Robotics und Machine Learning (ML)

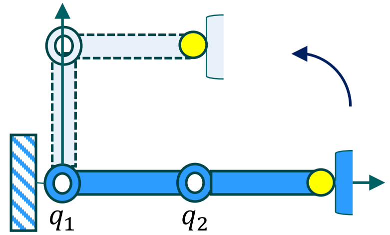
Prof. Dr.-Ing. Eric Kaigom

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Learning from experience – What are the coordinates of ● for $q_1 = \frac{\pi}{2}$ and $q_2 = -\frac{\pi}{2}$?

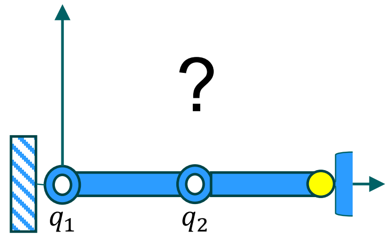
I know **from experience** that the coordinates of ● are (x = 1, y = 1)!

Human



What ???

Computer



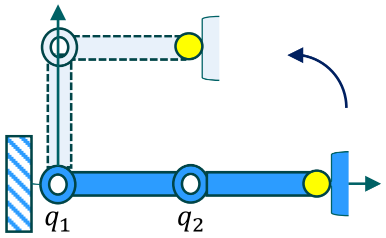
Can a computer (or in general a machine) do the same, **i.e, learn from experience** like a human?

Eric Kaigom
kaigom@fb2.fra-uas.de

Learning from experience – What are the coordinates of ● for $q_1 = \frac{\pi}{2}$ and $q_2 = -\frac{\pi}{2}$?

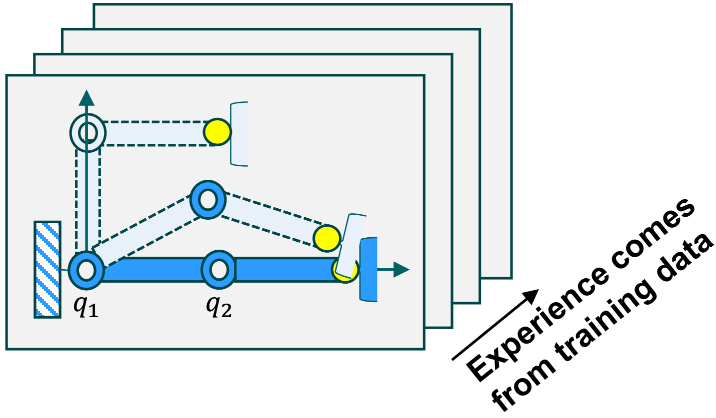
I know **from experience** that the coordinates of ● are (x = 1, y = 1)!

Human



I predict **after being trained** that the coordinates of ● are (x = 0.99, y = 1.01)!

Computer



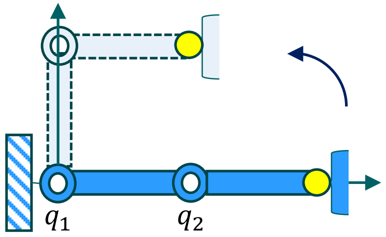
Can a computer (or in general a machine) do the same, **i.e, learn from experience** like a human?? **Yes!**

Eric Kaigom
kaigom@fb2.fra-uas.de

Learning from experience – What are the coordinates of ○ for $q_1 = \frac{\pi}{4}$ and $q_2 = -\frac{\pi}{4}$?

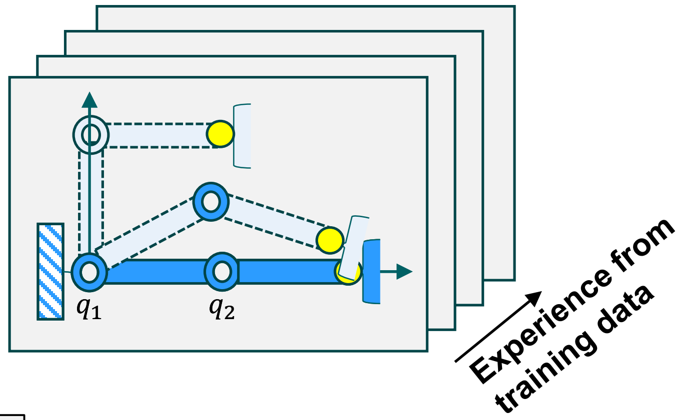
Well, …. …. I can imagine, but…
…..**actually, I have no idea**!

Human



I **predict after training** that the coordinates of ○ are (x = 1.7, y = 0.7)!

Computer



A trained **machine** can **outperform** human capabilities in e.g. non-nominal cases by **learning from data** (**entailing a rich and diversified experience**).

Eric Kaigom
kaigom@fb2.fra-uas.de

Machine Learning

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Machine Learning – A brief overview

What

Machine learning is the **process** of using **algorithms** to learn a **model** that helps **predict** the outcomes of previously **unseen events** (i.e., model input data) by using experiences condensed in **training data.**

How

Challenge and constraints understanding, selection of a learning approach

Data management

- Collection
- Pre-processing
- etc

Model development

- Hyperparameterization
- Optimization
- Validation, etc

Applications

Our focus

- Forward kinematics
- Backward kinematics
- etc

Eric Kaigom
kaigom@fb2.fra-uas.de

Understanding challenges, constraints, and opportunities - example

Why and what

- Forward kinematics (FK) and backward kinematics (BK) not available in a closed form (e.g., refurbished robots)
- FK and BK prone to uncertainties (e.g., geometry, structural properties, noise, etc.)
- Access to input (e.g., joint pos.) and output (e.g., end-effector pos.) data via measurements and data acquisition

How

- Instead of a model-based (i.e., analytical) derivation of the input-output or output-input mapping in closed form, data samples are leveraged to capture the mapping model (FK and BK)
- Captured models (FK and BK) can be used in arbitrary (e.g., physical or virtualized) applications and even shared via mail („FK and BK to go!")

Eric Kaigom
kaigom@fb2.fra-uas.de

Selecting a suitable machine learning approach

Our focus is **general**, i.e., encompassing the **non-linear case**!

Supervised learning

- Input example data known
- Corresponding/labelled output example data known
- **Learning goal**: retrieve noised mapping/association function from input and output examples

Our focus

- **Regression** ($x = f(u)$+noise)
- **Classification** ($x \in C_1 / \cdots / C_n$)
- **Etc.**

Unsupervised learning

- Input example data known
- There is no corresponding/labelled output example data available
- **Learning goal**: retrieve some properties in/from input data

- **Clustering** (captures characteristics from input data to infer properties)
- **Dimension reduction** (learns (statistically) a representation of input data in a lower dimension space (through e.g. projection) for e.g. efficiency reasons)
- **Etc.**

Semi-supervised learning

- Input example data known
- Partially labelled output example data known
- **Learning goal**: retrieve labelling from output examples.

- **Transductive Super Vector Machines** (harnesses a low density region between class distributions to find decision boundary for separation and generalization)
- **Generative adversarial networks** (Discriminator generates class labels)
- **Etc.**

Reinforcement learning

- AI system-environment-interaction for the AI system to take action
- If feedback (environm. to learning system), actions are +/- rewarded
- **Learning goal**: retrieve a policy to meet a goal (by increasing reward)

- **Q-Learning** (model-free, set of discrete actions selected via max. Q-value)
- **Policy gradient** (has a large continuous space of actions and probability-based policies)
- **Actor-critic** (combines advantages of Q-Learning and Policy gradient)
- **Etc.**

Eric Kaigom
kaigom@fb2.fra-uas.de

Machine Learning – Supervised Learning (Big Picture)

Hint: The trained AI model predicts (i.e., approximates, hence „ FK(q) $\approx X_E$“) the reality (where FK(q) $= X_E$ is assumed to be true).

Measurements potentially prone to noise!

Known and **pertinent output example data** ($X_E = FK(q)$) associated with **pertinent input example data**

4

Result: a trained digital model used for e.g. prediction purposes (e.g., FK(q) $\approx X_E$) in other applications (e.g., robotized indust. automation)

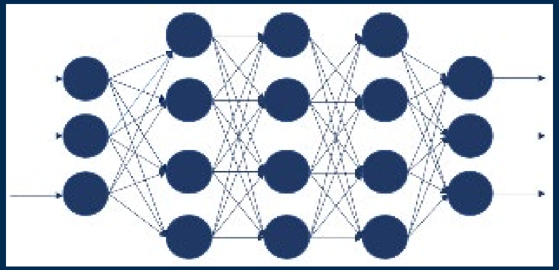
1

Pertinent data (e.g., q) as **input examples**

Why pertinent?: well, because otherwise garbage in and, unfortunately, garbage out!

Working on the physical systems in distributed labs / shop floor / at home

Model capture using maths, statistics, and data, etc



Model training (i.e., optimization of model parameters to capture the mapping between input-output examples and generalize)

2

3

Model output during training

−

Deviation

Optimization of parameters of the trained model to learn underlying mapping using labeled output data (hence „Supervised learning“)

5

Using a computer with solid computation capabilities (e.g., graphic card) for accelerated data processing

Eric Kaigom
kaigom@fb2.fra-uas.de

Ethics: Although compelling from an anthropocentric (human-centered) viewpoint, the related energy and ressource consumption for model training might be (at scale) questionable from an ecocentric perspective (nature with own rights)! How to globally and sustainably address this challenge for the biosphere and a shared well-being?!

Pivotal future research and application avenues YOU might want to streamline and shape!

Preparing data

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Why data preparation?

- An important step before data analysis (…in order to learn an unkown mapping or correspondence)
- Pivotal point in Machine Learning: How **features** are **distributed**
- **Implications:**
  - Preserve how the data samples (e.g., joint positions) relate to each other
  - Accommodate the impact of units
  - Avoid effects of relative scale (otherwise: learning performance can drop, i.e., convergence takes longer due to overshoot. More later on.)
- **Challenges:**
  - Impacts of outliers
  - etc

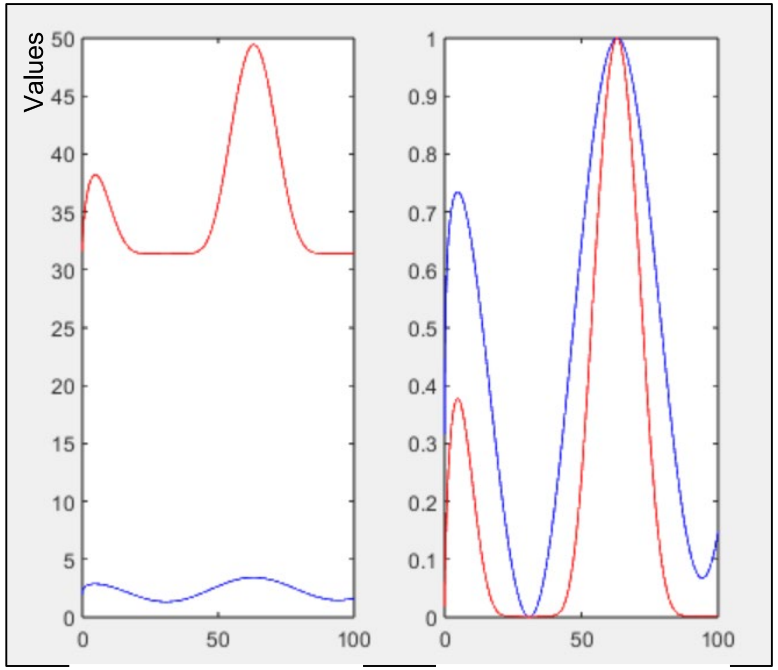
Eric Kaigom
kaigom@fb2.fra-uas.de

Data preparation - Minimum-Maximum Normalization

- $x = [x_1\ x_2 \ldots x_n]$ the vector of n scalars
- x_{min} and x_{max} is minimum and maximum value in x
- $\hat{x}$ is the min-max-normalized value of x:

$$\hat{x} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- Values in $\hat{x}$ are squeezed between 0 and 1, i.e., within [0 1]
- Is it possible to squeeze values between scalars z_0 and z_1 with $z_0 < z_1$?



Initial values

Min-max-normalised

Eric Kaigom
kaigom@fb2.fra-uas.de

Application #1

1) Use Matlab to randomly generate a vector V of 100 scalars between -100 and 100.
2) Carry out the min-max normalization of this vector and store the result in Vn.
3) Set the value at index 20 in V to 500. Store the result in Vo.
4) Carry out the min-max normalization of Vo and store the result in Von.
5) Plot V, Vn and Von in three subplots. What do you observe?

$$\hat{x} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

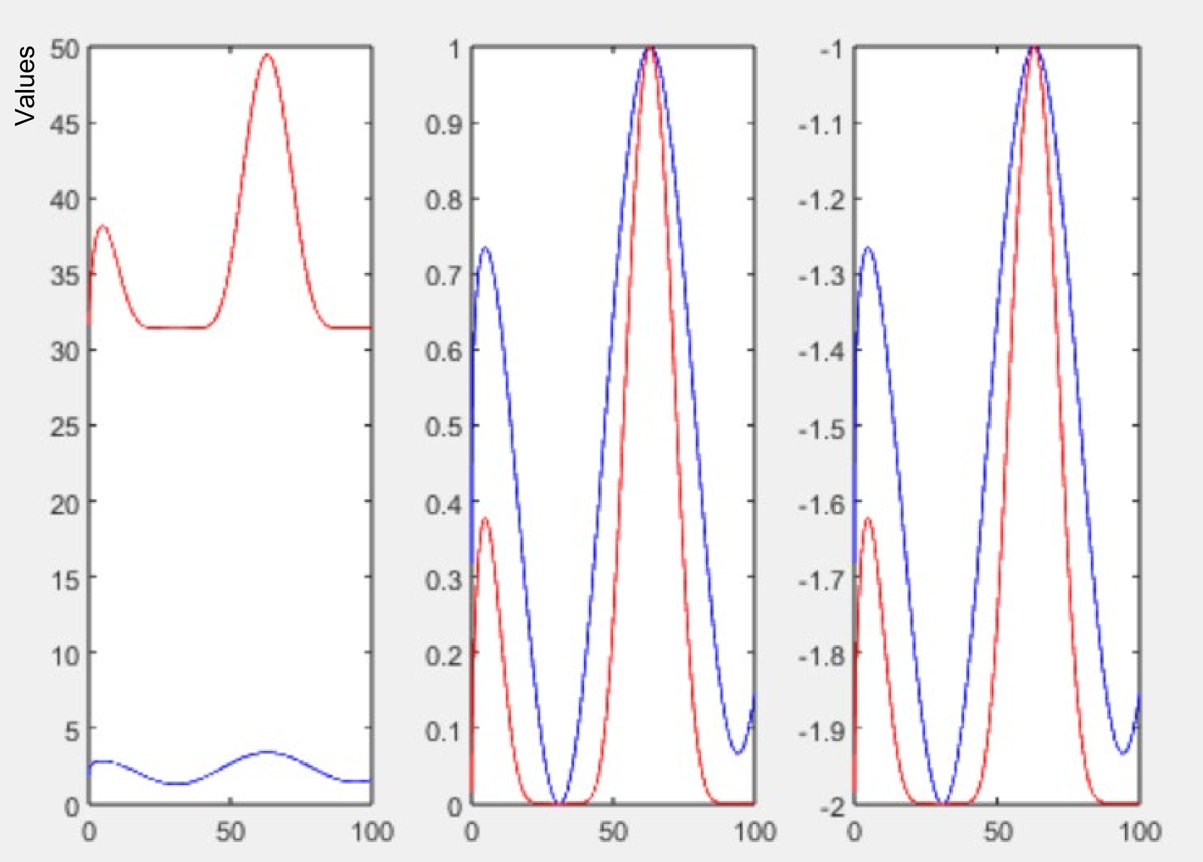
Eric Kaigom
kaigom@fb2.fra-uas.de

Data preparation - Minimum-Maximum Normalization

- $x = [x_1\ x_2 \dots x_n]$ the vector of n scalar data
- x_{min} and x_{max} is minimum and maximum value in x
- $\hat{x}$ is the min-max-normalized value of x:

$$\hat{x} = z_0 + (z_1 - z_0)\frac{x - x_{min}}{x_{max} - x_{min}}$$

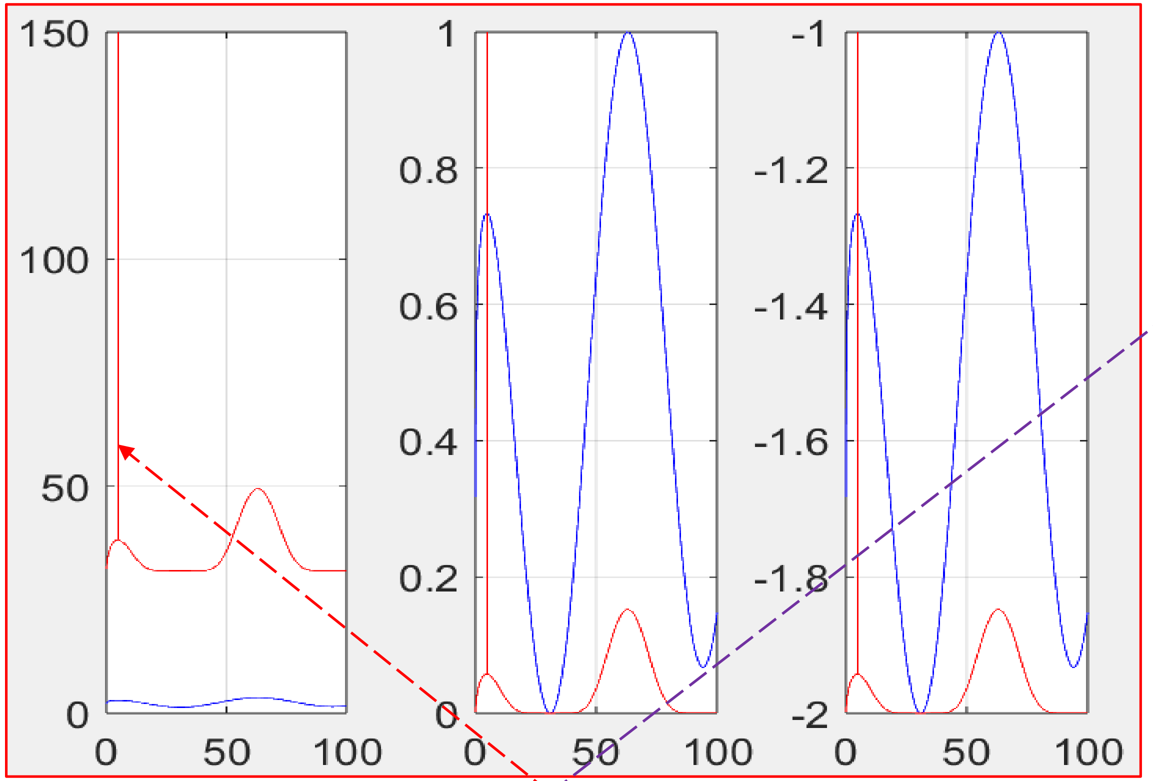
- Values in $\hat{x}$ are squeezed between z_0 and z_1

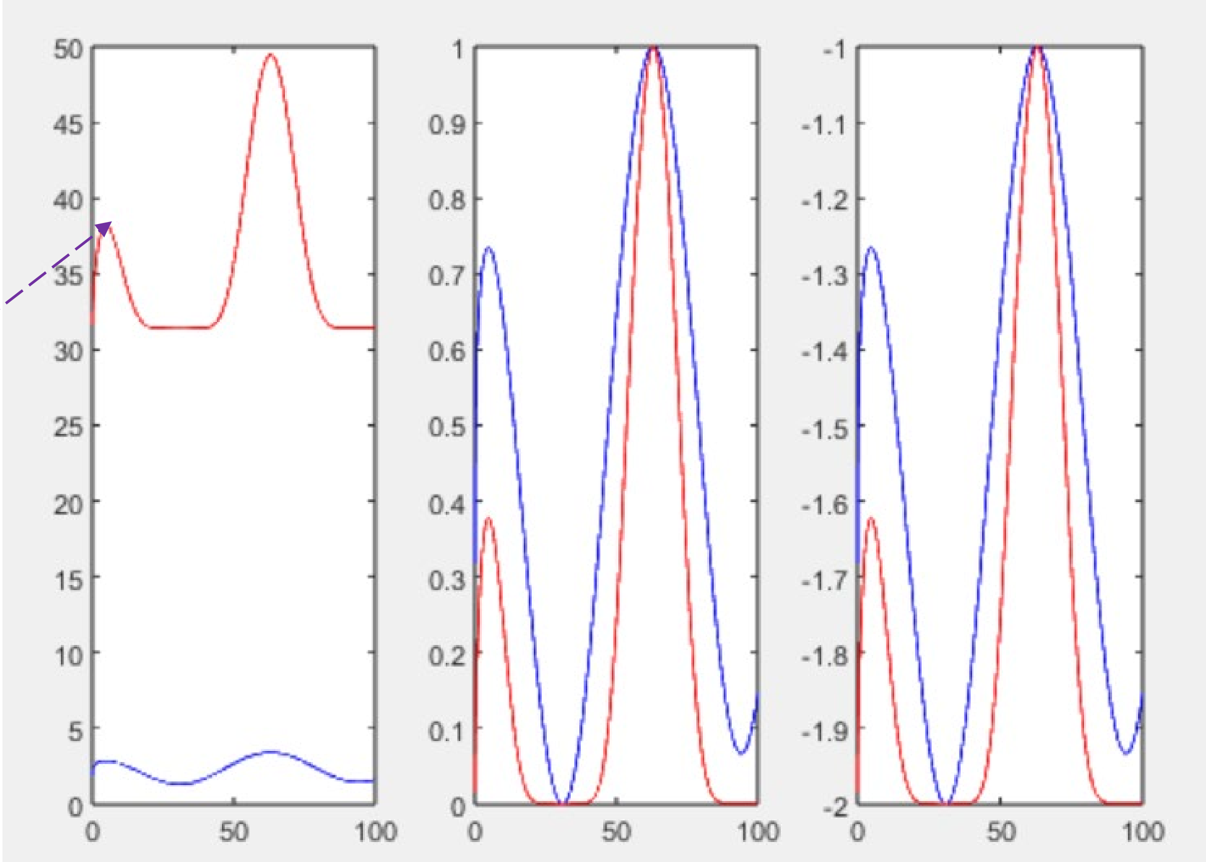


Initial values Min-max-normalised [0 1] Min-max-normalised [-2 -1]

Eric Kaigom
kaigom@fb2.fra-uas.de

Data preparation - Minimum-Maximum Normalization





Initial values | Min-max-normalised [0 1] | Min-max-normalised [-2 -1]

-
-
-
-
-
  - Min-max normalization faces outliers issues!
  - Shape (of the **distribution**) is preserved
  - How to influence **mean value** and **standard deviation**?

Eric Kaigom
kaigom@fb2.fra-uas.de

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Data preparation - z-Score Normalization

- $x = [x_1\ x_2 \ldots x_n]$ the vector of n scalar data
- Mean value

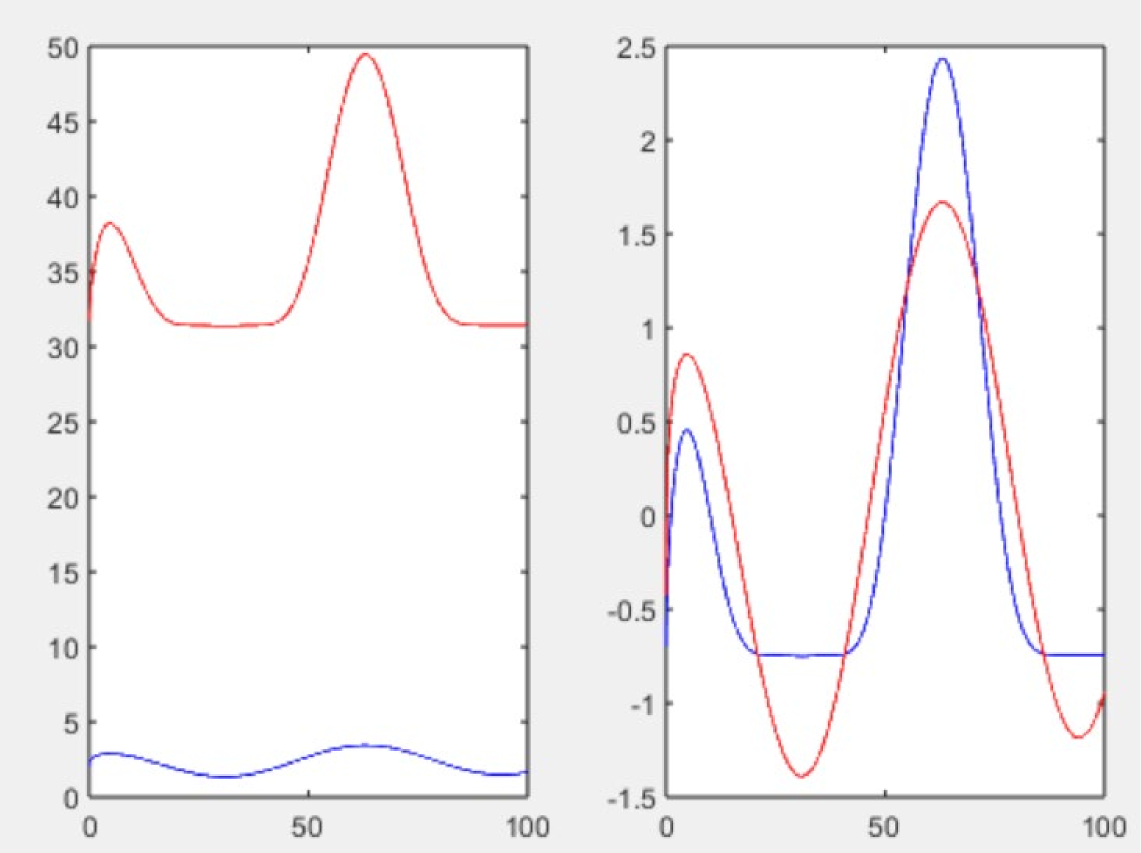
$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

- Standard deviation

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2}$$

- Normalization:

$$\hat{x} = \frac{x-\mu}{\sigma}$$



Initial values Normalised

- **Hints**:
  – The feature $\hat{x}$ hat mean $\hat{\mu} = 0$ and standard dev. $\hat{\sigma} = 1$
  – Data are not necessary confined within [0 1] (during training), which is helpful for robustness (w.r.t new test data outside [-1 1])!

Eric Kaigom
kaigom@fb2.fra-uas.de

Data preparation - z-Score Normalization

- $x = [x_1\ x_2 \ldots x_n]$ the vector of n scalar data
- Mean value

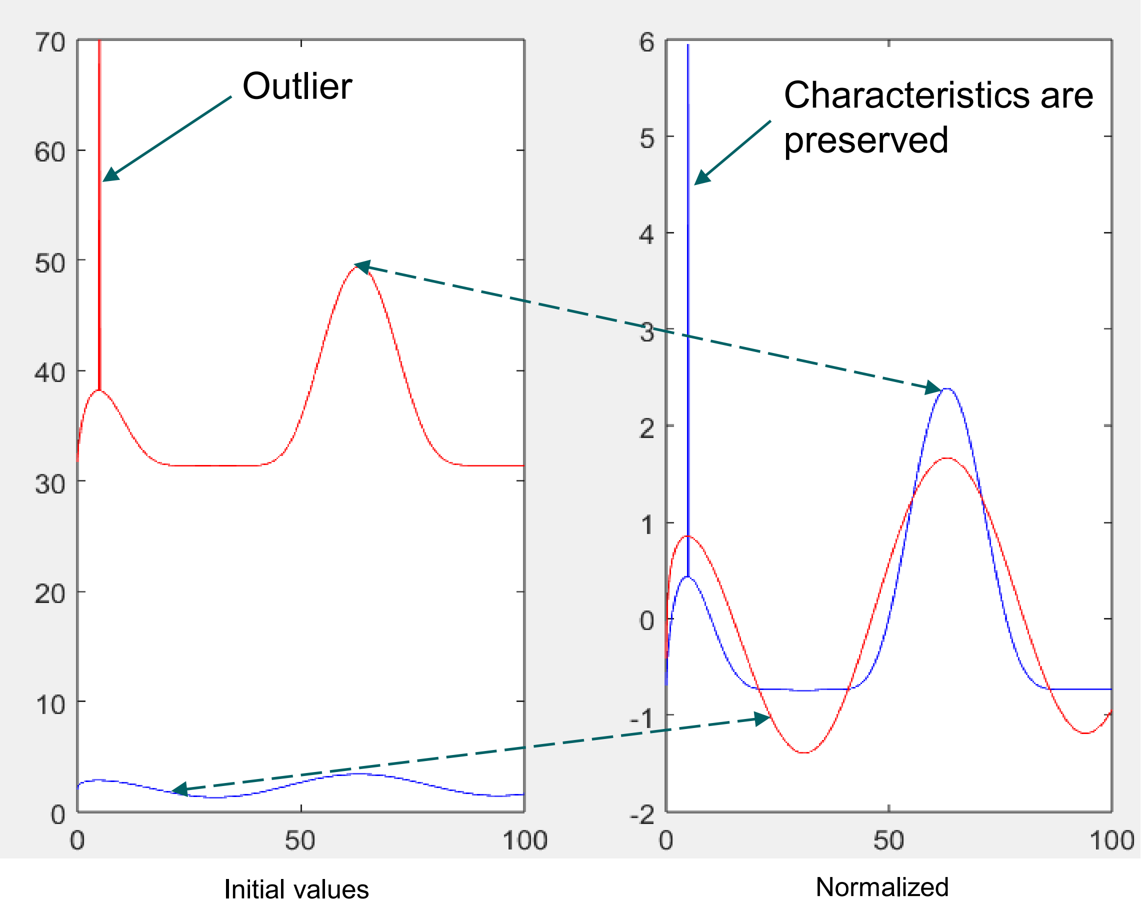
$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

- Standard deviation

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\mu)^2}$$

- Normalisation:

$$\hat{x} = \frac{x-\mu}{\sigma}$$



- **Hints**:
  - Useful properties about outliers are kept!
  - Impact of outliers on normalized data is mitigated (hence robustness!)

Eric Kaigom
kaigom@fb2.fra-uas.de

Application #2:

- Assume that $u = [x_1, \cdots, x_n]^T$ is a dataset
- Calculate Min-Max(z-Score(x))

Solution:

$$\hat{x} = \frac{x - x_{min}}{x_{max} - x_{min}} = \frac{\frac{x-\mu}{\sigma} - \frac{x_{min}-\mu}{\sigma}}{\frac{x_{max}-\mu}{\sigma} - \frac{x_{min}-\mu}{\sigma}} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

```
>> n=10; a=rand(n,1)'; zm = normalize(a,'range')- normalize(normalize(a),'range')

zm =

   1.0e-15 *

  -0.1110   -0.0555         0         0         0         0         0         0         0   -0.1110
```

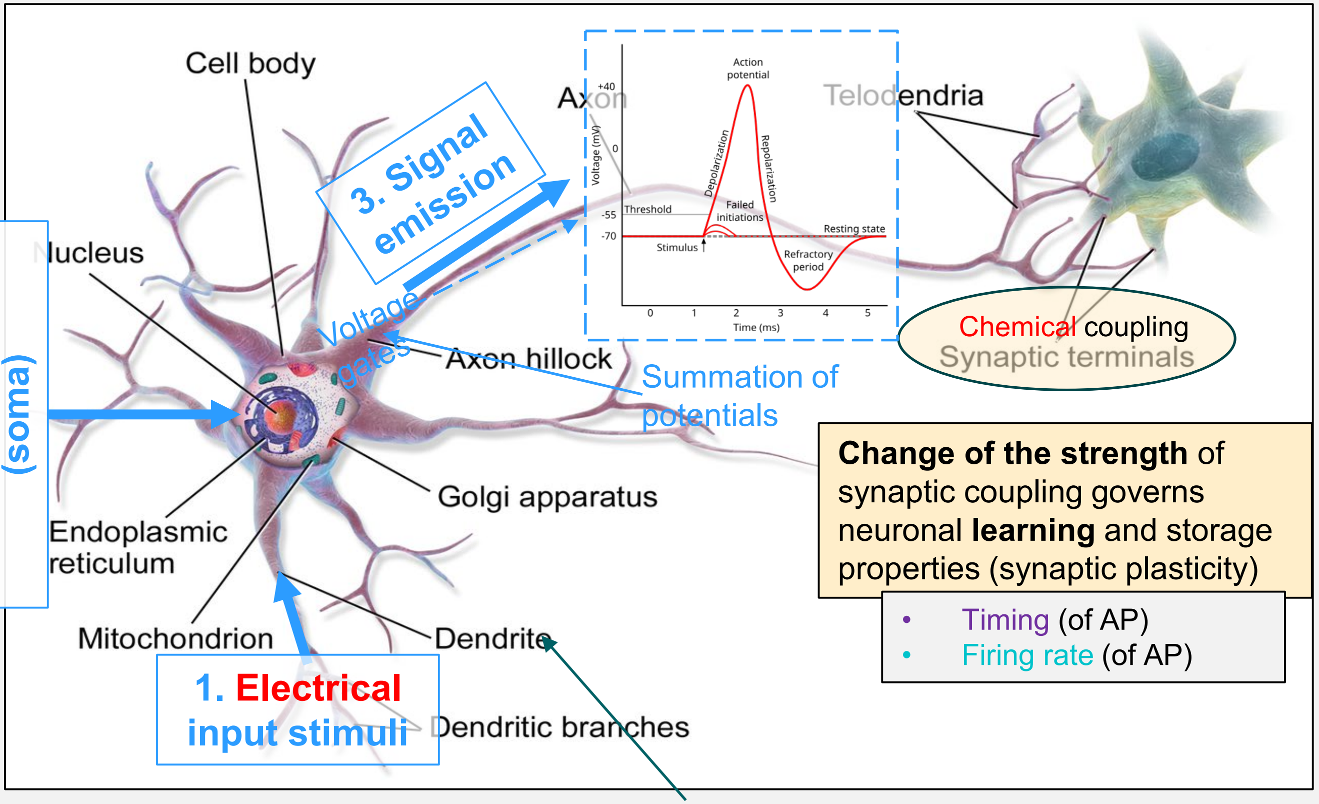
Min-max
Between [0 1]

Z-score
Between [0 1]

Eric Kaigom
kaigom@fb2.fra-uas.de

Using neurons to reason

Neuron model – Synapse dynamics



Change of the strength of synaptic coupling governs neuronal **learning** and storage properties (synaptic plasticity)

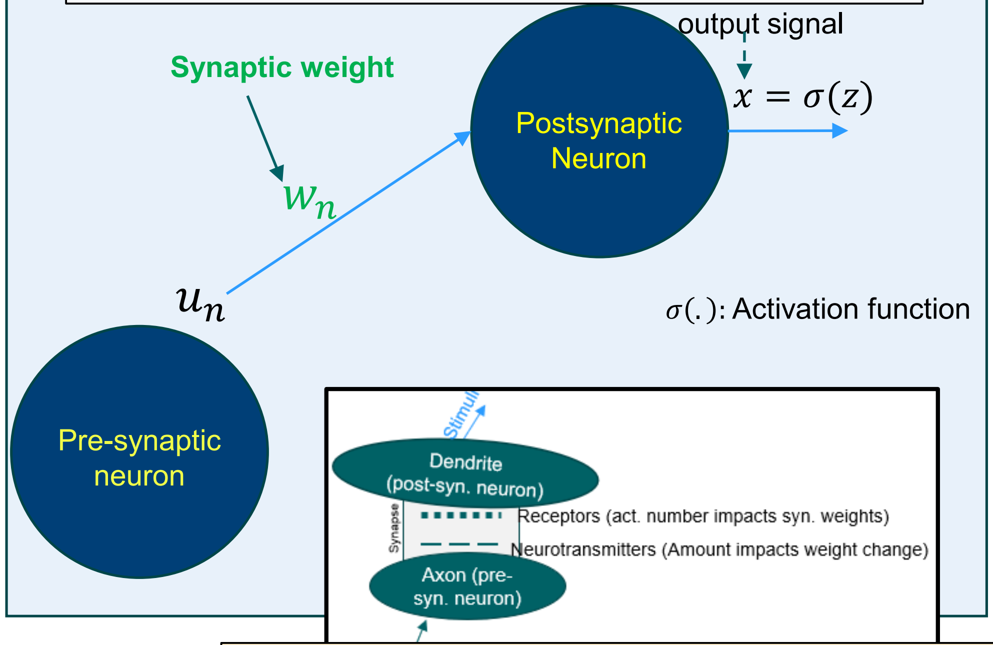
- Timing (of AP)
- Firing rate (of AP)

Receives stimuli from other neurons

Image source: https://en.wikipedia.org/wiki/Neuron#/media/File:Blausen_0657_MultipolarNeuron.png
Souce action potential image: https://en.wikipedia.org/wiki/Action_potential

Spike-timing-dependent plasticity

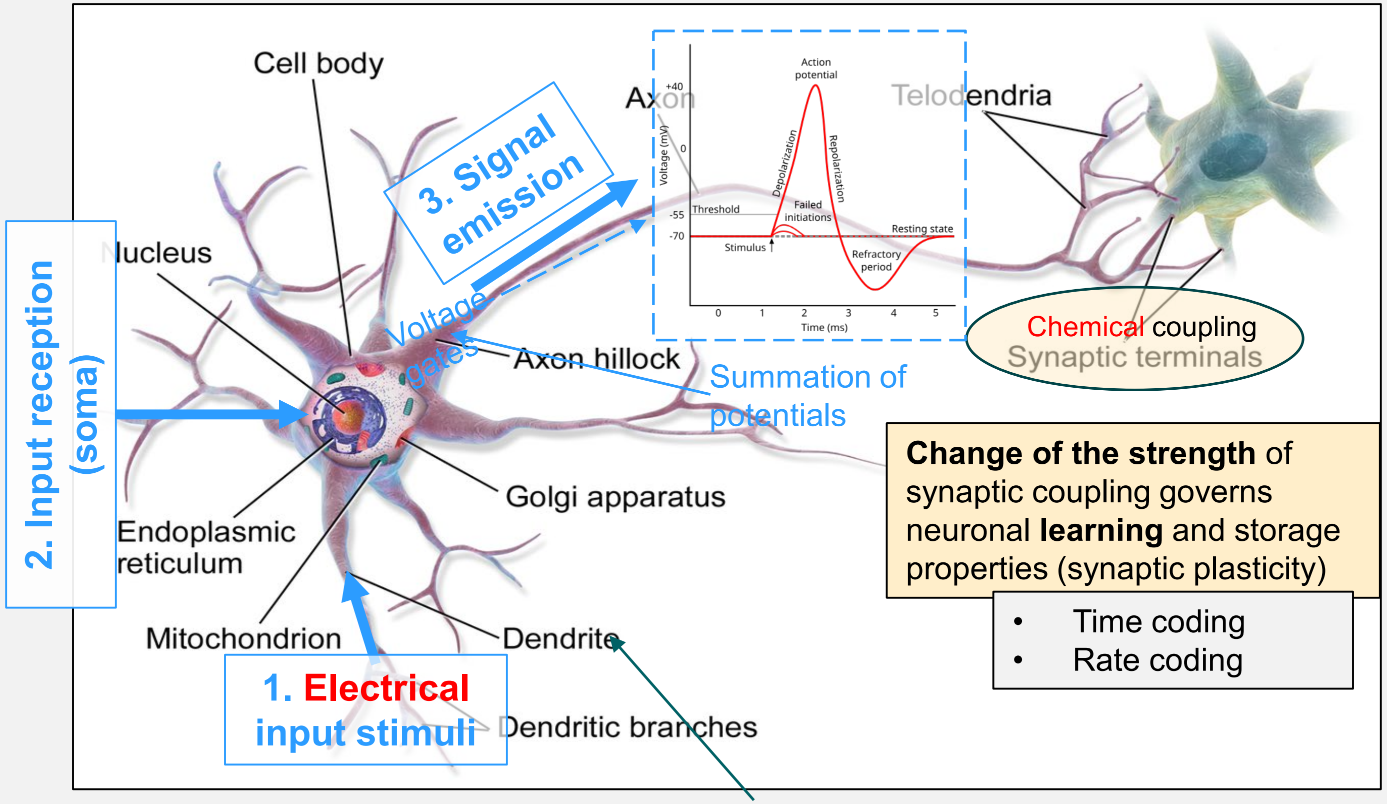
- Process associated with the modification (optimization) of the synaptic weights
- Depends upon temporal dynamics (e.g., latency) between action potential (AP) in pre- and postsynaptic neurons (+/- strength of long-term potentiation resp. depression)



Pre-synaptic firing rate impacts amount of neurotransmitters

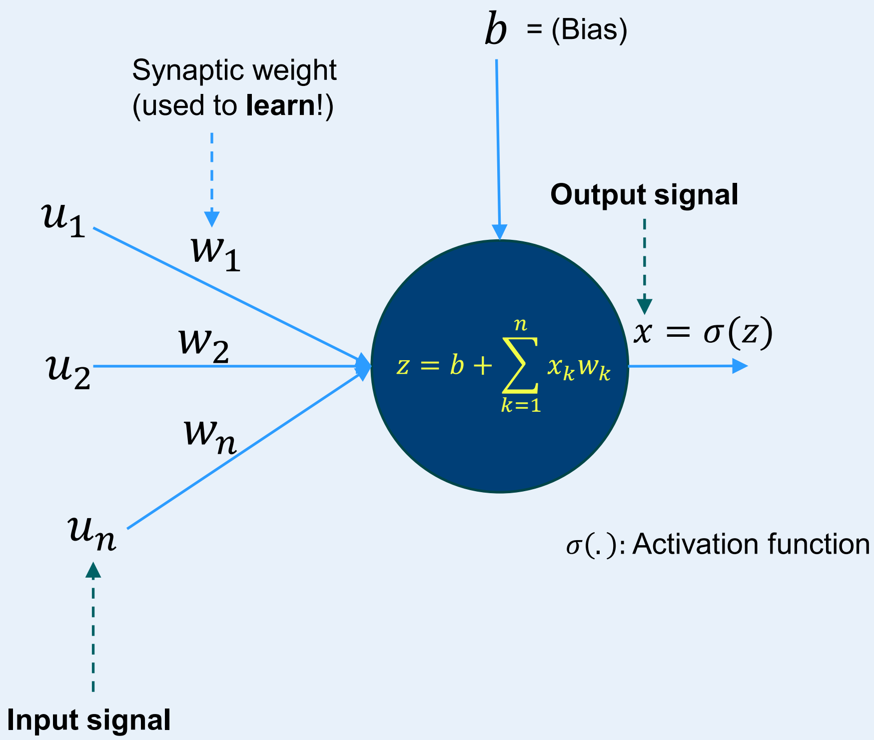
Eric Kaigom
kaigom@fb2.fra-uas.de

Neuron model – Synapse dynamics



Receives stimuli from other neurons

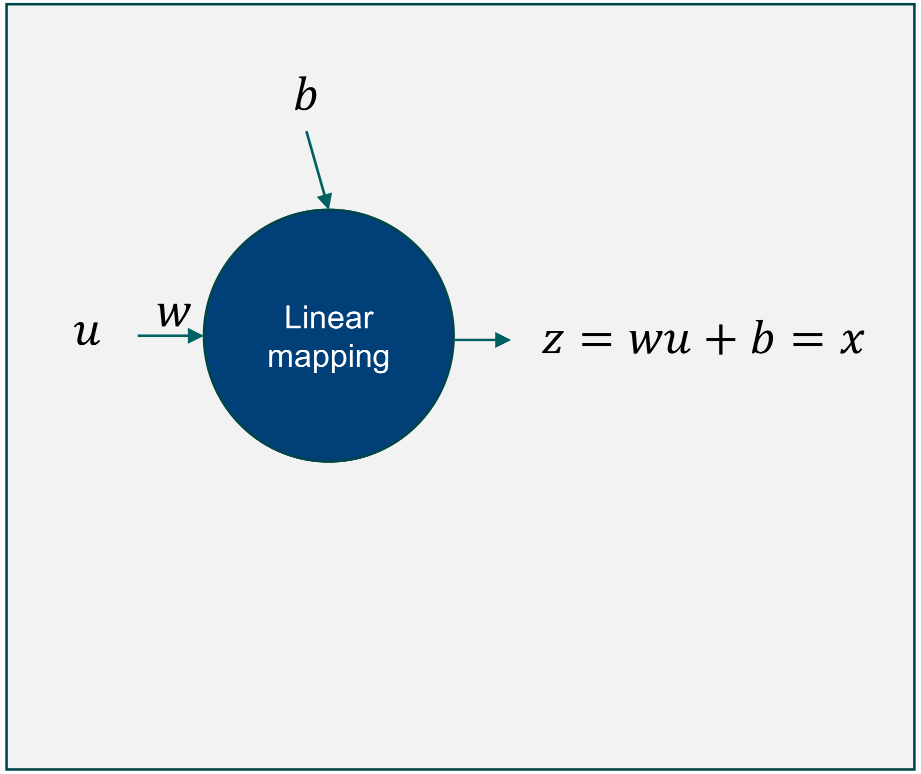
Image source: https://en.wikipedia.org/wiki/Neuron#/media/File:Blausen_0657_MultipolarNeuron.png
Souce action potential image: https://en.wikipedia.org/wiki/Action_potential



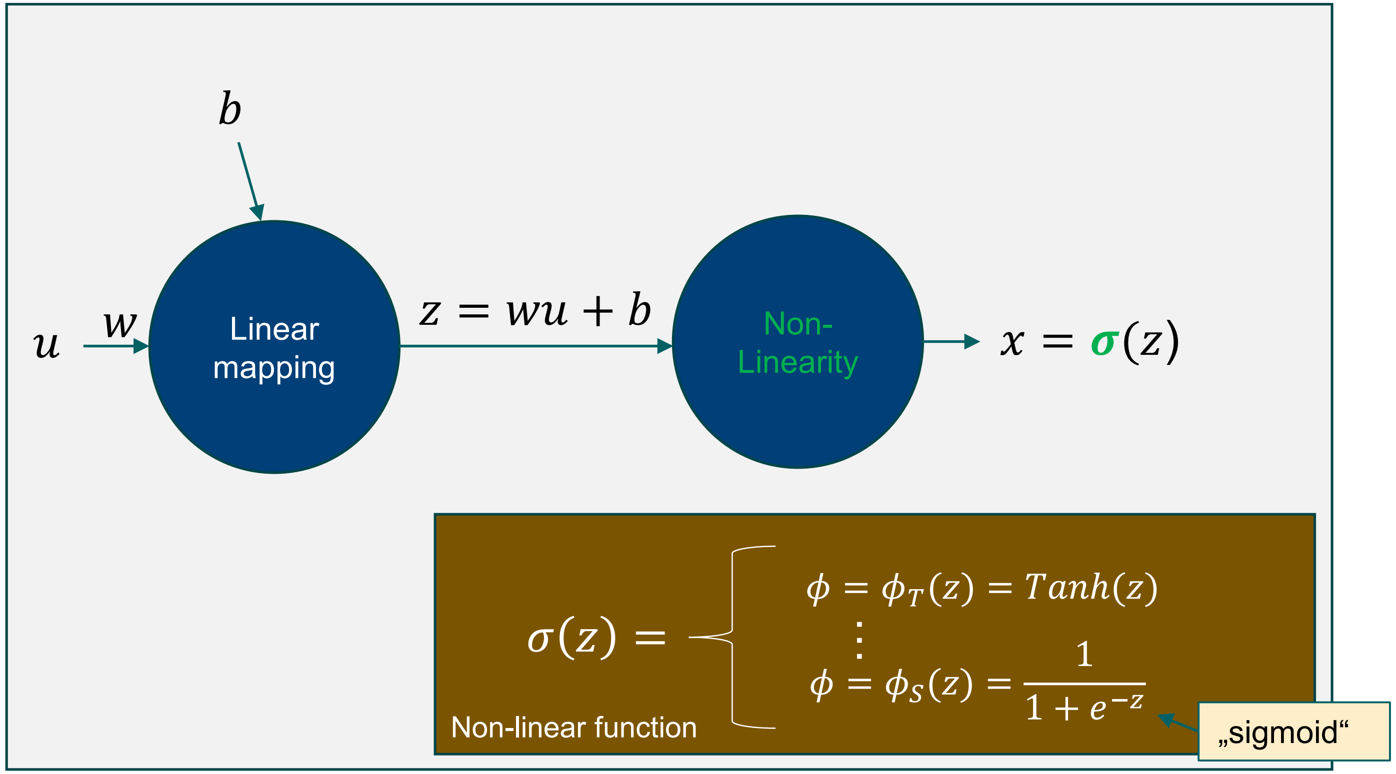
Neuron model

Eric Kaigom
kaigom@fb2.fra-uas.de

Artificial neuron



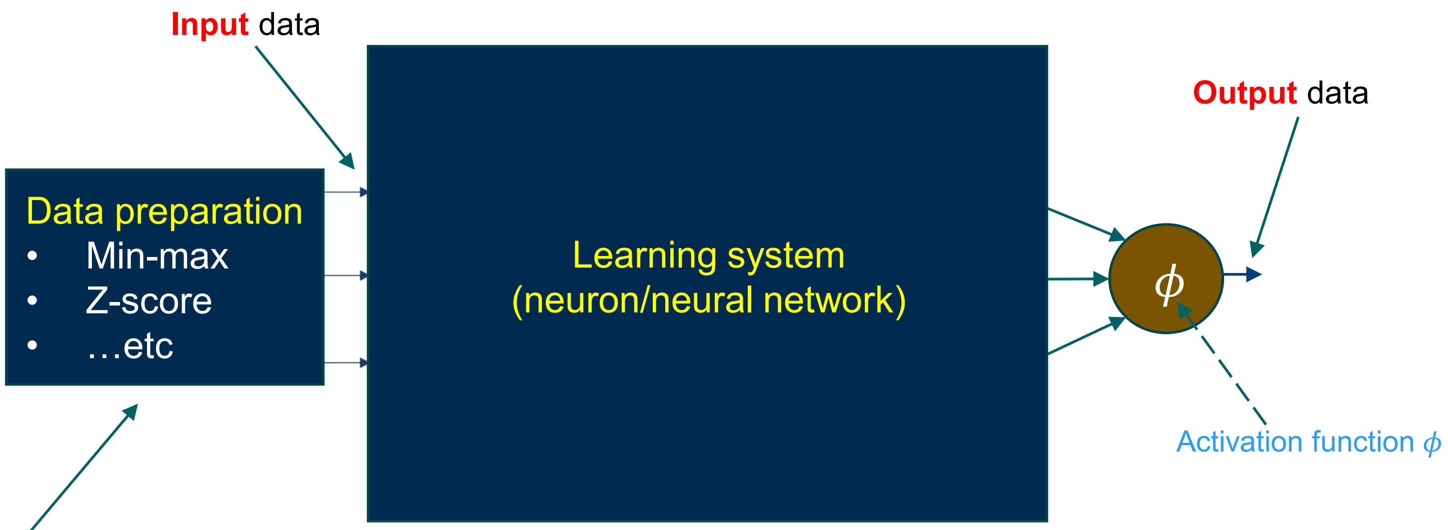
Identity activation

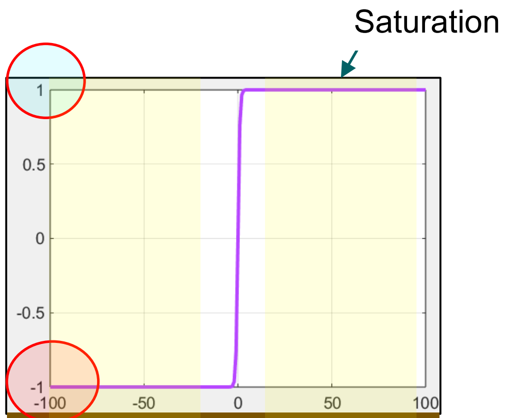


From identity activation to **non-linear** activation

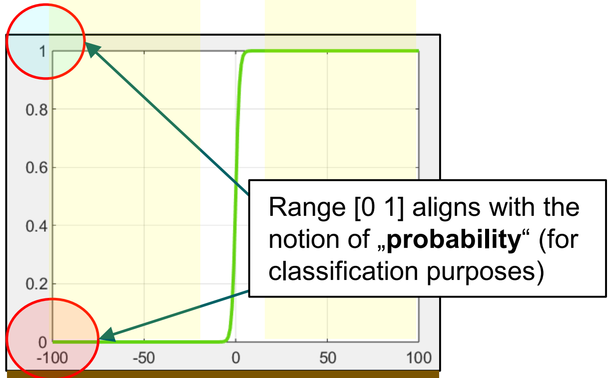
Eric Kaigom
kaigom@fb2.fra-uas.de
FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Data preparation and activation function





$$\phi = \phi_T(u) = Tanh(u)$$



$$\phi = \phi_S(u) = \frac{1}{1 + e^{-u}}$$

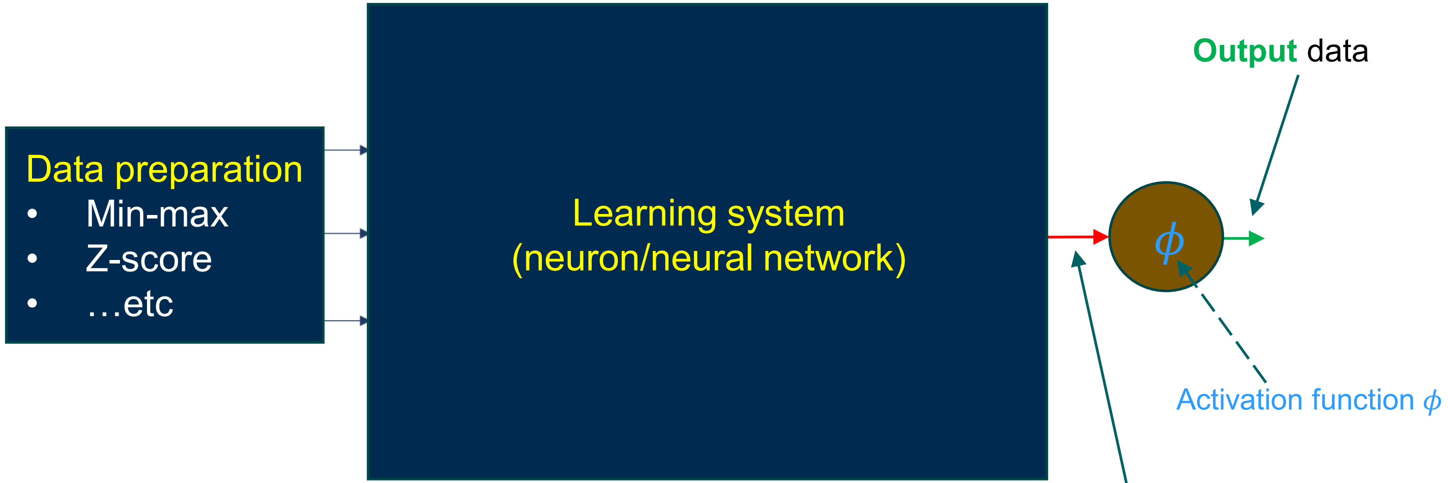
Called „Sigmoid“

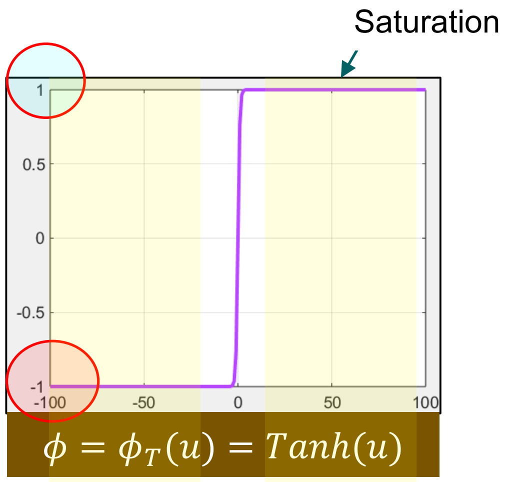
Data preparation helps

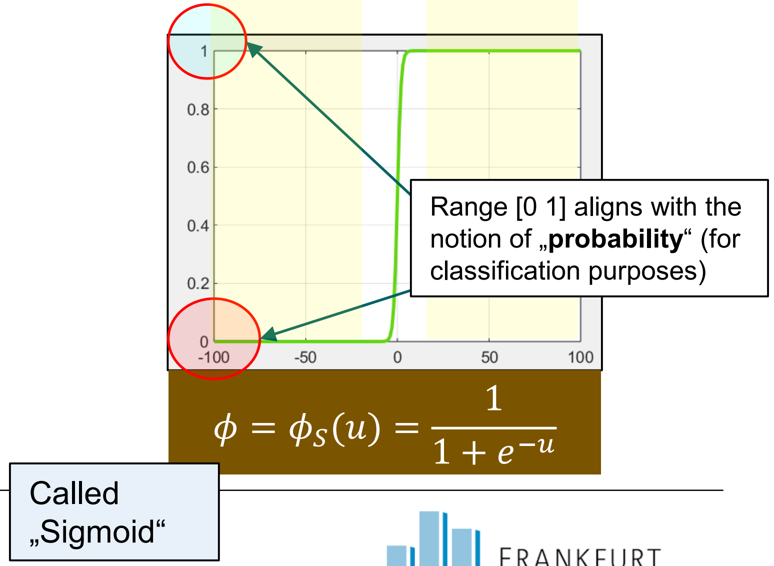
- keep the input training data in the **range** of the activation function ϕ
- **accelerate** the training process

Eric Kaigom
kaigom@fb2.fra-uas.de

Activation function - Saturation





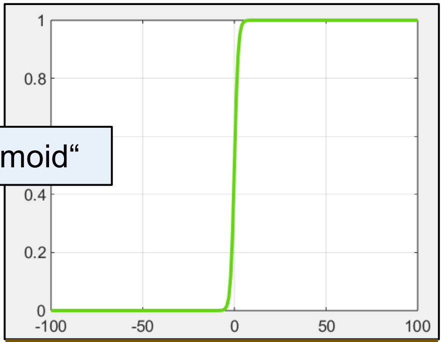


When saturated

- **Output** of the activation function ϕ does not respond to variations of its input
- learning capabilities drop, as potentially captured non-linear mappings are limited

Eric Kaigom
kaigom@fb2.fra-uas.de

Activation function – Product of derivatives

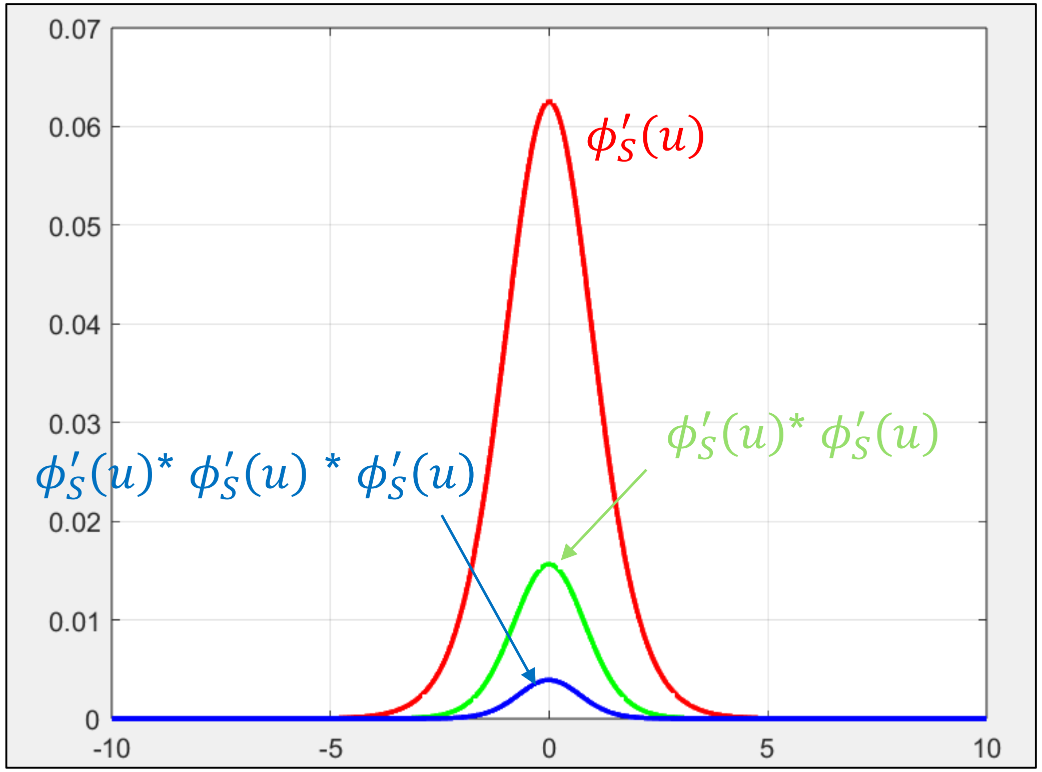


„Sigmoid“

$$\phi = \phi_S(u) = \frac{1}{1+e^{-u}}$$

„Sigmoid derivative“

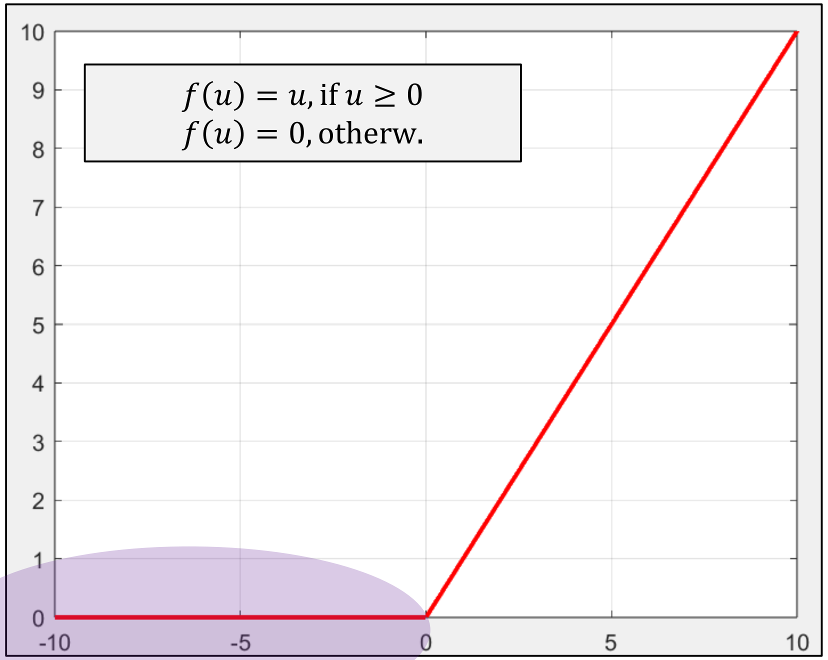
$$\phi'_S(u) = \left(1-\phi_S(u)\right)\phi_S(u)$$



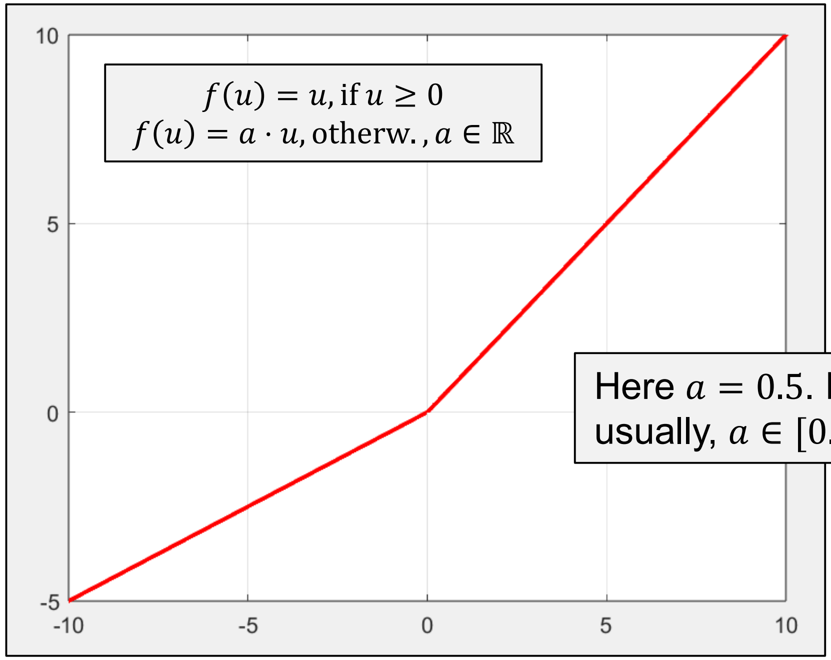
Observation: The amplitude of the product of the derivative $\phi'_S(u)$ by itself decreases with the increasing number of product terms.

Application#3: verify above-mentioned behavior ($\phi'_S(u) * \cdots * \phi'_S(u)$) of the sigmoid by writing and executing a matlab code for three product terms.

Activation function – Fixing saturation issues



Rectified Linear Unit (**ReLu**) Activation



Here $a = 0.5$. However, usually, $a \in [0.01, 0.1]$

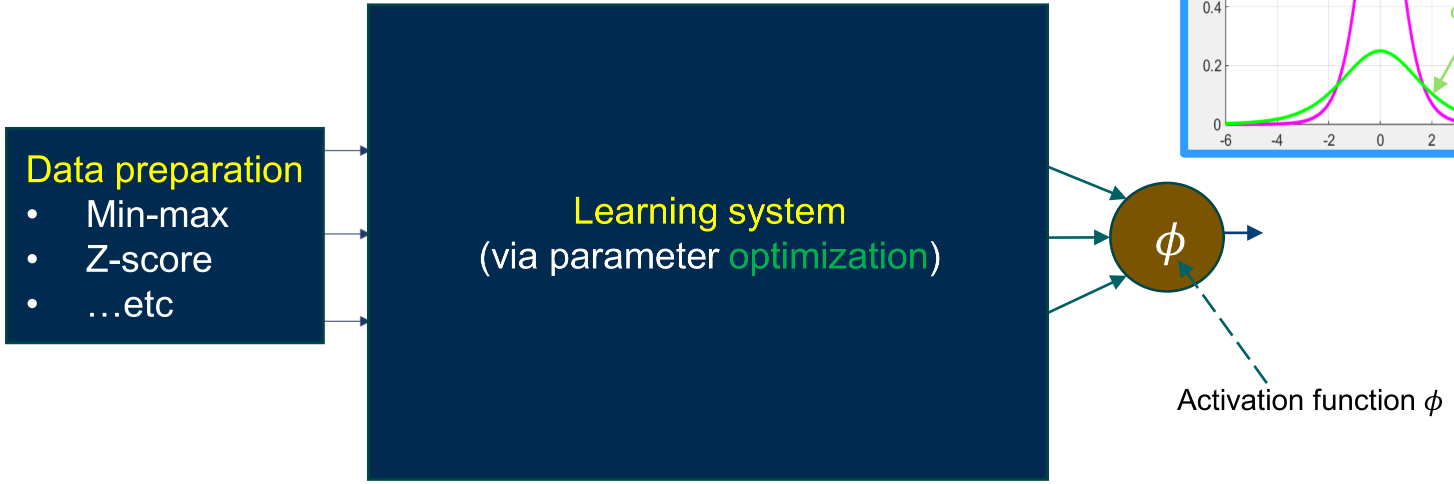
Leaky Rectified Linear Unit (**LReLu**)

- Not sensitive to negative inputs (might lead to dead neuron!)
- Unbounded for positive inputs

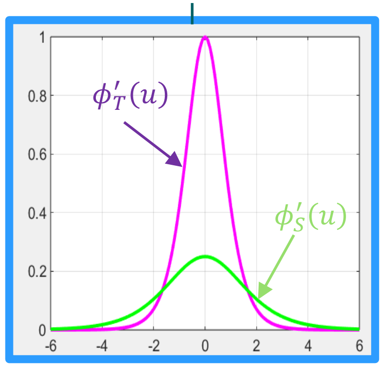
- Customized response (via a) to negative inputs (helps control the amplitude of derivative for negative inputs)
- Unbounded and no saturation hazard (thus: enhances learning rate during (optimization-based**)** learning)

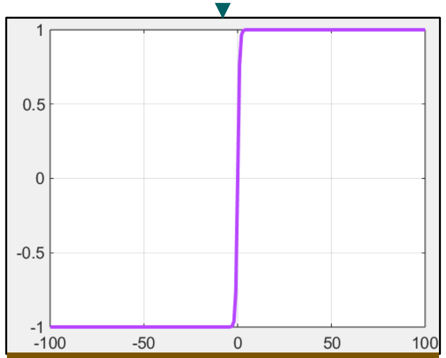
Eric Kaigom
kaigom@fb2.fra-uas.de

Activation function – Fixing derivative issues

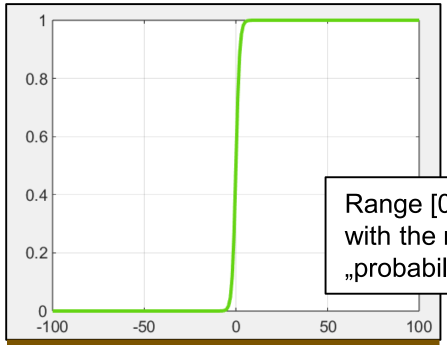


$\phi_T(u)$ provides a larger derivative (i.e., “gradient”, more in short)





$\phi = \phi_T(u) = Tanh(u)$



Range [0 1] aligns with the notion of „probability“

$\phi = \phi_S(u) = \dfrac{1}{1+e^{-u}}$

Derivative of ϕ_T and ϕ_S:

$$\phi_T'(u) = 1 - \tanh^2(u)$$

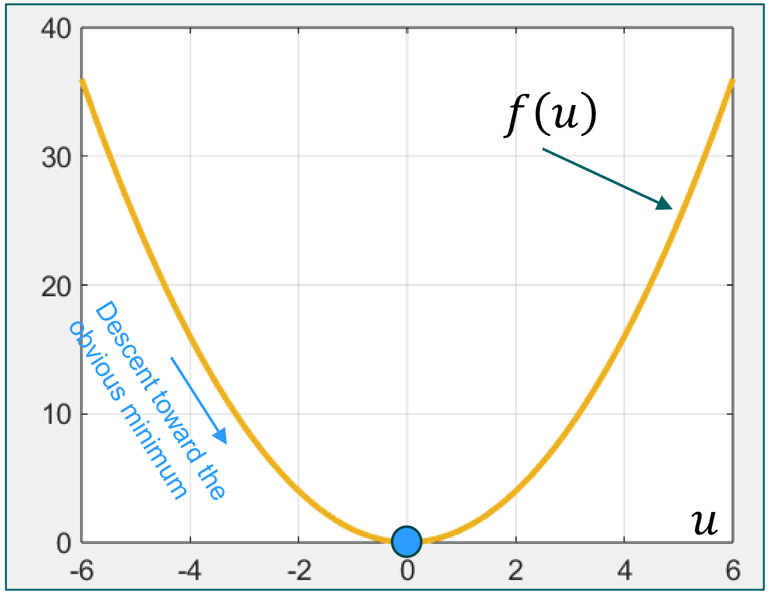
„Sigmoid derivative“

$$\phi_S'(u) = \big(1 - \phi_S(u)\big)\phi_S(u)$$

Eric Kaigom
kaigom@fb2.fra-uas.de

What is an „optimization“ and how does it work?

Goal: looking for a location with „maximum“ or „minimum“ function value



Optimization goal: find u^* that e.g. minimizes the 1D (or n-D) loss function $f(u)$.

$$u_{k+1} = u_k - rd$$

Optimization is an integral part of our daily life

Whereas humans tend to perceive optimum in some cases from experiences (see, e.g., r.h.s), the gradient can help machines do so in a systematic way!

Eric Kaigom
kaigom@fb2.fra-uas.de

What is an „optimization“ and how does it work?

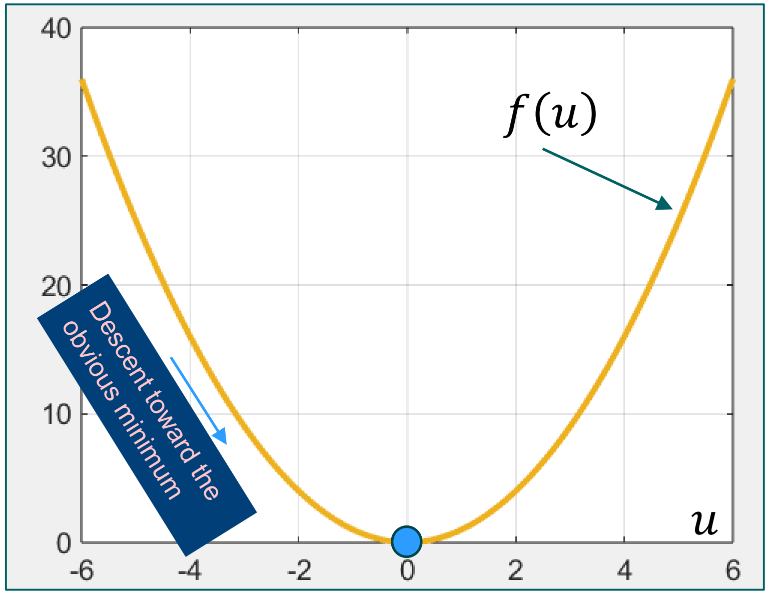
Human

Machine

$$u_{k+1} = u_k - rd$$

Computational methods

Optimization is an integral part of our daily life



Optimization goal: find u^* that e.g. minimizes the 1D (or n-D) loss function $f(u)$.

While humans tend to perceive an optimum in some cases (see, e.g., l.h.s), the gradient d can help machines do so in a systematic way with rate r!

Eric Kaigom
kaigom@fb2.fra-uas.de

What is a gradient?

- Preliminary: $x = f(u)$ is a real function of a real variable u.
- Gradient $\nabla f(u)_{u_0}$ of $f(u)$ w.r.t u at position $u = u_o$ is given by

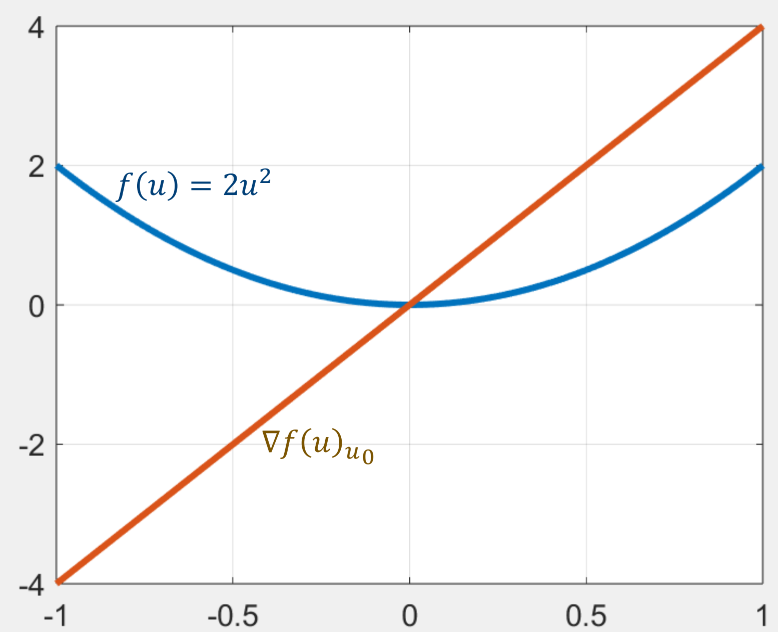
$$\nabla f(u)_{u_0} = \frac{\partial f(u)}{\partial u}\bigg|_{u=u_0}$$

f is differentiable at u_0

1.Order derivative w.r.t u at $u = u_o$

In the *n*-dimensional space: f: $\mathbb{R}^n \to \mathbb{R}$

- $u = (u_1, \cdots, u_n)$
- $\nabla f(u)_{u_0} = \left[\frac{\partial f(u)}{\partial u_1}, \cdots, \frac{\partial f(u)}{\partial u_n}\right]^T_{u=u_0}$



Gradient example

Eric Kaigom
kaigom@fb2.fra-uas.de

What is a gradient?

- Preliminary: $x = f(u)$ is a real function of a real variable u.
- Gradient $\nabla f(u)_{u_0}$ of $f(u)$ w.r.t u at position $u = u_o$ is given by

$$\boxed{\nabla f(u)_{u_0} = \frac{\partial f(u)}{\partial u}\Big|_{u=u_0}}$$

1.Order derivative w.r.t u at $u = u_o$

Application #4:

- $f(u) = u^2 \rightarrow \nabla f(u)_{u_{0=1}} = \frac{\partial f(u)}{\partial u}|_{u=1} = \frac{\partial u^2}{\partial u}|_{u=1} = 2u|_{u=1} = 2 \cdot 1 = 2$
- $f(u) = \frac{1}{u-1} \rightarrow \nabla f(u)_{u_{0=0}} = \frac{\partial f(u)}{\partial u}|_{u=0} = \frac{0-1}{(u-1)^2}|_{u=0} = -1$

Eric Kaigom
kaigom@fb2.fra-uas.de

Application #5

Challenge:

$$x = f(u_1, u_2) = 2u_1^3 + \ln(u_2)$$

Calculate the gradient of $f(u_1, u_2)$ at $u = (u_1, u_2)$

Solution:

$$\boxed{x = f(u_1, \dots, u_n) \rightarrow \nabla f = \begin{bmatrix} \frac{\partial f}{\partial u_1} \\ \vdots \\ \frac{\partial f}{\partial u_n} \end{bmatrix}} \Longrightarrow \nabla f = \begin{bmatrix} 6u_1^2 \\ \frac{1}{u_2} \end{bmatrix}$$

Eric Kaigom
kaigom@fb2.fra-uas.de

Application #6

Challenge:

$$x = f(u_1, u_2) = -u_2^2 + u_2 + 17$$

Calculate the gradient of $f(u_1, u_2)$ at $u = (u_1, u_2)$

Solution:

$$x = f(u_1, \dots, u_n) \rightarrow \nabla f = \begin{bmatrix} \frac{\partial f}{\partial u_1} \\ \vdots \\ \frac{\partial f}{\partial u_n} \end{bmatrix}$$

$$\Rightarrow \quad \nabla f = \begin{bmatrix} 0 \\ -2u_2 + 1 \end{bmatrix}$$

Eric Kaigom
kaigom@fb2.fra-uas.de

Application #7

Challenge:

$$x = f(u_1, u_2) = -u_1 u_2^2 + u_1 u_2 + 17 u_2$$

Calculate the gradient of $f(u_1, u_2)$ at $u = (u_1, u_2)$

Solution:

$$x = f(u_1, \dots, u_n) \rightarrow \nabla f = \begin{bmatrix} \frac{\partial f}{\partial u_1} \\ \vdots \\ \frac{\partial f}{\partial u_n} \end{bmatrix}$$

$$\Rightarrow$$

$$\nabla f = \begin{bmatrix} -u_2^2 + u_2 \\ -2u_1 u_2 + u_1 + 17 \end{bmatrix}$$

Eric Kaigom
kaigom@fb2.fra-uas.de

Application #8

Challenge:

$$x = f(u_1, u_2, u_3) = -u_1 u_2^2 u_3 + u_1 u_2 + 17u_2 + u_3$$

Calculate the gradient of $f(u_1, u_2, u_3)$ at $u = (u_1, u_2, u_3)$

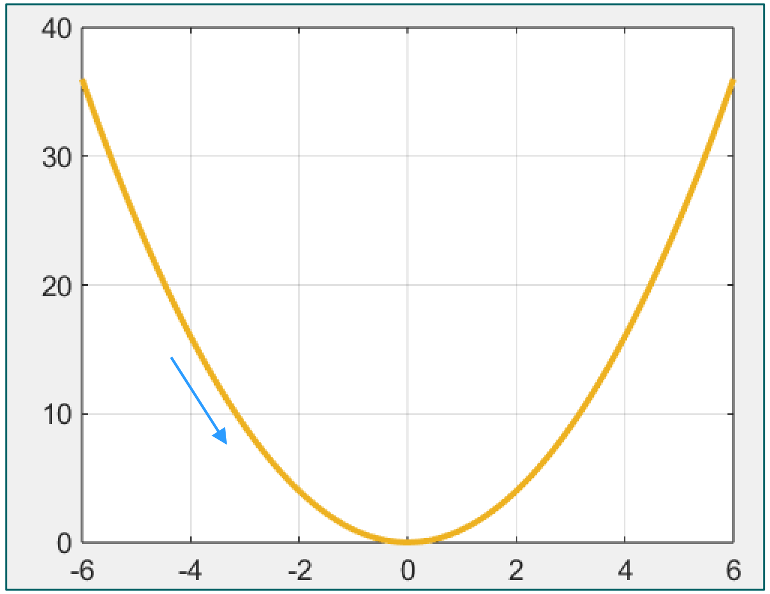
Solution:

$$x = f(u_1, \dots, u_n) \rightarrow \nabla f = \begin{bmatrix} \frac{\partial f}{\partial u_1} \\ \vdots \\ \frac{\partial f}{\partial u_n} \end{bmatrix} \Longrightarrow \nabla f = \begin{bmatrix} -u_2^2 u_3 + u_2 \\ -2u_1 u_3 u_2 + u_1 + 17 \\ -u_1 u_2^2 + 1 \end{bmatrix}$$

Eric Kaigom
kaigom@fb2.fra-uas.de

Why is the gradient useful?

- Suppose that we have a function $x = f(u) = u^2$



How to find u^* that minimize $f(u)$?

Approach:

$$u_{k+1} = u_k - r\overbrace{\nabla f(u)_{u_k}}^{= d} = u_k - r \left.\frac{\partial f(u)}{\partial u}\right|_{u=u_k}$$

- r is a **hyperparameter** that reflects the **learning rate**
- The **fastest decay** of $f(u)$ is the **opposite direction** of the **gradient** $\nabla f(u)$ of $f(u)$
- Hence, $-r\nabla f(u)$ steers the series until the convergence $u_{k+1} \rightarrow u^*$

Source: https://cdn.britannica.com/32/124632-004-1C08C796/craters-Mount-Cameroon.jpg

Caution: Many local optima might exist

Eric Kaigom
kaigom@fb2.fra-uas.de

Optimization routine based upon the gradient

Goal: find minimum of $x = f(u) = u^2$

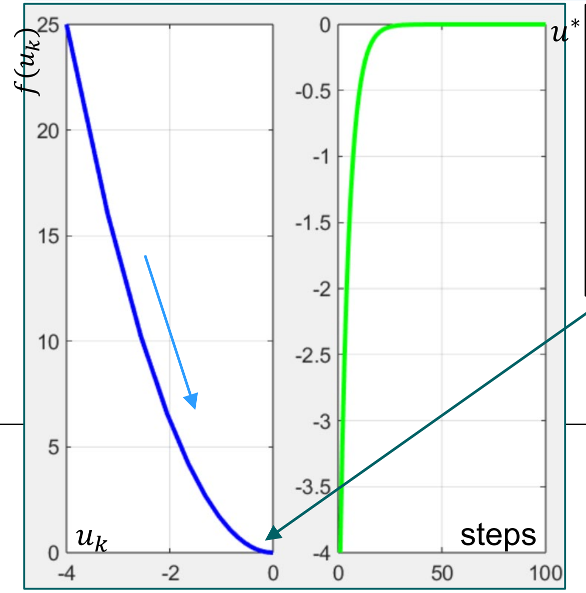
You are welcome to optimize and share the code in our lecture forum!

```
clear;
syms u              %symbolic variable
f =u^2;             %function to be optimized
c = gradient(f,u); %symbolic gardient
uk = -5;          % initial uk value
r = 0.1;          % learning rate
interval=1:100; % 100 steps
U=[]; functionValue=[];%container
for i=interval
 u = uk;
 uk = uk - r*subs(c); %main iteration loop
 U=[U uk];          % for plotting
 functionValue = [functionValue (subs(f))];
end
subplot(1,2,1); plot(U,functionValue,'b');
subplot(1,2,2); plot(interval,U,'g');
```

Approach:

$$u_{k+1} = u_k - r\nabla f(u)_{u_k} = u_k - r\left.\frac{\partial f(u)}{\partial u}\right|_{u=u_k}$$

- r is a **hyperparameter** that reflects the learning rate
- The **fastest decay** of $f(u)$ is the **opposite direction** of the **gradient** $\nabla f(u)$ of $f(u)$
- Hence, $-r\nabla f(u)$ steers the series until the convergence $u_{k+1} \to u^*$



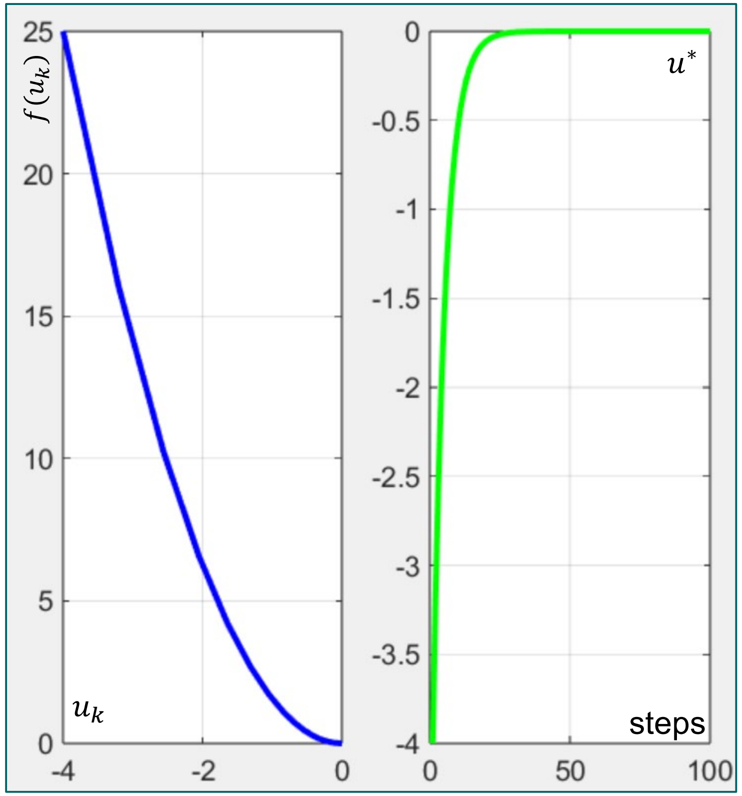
Observations:
- Goal-oriented decay toward minimum value
- Minimum position u^* is attained
- However …

Eric Kaigom
kaigom@fb2.fra-uas.de

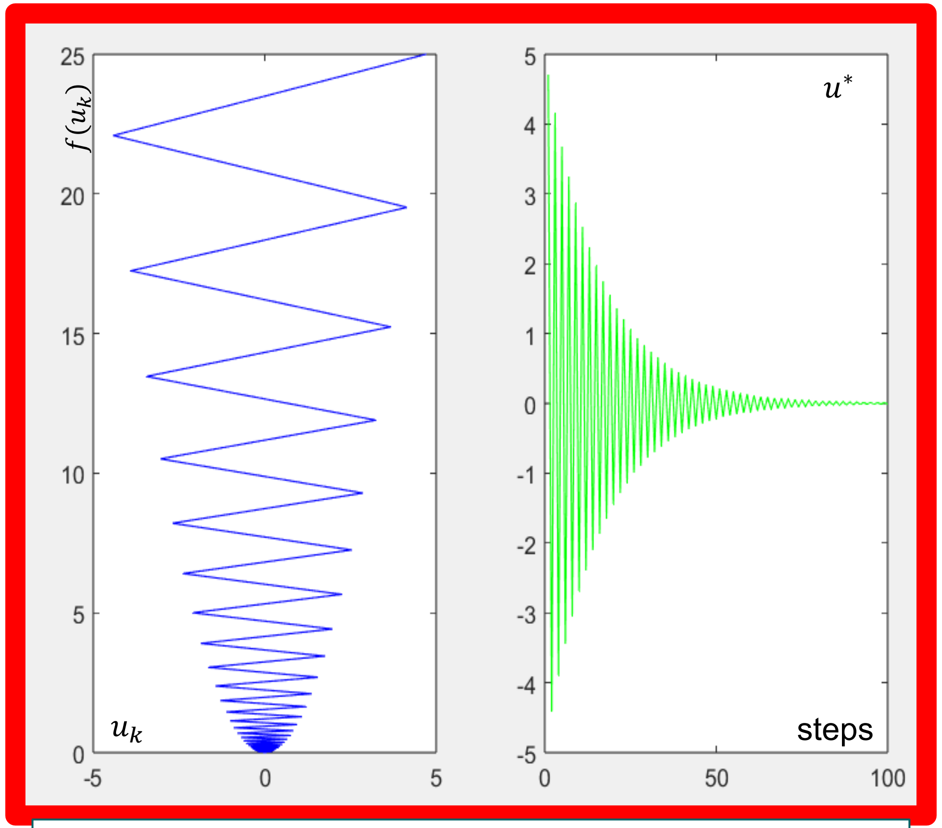
FRANKFURT UNIVERSITY OF APPLIED SCIENCES

We are aware of the gradient usefulness. The path toward usefulness can be struggeling! Why?

- Minimizing $x = f(u) = u^2$ to find minimum $u = u^*$



Learning rate $r = 0.1$. Optimization remains smooth and fast.



Learning rate $r = 0.97$. Optimimization takes longer to converge while oscillating!

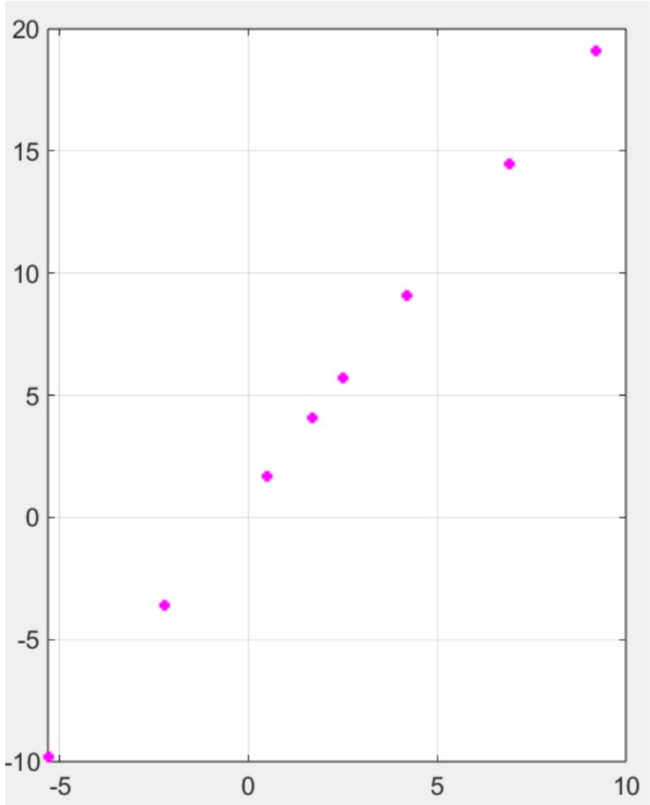
Observation #1: Values of hyperparameters r **impact** optimization processes!

Eric Kaig
kaigom@fb2.fra-uas.de

Linear regression

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Objective of regression – Find a function that captures the set of points

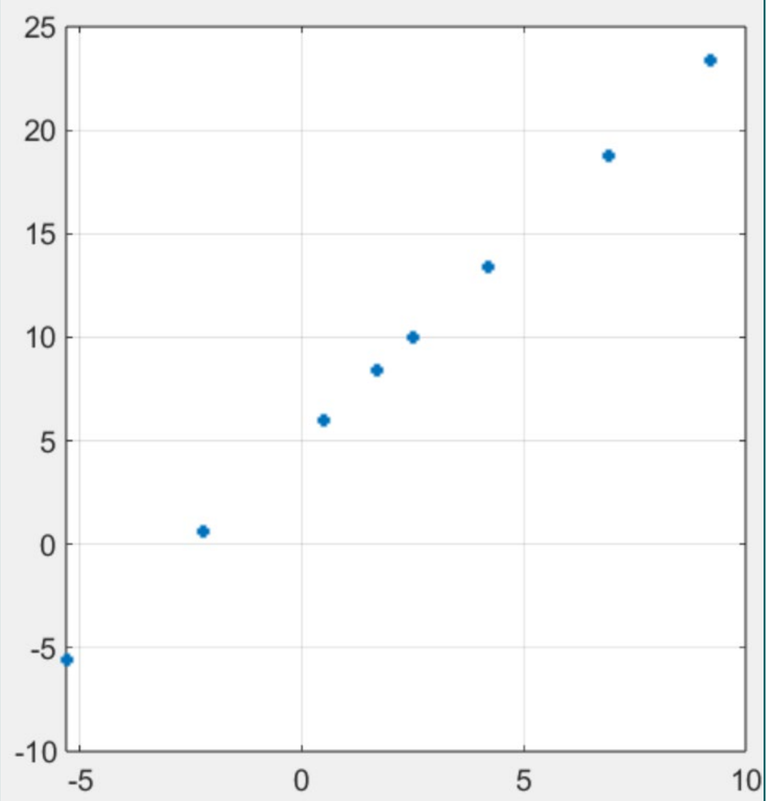


- Input: u = [-5.3 -2.2 0.5 1.7 2.5 4.2 6.9 9.2]^T;
- Output: x = [-9.8 -3.6 1.70 4.1 5.7 9.1 14.5 19.1]^T;

- Points A (u,x) potentially line up with a line L
- If so, then approxim. of L should read: $\hat{x} = wu + b$
- w (a scalar) is the slope, given by the **gradient**
- b (also a scalar) is the „bias“

Eric Kaigom
kaigom@fb2.fra-uas.de

Objective of regression – Find a function that captures the set of points



- u = [-5.3 -2.2 0.5 1.7 2.5 4.2 6.9 9.2 $]^T$;
- x = [-5.6 0.6 6.0 8.4 10.0 13.4 18.8 23.4 $]^T$;

prediction

- Line $\hat{x} = wu + b$
- Loss $L(w,b) = \sum_{i=1}^{n}(\hat{x}_i - x_i)^2$

Goal: learning w and b that min. $L(w,b)$

- Gradient $\nabla L(w,b) = \begin{bmatrix} \frac{\partial L(w,b)}{\partial w} \\ \frac{\partial L(w,b)}{\partial b} \end{bmatrix} = \sum_{i=1}^{n} \begin{bmatrix} 2u(\hat{x}_i - x_i) \\ 2(\hat{x}_i - x_i) \end{bmatrix}$
- $u_{k+1} = u_k - r\nabla L(u)_{u_k}, u_k = [w_k\ b_k]^T$

$\frac{\partial f}{\partial u_1}$ is growth rate of $f([u_1 \cdots u_n])$ in the u_1 direction

Hint:

$x = f(u_1, \dots, u_n) \rightarrow \nabla f = \begin{bmatrix} \frac{\partial f}{\partial u_1} \\ \vdots \\ \frac{\partial f}{\partial u_n} \end{bmatrix}$ is a vector of $\mathbb{R}^n$!

∇f is the direction of the steepest **ascent** (hence, -∇f is the direction of steepest **descent**) at u.

Hint:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{x}}\frac{\partial \hat{x}}{\partial w}$$
$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial \hat{x}}\frac{\partial \hat{x}}{\partial b}$$

Eric Kaigom
kaigom@fb2.fra-uas.de

Optimization pitfall

Source: https://cdn.britannica.com/32/124632-004-1C08C796/craters-Mount-Cameroon.jpg

Eric Kaigom
kaigom@fb2.fra-uas.de

When to update the parameters?

Hint:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{x}} \frac{\partial \hat{x}}{\partial w}$$
$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial \hat{x}} \frac{\partial \hat{x}}{\partial b}$$

- Line $\hat{x} = wu + b$
- Loss $L(w,b) = \sum_{i=1}^{n} (\hat{x}_i - x_i)^2$
- Gradient $\nabla L(w,b) = \begin{bmatrix} \frac{\partial L(w,b)}{\partial w} \\ \frac{\partial L(w,b)}{\partial b} \end{bmatrix} = \sum_{i=1}^{n} \begin{bmatrix} 2u(\hat{x}_i - x_i) \\ 2(\hat{x}_i - x_i) \end{bmatrix}$
- $u_{k+1} = u_k - r\nabla L(u)_{u_k}; u_k = [w_k \; b_k]^T$

Gradient descent:

Update after a loss evalation related to all training data samples (after each **epoch**)

Stochastic gradient descent:

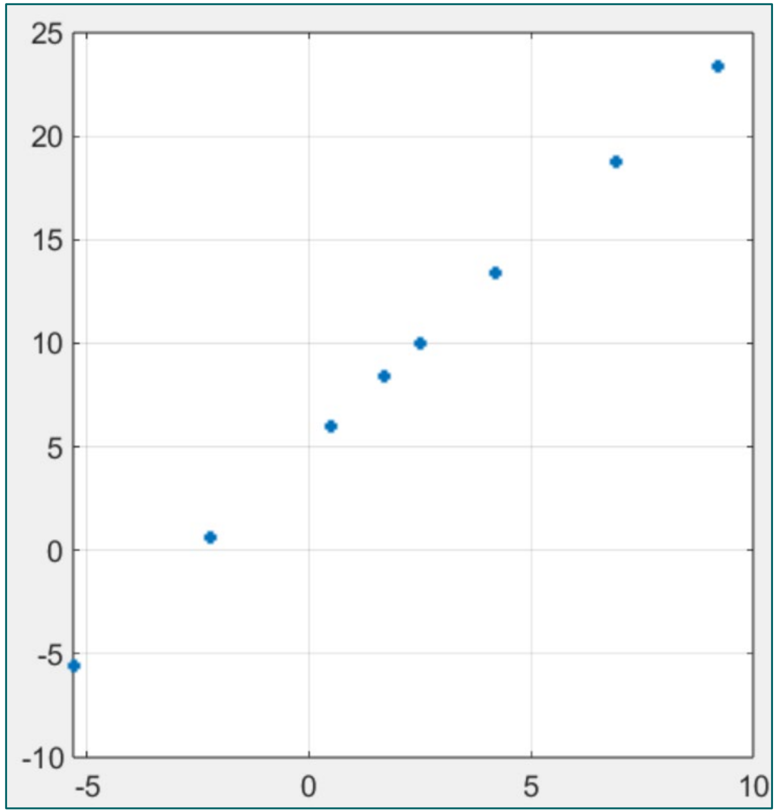
Update after the loss related to a training data sample has been computed

Mini-batch stochastic gradient descent:

Update after the loss related to a number of training data samples has been computed

Eric Kaigom
kaigom@fb2.fra-uas.de

Application #9: Write and execute a Matlab code that learns w and b!



- u = [-5.3 -2.2 0.5 1.7 2.5 4.2 6.9 9.2 $]^T$;
- x = [-5.6 0.6 6.0 8.4 10.0 13.4 18.8 23.4 $]^T$;

prediction

- Line $\hat{x} = wu + b$
- Loss $L(w,b) = \sum_{i=1}^{n}(\hat{x}_i - x_i)^2$

Goal: learning w and b that min. $L(w,b)$

- Gradient $\nabla L(w,b) = \begin{bmatrix} \frac{\partial L(w,b)}{\partial w} \\ \frac{\partial L(w,b)}{\partial b} \end{bmatrix} = \sum_{i=1}^{n} \begin{bmatrix} 2u(\hat{x}_i - x_i) \\ 2(\hat{x}_i - x_i) \end{bmatrix}$
- $u_{k+1} = u_k - r\nabla L(u)_{u_k}, u_k = [w_k\ b_k]^T$

$\frac{\partial f}{\partial u_1}$ is growth rate of $f([u_1 \cdots u_n])$ in the u_1 direction

Hint:

$$x = f(u_1, \ldots, u_n) \rightarrow \nabla f = \begin{bmatrix} \frac{\partial f}{\partial u_1} \\ \vdots \\ \frac{\partial f}{\partial u_n} \end{bmatrix}$$ is a vector of $\mathbb{R}^n$!

∇f is the direction of the steepest **ascent** (hence, -∇f is the direction of steepest **descent**) at u.

Hint:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{x}}\frac{\partial \hat{x}}{\partial w}$$
$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial \hat{x}}\frac{\partial \hat{x}}{\partial b}$$

Eric Kaigom
kaigom@fb2.fra-uas.de

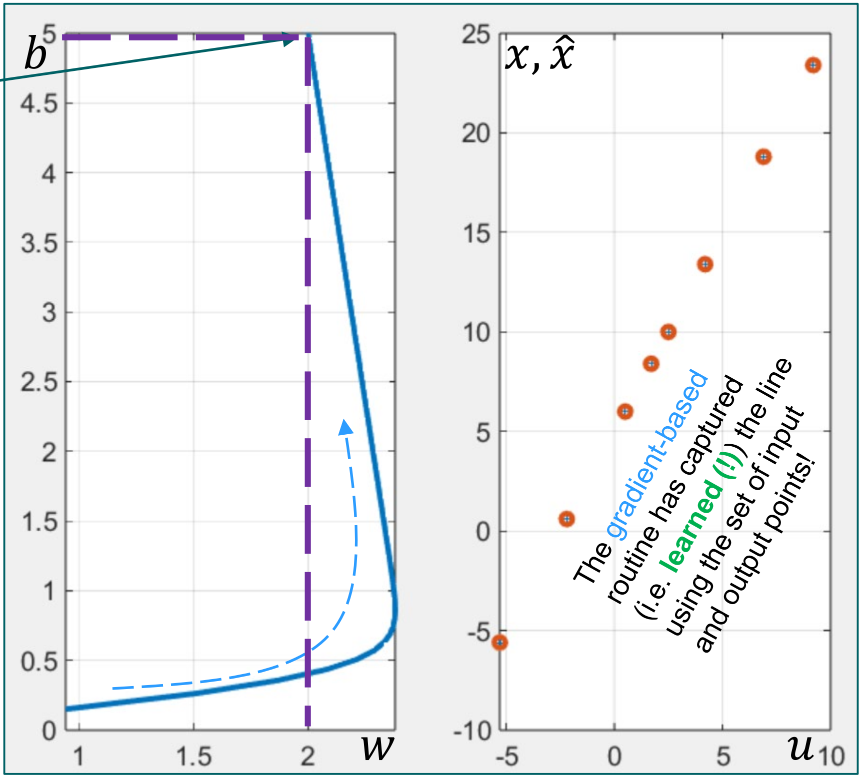
Objective of regression – Find a function that captures the set of points

```
clear; clc;
input = [-5.3 -2.2 0.5 1.7 2.5 4.2 6.9 9.2];
output=[-5.6 0.6 6.0 8.4 10.0 13.4 18.8 23.4];%ou = 2*in+5
uk=[0;0];
r = 0.001; U = []; w = 0; b = 0;
for i = 1:10000
uk = uk -r*gradientloss(uk,input,output);
U = [U uk]; w = uk(1); b = uk(2);
end

subplot(1,2,1); plot(U(1,:),U(2,:));
subplot(1,2,2); plot(input,output,'+'); hold on; plot(input,w*input+b,'o');

function [r] =gradientloss (uk, in, out)
r1 = sum(2*(uk(1)*in+uk(2)-out).*in);
r2 = sum(2*(uk(1)*in+uk(2)-out)    );
r=[r1;r2];
end
```

Ground truth



Once we have the weights w and b, we have **captured** the mapping between input and output data!

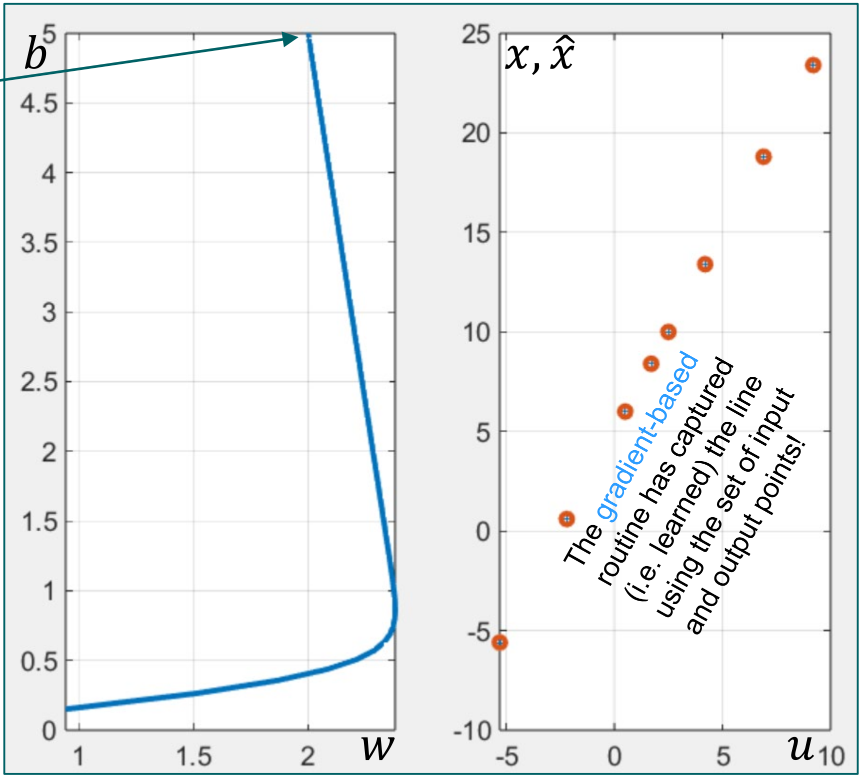
Eric Kaigom
kaigom@fb2.fra-uas.de

Objective of regression – Find a function that captures the set of points

```
clear; clc;
input = [-5.3 -2.2 0.5 1.7 2.5 4.2 6.9 9.2];
output=[-5.6 0.6 6.0 8.4 10.0 13.4 18.8 23.4]; %ou = 2*in+5
uk=[0;0];
r = 0.001; U = []; w = 0; b = 0;
for i = 1:10000
uk = uk -r*gradientloss(uk,input,output);
U = [U uk]; w = uk(1); b = uk(2);
end

subplot(1,2,1); plot(U(1,:),U(2,:));
subplot(1,2,2); plot(input,output,'+'); hold on; plot(input,w*input+b,'o');

function [r] =gradientloss (uk, in, out)
r1 = sum(2*(uk(1)*in+uk(2)-out).*in);
r2 = sum(2*(uk(1)*in+uk(2)-out)    );
r=[r1;r2];
end
```

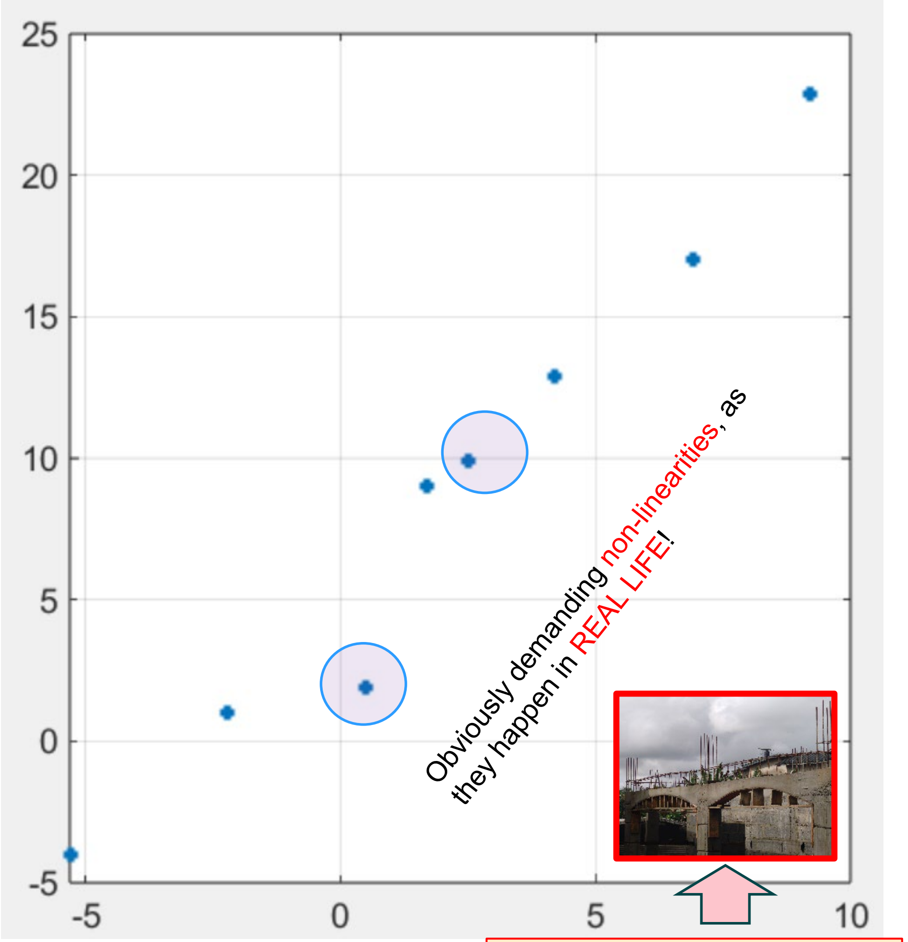


Observation #2: **Gradient** helps **LEARN** the mapping (Here: $\hat{x} = wu + b$) using input & output data! **However…**

Eric Kaigom
kaigom@fb2.fra-uas.de

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Objective – Find a function that captures the set of points



Recall: the real world (and thus application data…) is (mostly) non-linear.

- u = [-5.3 -2.2 0.5 1.7 2.5 4.2 6.9 9.2];
- x = [-4.0 1.0 1.9 9.0 9.90 12.9 17.0 22.9];

Hint:

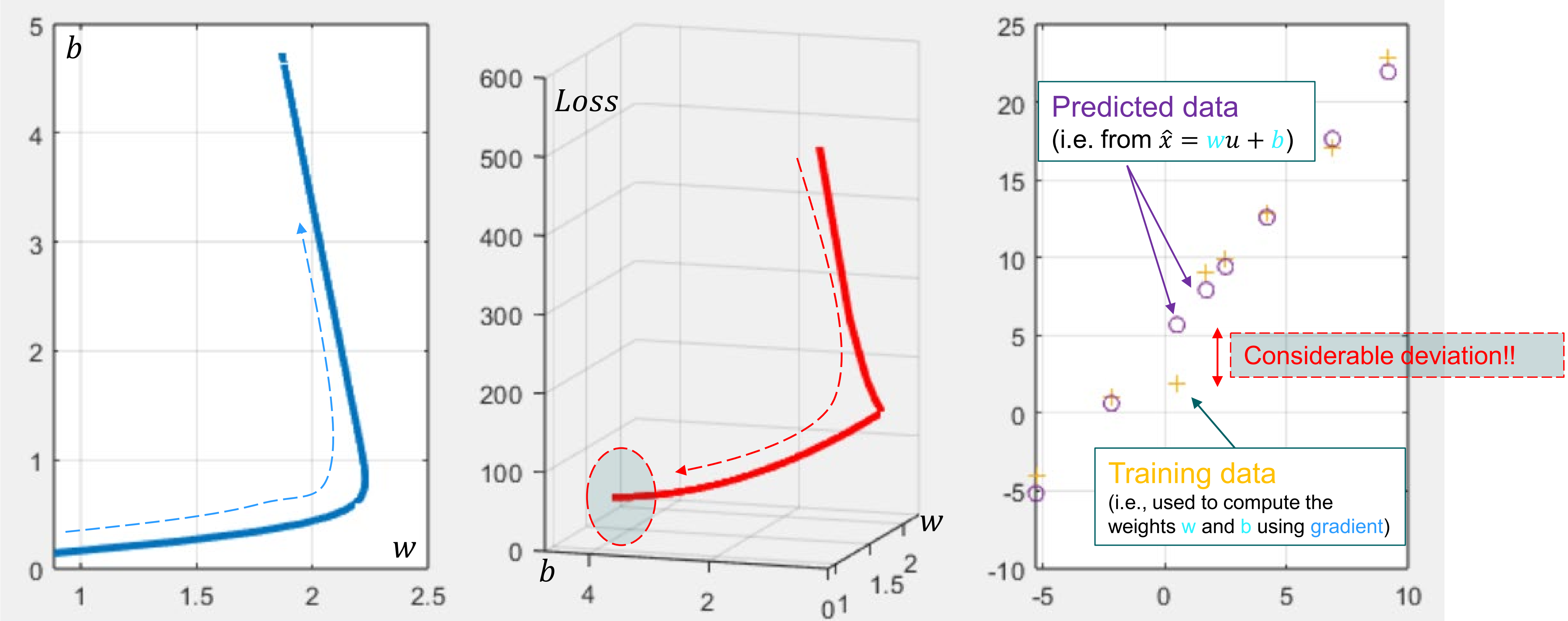
$$x = f(u_1, \dots, u_n) \rightarrow \nabla f = \begin{bmatrix} \frac{\partial f}{\partial u_1} \\ \vdots \\ \frac{\partial f}{\partial u_n} \end{bmatrix}$$

- Still a line $\hat{x} = wu + b$?!
- Loss $L(w,b) = \sum_{i=1}^{n} (\hat{x}_i - x_i)^2$
- Gradient $\nabla L(w,b) = \begin{bmatrix} \frac{\partial L(w,b)}{\partial w} \\ \frac{\partial L(w,b)}{\partial b} \end{bmatrix} = \sum_{i=1}^{n} \begin{bmatrix} 2u(\hat{x}_i - x_i) \\ 2(\hat{x}_i - x_i) \end{bmatrix}$
- $u_{k+1} = u_k - r\nabla L(u)_{u_k}, u_k = [w_k \; b_k]^T$

…**how good** does this approach with **linearity assumption** perform?!

Eric Kaigom
kaigom@fb2.fra-uas.de

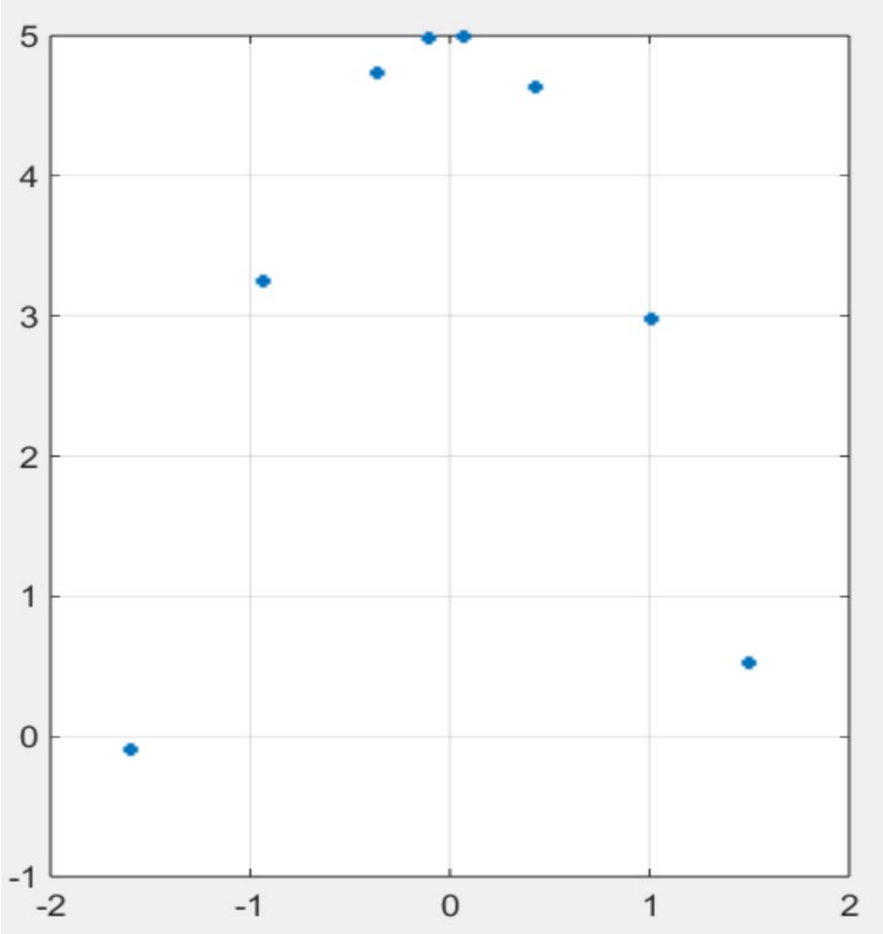
Objective – Find a function that captures the set of points



Dynamics of optimization of weights

Eric Kaigom
kaigom@fb2.fra-uas.de

Objective – Find a function that captures the set of points



- u = [-1.5967 -0.9356 -0.3599 -0.1040 0.0666 0.4292 1.0049 1.4954];
- x = [-0.0990 3.2492 4.7410 4.9784 4.9911 4.6316 2.9802 0.5275];

- Obviously not a line $\hat{x} = wu + b$!
- Loss $L(w,b) = \sum_{i=1}^{n} (\hat{x}_i - x_i)^2$
- Gradient $\nabla L(w,b) = \begin{bmatrix} \frac{\partial L(w,b)}{\partial w} \\ \frac{\partial L(w,b)}{\partial b} \end{bmatrix} = \sum_{i=1}^{n} \begin{bmatrix} 2u(\hat{x}_i - x_i) \\ 2(\hat{x}_i - x_i) \end{bmatrix}$
- $u_{k+1} = u_k - r\nabla L(u)_{u_k}, u_k = [w_k \; b_k]^T$

Hint:

$$x = f(u_1, \dots, u_n) \rightarrow \nabla f = \begin{bmatrix} \frac{\partial f}{\partial u_1} \\ \vdots \\ \frac{\partial f}{\partial u_n} \end{bmatrix}$$

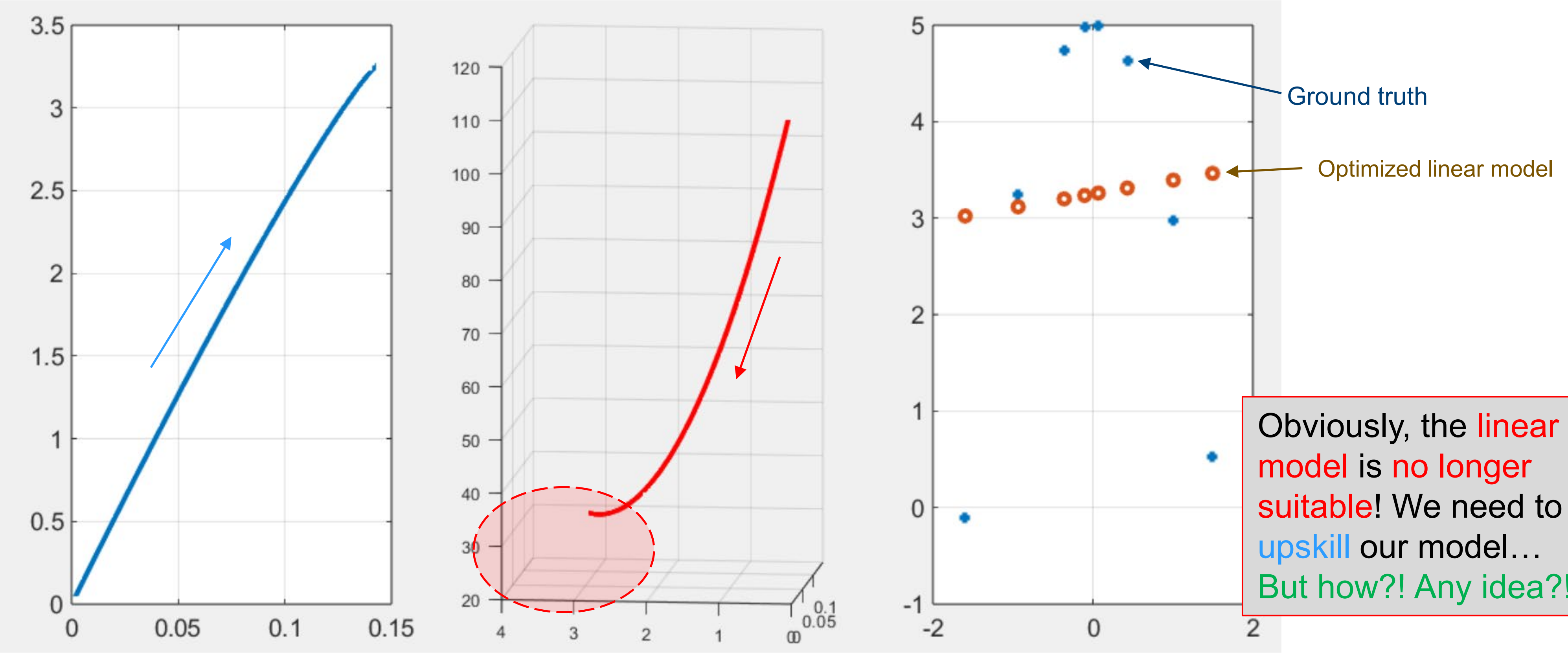
…how good does this approach with linearity assumption perform?!

Eric Kaigom
kaigom@fb2.fra-uas.de

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Robotics and Machine Learning – Model Capture

Objective – Find a function that captures the set of points



Obviously, the linear model is no longer suitable! We need to upskill our model… But how?! Any idea?!

Dynamics of optimization of weights

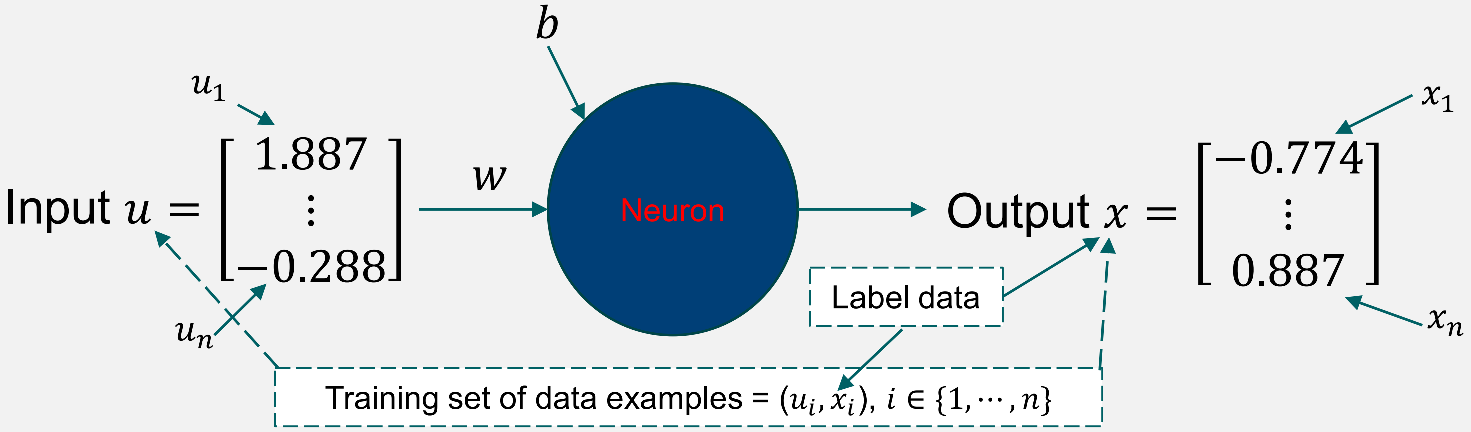
Loss

Training and prediction data

Observation #3: The performance of $\hat{x} = wu + b$ drops as non-linearities and outliers are involved in training data!

Learning using a neuron

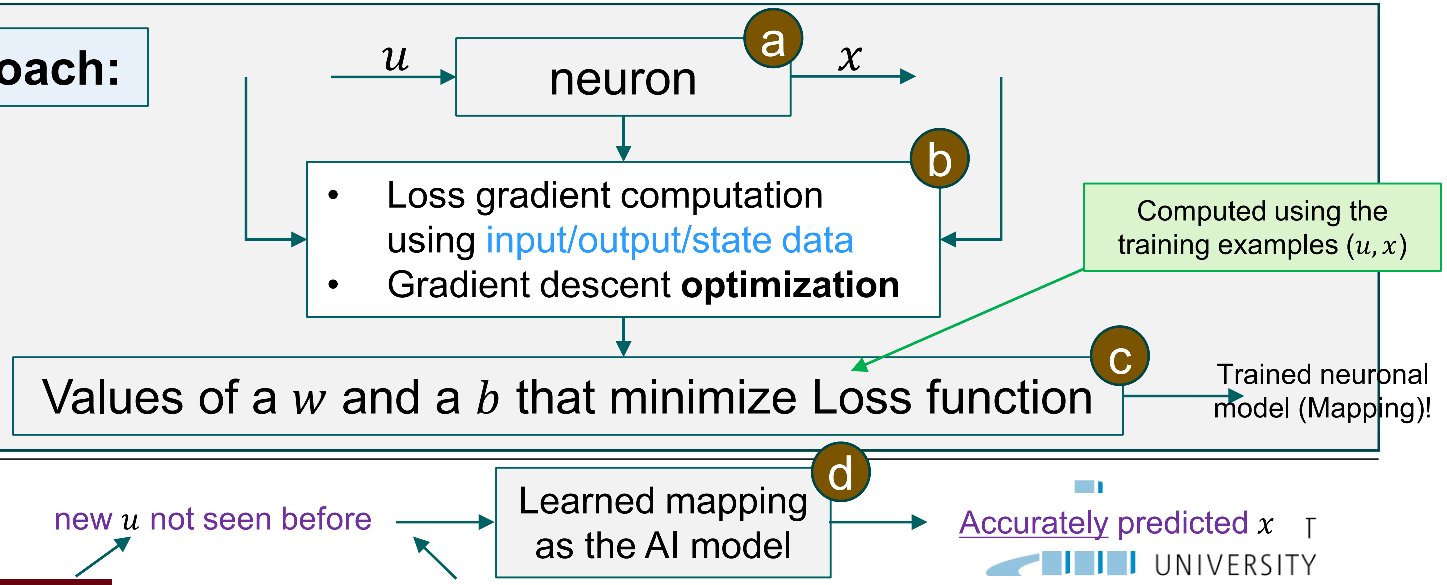
1 neuron case : Optimization of the weights using (Input data, Output data) and gradient



Objectives: How to find suitable w and b, two scalars, that (learn to) **associate (or map)** multiple input examples in u with corresponding output examples (i.e., labels) in x ?

.. and with good generalization!

Approach:



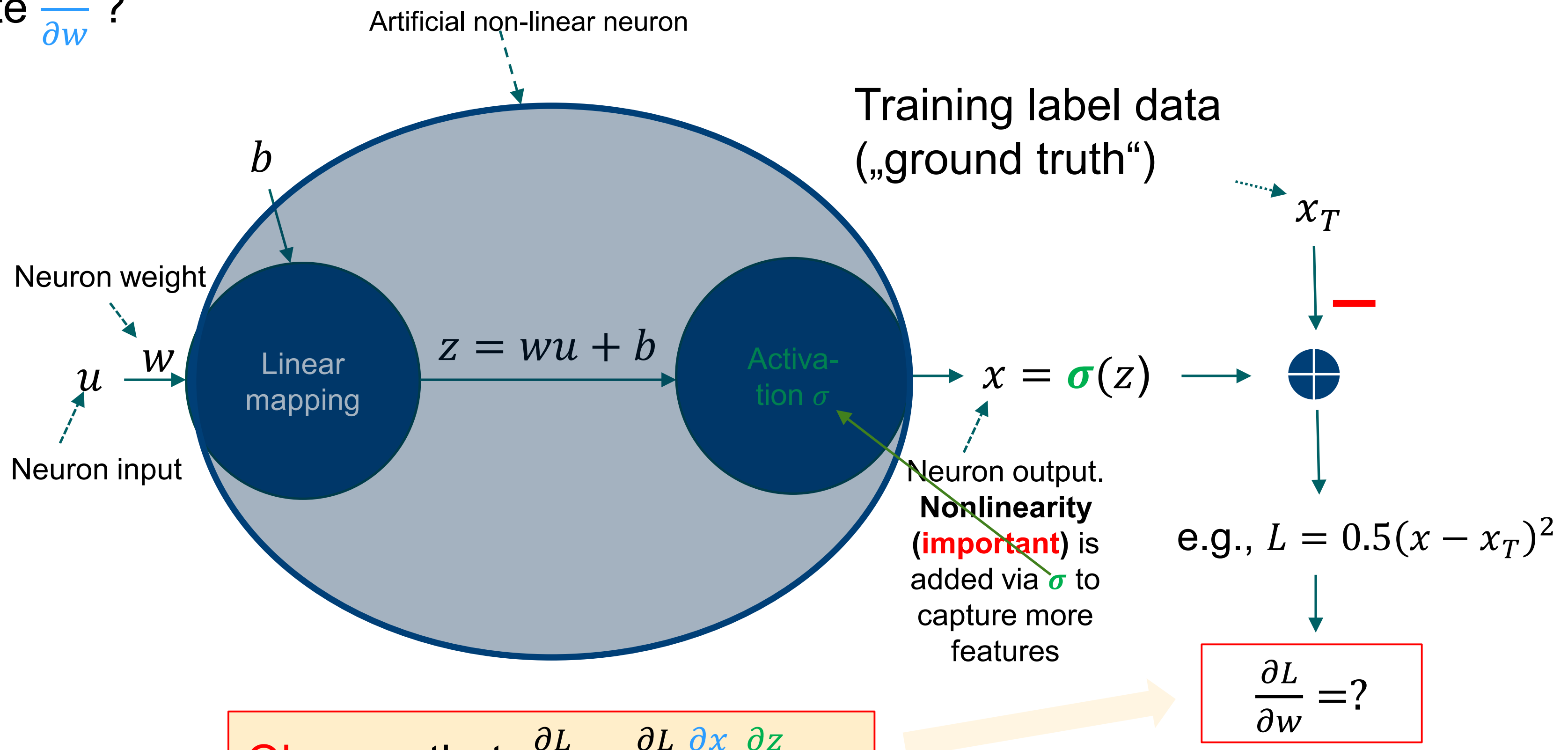
(Use in tests/applications testdata **normalized** with same normalization parameters as training)

Note: Similar distribution usually assumed (realistic?). However, out of the scope of this lecture!

Eric Kaigom
kaigom@fb2.fra-uas.de

Neuron-related gradient computation

Objective: How to compute $\frac{\partial L}{\partial w}$?



Observe that: $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial x}\frac{\partial x}{\partial z}\frac{\partial z}{\partial w}$

Eric Kaigom
kaigom@fb2.fra-uas.de
FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Neuron-related gradient computation

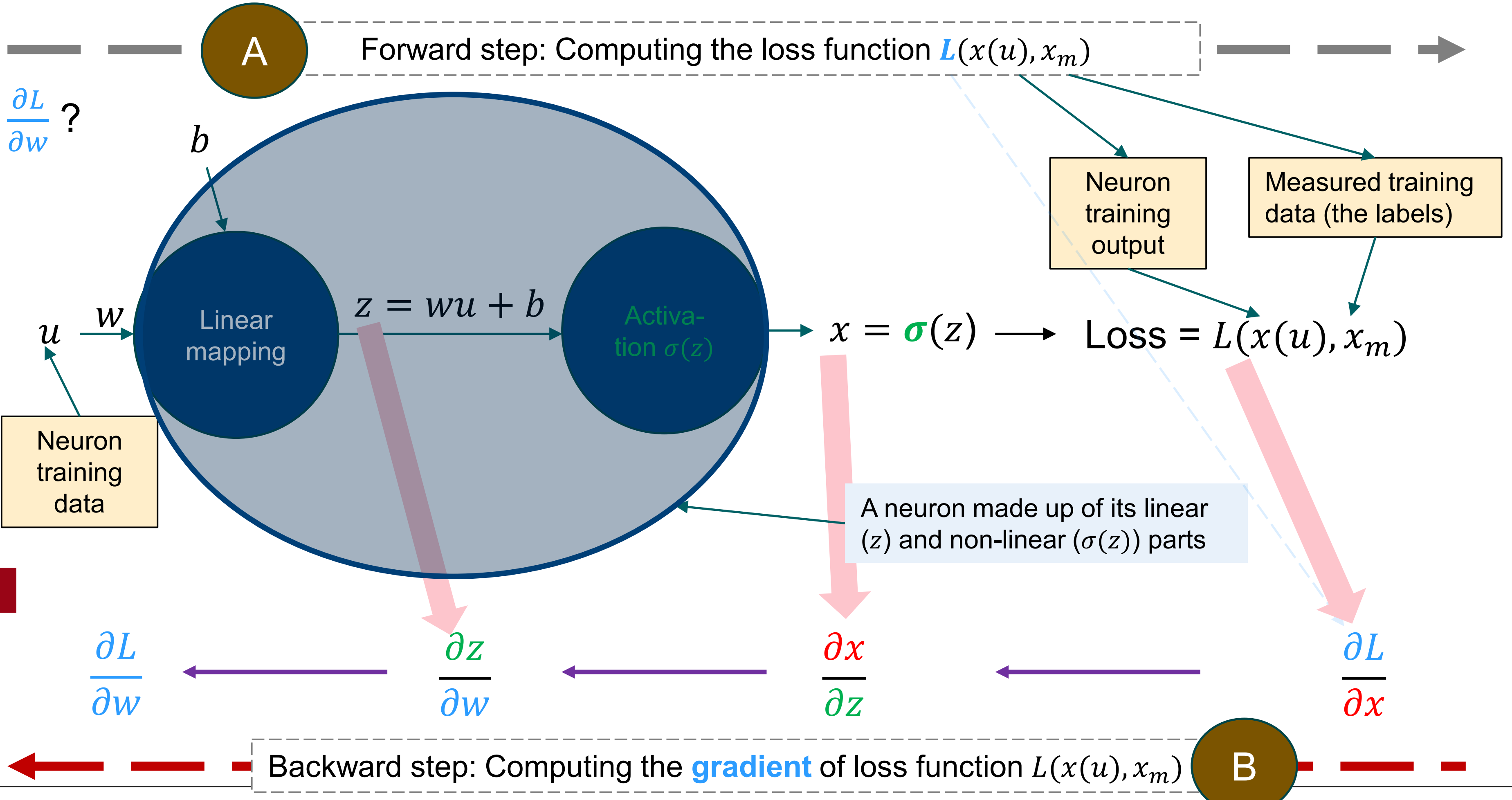
Objective: How to compute $\frac{\partial L}{\partial w}$?

Rationale: We need the derivative (i.e., $\frac{\partial L}{\partial w}$) to be aware of the **direction of fastest decrease** (given by $-\frac{\partial L}{\partial w}$) of L as a function of w to update w_{k+1}. Recall that we strive to find $w_{k+1} = w_k - r\frac{\partial L}{\partial w} \rightarrow w^*$ that minimizes L!

This has implications:

How? Observe that:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial x}\frac{\partial x}{\partial z}\frac{\partial z}{\partial w}$$



Observation: Gradient $\frac{\partial L}{\partial w}$ computed (to optimize w) through a **backward propagation**!

Neuron-related gradient computation

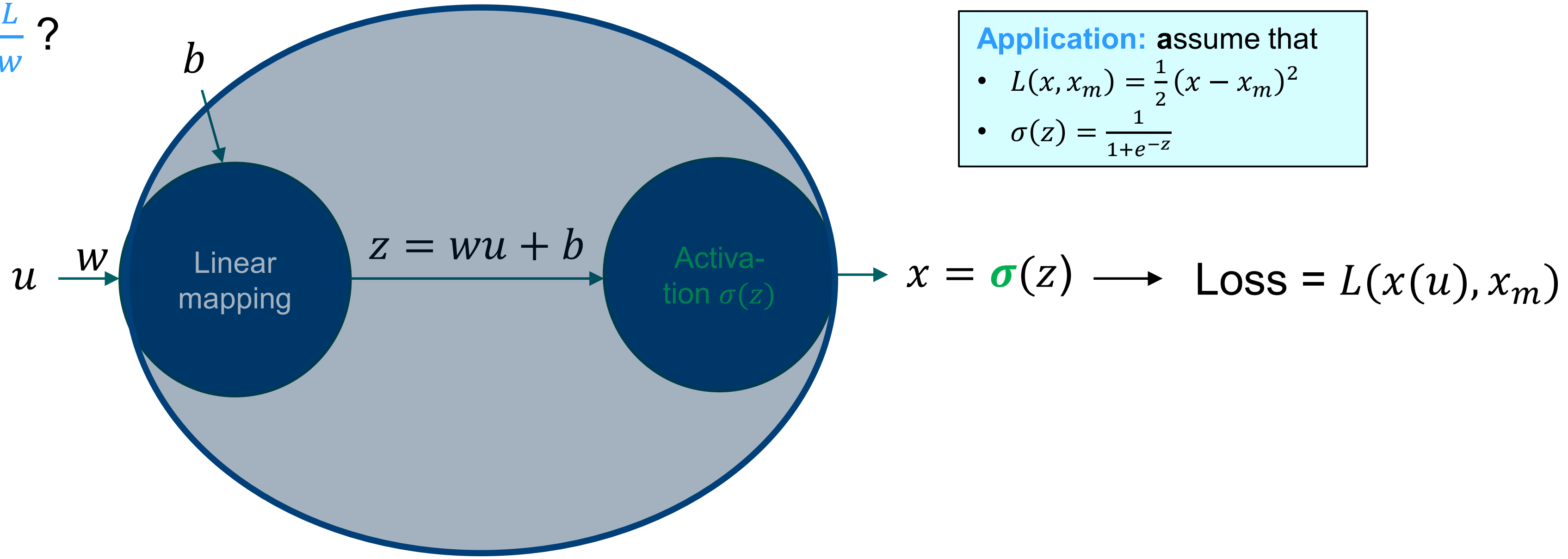
Forward step: Computing the loss function $L(x(u), x_m)$

Objective: How to compute $\frac{\partial L}{\partial w}$?

Rationale: We need the derivative (i.e., $\frac{\partial L}{\partial w}$) to be aware of the **direction of fastest decrease** (given by $-\frac{\partial L}{\partial w}$) of L as a function of w to update w_{k+1}. Recall that we strive to find

$$w_{k+1} = w_k - r\frac{\partial L}{\partial w} \to w^*$$

that minimizes L!



Application: assume that
- $L(x, x_m) = \frac{1}{2}(x - x_m)^2$
- $\sigma(z) = \frac{1}{1+e^{-z}}$

How? Observe that:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial x}\frac{\partial x}{\partial z}\frac{\partial z}{\partial w}$$

$$\frac{\partial L}{\partial w} \leftarrow \frac{\partial z}{\partial w} \leftarrow \frac{\partial x}{\partial z} \leftarrow \frac{\partial L}{\partial x}$$

Backward step: Computing the **gradient** of loss function $L(x(u), x_m)$

$$\boxed{\frac{\partial L}{\partial w} = (x - x_m)\sigma(z)\big(1 - \sigma(z)\big)u}$$

$$\frac{\partial z}{\partial w} = u$$

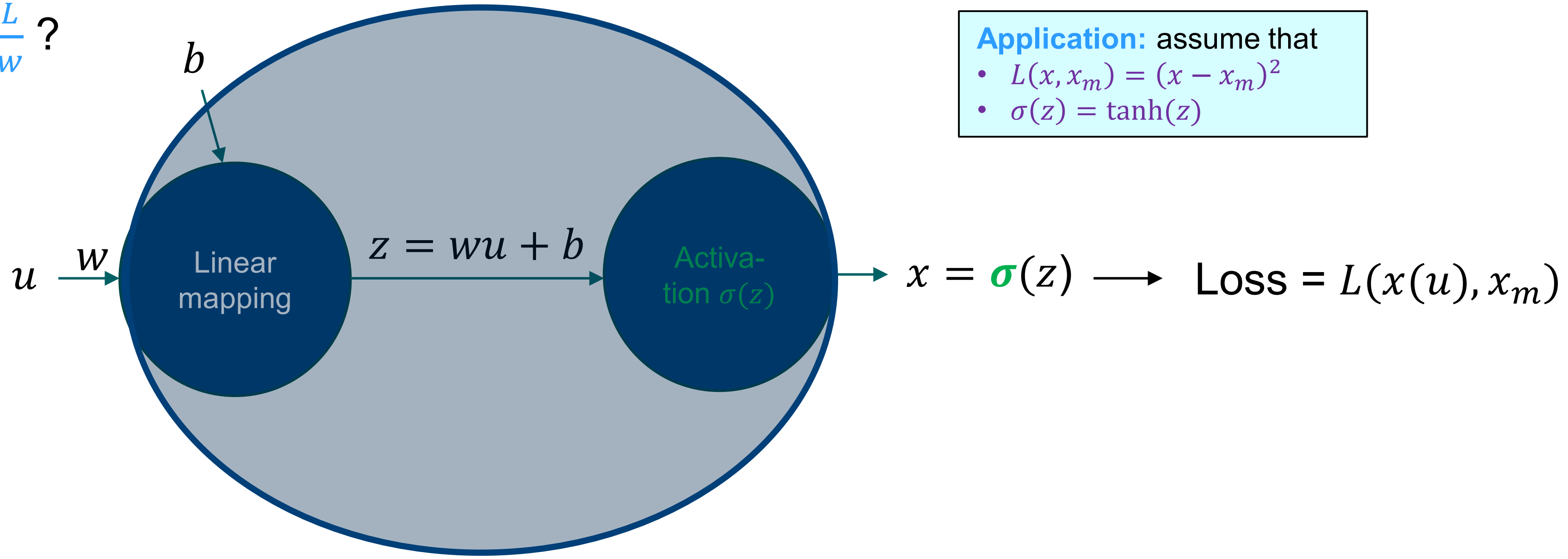
$$\frac{\partial x}{\partial z} = \sigma(z)\big(1 - \sigma(z)\big)$$

$$\frac{\partial L}{\partial x} = (x - x_m)$$

Neuron-related gradient computation

Forward step: Computing the loss function $L(x(u), x_m)$

Objective: How to compute $\frac{\partial L}{\partial w}$?



Application: assume that
- $L(x, x_m) = (x - x_m)^2$
- $\sigma(z) = \tanh(z)$

Rationale: We need the derivative (i.e., $\frac{\partial L}{\partial w}$) to be aware of the **direction of fastest decrease** (given by $-\frac{\partial L}{\partial w}$) of L as a function of w to update w_{k+1}.

Recall that we strive to find
$w_{k+1} = w_k - r\frac{\partial L}{\partial w} \rightarrow w^*$
that minimizes the loss L!

How? Observe that:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial x}\frac{\partial x}{\partial z}\frac{\partial z}{\partial w}$$

$$\frac{\partial L}{\partial w} \leftarrow \frac{\partial z}{\partial w} \leftarrow \frac{\partial x}{\partial z} \leftarrow \frac{\partial L}{\partial x}$$

Backward ste Computing the **gradient** c oss function $L(x(u), x_m)$

$$\boxed{\frac{\partial L}{\partial w} = 2(x - x_m)(1 - \sigma^2(z))u}$$

$\frac{\partial z}{\partial w} = u$

$\frac{\partial x}{\partial z} = 1 - \sigma^2(z)$

$\frac{\partial L}{\partial x} = 2(x - x_m)$

Eric K
kaigor

Neuron-related gradient computation

Known activation function $\sigma(z)$

Assume that
- $L(x, x_m) = (x - x_m)^2$
- $\sigma(z) = \tanh(z)$

$$\frac{\partial L}{\partial w} \leftarrow \frac{\partial z}{\partial w} \leftarrow \frac{\partial x}{\partial z} \leftarrow \frac{\partial L}{\partial x}$$

Backward step: Computing the **gradient** of loss function $L(x(u), x_m)$

$$\boxed{\frac{\partial L}{\partial w} = \mathbf{2(x - x_m)(1 - \sigma^2(z))}u} \qquad \frac{\partial z}{\partial w} = u \qquad \frac{\partial x}{\partial z} = 1 - \sigma^2(z) \qquad \frac{\partial L}{\partial x} = 2(x - x_m)$$

Assume that
- $L(x, x_m) = \frac{1}{2}(x - x_m)^2$
- $\sigma(z) = \frac{1}{1+e^{-z}}$

$$\frac{\partial L}{\partial w} \leftarrow \frac{\partial z}{\partial w} \leftarrow \frac{\partial x}{\partial z} \leftarrow \frac{\partial L}{\partial x}$$

Backward step: Computing the **gradient** of loss function $L(x(u), x_m)$

$$\boxed{\frac{\partial L}{\partial w} = \mathbf{(x - x_m)\sigma(z)(1 - \sigma(z))}u} \qquad \frac{\partial z}{\partial w} = u \qquad \frac{\partial x}{\partial z} = \sigma(z)(1 - \sigma(z)) \qquad \frac{\partial L}{\partial x} = (x - x_m)$$

Observe that:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial x}\frac{\partial x}{\partial z}\frac{\partial z}{\partial w} = p\frac{\partial z}{\partial w}$$

with $p = \frac{\partial L}{\partial x}\frac{\partial x}{\partial z} = \frac{\partial L}{\partial z}$

Impact of the activation function on p is kept hidden!

Unknown activation function $\sigma(z)$

Robotics and Machine Learning – Model Capture

Neuron-related gradient computation

Known activation function $\sigma(z)$

Assume that
- $L(x, x_m) = (x - x_m)^2$
- $\sigma(z) = \tanh(z)$

$\frac{\partial L}{\partial w} \leftarrow \frac{\partial z}{\partial w} \leftarrow \frac{\partial x}{\partial z} \leftarrow \frac{\partial L}{\partial x}$

Backward step: Computing the **gradient** of loss function $L(x(u), x_m)$

$\frac{\partial L}{\partial x} = 2(x - x_m)$

$\frac{\partial x}{\partial z} = 1 - \sigma^2(z)$

$\frac{\partial z}{\partial w} = u$

$\boxed{\frac{\partial L}{\partial w} = \mathbf{2(x - x_m)(1 - \sigma^2(z))}u}$

Assume that
- $L(x, x_m) = \frac{1}{2}(x - x_m)^2$
- $\sigma(z) = \frac{1}{1+e^{-z}}$

$\frac{\partial L}{\partial w} \leftarrow \frac{\partial z}{\partial w} \leftarrow \frac{\partial x}{\partial z} \leftarrow \frac{\partial L}{\partial x}$

Backward step: Computing the **gradient** of loss function $L(x(u), x_m)$

$\frac{\partial L}{\partial x} = (x - x_m)$

$\frac{\partial x}{\partial z} = \sigma(z)(1 - \sigma(z))$

$\frac{\partial z}{\partial w} = u$

$\boxed{\frac{\partial L}{\partial w} = \mathbf{(x - x_m)\sigma(z)(1 - \sigma(z))}u}$

Observe that:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial x}\frac{\partial x}{\partial z}\frac{\partial z}{\partial w} = pu$$

with $p = \frac{\partial L}{\partial x}\frac{\partial x}{\partial z} = \frac{\partial L}{\partial z}$ ←

Weight update: $w_{k+1} = w_k - r\frac{\partial L}{\partial w} = w_{k+1} + w_k - rpu$

Unknown activation function $\sigma(z)$

Application

1) Generate the **set** $\boldsymbol{u}$ of **input training data** made up of **10000 random numbers** between -1 and 1

2) Generate the set $y = x_m$of label data with $\boldsymbol{y} = \boldsymbol{x_m}$ = **243.5∗$\boldsymbol{u}$ − 162**

3) Implement a neuron with **input** $\boldsymbol{u}$, **weight** $\boldsymbol{w}$**, bias** $\boldsymbol{b}$**, state** $\boldsymbol{z}$**, identity output** $\boldsymbol{x}$ as activation (i.e., $x(z) = \sigma(z) = z$) in Matlab. The neuron maps the input signal $\boldsymbol{u}$ to an ouput signal $\boldsymbol{x}$

4) Write a loop (the maximum number of iterations is $\mathbf{10^4}$) that uses the neuron to run the
   a) **Forward propagation** (from u to the current neural network output x)
   b) **Compute the gradient of the loss function** $\boldsymbol{L}$ (what is a **useful** one?) by **using** $\boldsymbol{u}, \boldsymbol{x}$**, and** $\boldsymbol{x_m}$
   c) **Update the weights (**$\boldsymbol{w}, \boldsymbol{b}$**)** of the neuron

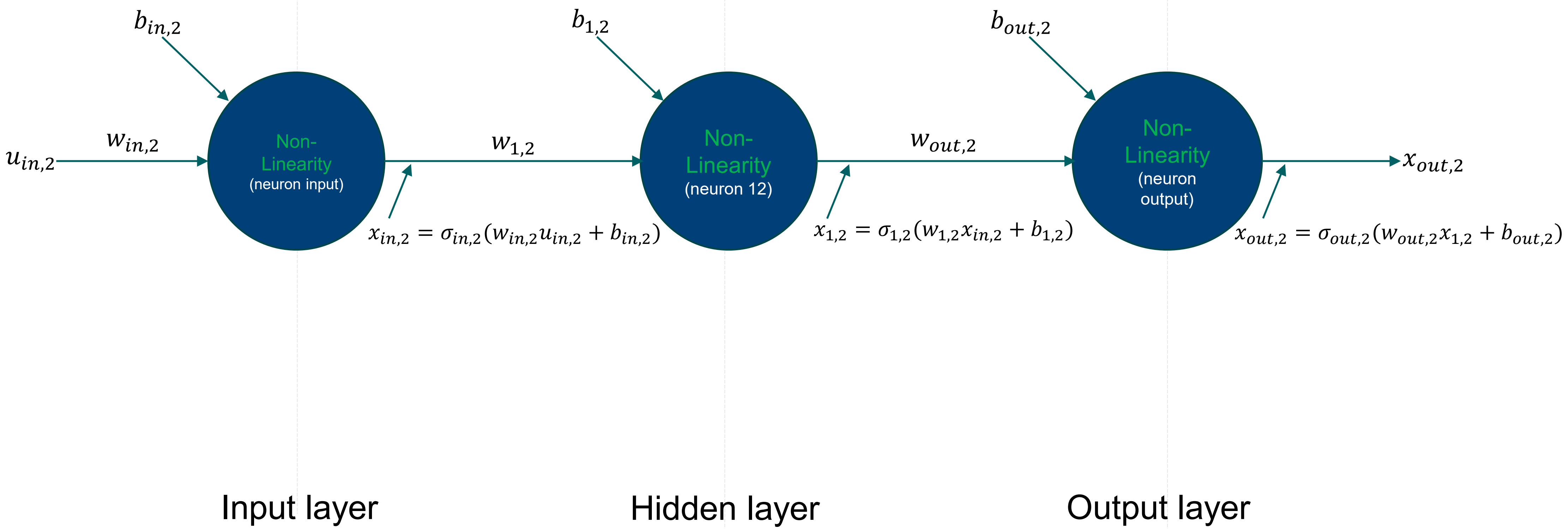
5) Did you retrieve ($w = 243.5, b = 162$)? What do you observe while increasing/decrasing the learning rate r?

6) Repeat from 2) with $x_m = 2u^2$+1. What did you observe?

Eric Kaigom
kaigom@fb2.fra-uas.de

Learning with several fully connected neurons

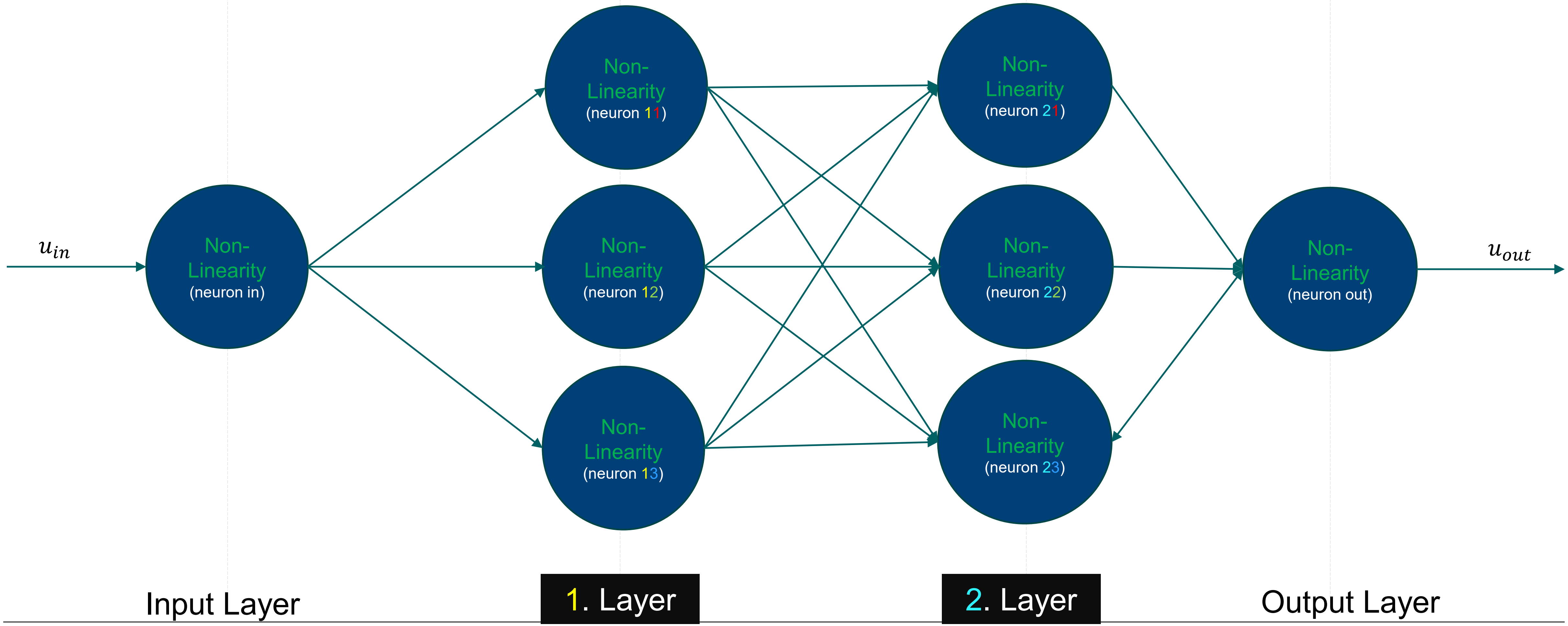
Neural network



Eric Kaigom
kaigom@fb2.fra-uas.de

Robotics and Machine Learning – Model Capture

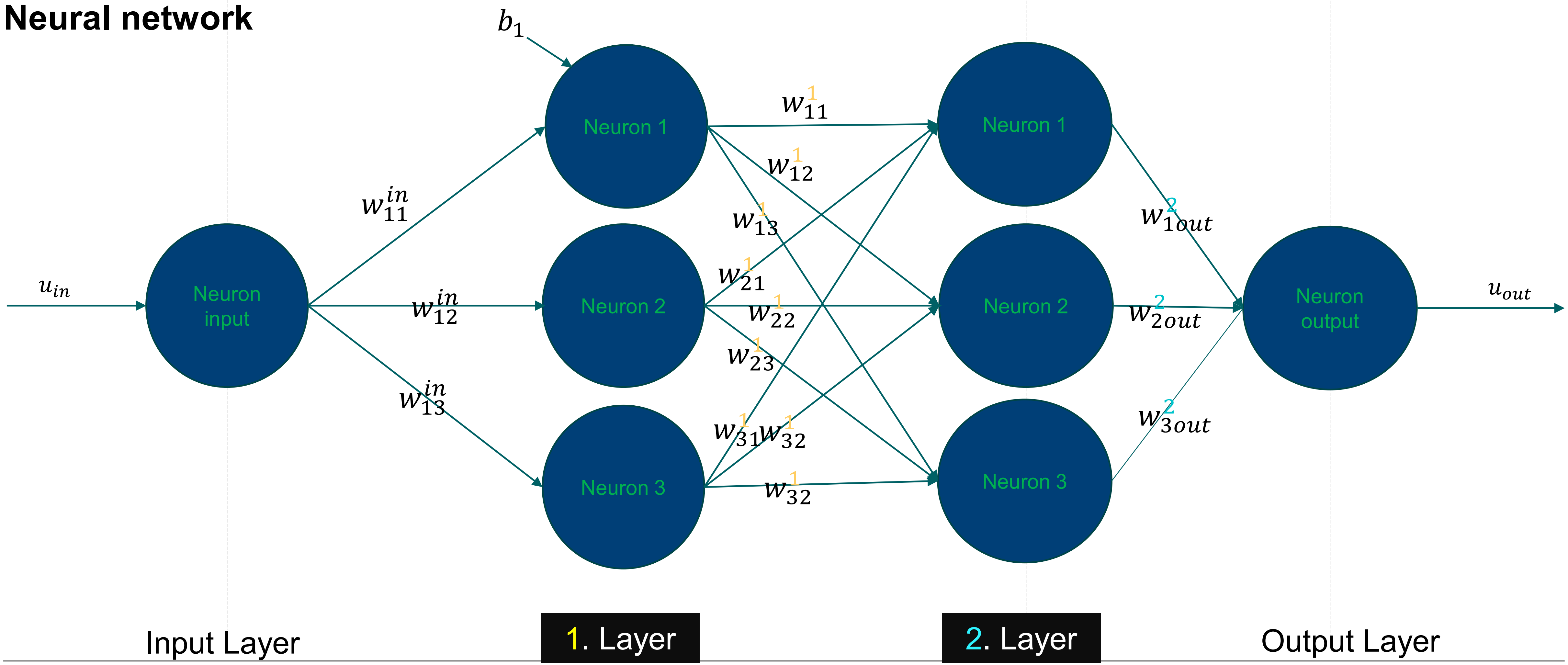
Fully connected neural network (NN)



MiMo (Multiple Input and Multiple Output) **are usual** (e.g., **2 joint positions as input** and **2 Cartesian positions as output…)**.

Eric Kaigom
kaigom@fb2.fra-uas.de

Neural network



Eric Kaigom
kaigom@fb2.fra-uas.de

Gradient via backpropagation

Recall that:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial x}\frac{\partial x}{\partial z}\frac{\partial z}{\partial w}$$

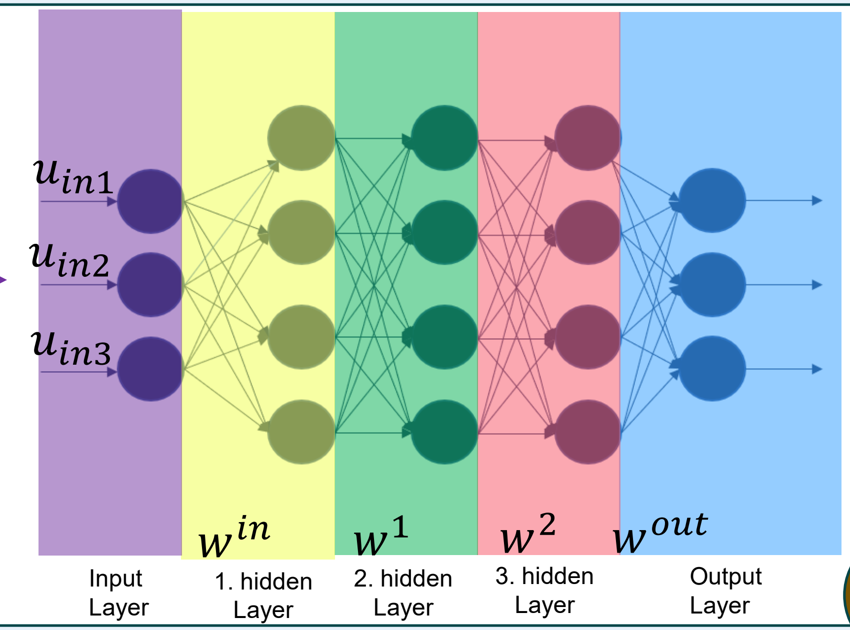
$$\frac{\partial L}{\partial w^4} = \frac{\partial L}{\partial x^3}\frac{\partial x^3}{\partial z^3}\frac{\partial z^3}{\partial w^4}$$

$$\frac{\partial L}{\partial w^3} = \frac{\partial L}{\partial x^3}\frac{\partial x^3}{\partial z^3}\frac{\partial z^3}{\partial x^2}\frac{\partial x^2}{\partial z^2}\frac{\partial z^2}{\partial w^3}$$

$$\frac{\partial L}{\partial w^2} = \frac{\partial L}{\partial x^3}\frac{\partial x^3}{\partial z^3}\frac{\partial z^3}{\partial x^2}\frac{\partial x^2}{\partial z^2}\frac{\partial z^2}{\partial x^1}\frac{\partial x^1}{\partial z^1}\frac{\partial z^1}{\partial w^2}$$

w^n means weights w w.r.t n-th layer

1 Forward input propagation … w^1, z^1, x^2 → w^2, z^2, x^3 → w^3, z^3, x^4 →



2 **Backward gradient propagation …** $\frac{\partial L}{\partial w^1}$ ← $\frac{\partial L}{\partial w^2}$ ← $\frac{\partial L}{\partial w^{out}}$ ←

3 Weight update… $w^i_{k+1} = w^i_k - r\frac{\partial L}{\partial w^i}$

4 Feeding the neural network with training data…

Eric Kaigom
kaigom@fb2.fra-uas.de

Gradient via backpropagation

Recall that:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial x}\frac{\partial x}{\partial z}\frac{\partial z}{\partial w}$$

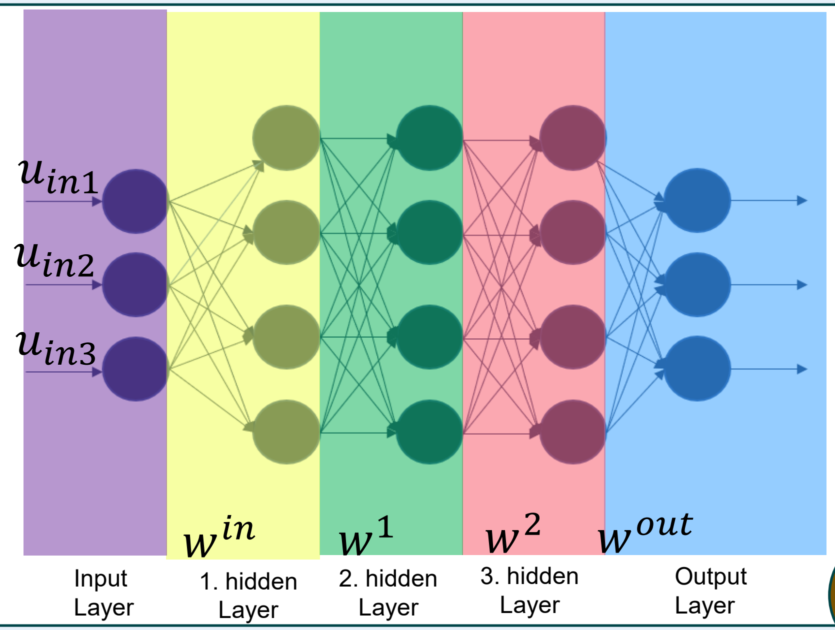
$$\frac{\partial L}{\partial w^4} = \frac{\partial L}{\partial x^3}\frac{\partial x^3}{\partial z^3}\frac{\partial z^3}{\partial w^4}$$

$$\frac{\partial L}{\partial w^3} = \frac{\partial L}{\partial x^3}\frac{\partial x^3}{\partial z^3}\frac{\partial z^3}{\partial x^2}\frac{\partial x^2}{\partial z^2}\frac{\partial z^2}{\partial w^3}$$

$$\frac{\partial L}{\partial w^2} = \frac{\partial L}{\partial x^3}\frac{\partial x^3}{\partial z^3}\frac{\partial z^3}{\partial x^2}\frac{\partial x^2}{\partial z^2}\frac{\partial z^2}{\partial x^1}\frac{\partial x^1}{\partial z^1}\frac{\partial z^1}{\partial w^2}$$

Observations:
Some **terms** appear **many times** in the gradient-related chain!

1. Forward input propagation … $w^1, z^1, x^2 \rightarrow w^2, z^2, x^3 \rightarrow w^3, z^3, x^4$



2. **Backward gradient propagation …** $\frac{\partial L}{\partial w^1} \leftarrow \frac{\partial L}{\partial w^2} \leftarrow \frac{\partial L}{\partial w^{out}}$

3. Weight update… $w^i_{k+1} = w^i_k - r\frac{\partial L}{\partial w^i}$

4. Feeding the neuronal network with training data…

Eric Kaigom
kaigom@fb2.fra-uas.de

Gradient via backpropagation

Recall that:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial x}\frac{\partial x}{\partial z}\frac{\partial z}{\partial w}$$

$$\frac{\partial L}{\partial w^4} = \frac{\partial L}{\partial x^3}\frac{\partial x^3}{\partial z^3}\frac{\partial z^3}{\partial w^4}$$

$$\frac{\partial L}{\partial w^3} = \frac{\partial L}{\partial x^3}\frac{\partial x^3}{\partial z^3}\frac{\partial z^3}{\partial x^2}\frac{\partial x^2}{\partial z^2}\frac{\partial z^2}{\partial w^3}$$

$$\frac{\partial L}{\partial w^2} = \frac{\partial L}{\partial x^3}\frac{\partial x^3}{\partial z^3}\frac{\partial z^3}{\partial x^2}\frac{\partial x^2}{\partial z^2}\frac{\partial z^2}{\partial x^1}\frac{\partial x^1}{\partial z^1}\frac{\partial z^1}{\partial w^2}$$

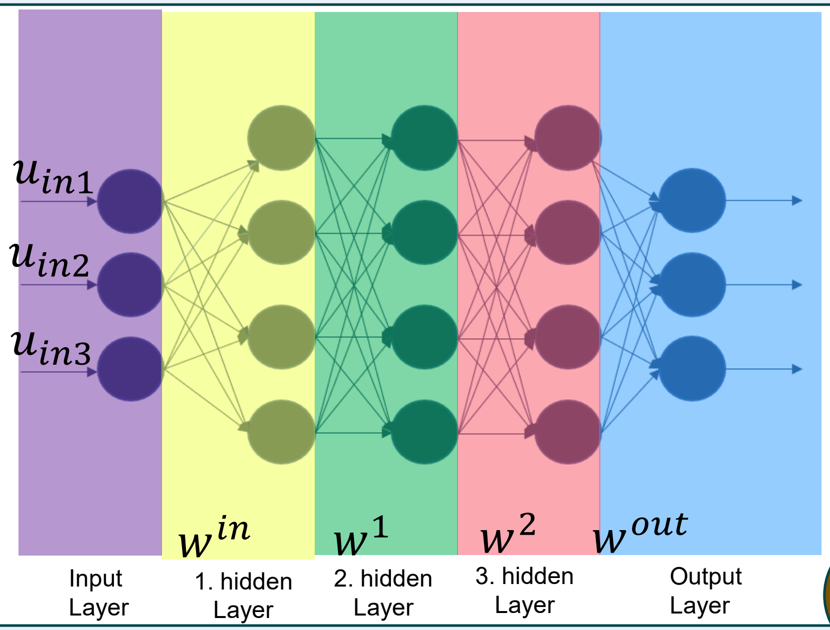
Observation:
If some of the $\frac{\partial ()}{\partial ()}$ (e.g., $\frac{\partial x^2}{\partial z^2}$) in the chain are too small or simply vanish, the **gradient** is likely to **disappear**!

What does it implicate?

Big challenge faced by **deep neural networks**!

Forward input propagation … w^1, z^1, x^2 w^2, z^2, x^3 w^3, z^3, x^4

1



2

Backward gradient propagation … $\frac{\partial L}{\partial w^1}$ $\frac{\partial L}{\partial w^2}$ $\frac{\partial L}{\partial w^{out}}$

Weight update… $w_{k+1}^i = w_k^i - r\frac{\partial L}{\partial w^i}$

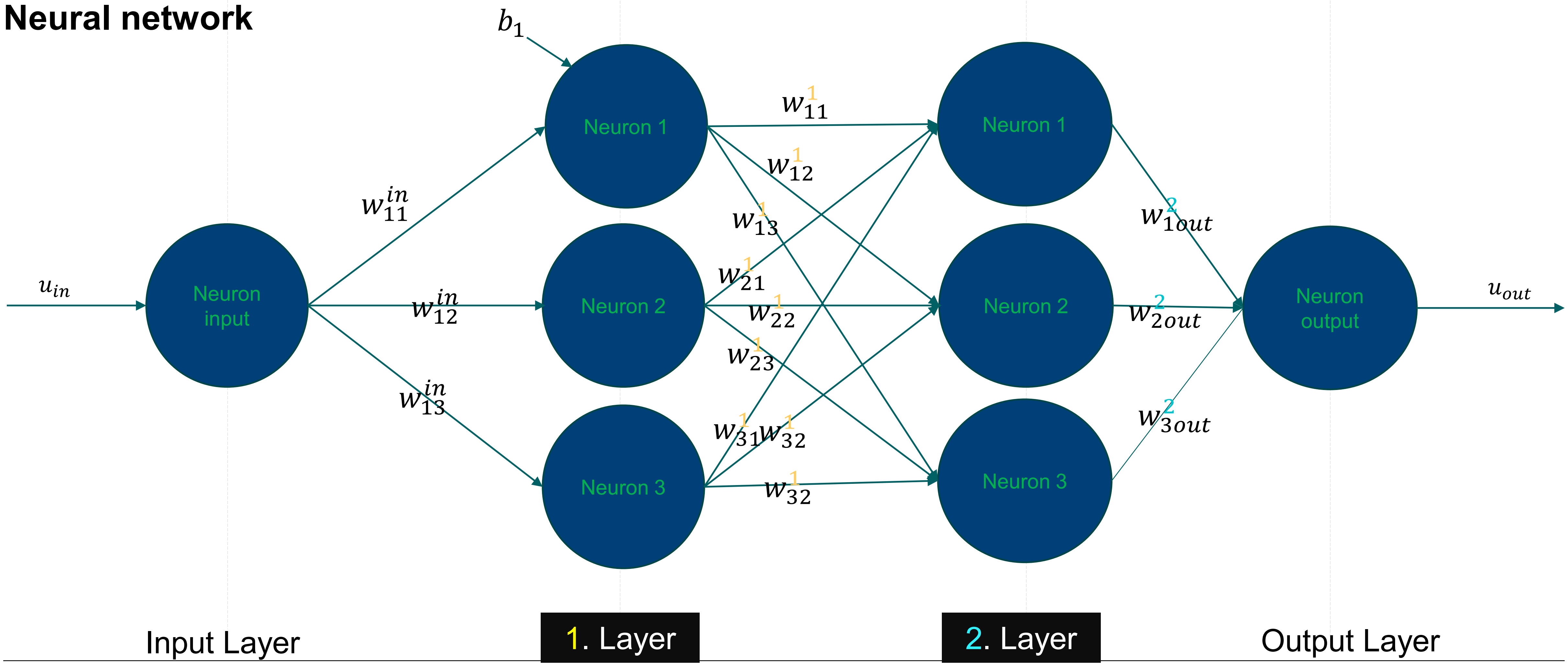
3

Feeding the neuronal network with training data…

4

Eric Kaigom
kaigom@fb2.fra-uas.de

Neural network



Eric Kaigom
kaigom@fb2.fra-uas.de

Application

1) Implement a matlab function **[x1, x2] = forward(q1, q2, w)** that reflects a fully connected neural network made up of **two hidden layers** with **two neurons** for each of them. The function receives the following parameters
   a) Two real inputs **q1** and **q2**
   b) A vector **w** of weights for all layers

   and returns two real outputs **x1** and **x2**

2) Implement a loop that runs 10^6 times, feeds **forward(q1, q2, w)** with randomly generated **q1, q2**, and **w**.

3) Set **q1** and **q2** as a vector of 1000 random values between -1 and 1. Furthermore, assume (ground truth) that
   - **X1 = q1 + q2**
   - **X2 = q1 – q2**
4) Run the loop while feeding the network (i.e., **forward (…)**) with **q1** and **q2** value pairwise and compute a meaningful loss function for each epoch using **X1** and **X2**.

Eric Kaigom
kaigom@fb2.fra-uas.de
FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Machine Learning – Updating the weights of the neural network in practice

```
% Compute the contribution to the gradient of the loss for each weight
for i = 1:length(trainedWeights)
    % Perturb the current weight using espsilonValue
    trainedWeightsPerturbedRight = trainedWeights;
    trainedWeightsPerturbedRight(i) = trainedWeightsPerturbedRight(i) + espsilonValue;
```

Measurements potentially prone to noise!

Data label

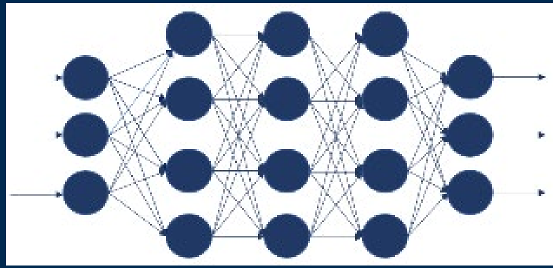
Result: a trained digital model used for e.g. prediction purposes (e.g., FK(q) $\approx X_E$) in other applications (e.g., robotized indust. automation)

4

```
% Approximate the gradient contribution using a difference
grad(i) = (costFunctionPertubedRight - costFunctionPertubedLeft)...
/ (2 * espsilonValue);
```

1 **Pertinent data** (e.g., q) as **input examples**

Model capture using maths, statistics, and data, etc



Model training (i.e., optimization of model parameters to capture the mapping between input-output examples and generalize)

2

3 **Model output during training**

Loss function f

- ϵ: small and positive number
- e_1: [1 0 0 0 0 ….]
- e_2: [0 1 0 0 0 ….]

5 Computation of the gradient ∇f of the loss function f w.r.t the weights of the neural network

$$\frac{\partial f(w)}{\partial w_1} = \frac{f(x+\epsilon e_1) - f(x-\epsilon e_1)}{2\epsilon}$$

$$\frac{\partial f(w)}{\partial w_2} = \frac{f(x+\epsilon e_2) - f(x-\epsilon e_2)}{2\epsilon}$$

$$\vdots$$

$$w_{k+1} = w_k - r\nabla f w_k$$

Weight update

$$x = f(w_1, \dots, w_n)_u \rightarrow \nabla f = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \vdots \\ \frac{\partial f}{\partial w_n} \end{bmatrix}$$

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ \frac{\partial f}{\partial w_3} \\ \vdots \\ \frac{\partial f}{\partial w_n} \end{bmatrix}$$

Physical

Digital

Eric Kaigom
kaigom@fb2.fra-uas.de

Machine Learning – Adapting the learning rate via moment estimation

- $m_{k+1} = \beta_1 m_k + (1-\beta_1)\nabla f$
- $v_{k+1} = \beta_2 v_k + (1-\beta_2)(\nabla f)^2$
- $\hat{m}_{k+1} = \frac{m_{k+1}}{1-\beta_1^{k+1}}$
- $\hat{v}_{k+1} = \frac{v_{k+1}}{1-\beta_2^{k+1}}$
- $w_{k+1} = w_k - \alpha \frac{1}{\sqrt{\hat{v}_k + \epsilon}} \hat{m}_k$

adaptation

β_1 and β_2 induce an exponential decay (from $\beta_1 = 0.9, \beta_2 = 0.999$)

- $\alpha = 0.001$
- $\epsilon = 10^{-8}$

```
% Compute the contribution to the gradient of the loss for each weight
for i = 1:length(trainedWeights)
    % Perturb the current weight using espsilonValue
    trainedWeightsPerturbedRight = trainedWeights;
    trainedWeightsPerturbedRight(i) = trainedWeightsPerturbedRight(i) + espsilonValue;
```

Data label

4

```
% Approximate the gradient contribution using a difference
grad(i) = (costFunctionPertubedRight - costFunctionPertubedLeft)...
/ (2 * espsilonValue);
```

3

Model output during training

5

- ϵ: small and positive number
- e_1: [1 0 0 0 0 ….]
- e_2: [0 1 0 0 0 ….]

$w_{k+1} = w_k - r\nabla f(w)_{w_k}$

Instead of …

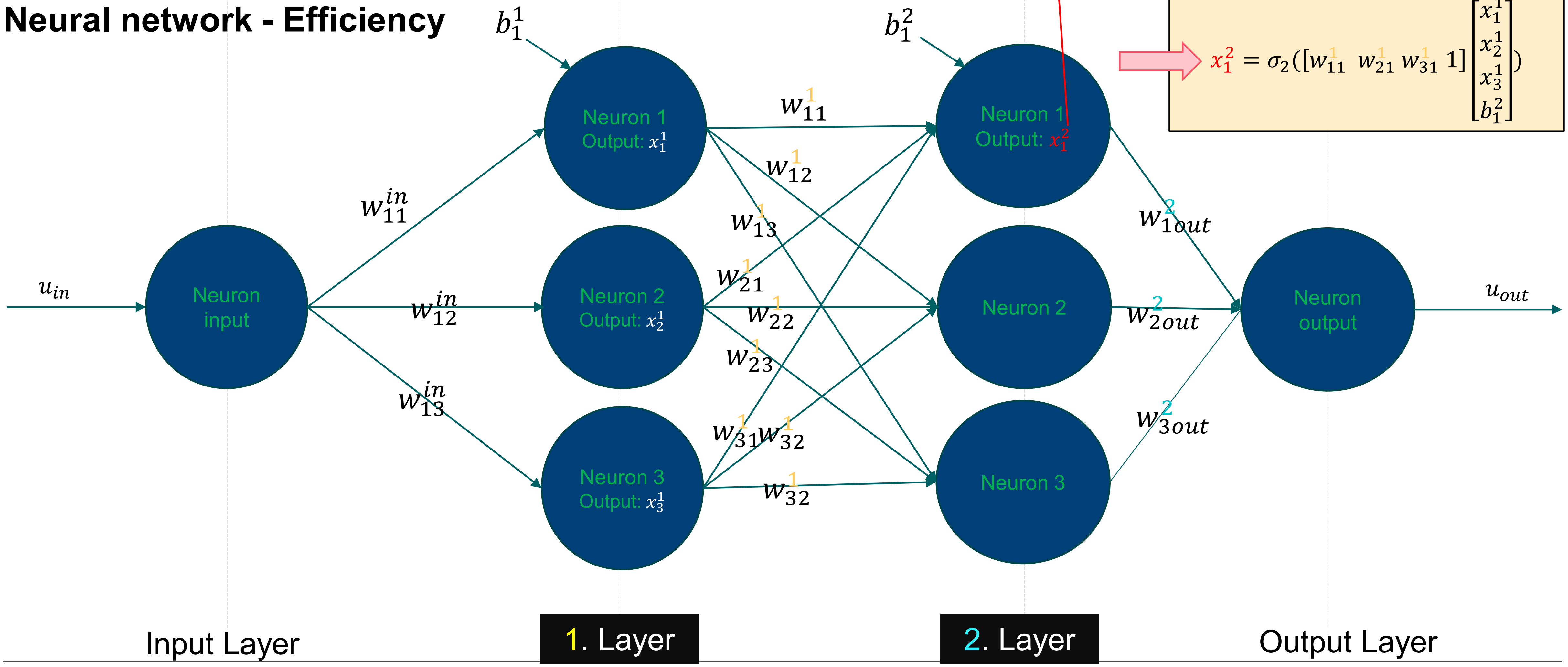
Eric Kaigom
kaigom@fb2.fra-uas.de

Neural network - Efficiency

….harness the following insight:

$$x_1^2 = \sigma_2(w_{11}^1 x_1^1 + w_{21}^1 x_2^1 + w_{31}^1 x_3^1 + b_1^2)$$

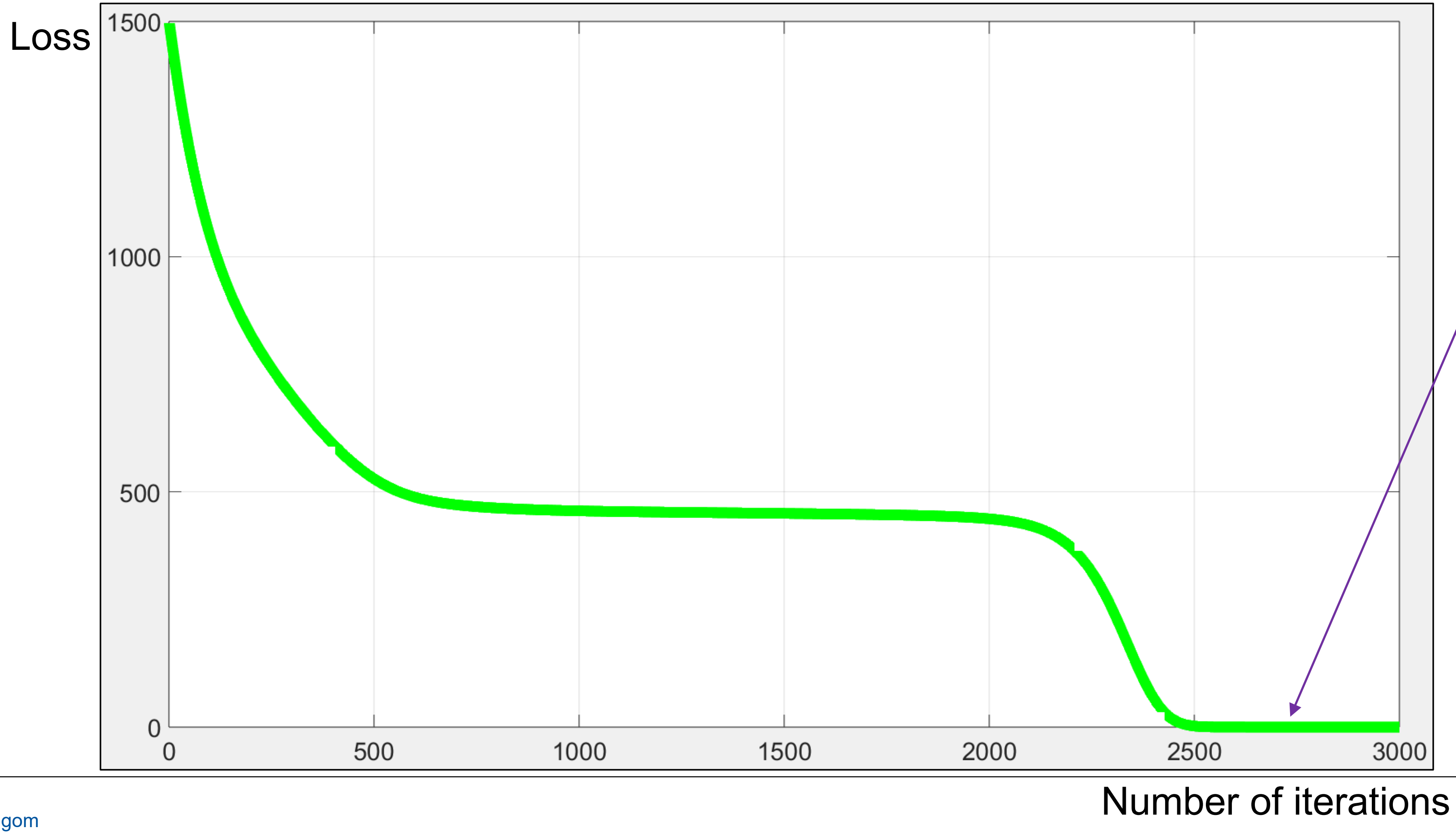
$$x_1^2 = \sigma_2\left([w_{11}^1 \;\; w_{21}^1 \;\; w_{31}^1 \;\; 1]\begin{bmatrix} x_1^1 \\ x_2^1 \\ x_3^1 \\ b_1^2 \end{bmatrix}\right)$$



Eric Kaigom
kaigom@fb2.fra-uas.de

Robotics and Machine Learning – Model Capture

Optimization of the prediction accuracy (under adaptive moment-based learning rate) - Result



Eric Kaigom
kaigom@fb2.fra-uas.de

Project (presentation: 29.01.25, 8:15 h, groups of min. 2 /max. 3 students, Duration max. 20 Min)

- Improve our model (see the file withStudentsNew.m) to learn
  - **X1 = q1 + q2 - 1**
  - **X2 = q1 – q2 + 1**

  and

  - **X1 = q1 + q2.*q2 - 1**
  - **X2 = q1 – q2 + 1**
- When and why is an enhancement necessary?
- How did you enhance the model?
- What did you observe? Explain your observation with your own words

Eric Kaigom
kaigom@fb2.fra-uas.de

Thank you
for your attention!