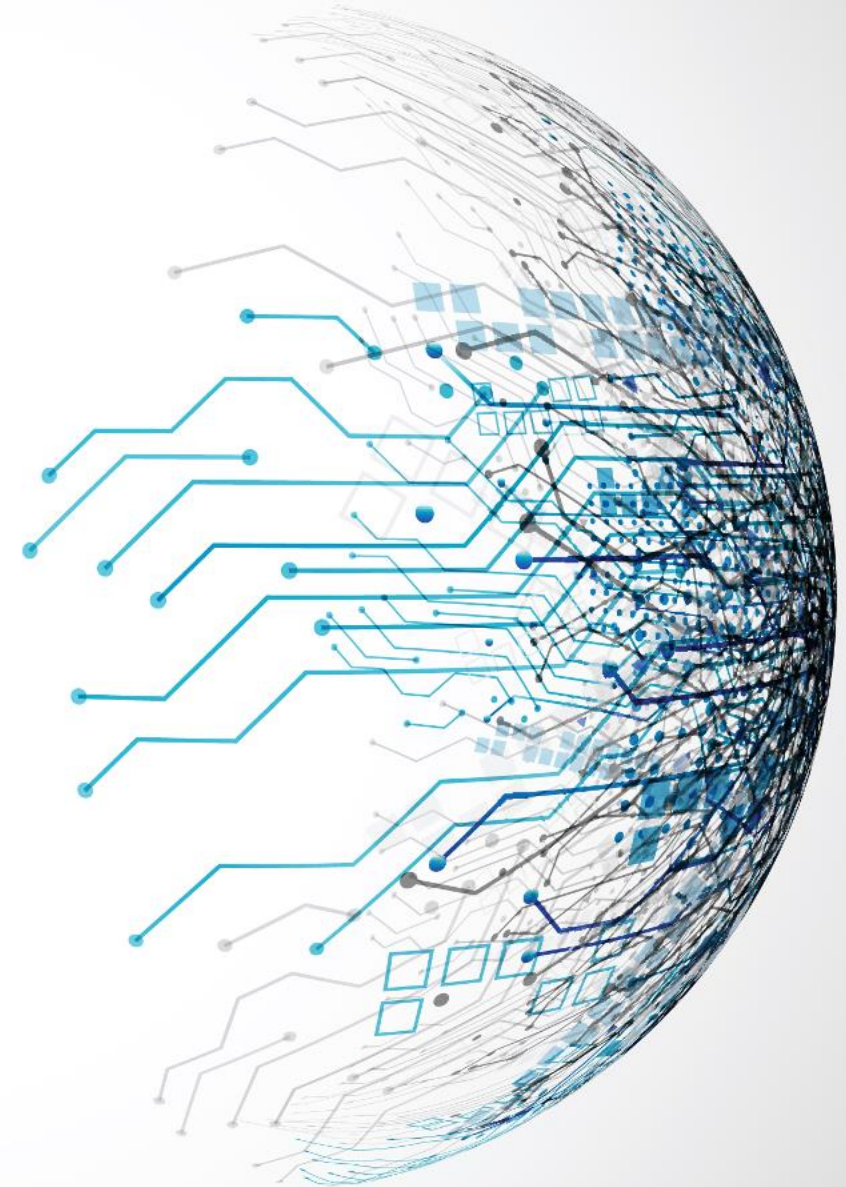


# Deep Learning

# Backpropagation

Dr. Mohammed Salah Al-Radhi  
(slides by: Dr. Bálint Gyires-Tóth)



# Copyright

Copyright © **Bálint Gyires-Tóth & Mohammed Salah Al-Radhi**, All Rights Reserved.

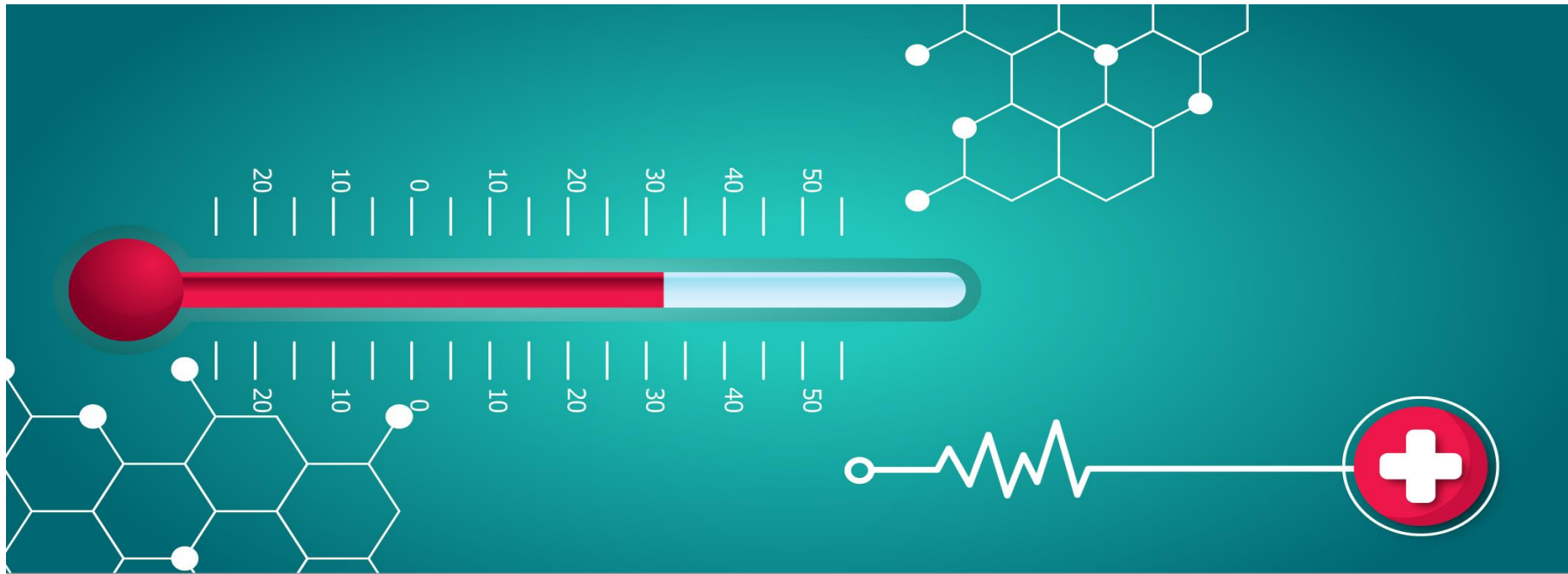
This presentation and its contents are protected by copyright law. The intellectual property contained herein, including but not limited to text, images, graphics, and design elements, are the exclusive property of the copyright holder identified above. Any unauthorized use, reproduction, distribution, or modification of this presentation or its contents is strictly prohibited without prior written consent from the copyright holder.

**No Recordings or Reproductions:** Attendees, viewers, and recipients of this presentation are expressly prohibited from making any audio, video, or photographic recordings, as well as screen captures, screenshots, or any form of reproduction, of this presentation, its content, or any related materials, whether during its live presentation or subsequent access. Violation of this prohibition may result in legal action.

For permissions, inquiries, or licensing requests, please contact: **{toth.b,malradhi}@tmit.bme.hu**

Unauthorized use, distribution, or reproduction of this presentation may result in civil and criminal penalties. Thank you for respecting the intellectual property rights of the copyright holder.

# Task



- Input: [temperature (°C), medicine (mg)] =  $X$
- Output:
  - Regression: [temperature in 2 hours] =  $\hat{y}$
  - Classification: [fever/normal within 2 hours]

# Train and test datasets

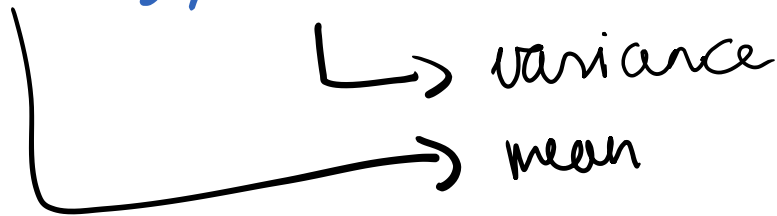
$$X = \begin{bmatrix} 38.6 & 25 \\ 37.8 & 25 \\ 37.9 & 50 \\ 38.2 & 50 \end{bmatrix}$$

$$y = \begin{bmatrix} 37.2 \\ 37.3 \\ 36.6 \\ 36.9 \end{bmatrix}$$

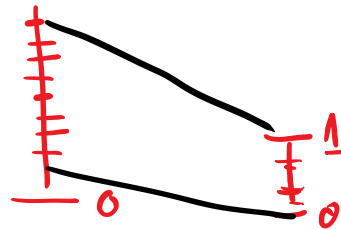
$$X_{\text{test}} = \begin{bmatrix} 38.3 & 35 \end{bmatrix}$$

$\hat{y} = ?$

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



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

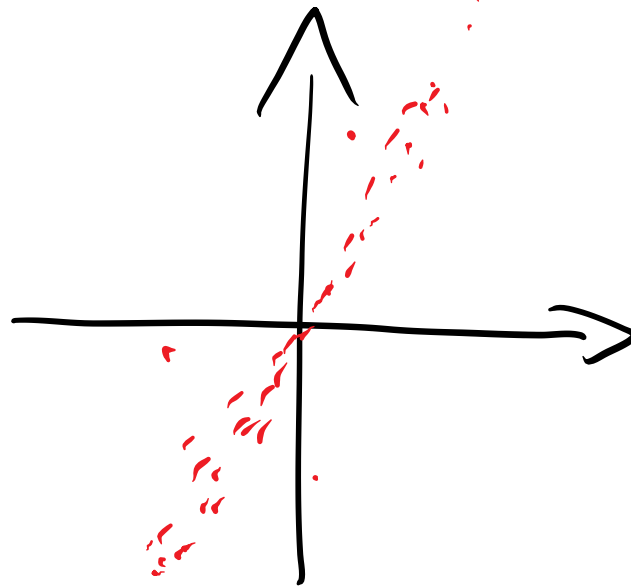
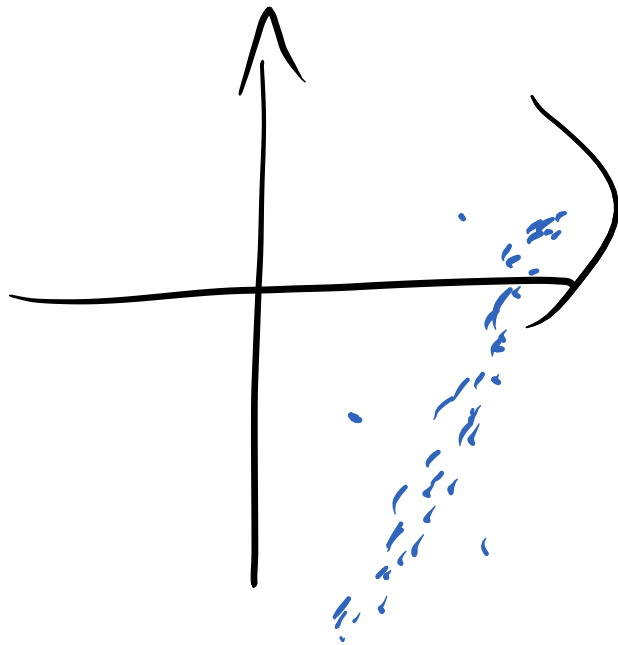


# Standardization

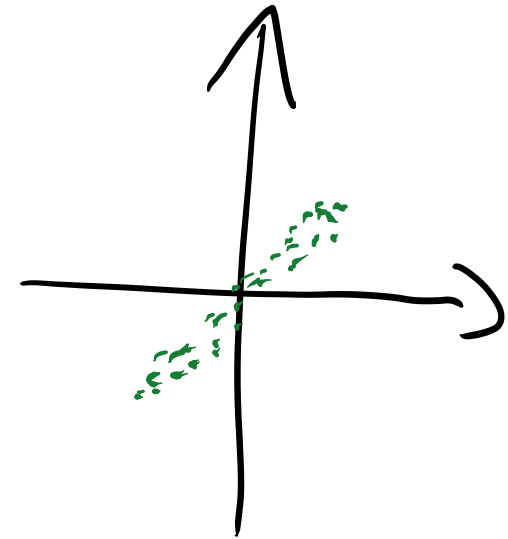
To have 0 mean, 1 variance

In practice: subtracting the mean and dividing by the variance

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



0 mean



0 mean, 1 variance

# 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.

$$y_{std} = (y - \min y) / (\max y - \min y)$$
$$y_{scaled} = y_{std} \cdot (\max - \min) + \min$$

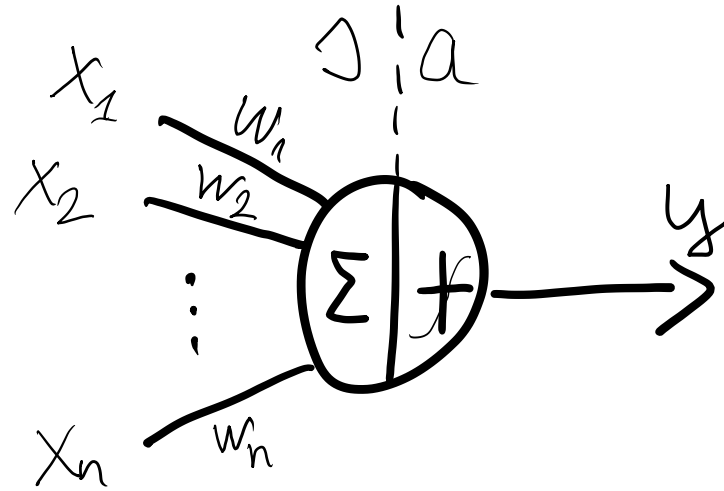
|  $\max = 1$   
 $\min = 0$



# Backpropagation algorithm



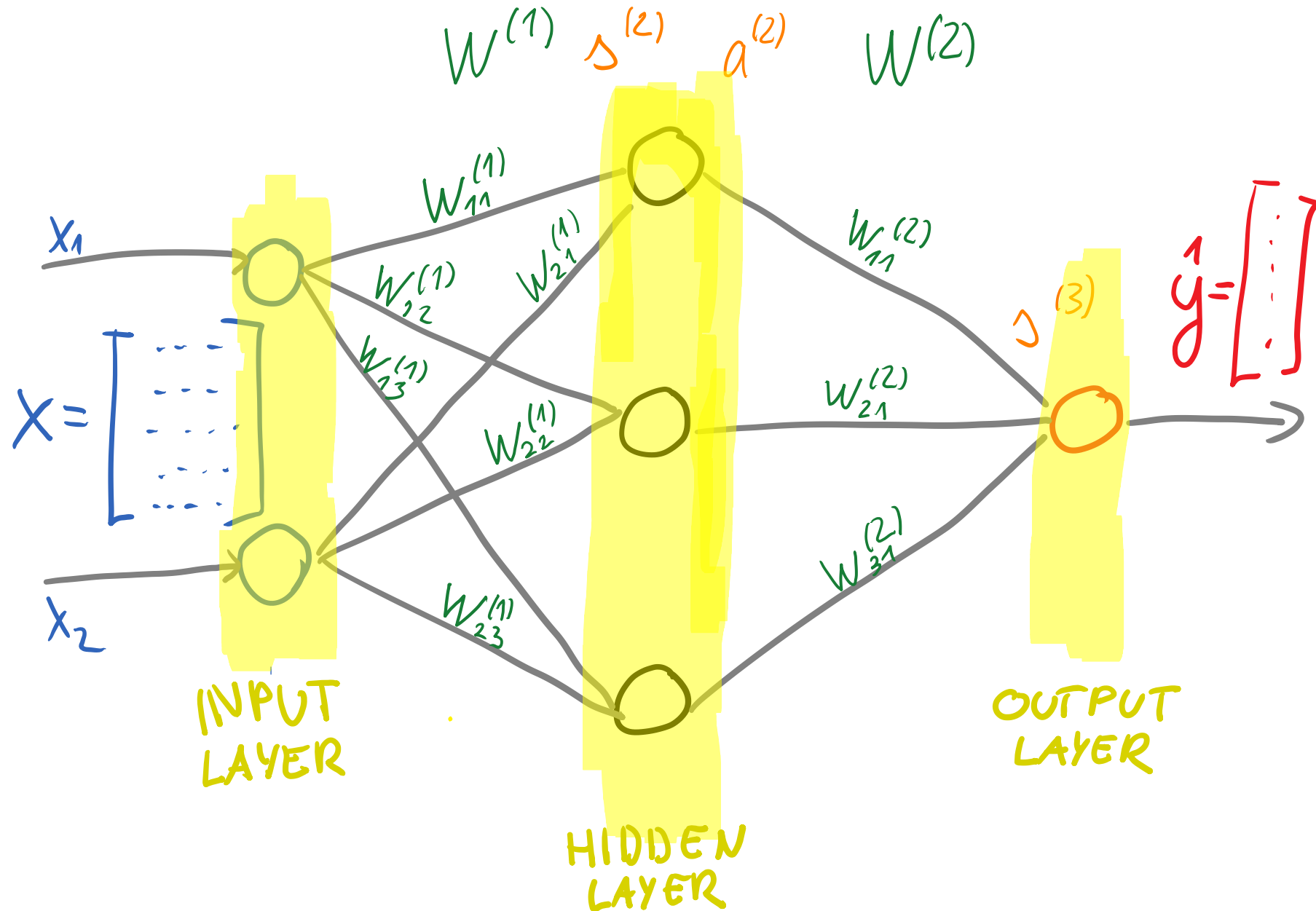
# Single neuron with non-linear activation



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

$$y = a(\Delta)$$

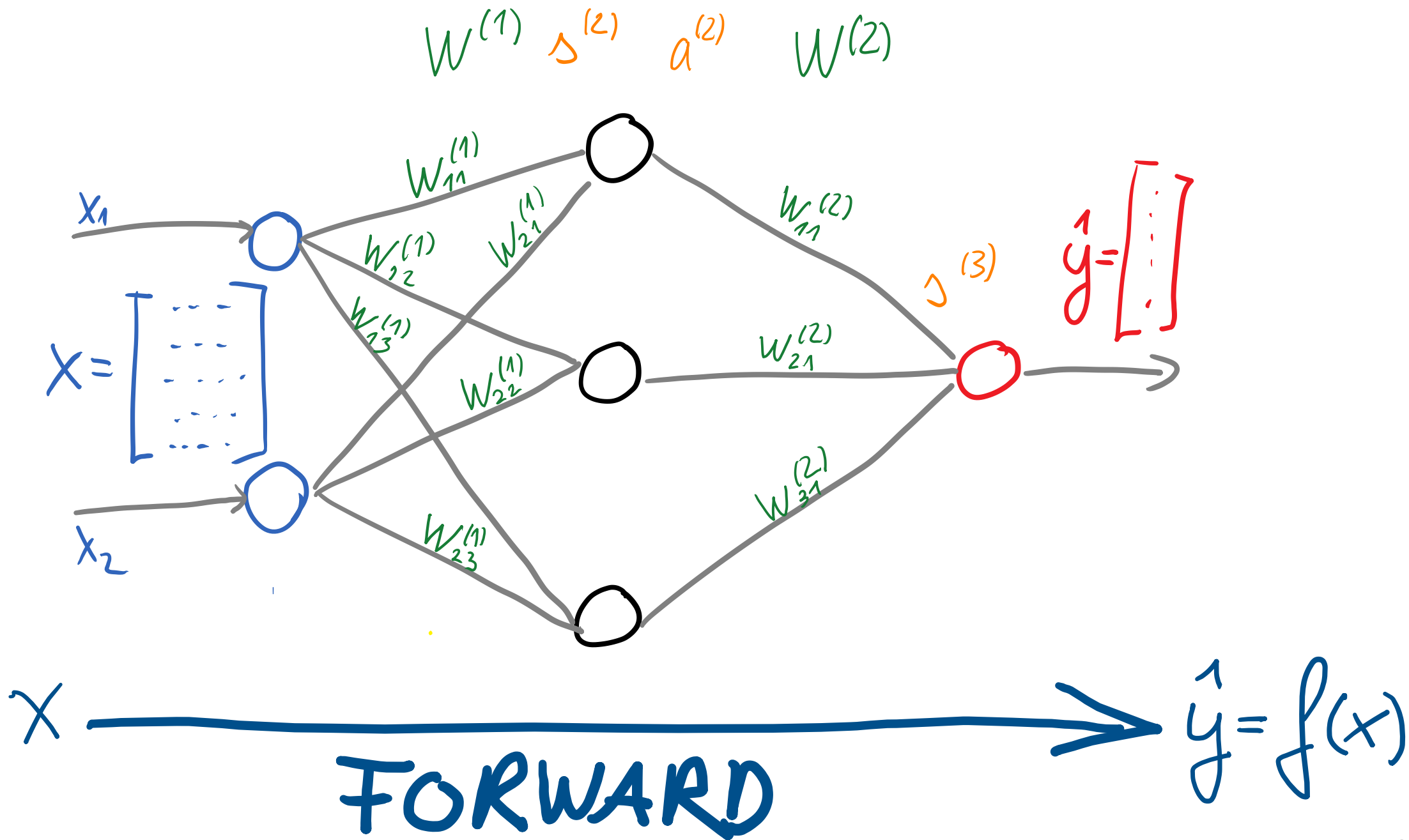
# Fully connected feedforward neural network



# Matrix algebra

- Matrix multiplication
- Transpose
- Partial derivative





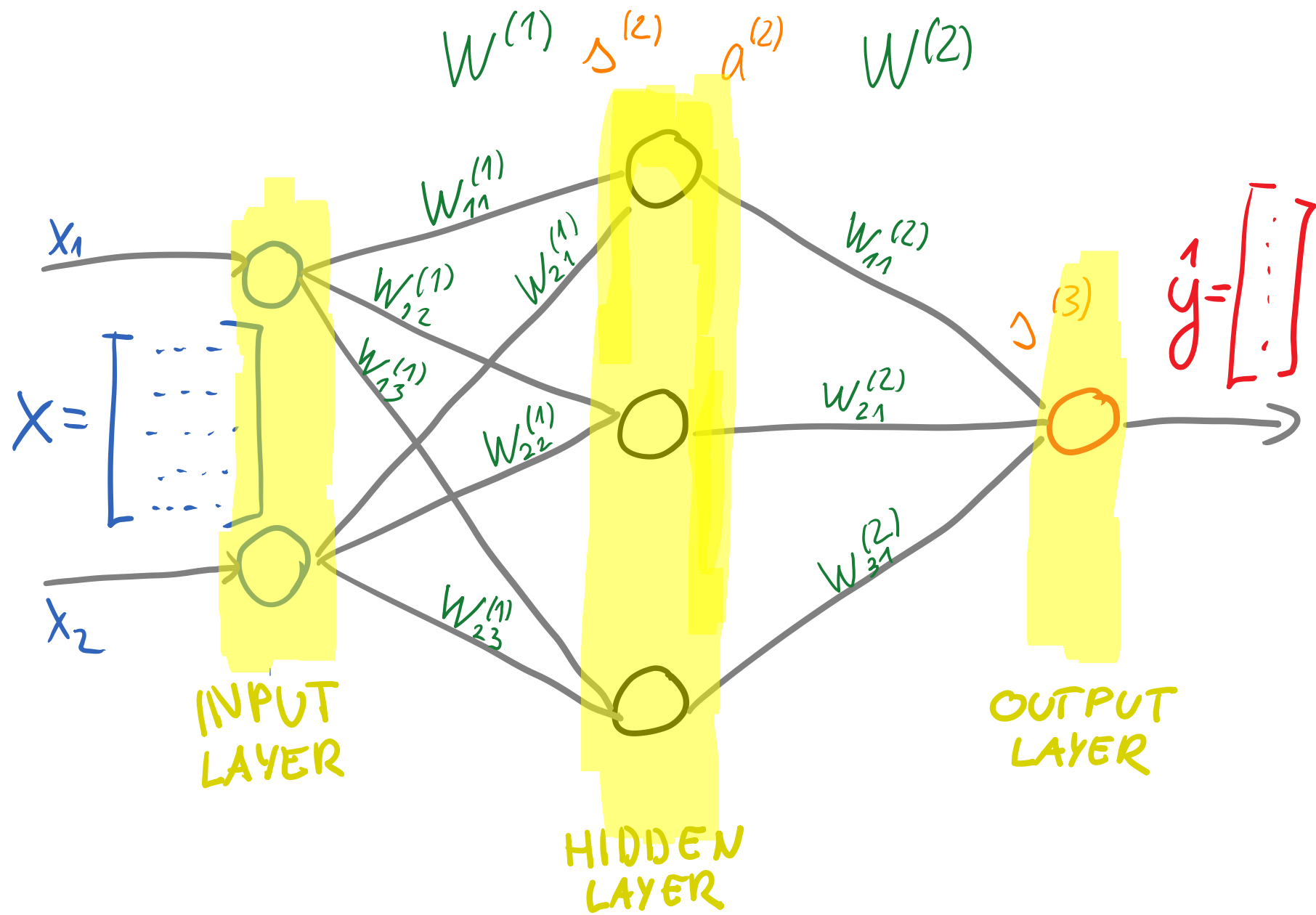
# Forward propagation – step 1

$$\textcircled{1} \quad \overset{(4 \times 2)}{X} \overset{(2 \times 3)}{W^{(1)}} = \overset{(4 \times 3)}{\Delta^{(2)}}$$

mint áe zámng

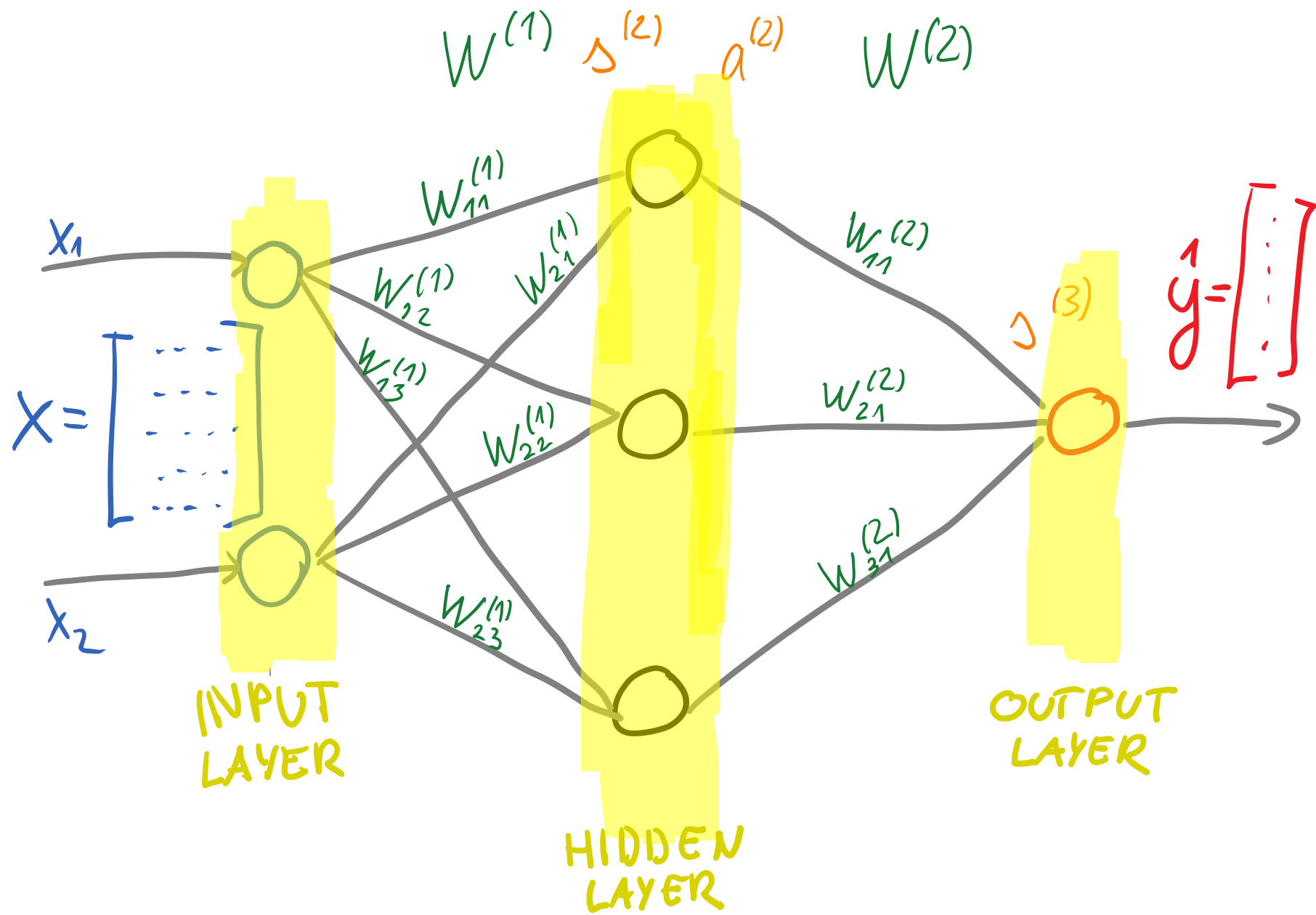
$$X = \begin{bmatrix} X_1^{(1)} & X_2^{(1)} \\ X_1^{(2)} & X_2^{(2)} \\ X_1^{(3)} & X_2^{(3)} \\ X_1^{(4)} & X_2^{(4)} \end{bmatrix}$$

tulajdonságok (features)



# Forward propagation – step 2

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



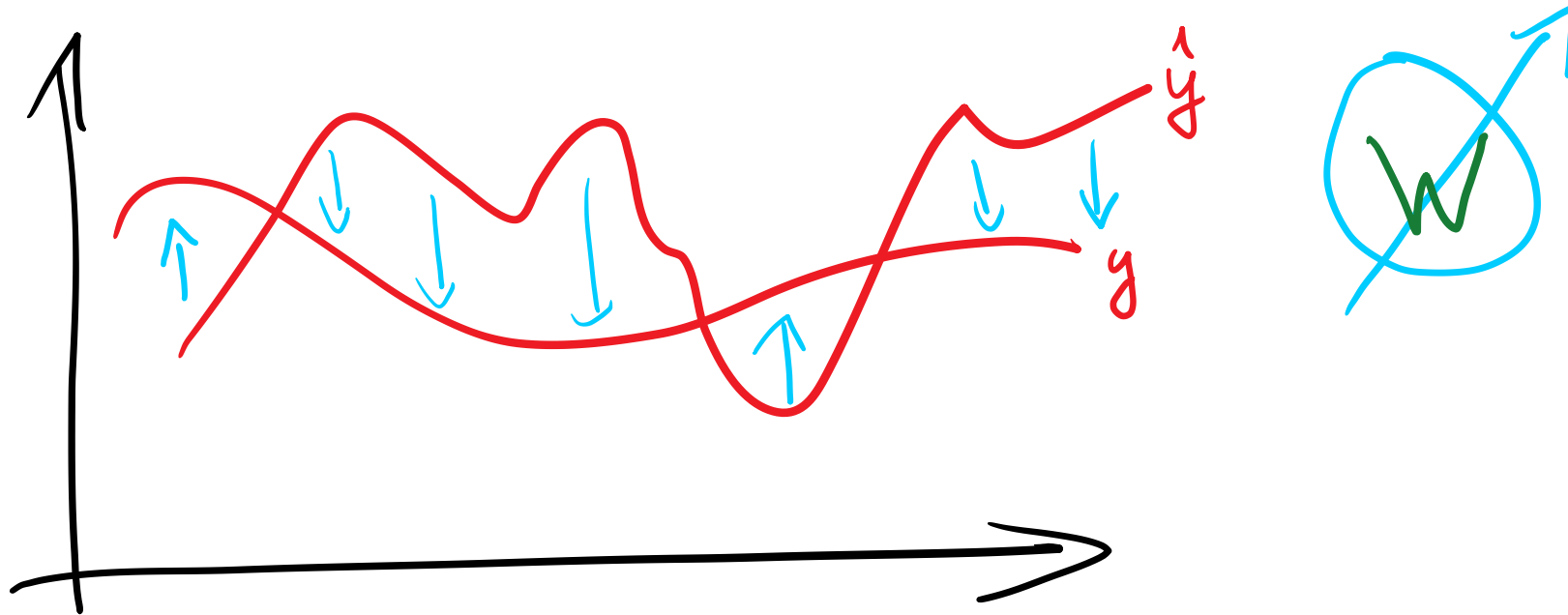


# Forward – step 3 Loss/cost/error function

For example:

- Mean squared error
- Cross entropy

$$\textcircled{5} \quad C = \sum \frac{1}{2} (y - \hat{y})^2$$



# Loss function optimization

- Random search 

- Numerical differentiation

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

- Gradient descent

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

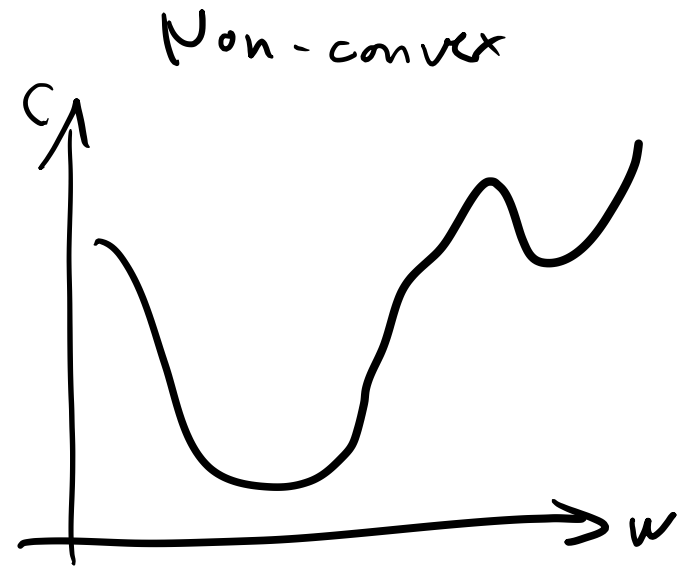
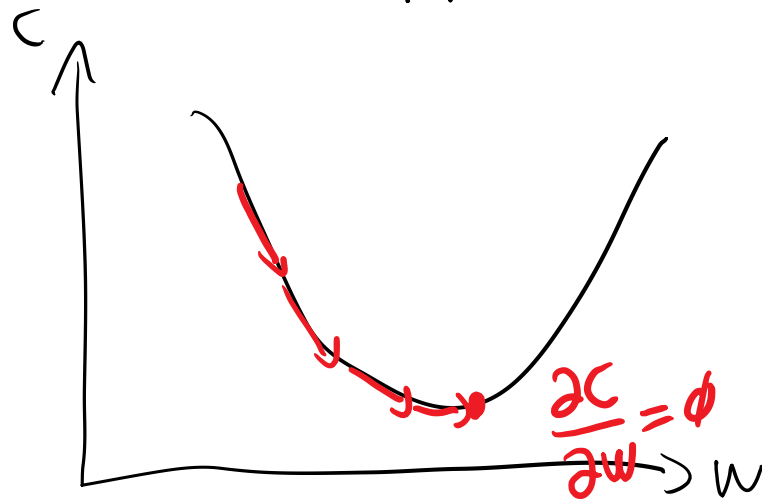


# Gradient descent

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

$$\frac{\partial C}{\partial w} > 0 \nearrow \quad < 0 \searrow$$

GOAL: approach  $\emptyset$



# Matrix calculus – cheatsheet

y scalar, x vector

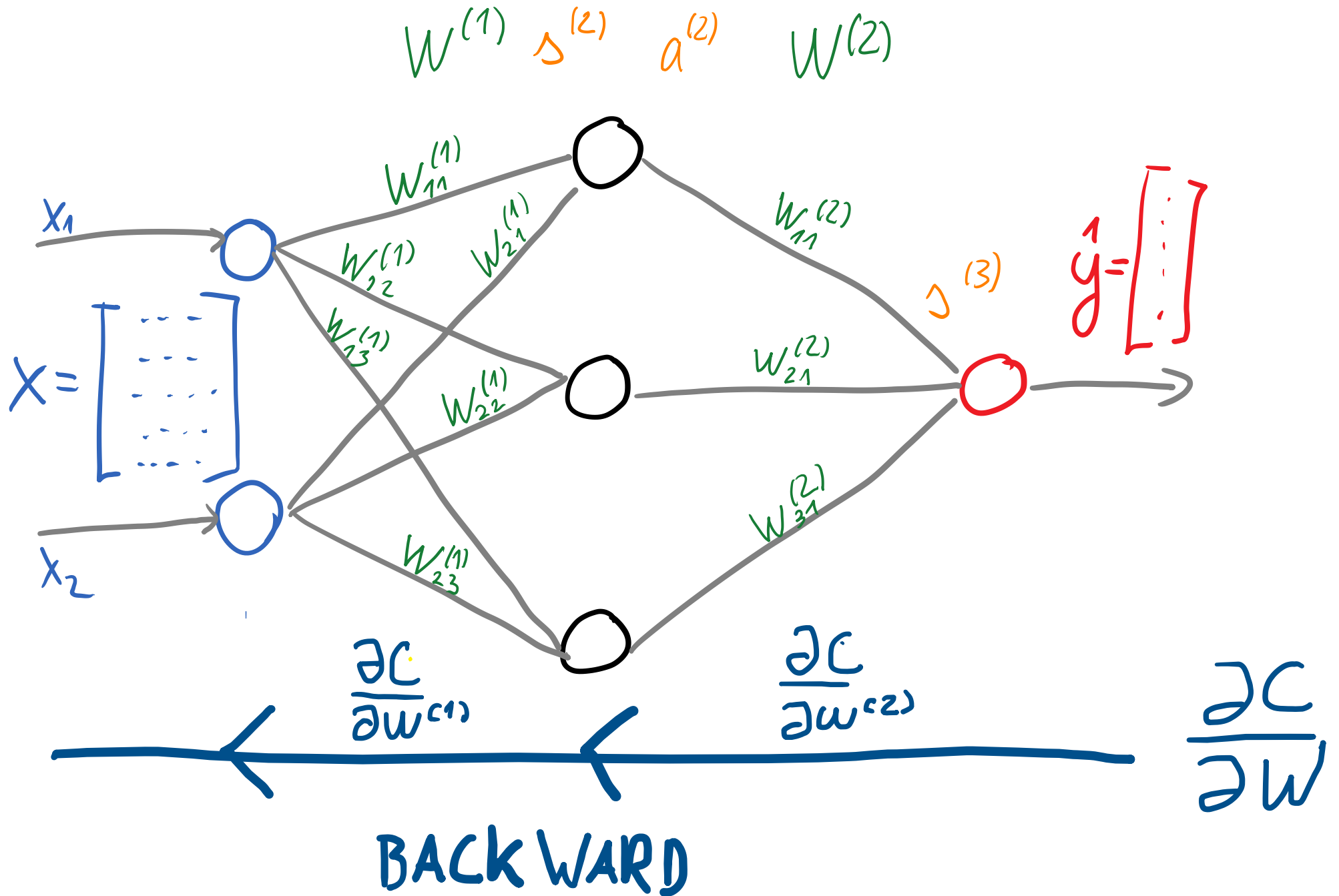
$$\frac{\partial y}{\partial x} = \left[ \frac{\partial y}{\partial x_1} \quad \frac{\partial y}{\partial x_2} \quad \dots \quad \frac{\partial y}{\partial x_n} \right]$$

y scalar, W matrix

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

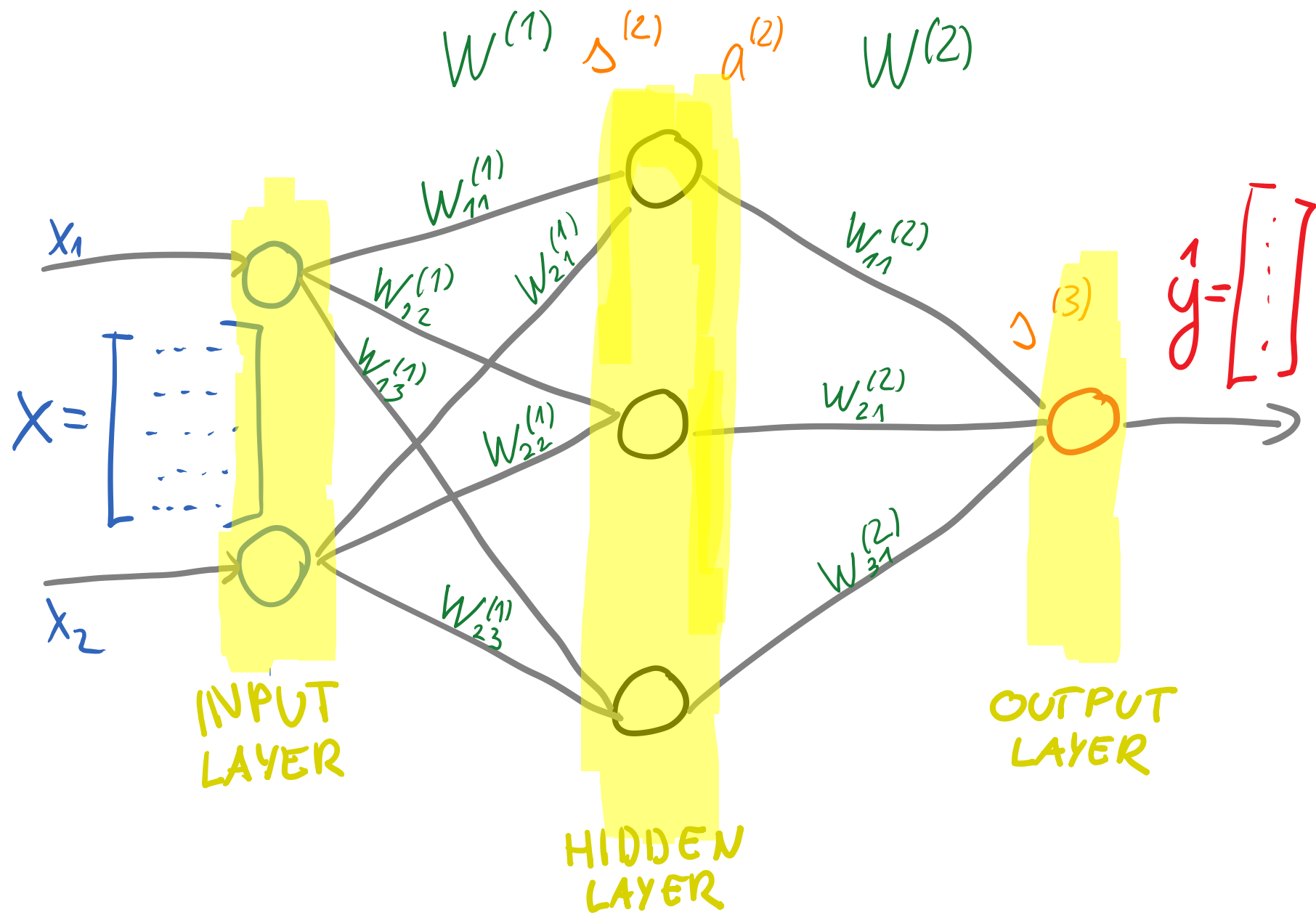
y vector, x vector

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



# Backward propagation – step 1

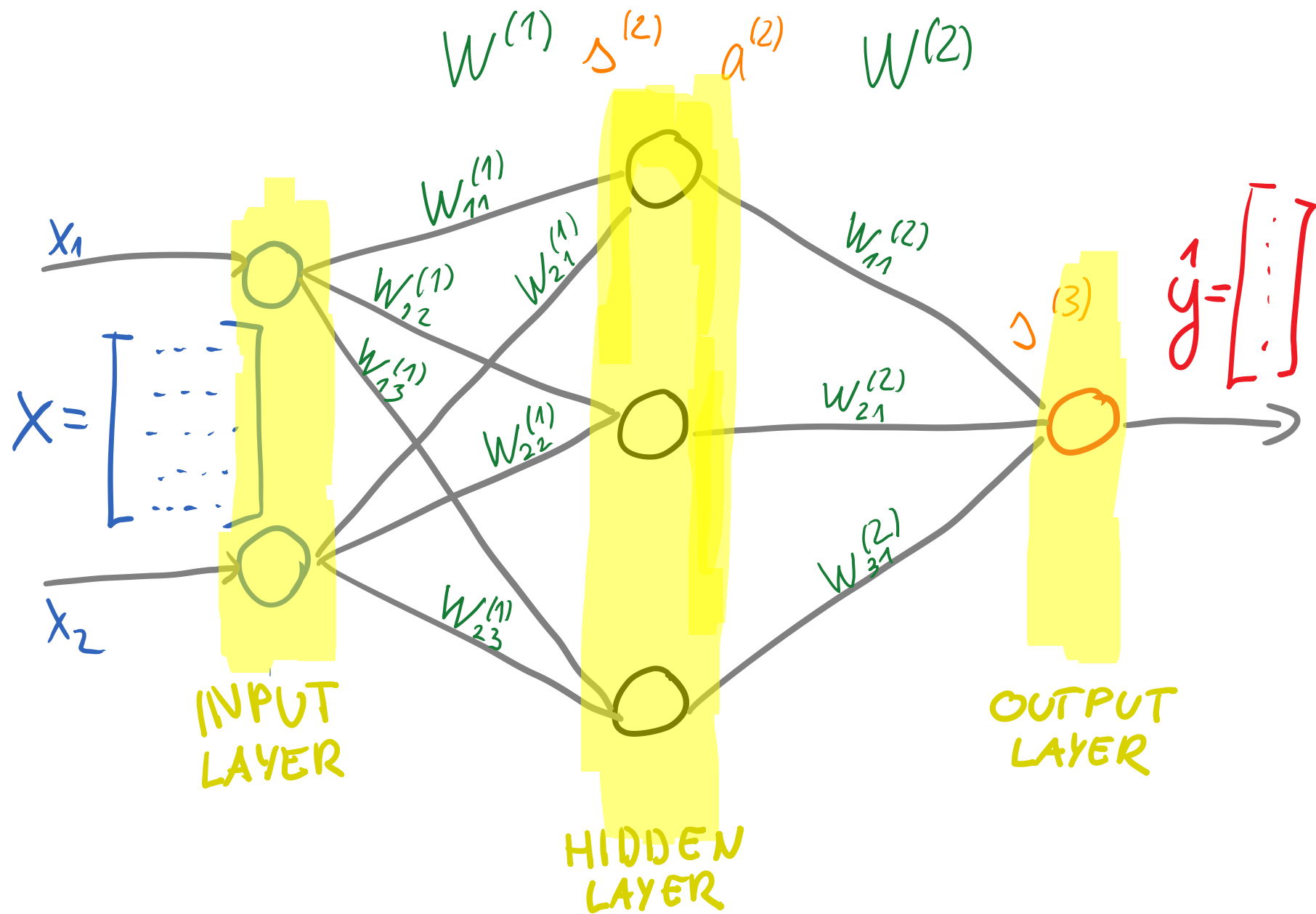
$$\frac{\partial C}{\partial w^{(2)}} = \frac{\partial \sum \frac{1}{2} (y - \hat{y})^2}{\partial w^{(2)}}$$



# Backward – step 2: batch gradient descent

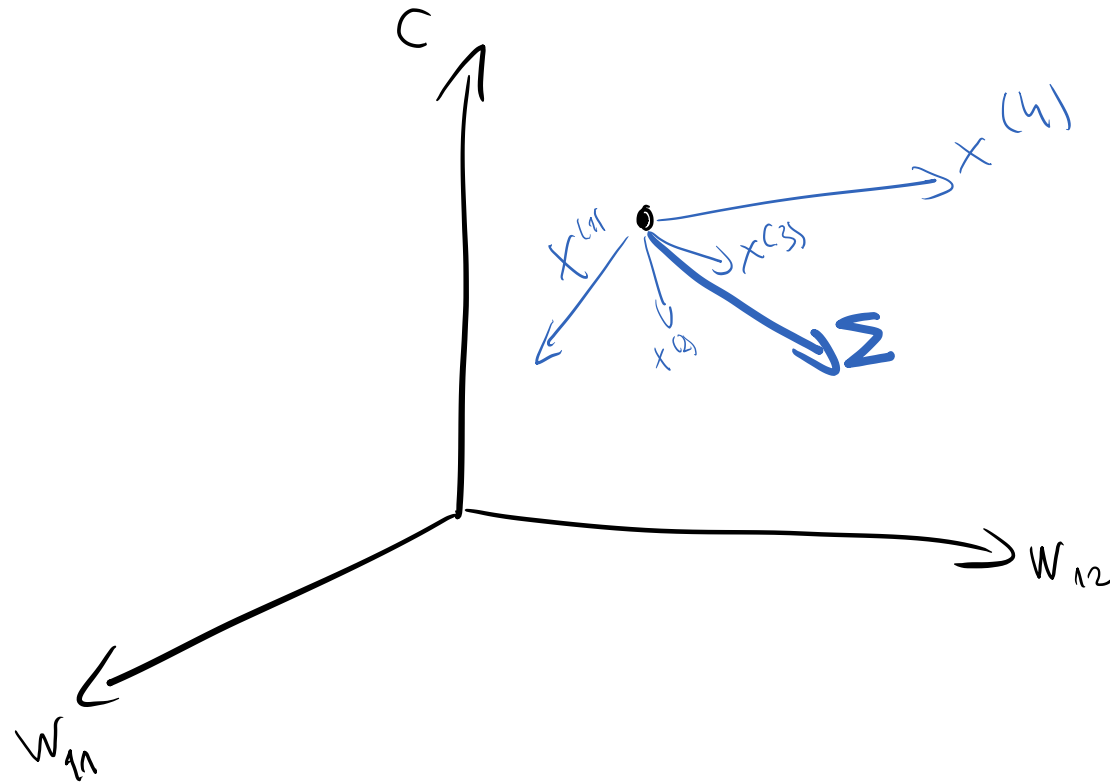
$$\textcircled{7} \quad \frac{\partial C}{\partial w^{(2)}} \stackrel{(3 \times 1)}{=} \sum \frac{\partial \frac{1}{2} (y - \hat{y})^2}{\partial w^{(2)}} = \stackrel{(3 \times 4)}{(a^{(2)})^T} \stackrel{(4 \times 1)}{\delta^{(3)}} =$$





# 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

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

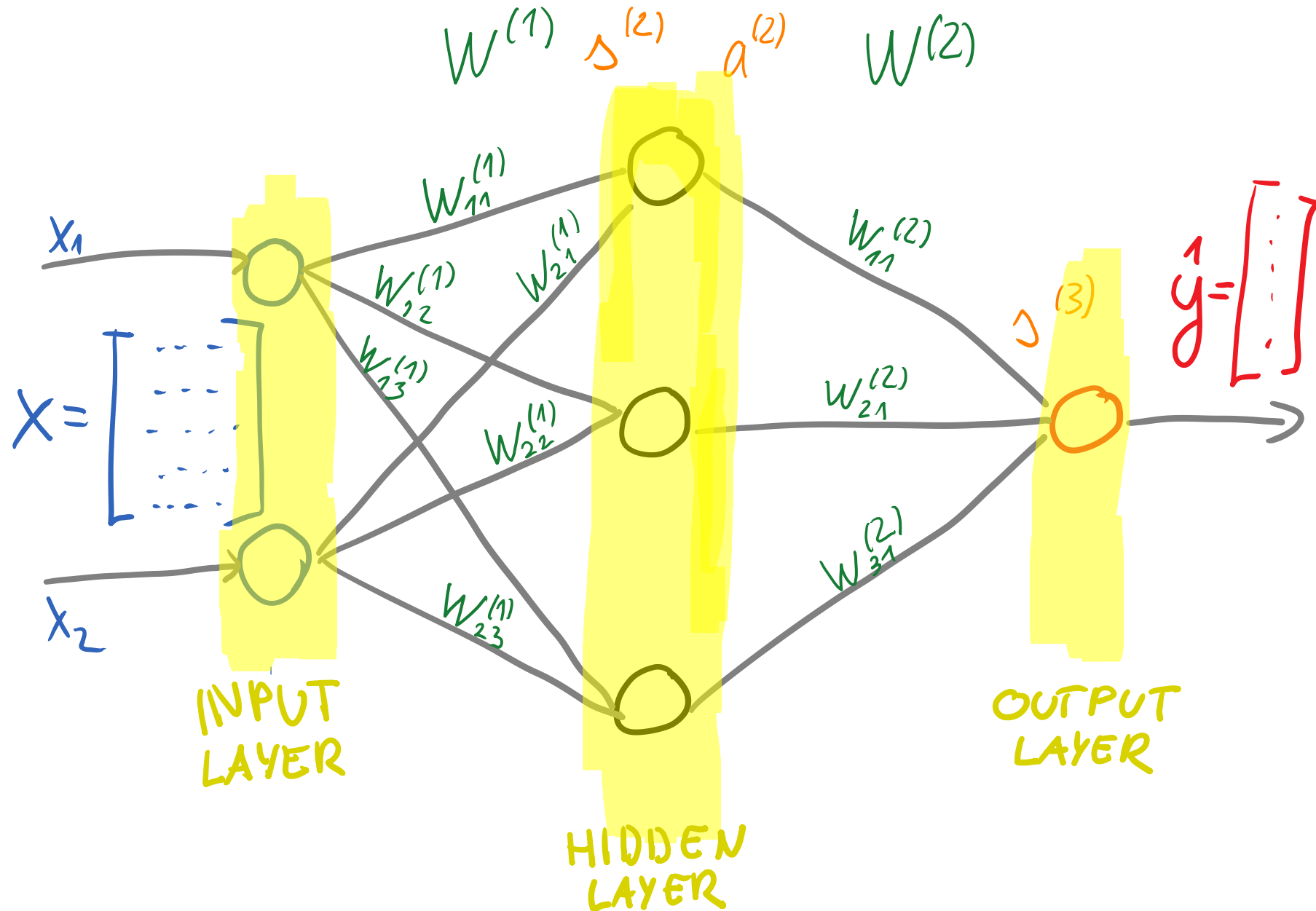
TRAINING

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

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

$\mu$ : learning rate

# Fully connected feedforward neural network

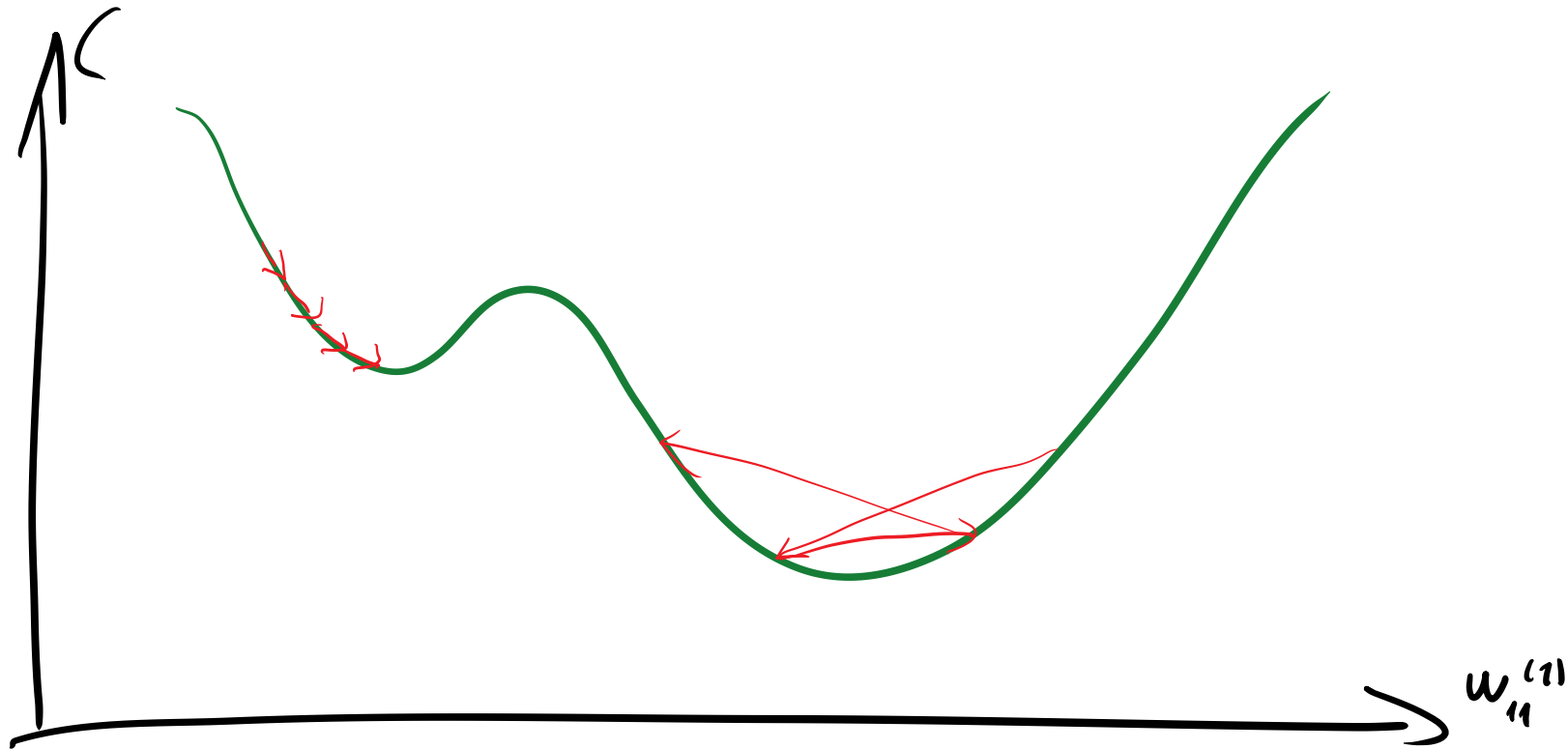


# Gradient descent

- **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 LeCun: 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](mailto:malradhi@tmit.bme.hu)

(slides by: Dr. Bálint Gyires-Tóth)

10 September 2024

