback:

https://bit.ly/bme-dl

Thank you
for your attention



Dr. Mohammed Salah Al-Radhi

malradhi@tmit.bme.hu

(slides by: Dr. Bálint Gyires-Tóth)

10 September 2024