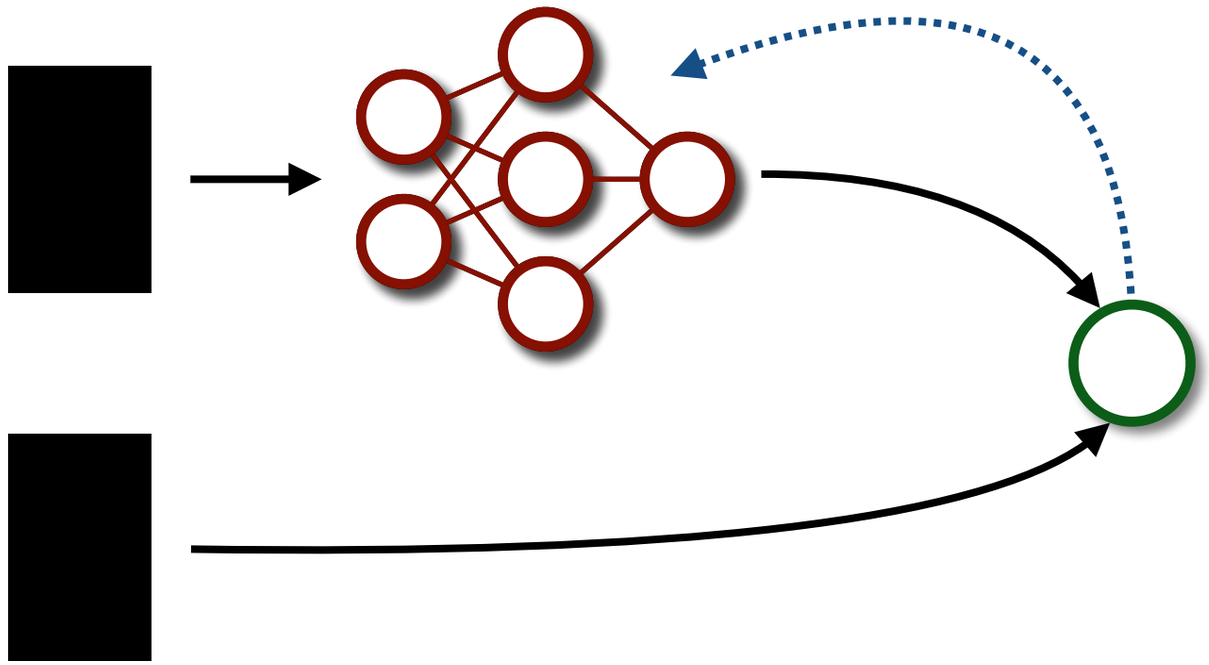


Neural Network Building Blocks (1/3)

Docent: Dennis Noll (RWTH)
Tutor: Boyang Yu (LMU)

Deep Learning School "Basic Concepts"

08.08.22

RWTH AACHEN UNIVERSITY

Artificial Intelligence Yesterday

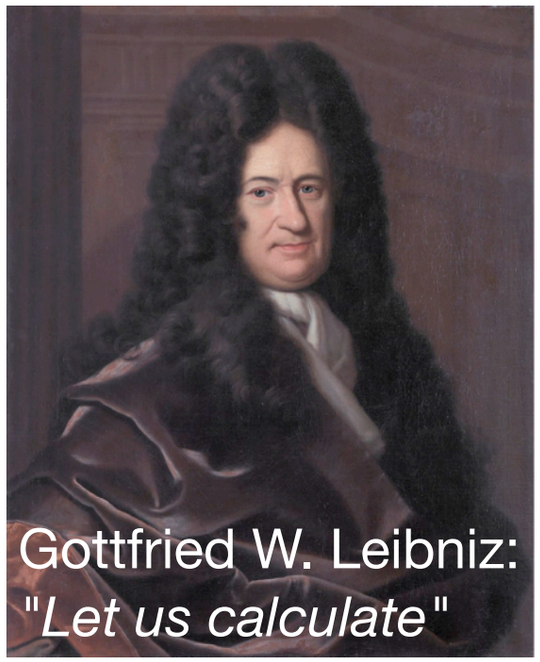
[2, 3]

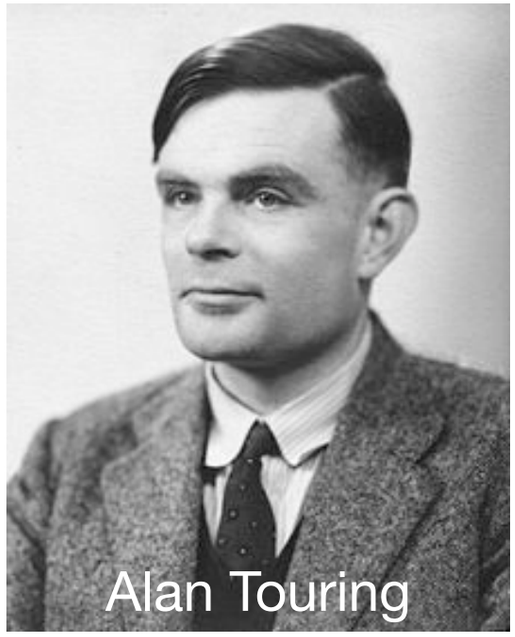
Artificial Intelligence: The ability of a machine to perform tasks commonly associated with intelligent beings


Deep Blue


Dartmouth Workshop


Adoption in industry


Gottfried W. Leibniz:
"Let us calculate"


Alan Touring


Arthur Samuel

1700 Reasoning

1940 Computing

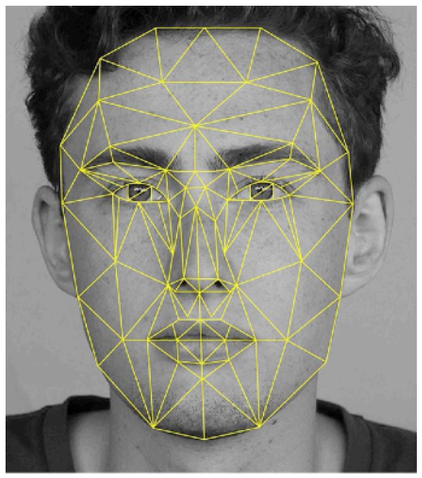
1950 1974 1987 Artificial Intelligence

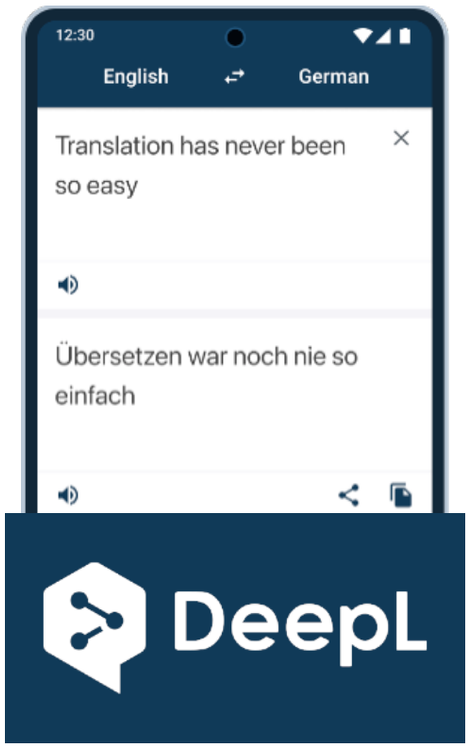
Artificial Intelligence Today

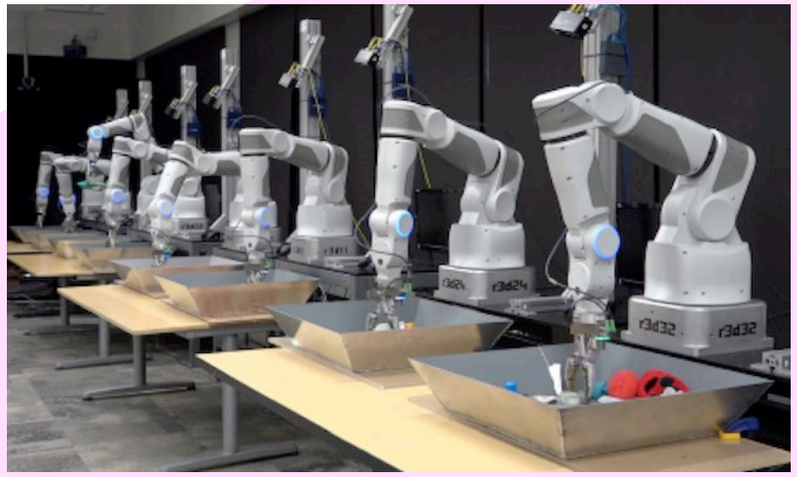
Dennis Noll - 08.08.22

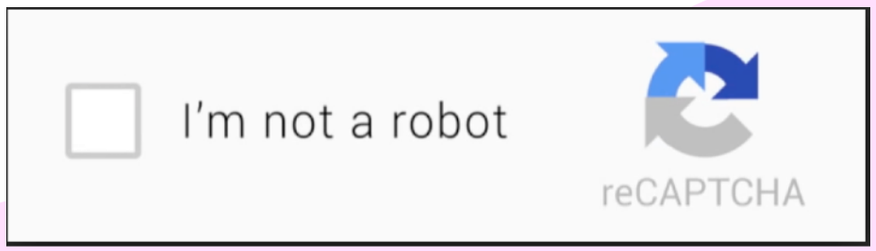
[4-10]

Artificial Intelligence: The ability of a machine to perform tasks commonly associated with intelligent beings









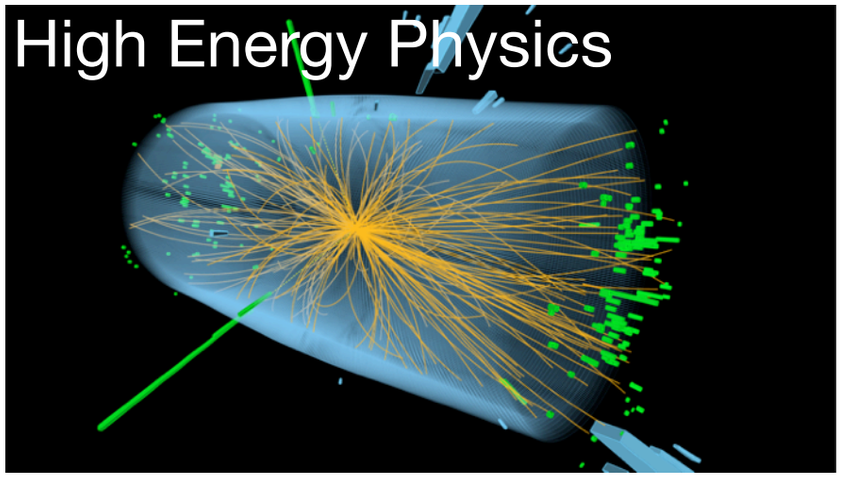
2012

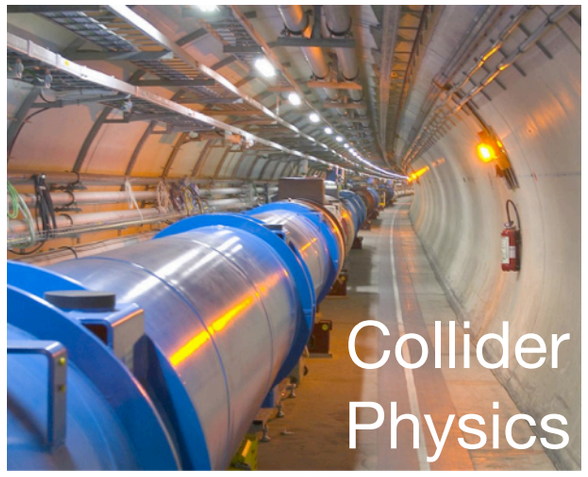
2022

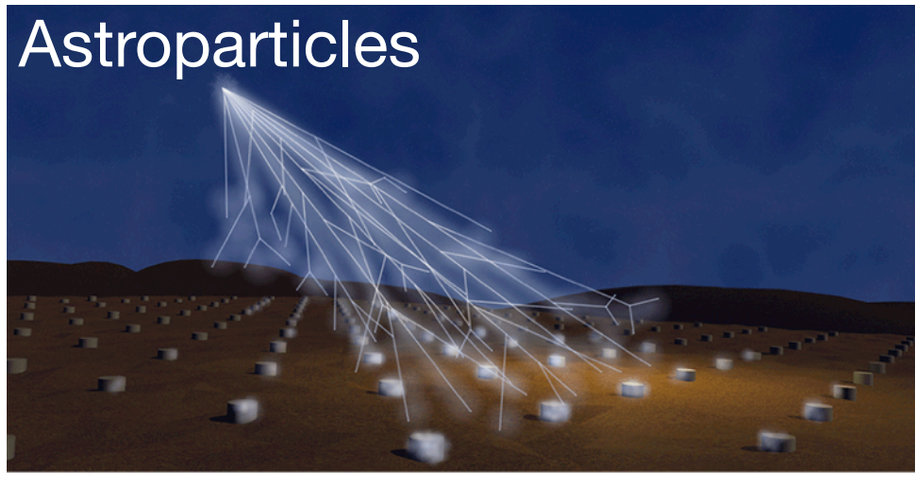
Artificial Intelligence In Physics

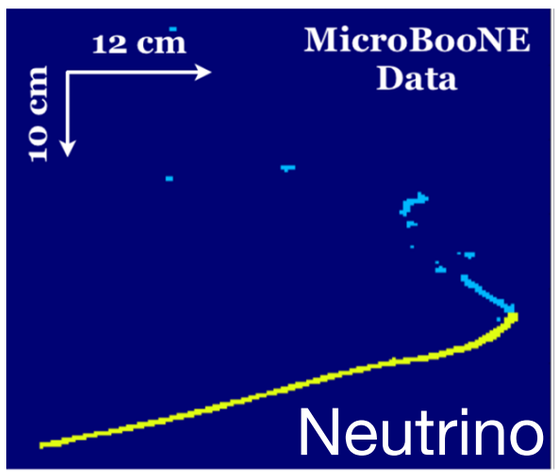
- Many AI applications in physics, and rising!
- Three main drivers:
  - Complex problems
  - Large amounts of data
  - Sufficient computing resources

[11-16]

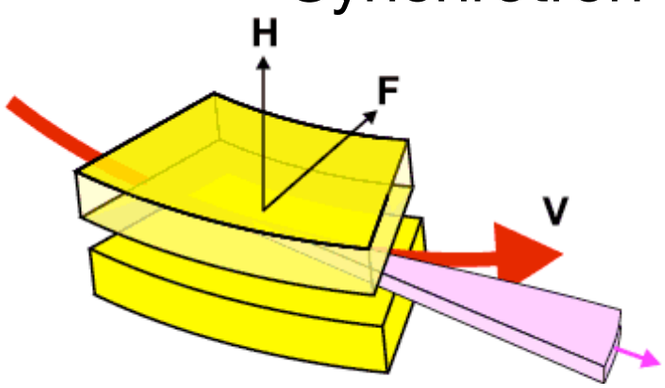




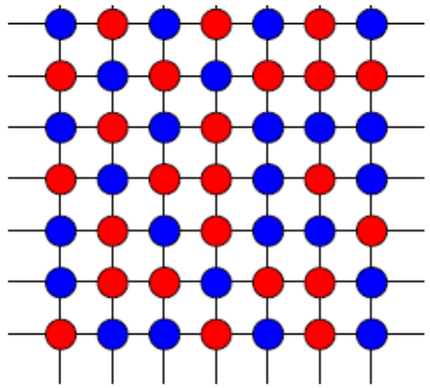


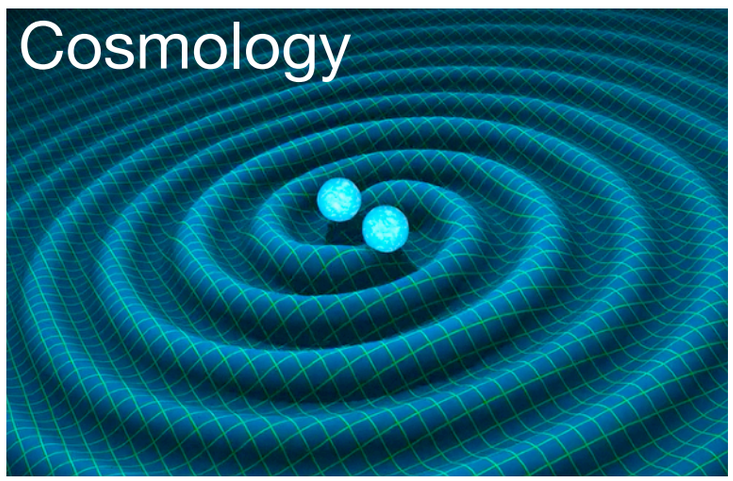


Synchrotron



Solid States





And many more!

About me

- Since >5 years working in data analysis @ CMS Experiment (LHC)

- Physics:
  - Higgs Boson Pair Production

- Deep Learning:
  - Supervised & Reinforcement Learning
  - Production-ready Deep Learning

- Computing:
  - Fast O(TB) Data Processing
  - Pipeline Computing

Contact:

RWTH AACHEN UNIVERSITY
Dennis Noll - 08.08.22

Data:

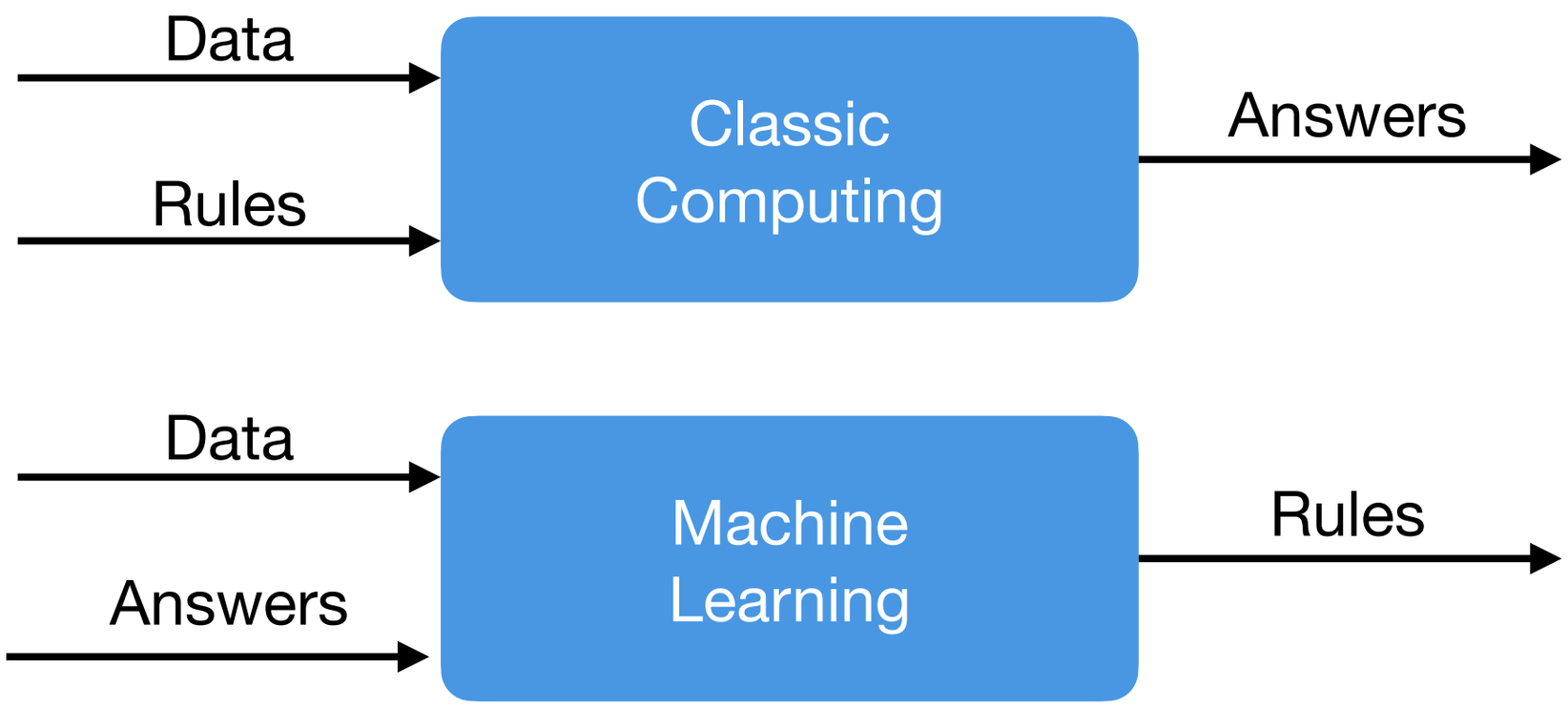
Answers: Rock Paper Scissors

From Classic Computing to Machine Learning

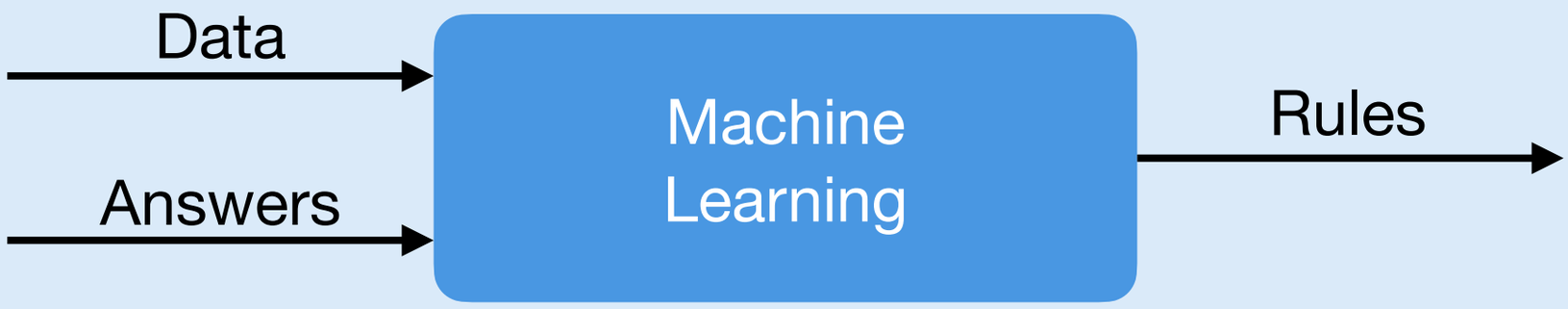
RWTH AACHEN UNIVERSITY
Dennis Noll - 08.08.22

Data:

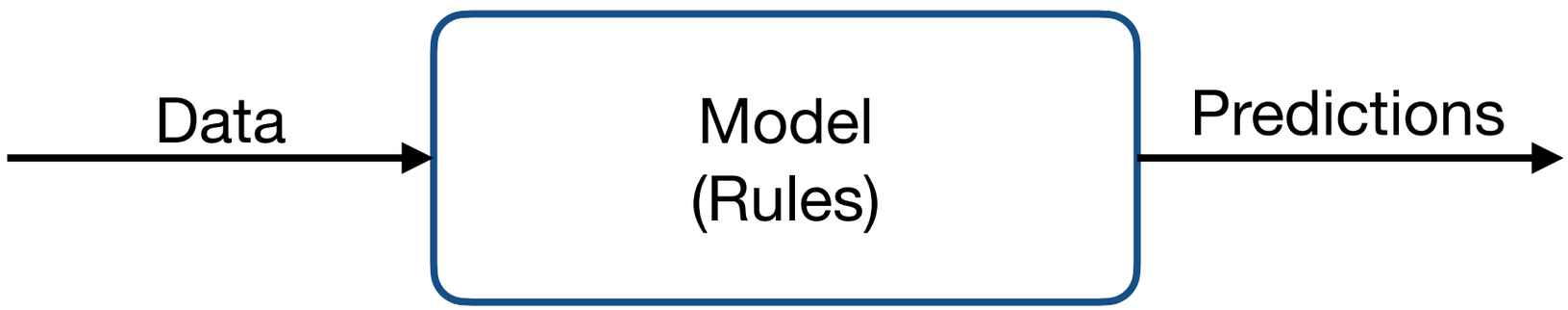
Answers: Rock Paper Scissors



From Classic Computing to Machine Learning



Training Phase



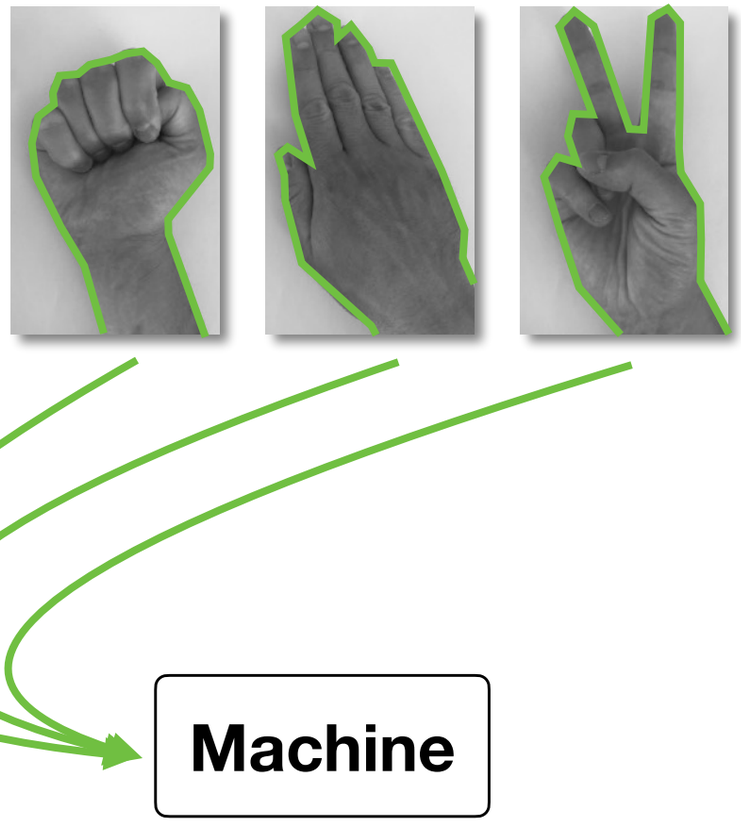
Inference Phase

Types of Machine Learning

From Machine Learning to Deep Learning

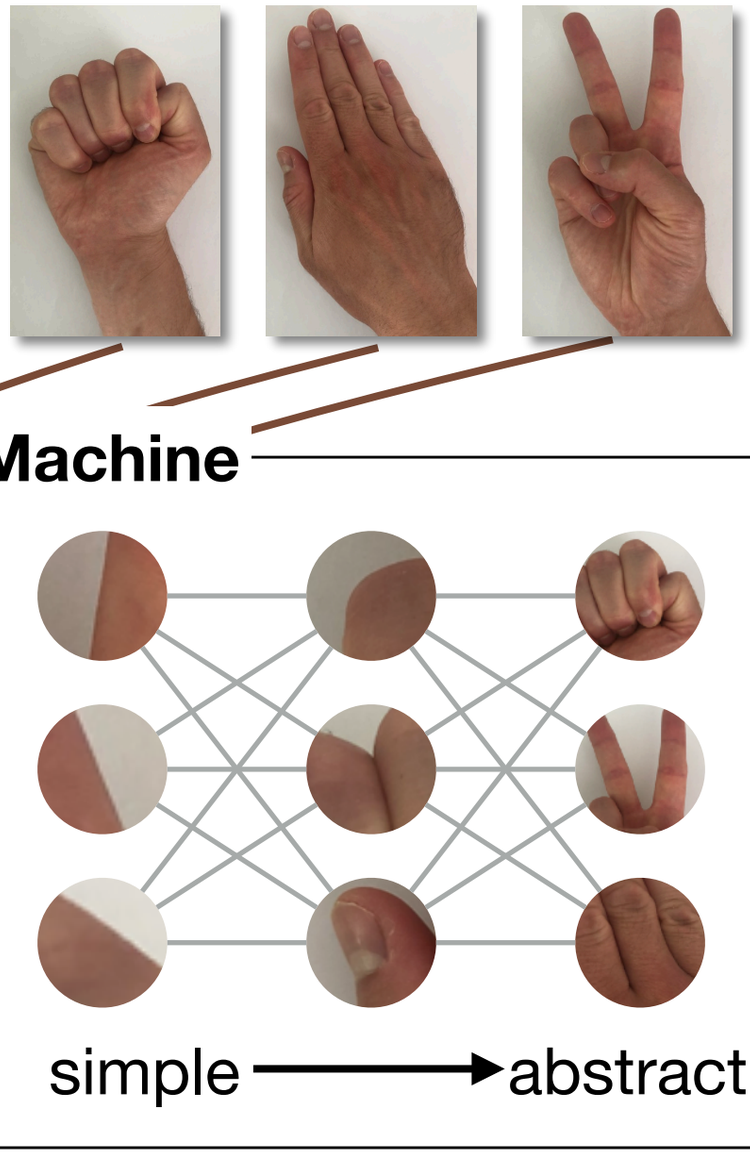
Classic Machine Learning

- Hand-engineered features
  - time consuming
  - not stable
  - not scalable

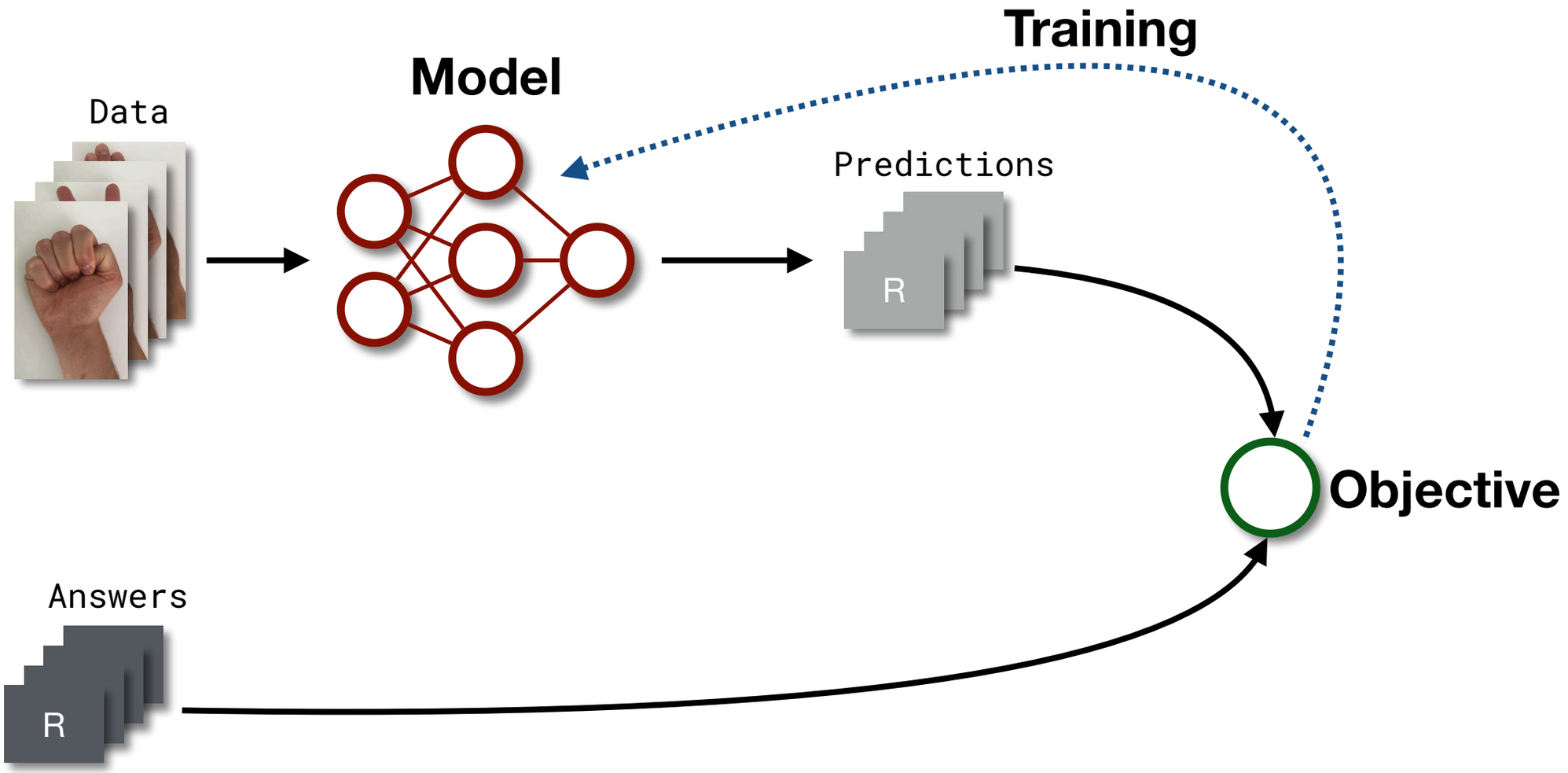


Deep Learning

- Machine extracts features
  - fast
  - stable
  - well scalable



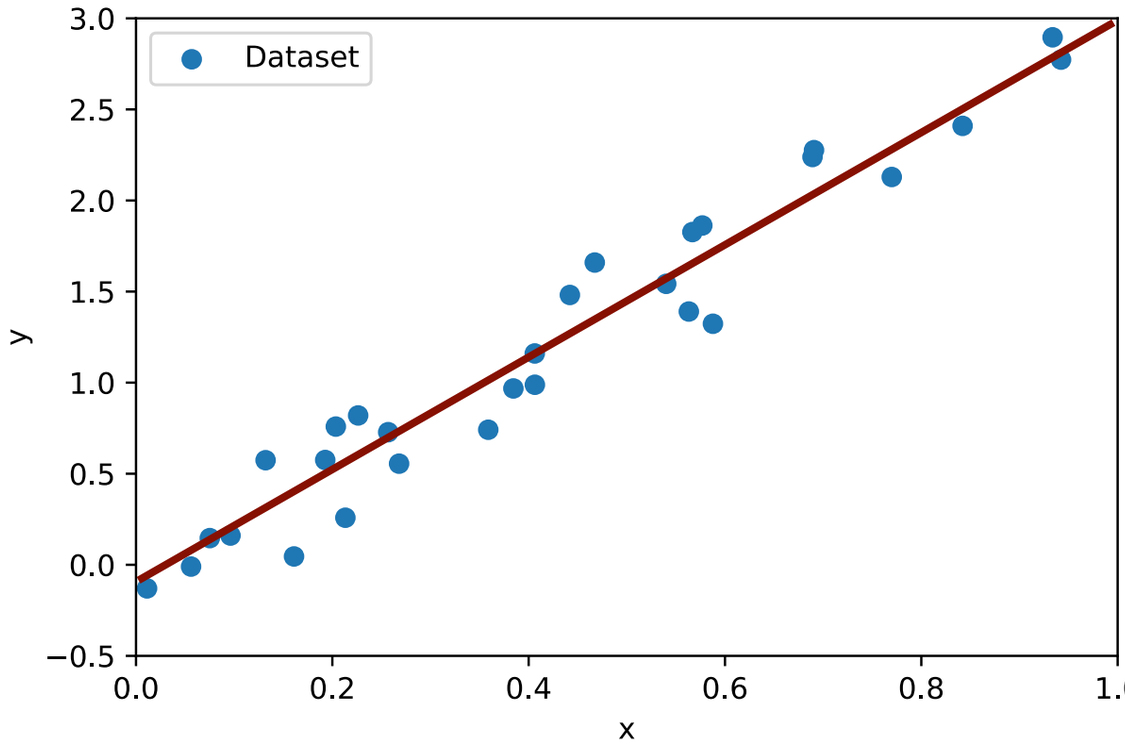
Machine Learning in a Nutshell

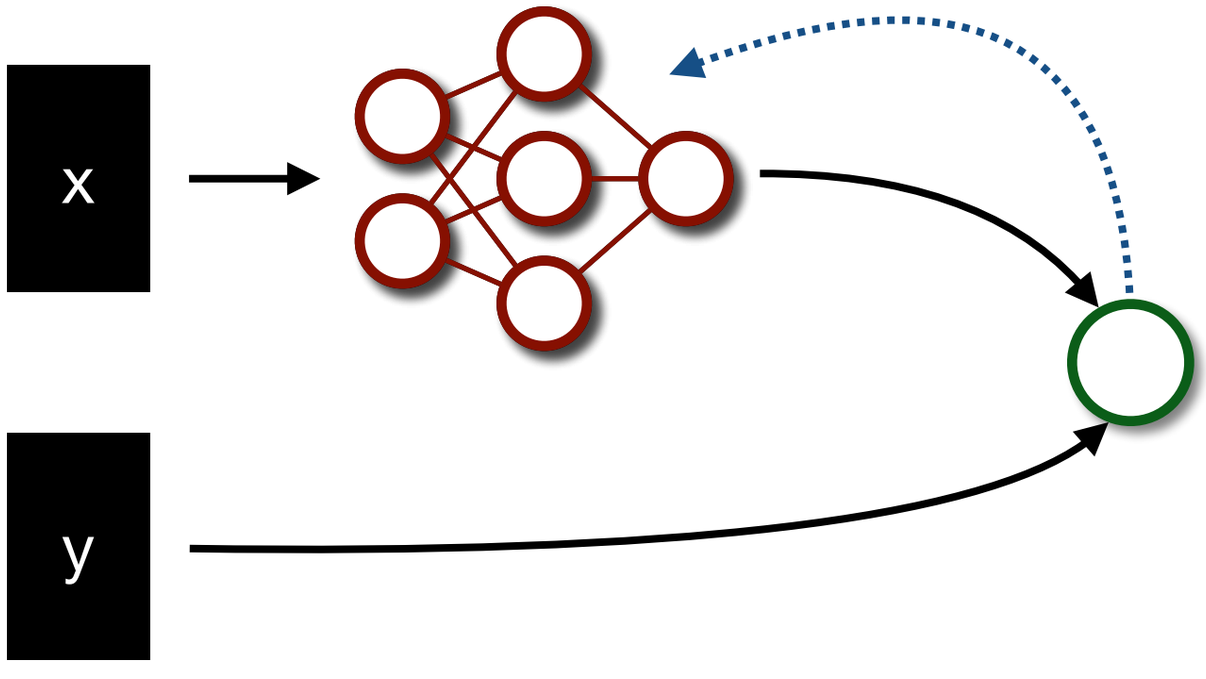


Beginners Example: Linear Regression

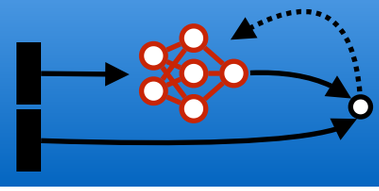
RWTH AACHEN UNIVERSITY
Dennis Noll - 08.08.22

- Find **rules** which connect Data→Answers (x→y)





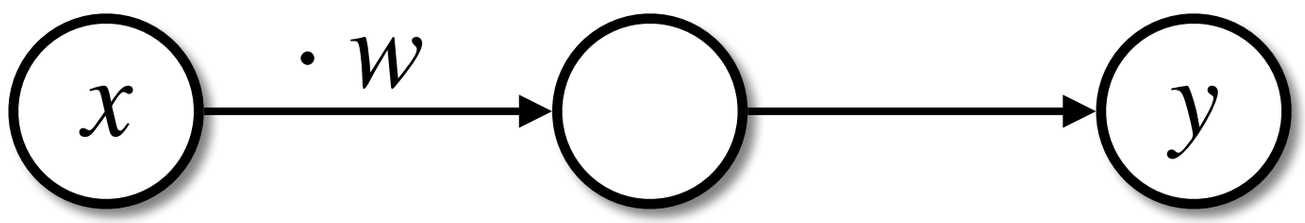
The Perceptron (1 Input, 1 Output)



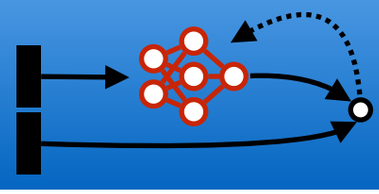
RWTH AACHEN UNIVERSITY
Dennis Noll - 08.08.22

- Approximate Linear Function $f : \mathbb{R}^1 \to \mathbb{R}^1$
- Parametrizable (1 parameter):
  - Weight: w

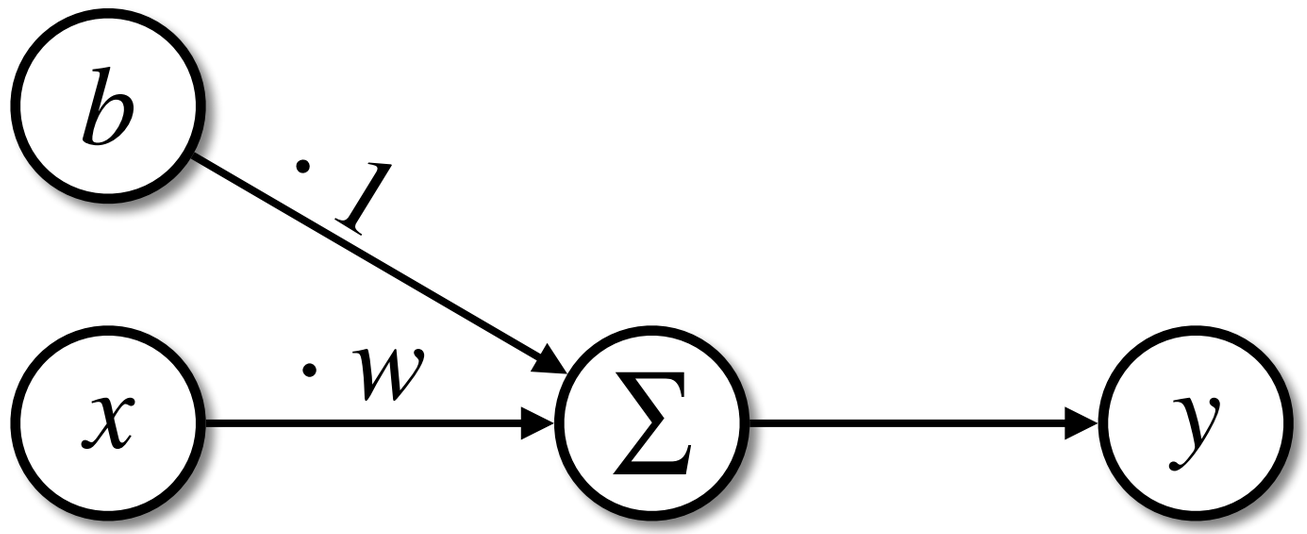
- Functional form: $y = w \cdot x$



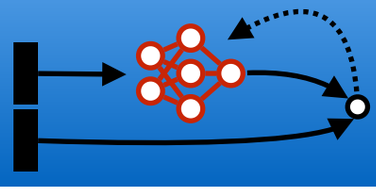
The Perceptron (1 Input, 1 Output)



- Approximate Linear Function $f : \mathbb{R}^1 \to \mathbb{R}^1$
- Parametrizable (2 parameters):
  - Weight: w
  - Bias: b
- Functional form: $y = w \cdot x + b$

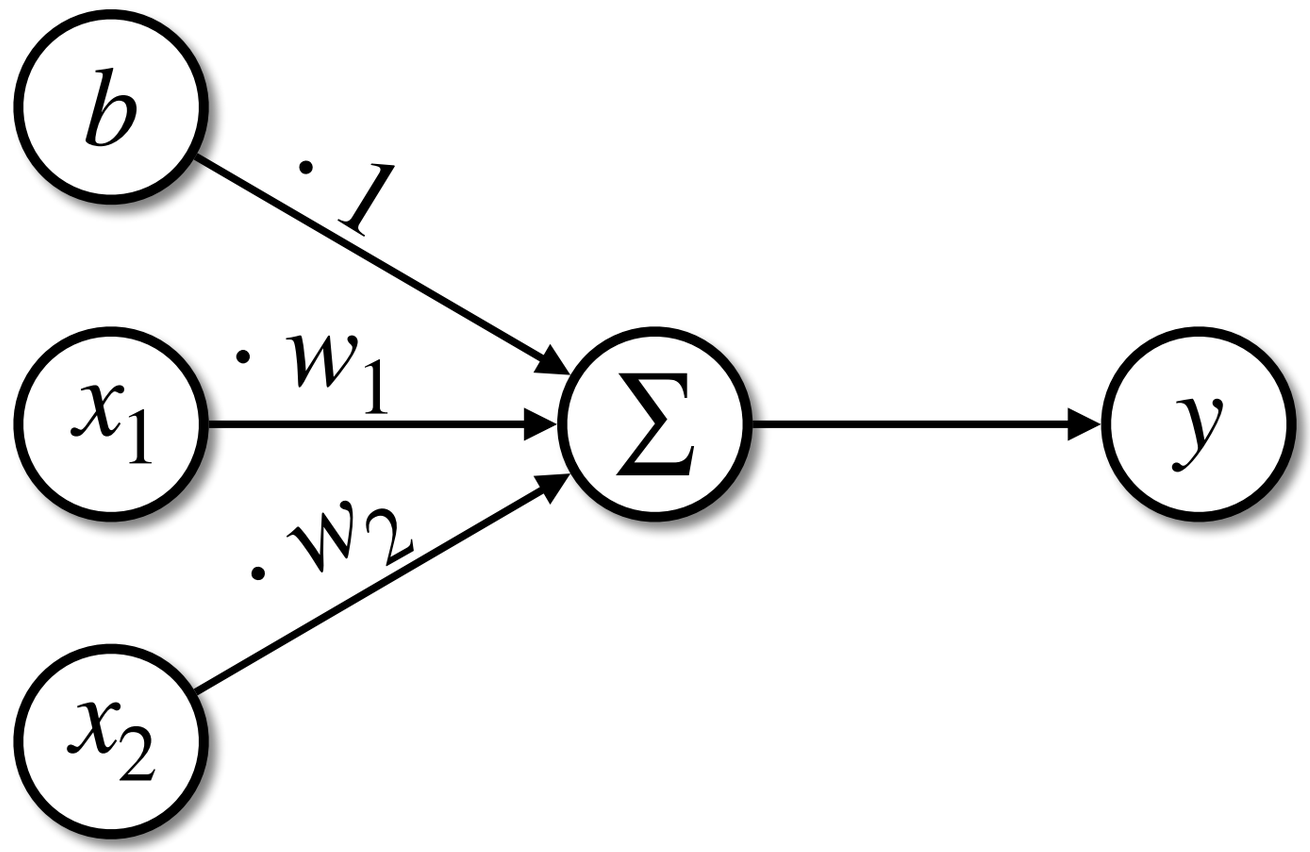


The Perceptron (N Inputs, 1 Output)



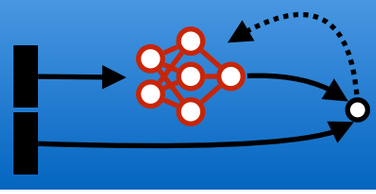
RWTH AACHEN UNIVERSITY
Dennis Noll - 08.08.22

- Approximate Linear Function $f : \mathbb{R}^N \to \mathbb{R}^1$
- Parametrizable (N+1 parameters):
  - Weights: w_1, w_2, …, w_N
  - Bias: b
- Functional form: $y = \sum_i w_i \cdot x_i + b$



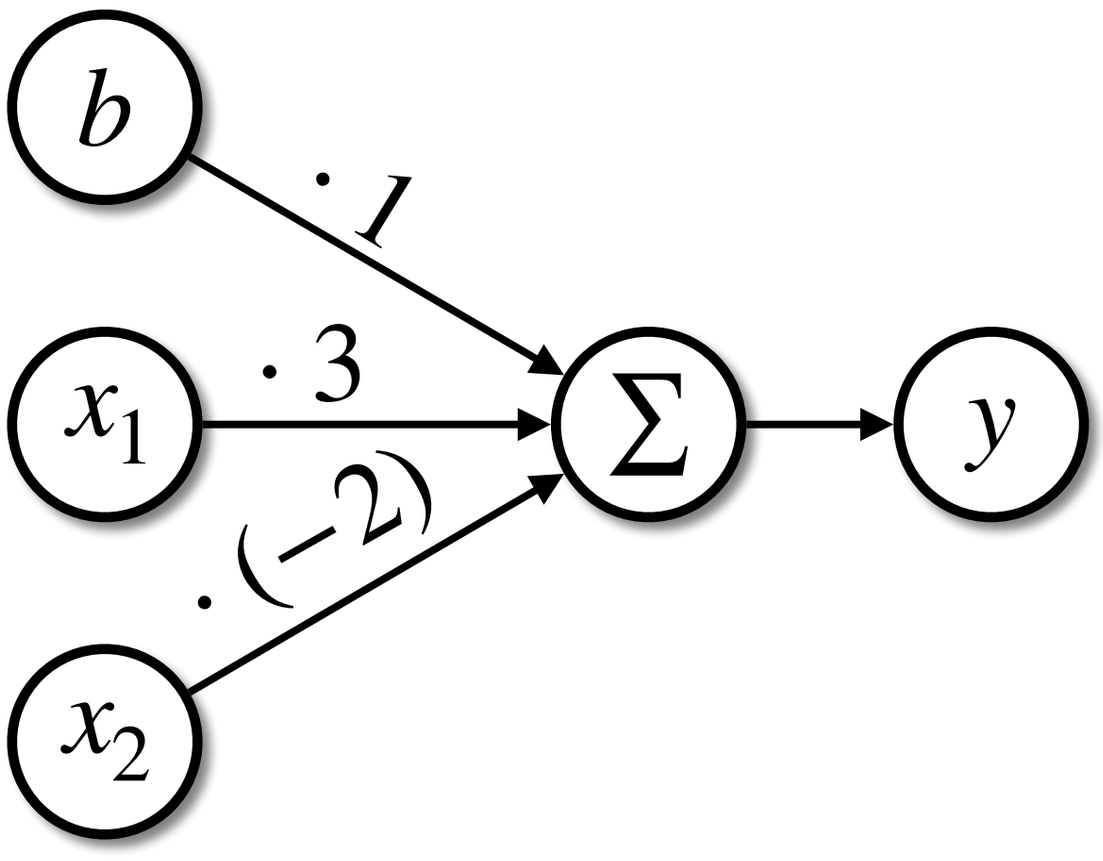
…

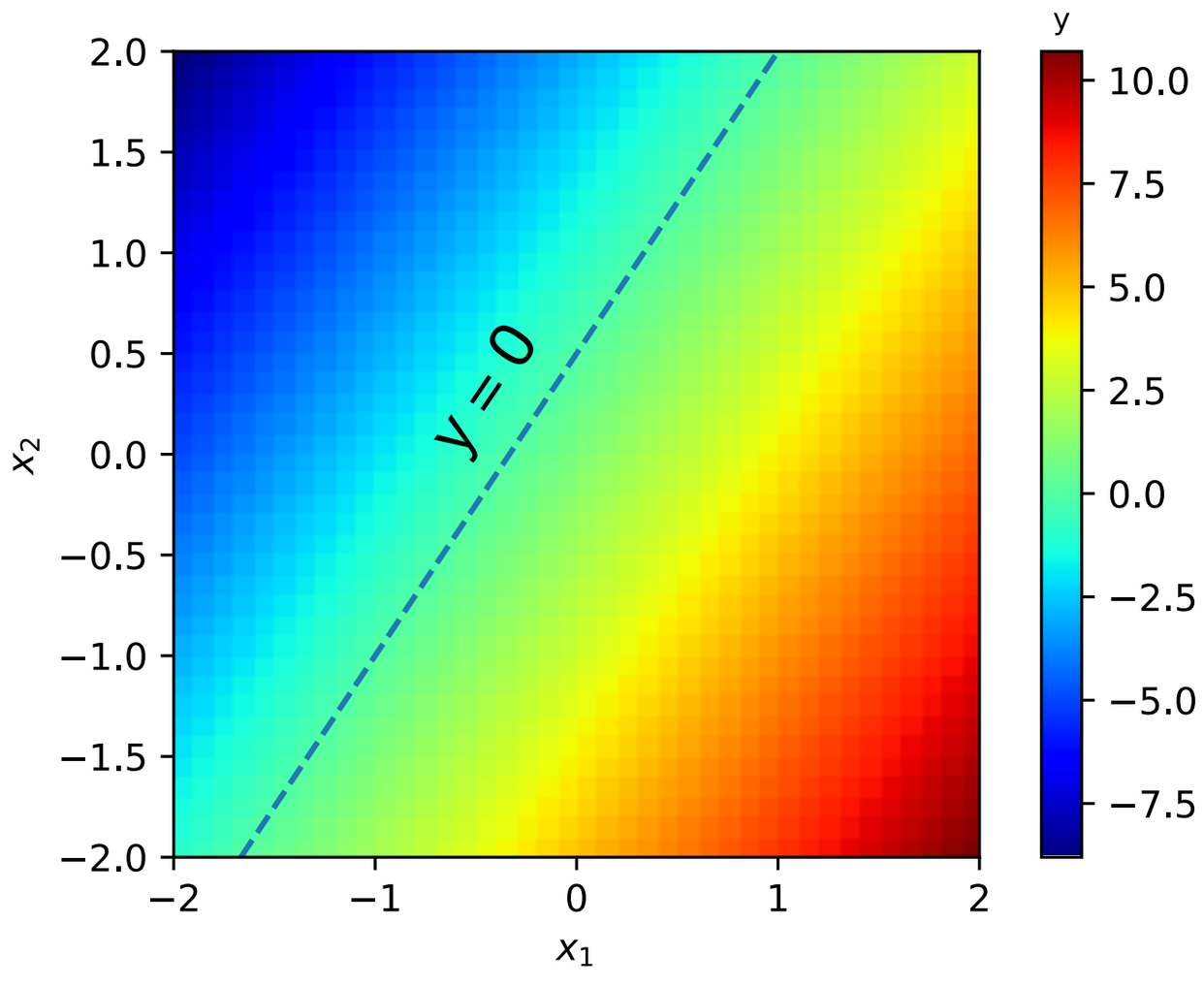
The Perceptron (2 Inputs, 1 Output)



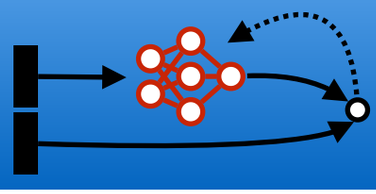
RWTH AACHEN UNIVERSITY
Dennis Noll - 08.08.22

- Approximate Linear Function $f : \mathbb{R}^2 \to \mathbb{R}^1$
- Parametrizable:
  - Weights: $w_1 = 3, w_2 = -2$
  - Bias: $b = 1$
- Functional form: $y = 3x_1 - 2x_2 + 1$



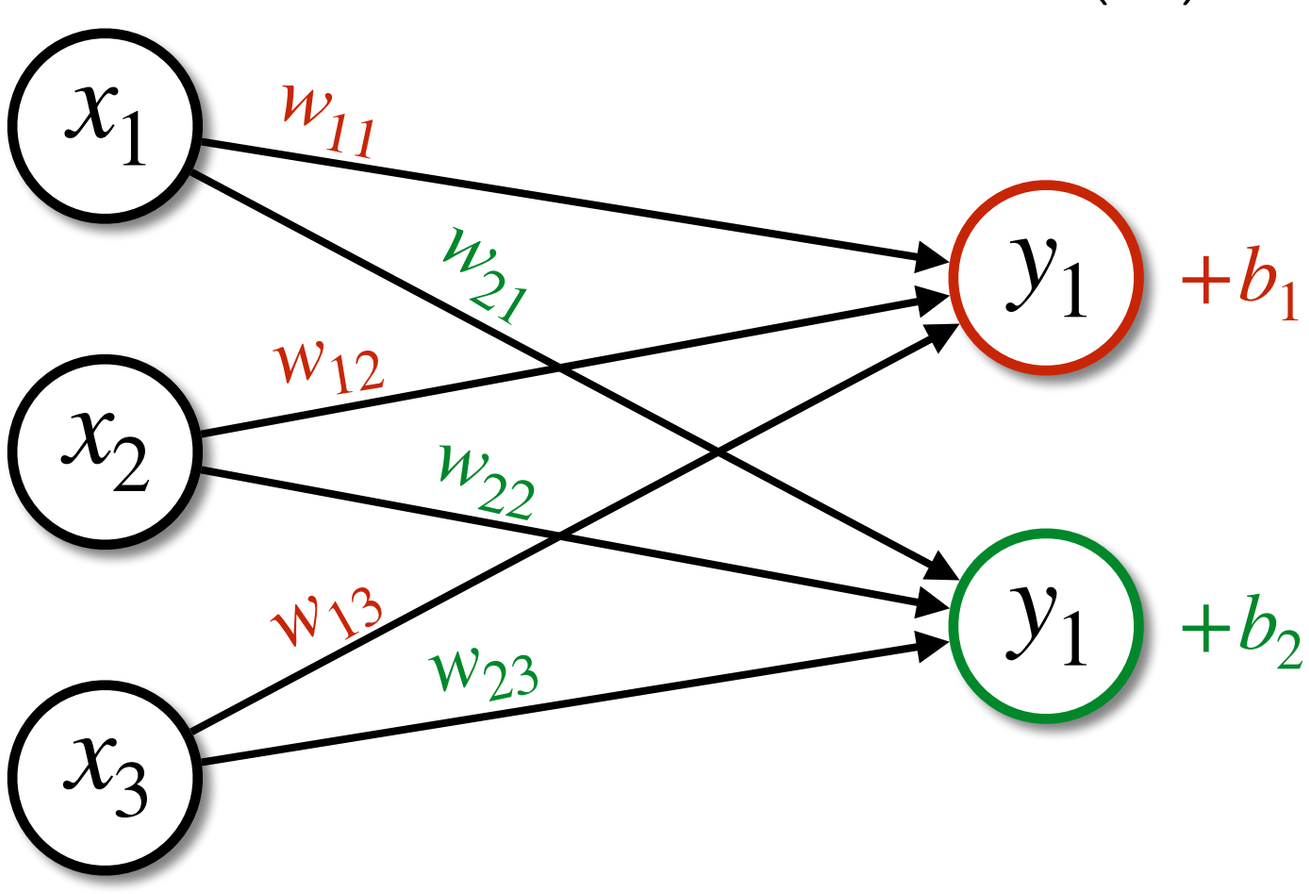


The Perceptron (N Inputs, M Outputs)

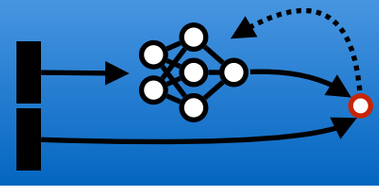


- Approximate Linear Function $f : \mathbb{R}^N \to \mathbb{R}^M$
- Multilayer perceptron with N inputs and M outputs
- Parametrizable (N·M+M parameters):
  - Weights: $w_{11}, w_{12}, \ldots, w_{MN}$
  - Biases: : $b_1, b_2, \ldots, b_M$
- Functional form:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

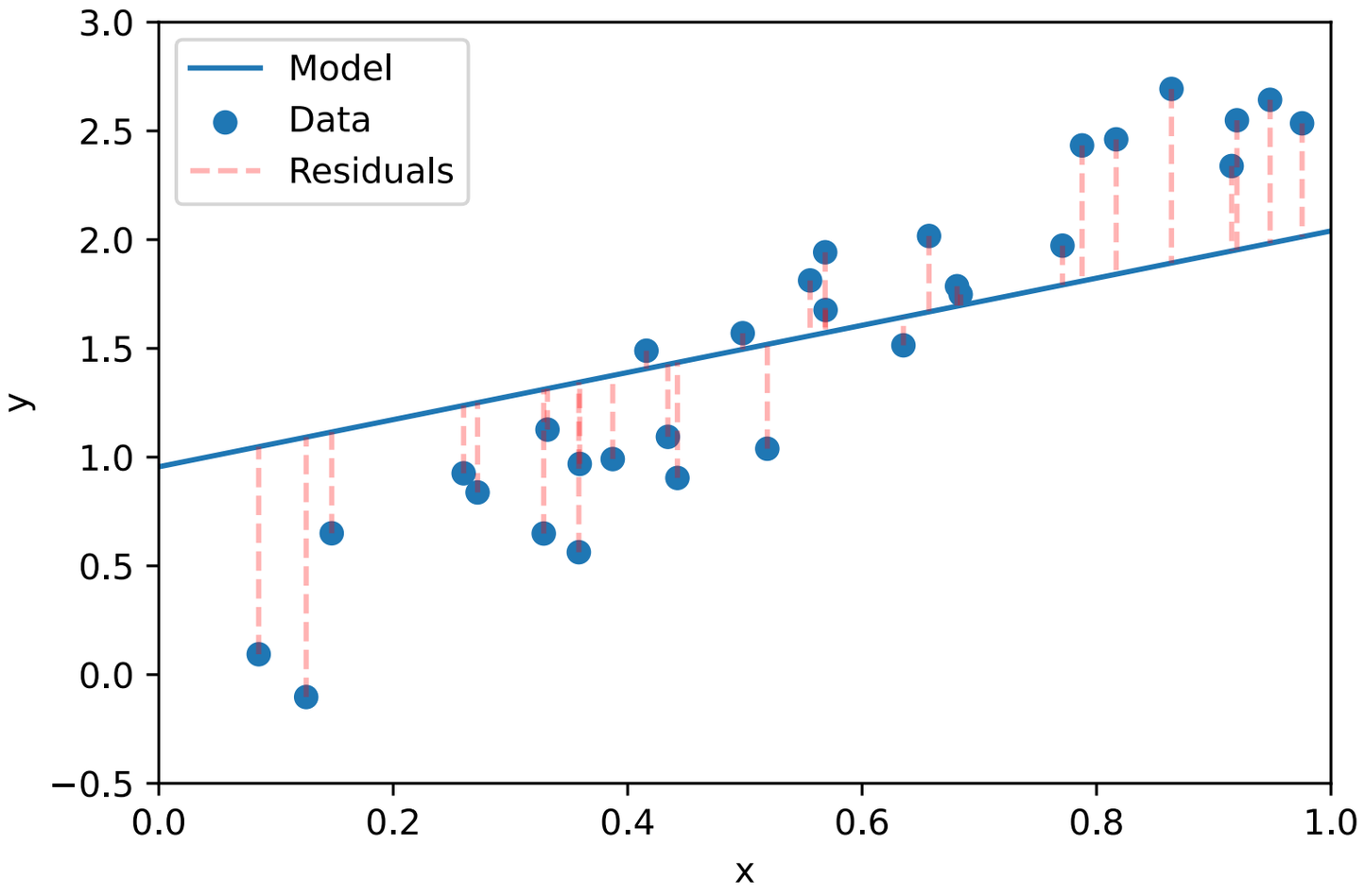


Objective



Dennis Noll - 08.08.22

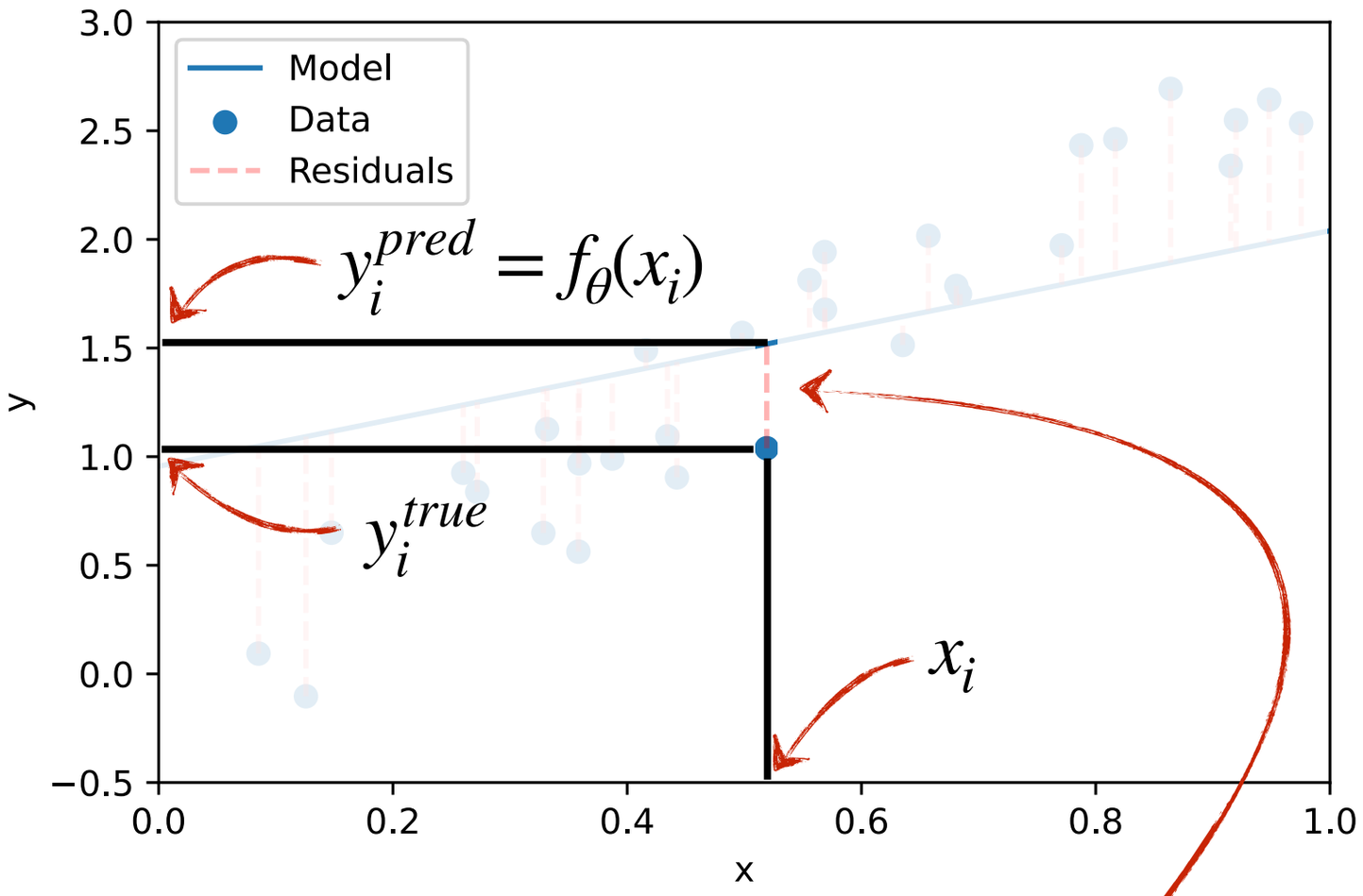
- How good is the prediction of the model?



Objective

Dennis Noll - 08.08.22

- How good is the prediction of the model?

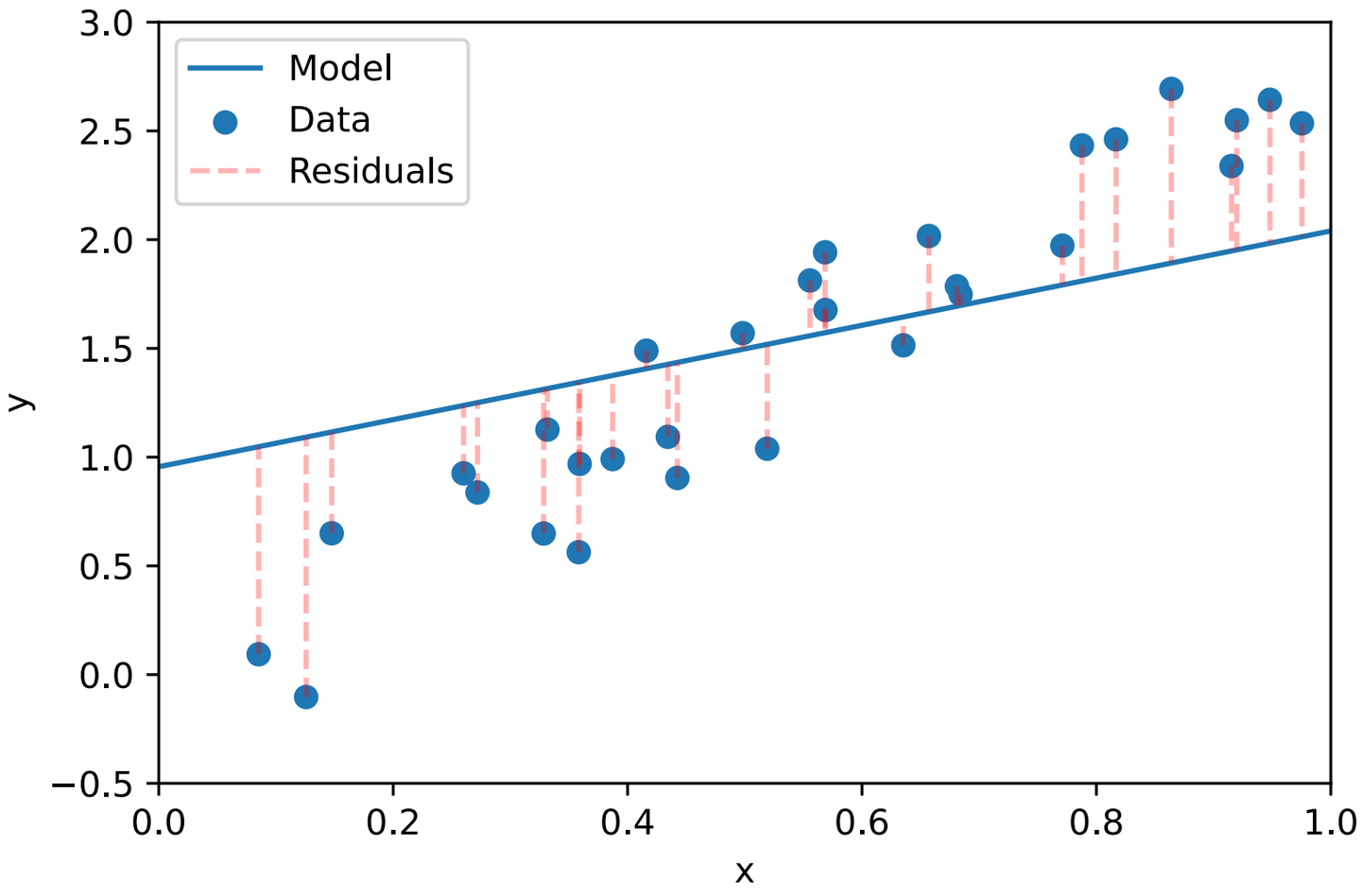


$$\mathscr{L} = \qquad (y_i^{true} - y_i^{pred})^2$$

Objective

Dennis Noll - 08.08.22

- How good is the prediction of the model?

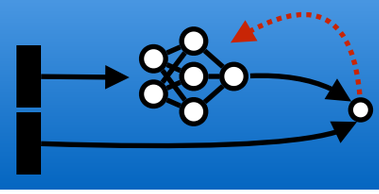


$$\mathscr{L} = \sum_i (y_i^{true} - y_i^{pred})^2$$

Small $\mathscr{L}$ = good modelling

Large $\mathscr{L}$ = bad modelling

Training (Gradient Descent)



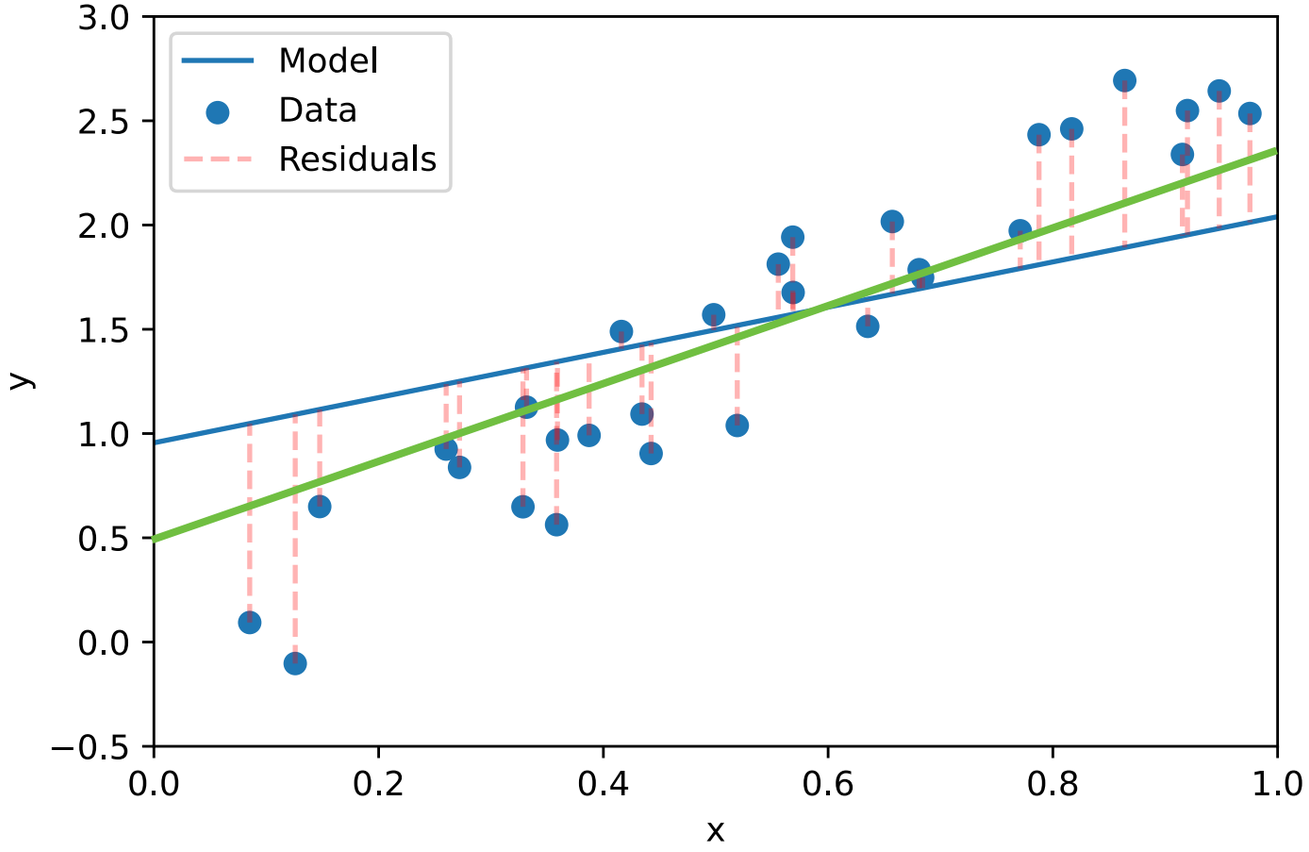
RWTH AACHEN UNIVERSITY
Dennis Noll - 08.08.22

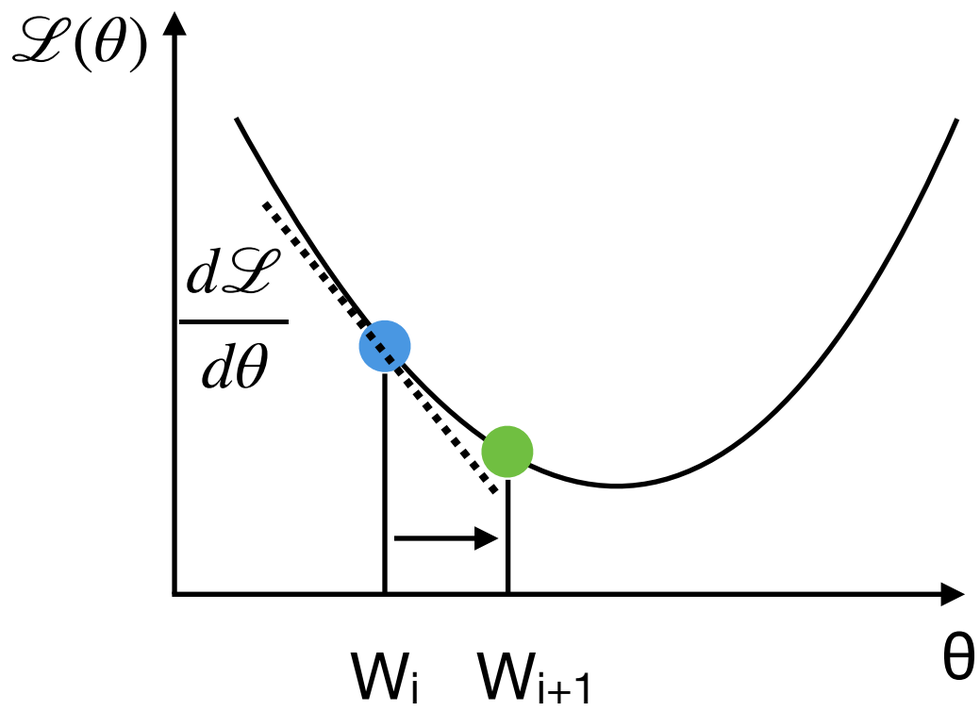
- Minimize objective function $\mathscr{L}(\theta)$
- Update model (θ) in opposite direction of gradient iteratively

$$\theta \rightarrow \theta - \alpha \frac{d\mathscr{L}}{d\theta}$$

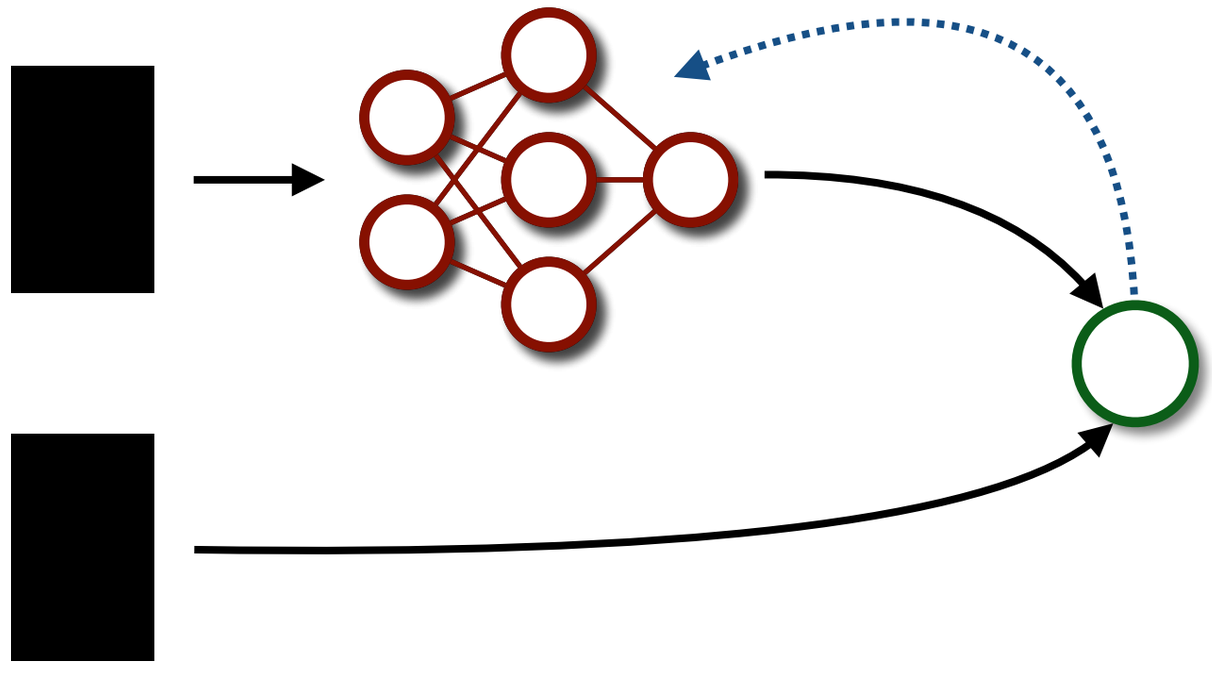
Step size (learning rate)

Gradient





Summary

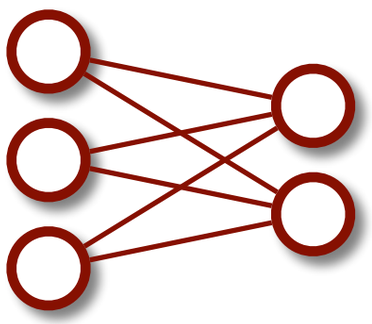


Model

- Linear Perceptron

  $y = w \cdot x + b$

  Parameter $\theta = (w, b)$



Objective

- Fit between training data and prediction
- Regression (mse):

  $\mathscr{L} = (y_{true} - y_{pred})^2$

Training

- Find parameters which minimize loss ($\mathscr{L}$)
- Gradient descent

  $\theta \to \theta - \alpha \frac{d\mathscr{L}}{d\theta}$

Python

- **High-level**, **interpreted** (no compilation), **multi-purpose** programming language
- Many applications:
  - Software development
  - Web development
  - Scientific computing
  - Machine learning
- Designed for readability
- Code structured in lines, scopes defined by indentation

```
def state_count(count):
  print(f"Count is {count}")

for i in range(2):
  state_count(i)
```

```
Count is 0
Count is 1
```

NumPy: Numerical Python

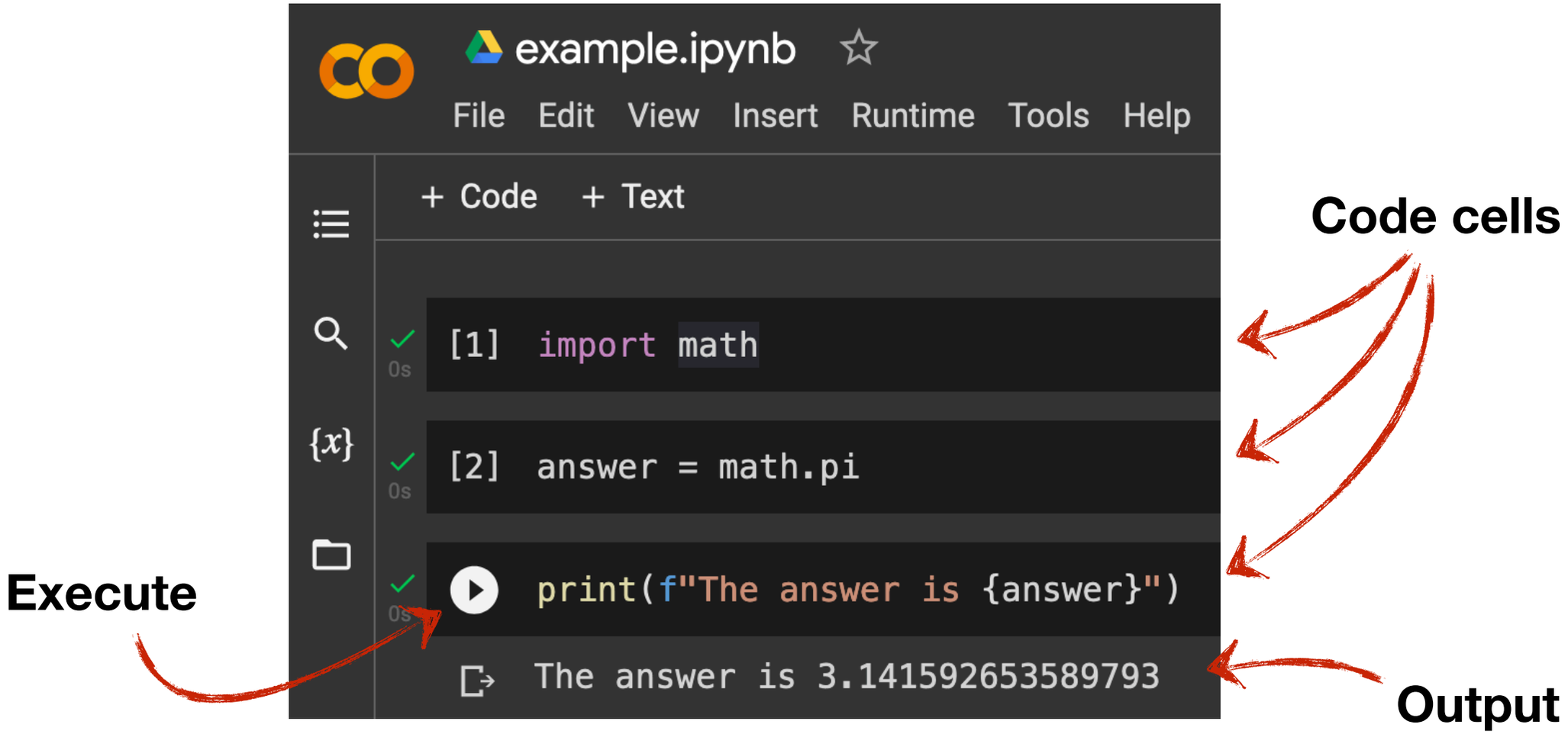
- The package for **scientific computing with Python**
- Implements:
  - Arrays
  - Matrices
  - Linear algebra
  - …
- Central element: ndarray (50x faster than Python List)
- Most low level operations implemented in C/C++ code

```python
import numpy as np

a = np.array([1, 2, 3])
a = a**2 + 1

print(a)
```

```
[ 2  5 10]
```

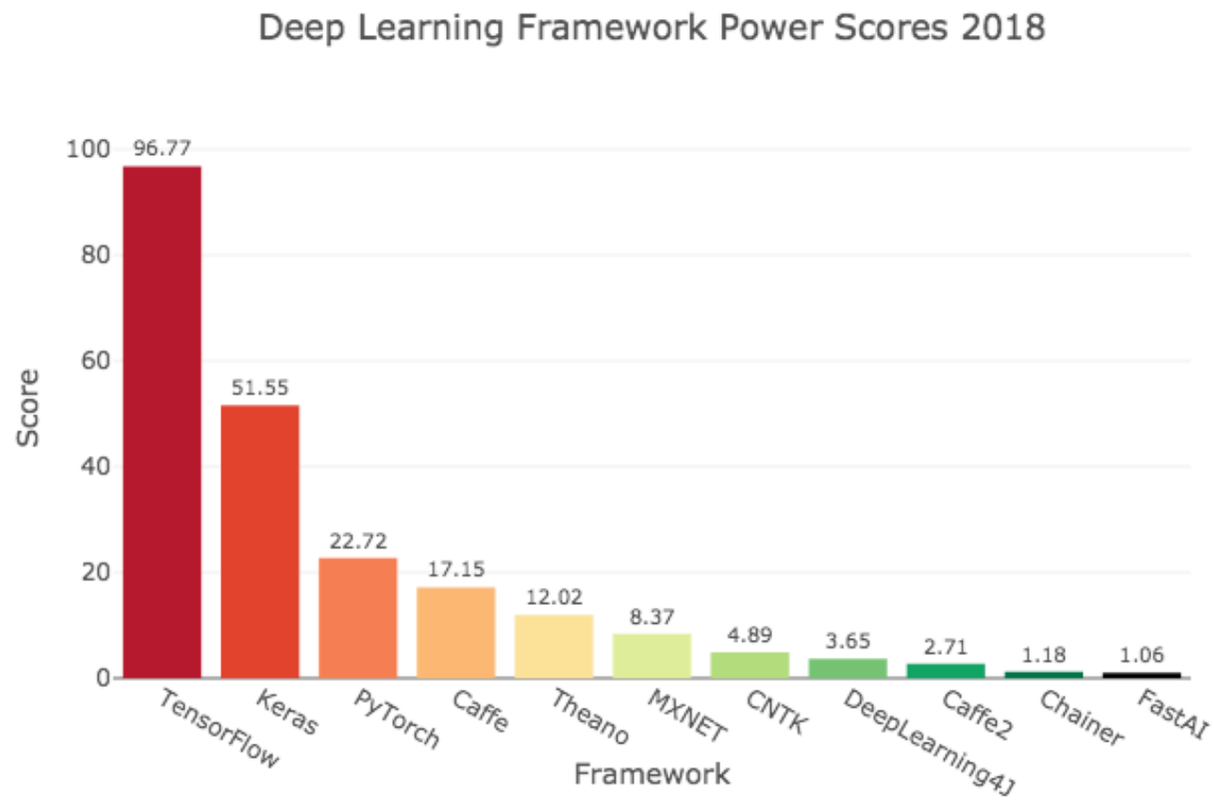
Jupyter Notebooks + Google Colab

- Exercises in Jupyter Notebooks
- Execute Python code in cells
- Computations performed by Kernel:
  - Runs on server, gets tasks by notebook
  - Stateful (variables, …)
- We will use the google colab servers



Deep Learning Libraries

- Various deep learning libraries available
- Many are open source
- Field dominated by industrial players



TensorFlow

- Developed by Google
- De-facto standard

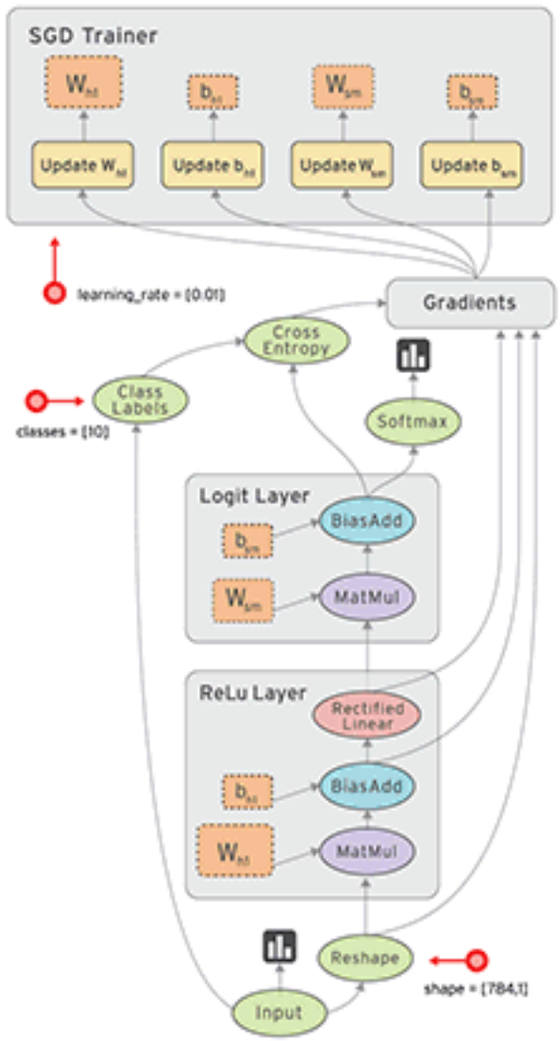
PyTorch

- Developed by Meta
- Pythonic feeling

[17]

TensorFlow

- TensorFlow: Open source Software library for numerical computations using dataflow graphs
  - Purpose:
    - Load & preprocess data
    - Build, train & reuse models
    - Deploy models
  - Central concepts:
    - Nodes: mathematical operations
    - Edges: tensors connecting nodes
- Keras integration for easy model building



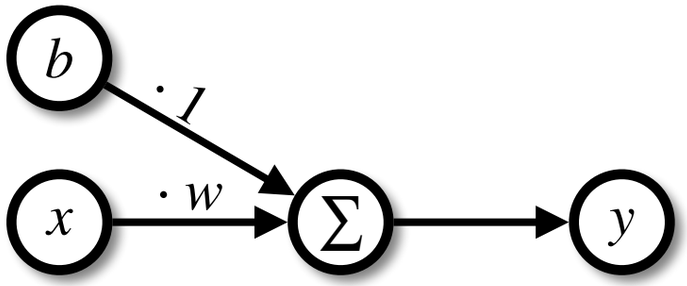
[18]

Our First Model (Creation)

[1]:
```
import tensorflow as tf
```

[2]:
```
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(1, input_shape=(1,)),
])
```

Create model:



[3]:
```
model.layers
```

[3]:
```
[<tensorflow.python.keras.layers.core.Dense at 0x7fb91c4f3b80>]
```

[4]:
```
for layer in model.layers:
    for weight in layer.weights:
        print(weight.name, weight.numpy())
```

```
dense/kernel:0 [[0.1140337]]
dense/bias:0 [0.]
```

Check weights:

$$w = 0.1140337$$
$$b = 0$$

[5]:
```
model.predict([[1]])
```

Make prediction:

[5]:
```
array([[0.1140337]], dtype=float32)
```

$$y = 0.1140337$$

Our First Model (Training)

[6]:
```
x = [[1], [2], [3], [4]]
y = [[2], [4], [6], [8]]
```

Define Training data

[7]:
```
model.compile(
    loss="mse"
    optimizer=tf.keras.optimizers.SGD(),
)
```

Define loss
Define optimizer

[8]:
```
model.fit(x, y, epochs=300, verbose=0)
```

Perform training

[8]:
```
<tensorflow.python.keras.callbacks.History at 0x7fb91c523d30>
```

[9]:
```
for layer in model.layers:
    for weight in layer.weights:
        print(weight.name, weight.numpy())
```

Check weights

```
dense/kernel:0 [[1.9204563]]
dense/bias:0 [0.23386857]
```

$w = 1.9204563$
$b = 0.23386857$

[10]:
```
model.predict([[1]])
```

Make prediction

[10]:
```
array([[2.1543248]], dtype=float32)
```

$y = 2.1543248$

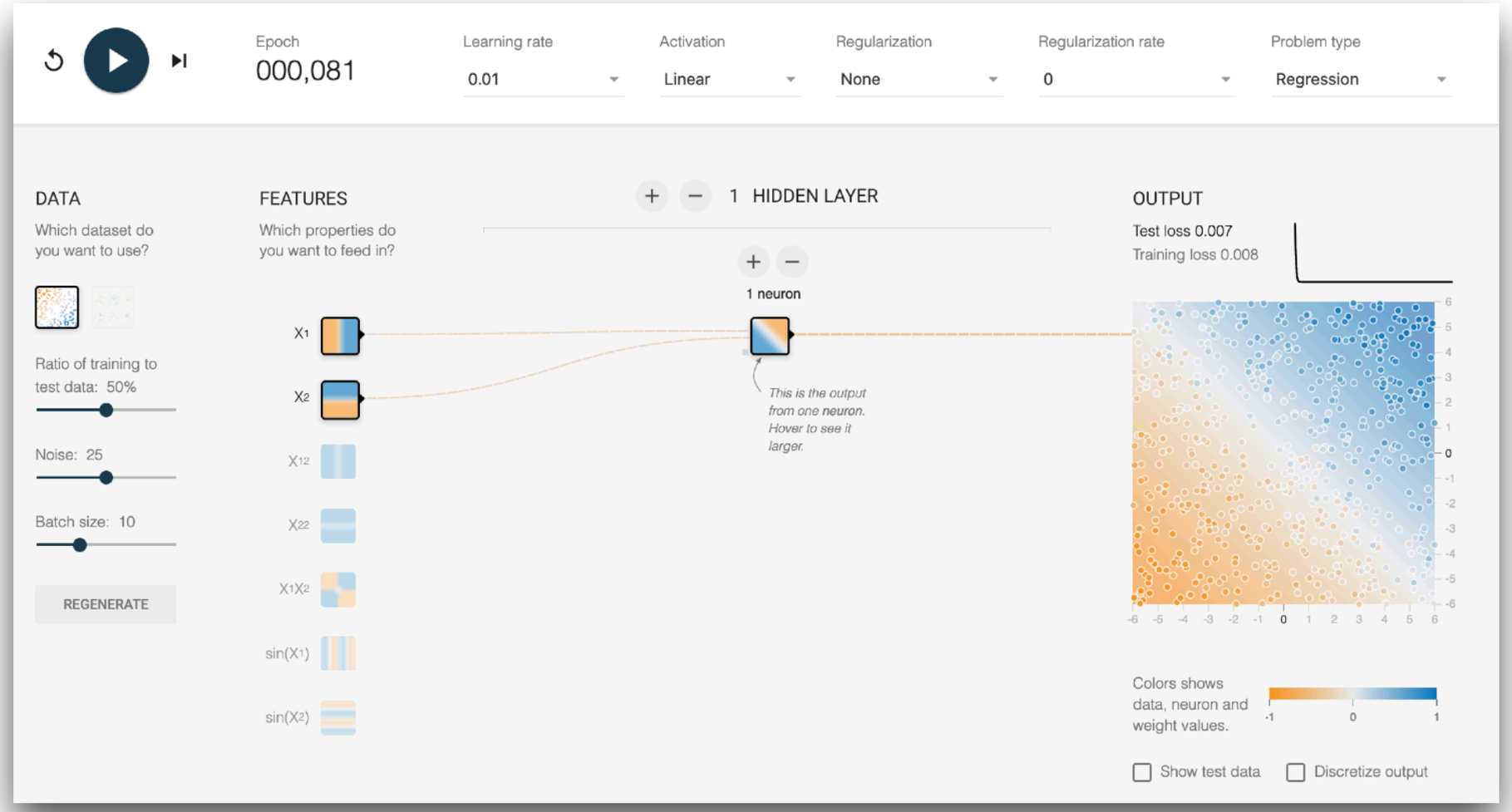
Warm Up: Python, TensorFlow, PyTorch

- To get you started with the basic tools, we have prepared basic tutorials on Python, TensorFlow and PyTorch. Visit the notebooks and work through the sections.
  - Python Tutorial:
    - Access here
    - Includes: Variables, Data-Types, Arithmetics, Comparison Operators, Control Flow, Functions, Classes and NumPy.
  - TensorFlow Tutorial:
    - Access here
    - Includes: Tensors, Variables, Fundamental Mathematical Operations
  - PyTorch Tutorial:
    - Access here
    - Includes: Tensors, Fundamental Mathematical Operations

Exercise 1.1 - Linear Regression (Playground)

RWTH AACHEN UNIVERSITY
Dennis Noll - 08.08.22

- Try the Linear Regression example at playground.tensorflow.org
  1. How many parameters does the model have?
  2. Train the model and state the final loss ("Training Loss").
  3. What happens for different inputs (X^1 & X^2, only X^1, only X^2)?
  4. What happens for different learning rates (0.0001, 0.001, 0.01 ,0.1, 1, 10)?



Exercise 1.1 - Linear Regression (Playground)

- **Notes/Solutions**:

Exercise 1.2 - Linear Regression (tf.keras)

- In this task we will use TensorFlow to perform a linear regression. We will first look at a one-dimensional case. Then we will raise the number of dimensions and learn how the used neural network performs in high dimensional spaces.
  - Open this notebook
  - Inspect the data
  - Build a model containing one node
  - Plot the predictions of your model
  - Compile the model (loss="mse", optimizer="sgd").
  - Train your model for 20 epochs. After each 2 epochs of training plot the predictions of your model. What do you observe?
  - Inspect the weights of your model. What do you observe?
  - What happens if the measured data (y) is uncertain (uncertainty > 0)? Explain your observation!
  - Vary the number of input dimensions to n=1/2/10. How do you need to change the model? Describe your observation.

Exercise 1.2 - Linear Regression (tf.keras)

- **Notes/Solutions**:

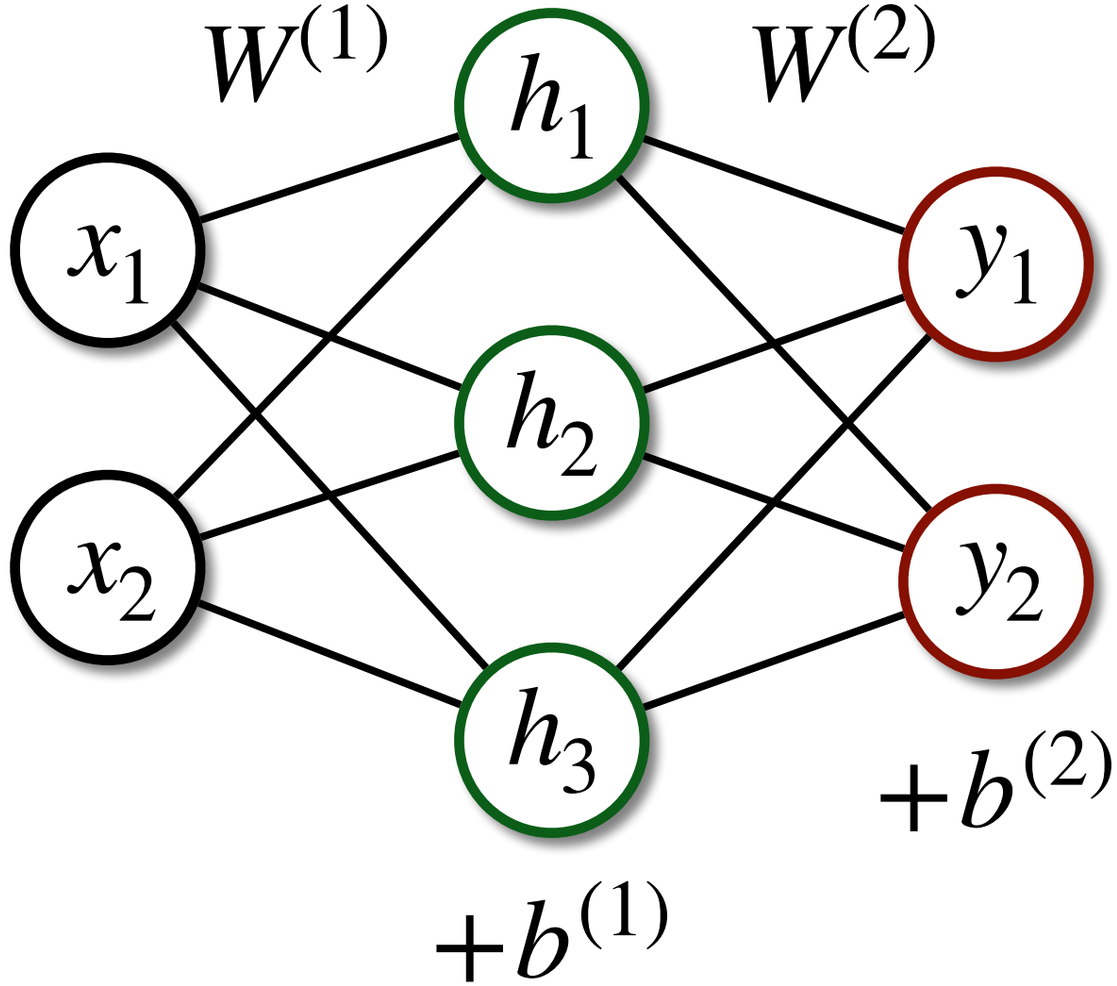
Exercise 1.3 [advanced] - Many Linear Layers

RWTH AACHEN UNIVERSITY
Dennis Noll - 08.08.22

- Using pen and paper, show that a chain of multiple linear layers, can always be represented as a single linear layer.
- **Hints:**
  - Revisit the matrix formulation of a linear layer.
  - You may start your proof with a chain of two linear layers.

$$h = W^1 x + b^1$$

$$y = ?$$



Exercise 1.3 [advanced] - Many Linear Layers

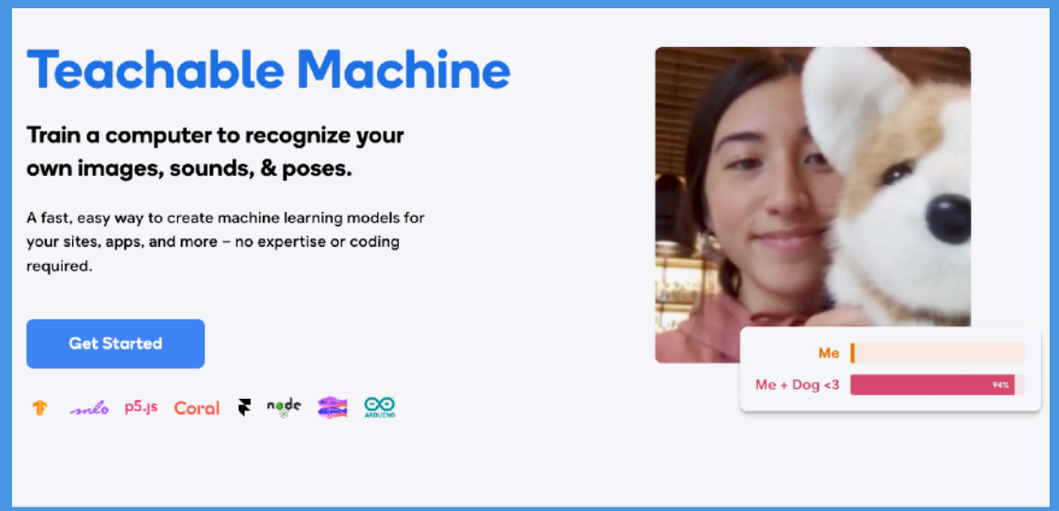
RWTH AACHEN UNIVERSITY
Dennis Noll - 08.08.22

- **Notes/Solutions**:

Exercise 1.4 [advanced, motivation] - Try Out!

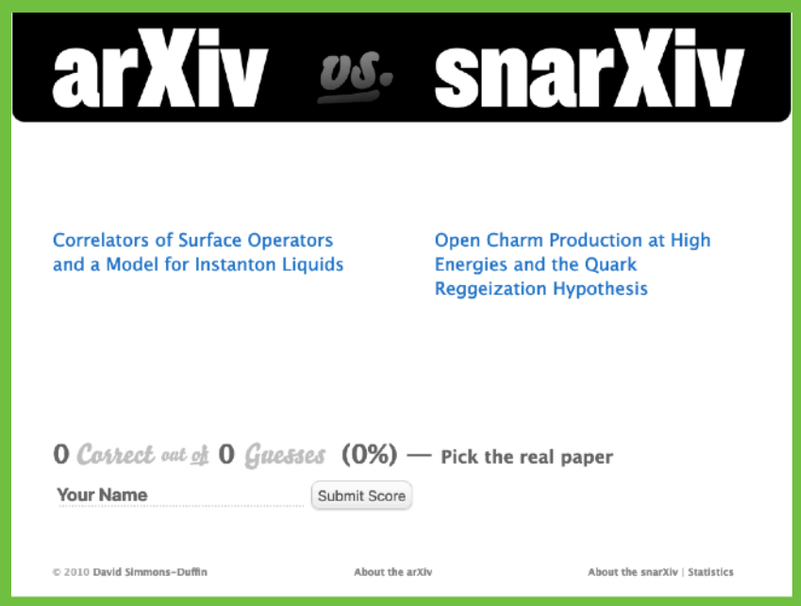
RWTH AACHEN UNIVERSITY
Dennis Noll - 08.08.22

Teach the machine!



Play with the machine!

Play agains the machine!



Do not get fooled by the machine!

Citations

1. **Deep Learning in Physics Research Lecture**, Martin Erdmann, Jonas Glombitza, Uwe Klemradt, Dennis Noll, RWTH Aachen University
2. **Dartmouth Workshop**: Photo by Margaret Minsky
3. **Garri Kasparow gegen Deep Blue** am 7. Mai 1997 in New York (Stand Honda/Getty Images)
4. **Cole Murray Medium**: Link (accessed 04.08.22)
5. **Amazon Echo**: Amazon Link (accessed 04.08.22)
6. **DeepL Translator**: DeepL Link (accessed 04.08.22)
7. **All Systems Go**: Nature, Volume 529 Issue 7587, 28 January 2016, Link (accessed 04.08.22)
8. **Training a robotik task with Reinforcement Learning**: Siemens, Link (accessed 04.08.22)
9. **Self-Driving Car**: Google
10. **Concious AI**: Shutterstock Willyam Bradberry
11. **CMS event displays of Higgs to two photon candidate**: The CMS Collaboration, Link (accessed 04.08.22)
12. **Large Hadron Collider**: CERN
13. **Pierre Auger Observatory**, ASPERA/G.Toma/A.Saftoiu
14. **MicroBooNE Event Display**, The MicroBooNE Collaboration
15. **Properties of Synchrotron Radiation**, Birkbeck College, University of London Link (accessed 04.08.22)
16. **Artist impression gravitational waves**, R. Hurt/Caltech-JPL
17. **Deep Learning Framework Power Scores**, Jeff Hale, Link (accessed 04.08.22)
18. **Tensorflow Graph**, Easy-TensorFlow Team Link (accessed 04.08.22)