Deep Learning for Physicists

Lecture #1: Introduction

Kosmas Kepesidis

Outline

- About the course
- Introduction to artificial intelligence (AI)
- Models for data analysis
- Building blocks of neural networks

About the course

About the course

- Things to keep in mind:

  - Comprehensive introductory course on deep learning – adapted for physicists

  - Target audience: BSc & MSc students of physics

  - Prior knowledge on the topic is not required!

  - Prerequisites: basic math and physics from the BSc in physics and some programing experience (preferably with Python)

About the course

- Course content:

    1. Basics of artificial neural networks

    2. Architectures of deep neural networks

    3. Interpretability, uncertainties, objectives

About the course

- Course content:
  - Basics
    - Data and analysis in physics
    - Building blocks of neural networks
    - Optimization of neural networks (model training)
    - Practical methodology (basics of machine-learning engineering)

About the course

- Course content:
  - Architectures of deep neural networks
    - Fully-connected neural networks
    - Convolutional neural networks (CNNs)
    - Recurrent neural networks (RNNs)
    - Graph networks
    - Hybrid architectures

About the course

- Course content:
  - Interpretability, uncertainties, objectives
    - Interpretability
    - Uncertainties
    - Objective functions: deeper insights

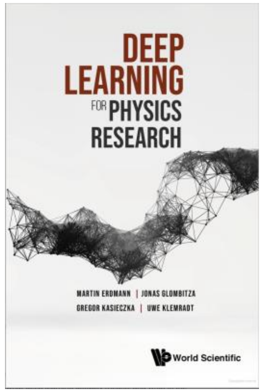
About the course

- Next semester: Advanced Deep Learning for Physicists
    - Advanced topics
        - Unsupervised and semi-supervised learning
        - Autoencoders
        - Generative models
        - Outlier and anomaly detection
        - Special topics
        - Mathematical foundations

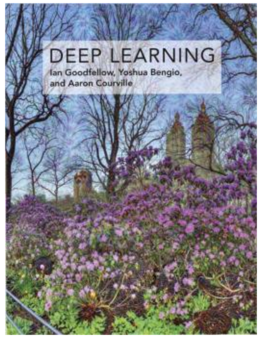
About the course

- Textbooks used for the lectures:

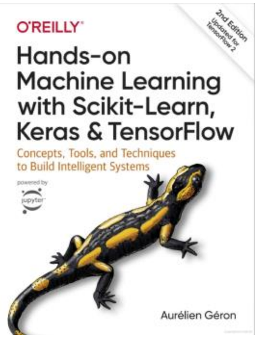
  - **Deep Learning for Physics Research**, Gregor Kasieczka, Jonas Glombitza, and Martin Erdmann (2021)



  - **Deep Learning**, Ian Goodfellow, Yoshua Bengio, Aaron Courville (2016)
    - Open access: https://www.deeplearningbook.org/



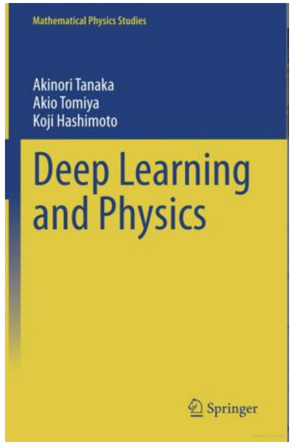
  - **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow** (2nd Edition), Aurélien Géron (2019)



About the course

- Another textbook… not used here but still interesting!

  - ➢ **Deep Learning and Physics**, Akinori Tanaka, Akio Tomiya, Koji Hashimoto (2021)



About the course

- Tutorials:
    - Problems sets
    - Computational assignment – Jupyter notebook in Python
        - Assignments will not contribute to the grade

- Written final exam :
        - Final exam will contribute to 100% of your grade

About the course

- Course material:
  - Moodle: “Deep Learning for Physicists” https://moodle.lmu.de/course/view.php?id=25091
    - Lecture slides
    - Problem sets (+ solutions)
    - Jupyter notebooks (+ solutions)
    - Useful papers

About the course

- Software:
  - Python language: https://docs.python.org/3.7/tutorial/index.html
    - Would be useful to get familiar within the first two weeks of the course
  - Anaconda distribution (open source): https://www.anaconda.com/
    - Recommend to download it – will be used in the tutorials

ANACONDA

Introduction to artificial intelligence (AI)

Introduction to artificial intelligence (AI)

- The field of AI is driven by the desire to create machines that think
- In the early days of AI, problems that are difficult for humans and easy for computers were tackled – **rule-based systems** (RBS)

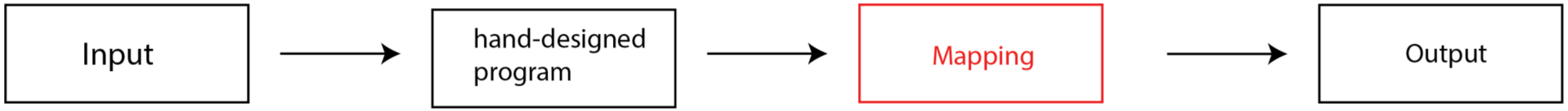
- Early success based on RBS: IBM's Deep Blue chess computer defeated world champion Garry Kasparov in 1997 (Hsu, 2002)

Introduction to artificial intelligence (AI)

- The real challenge of AI remained to tackle problems that are straightforward to humans
- **Knowledge based** approach to AI: knowledge about the world hard-coded in a computer program
- Cyc (Lenat and Guha, 1989): inference engine + database of statements
- Memorizes large collections of general knowledge
- Has proven challenging in practice (Linde, 1992)

Introduction to artificial intelligence (AI)

- Solution: ability to acquire own knowledge by discovering patterns in data
- This approach is known as **machine learning** (ML)
- Examples of *classical* ML algorithms: logistic regression, decision trees, support vector machines, naïve Bayes

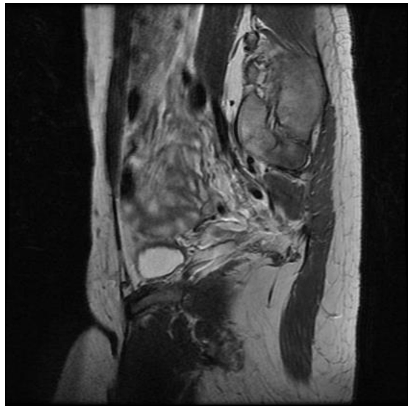


- Examples of problems ML solves:
  - ➢ recommend cesarean delivery (Mor-Yosef et al., 1990)
  - ➢ separate legitimate e-mail from spam e-mail

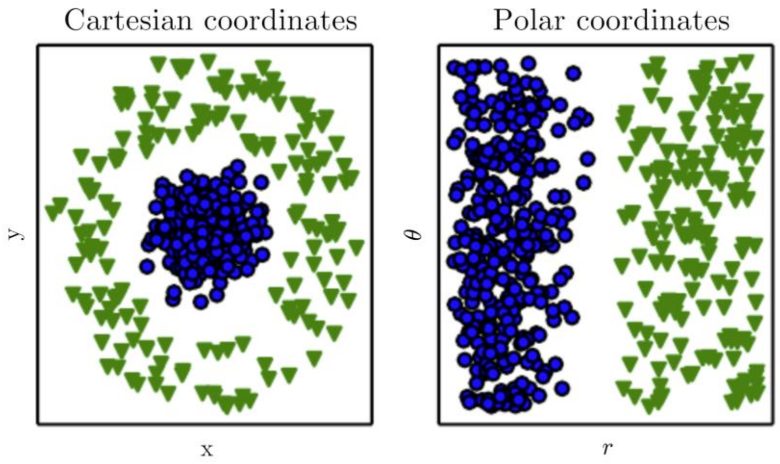
Introduction to artificial intelligence (AI)

- ML works well with predefined/preselected **features** (or **observables**) selected by researcher

- Limitation of ML: no influence on feature definition
  - Example: ML-based cesarean-delivery recommender depends on the doctors report (features), doctor needs to interpret MRI scan of patient



- **Representations** play crucial role in ML

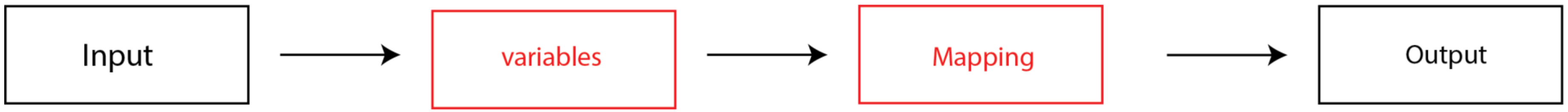


Source: Goodfellow et al., Deep Learning

Introduction to artificial intelligence (AI)

- Use machine learning to discover not only the mapping from representation but also the representation itself

- Approach known as **representation learning**

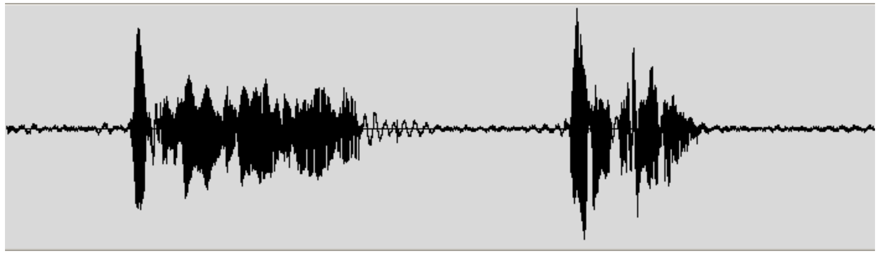


- Example from physics: extracting velocity of particle from location and time data

- Representation learning can be challenging – need to separate the **factors of variation** (FOV), which can be of high abstraction

Introduction to artificial intelligence (AI)

- Example: speech recording
  - ➢ FOV: age, gender, accent

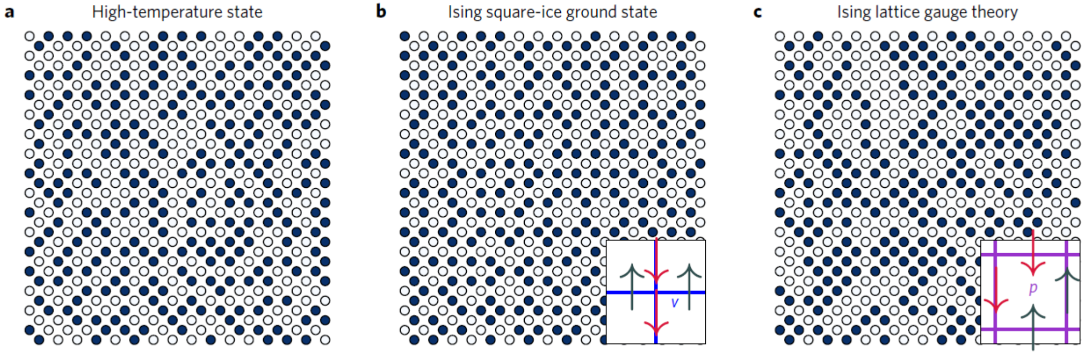


Source: https://cmusphinx.github.io/

- Example: image of a car
  - ➢ FOV: position of the car, its color, angle and brightness of the sun

Source: https://en.wiktionary.org/

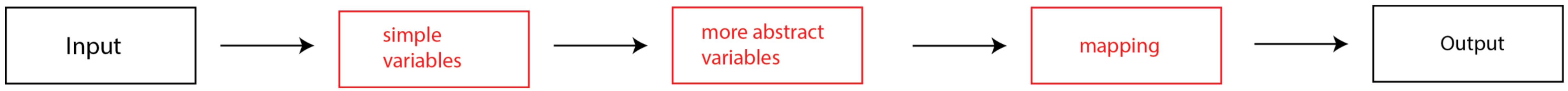
- Example: phases of matter
  - ➢ FOV: temperature



Nature Phys **13,** 431–434 (2017)

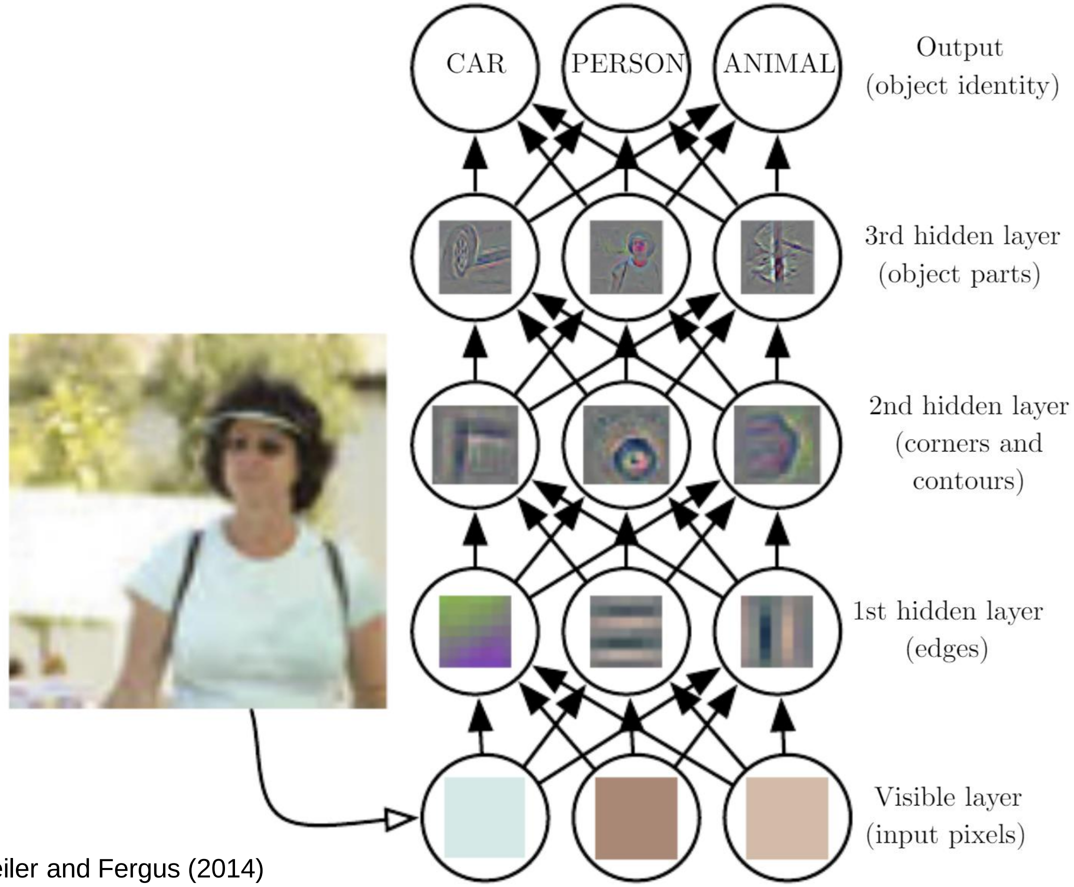
Introduction to artificial intelligence (AI)

- **Deep learning** (DL): constructing more abstract representations from simpler representations *hierarchically*



Introduction to artificial intelligence (AI)

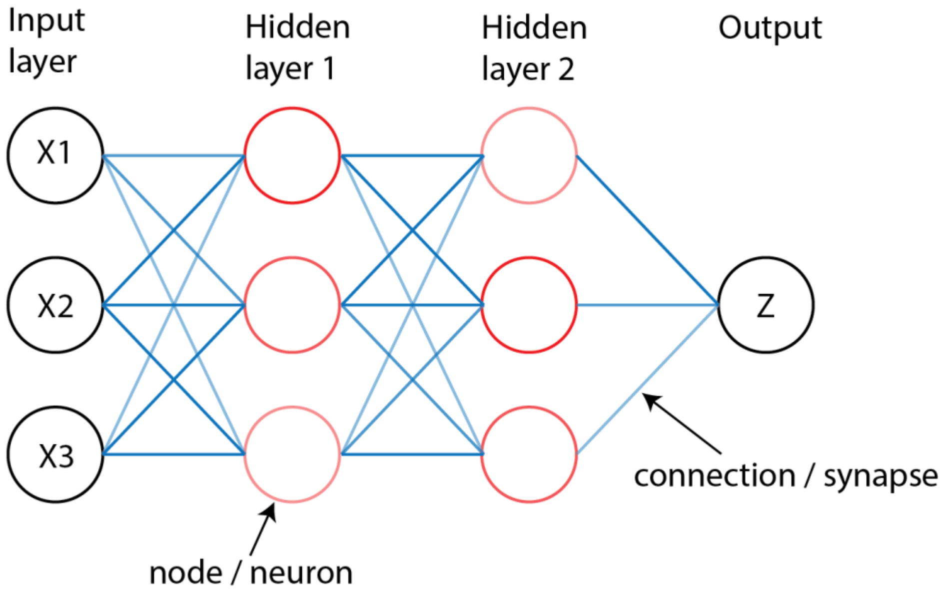
- **Deep learning** (DL): constructing more abstract representations from simpler representations *hierarchically*



Source: Zeiler and Fergus (2014)

Introduction to artificial intelligence (AI)

- **Artificial neural networks** (ANN):
  - Signal flows only in one direction: feedforward neural networks (FNN)



- **Deep neural networks** (DNN): ANNs with many hidden layers

"It's deep if it has more than one stage of non-linear feature transformation."

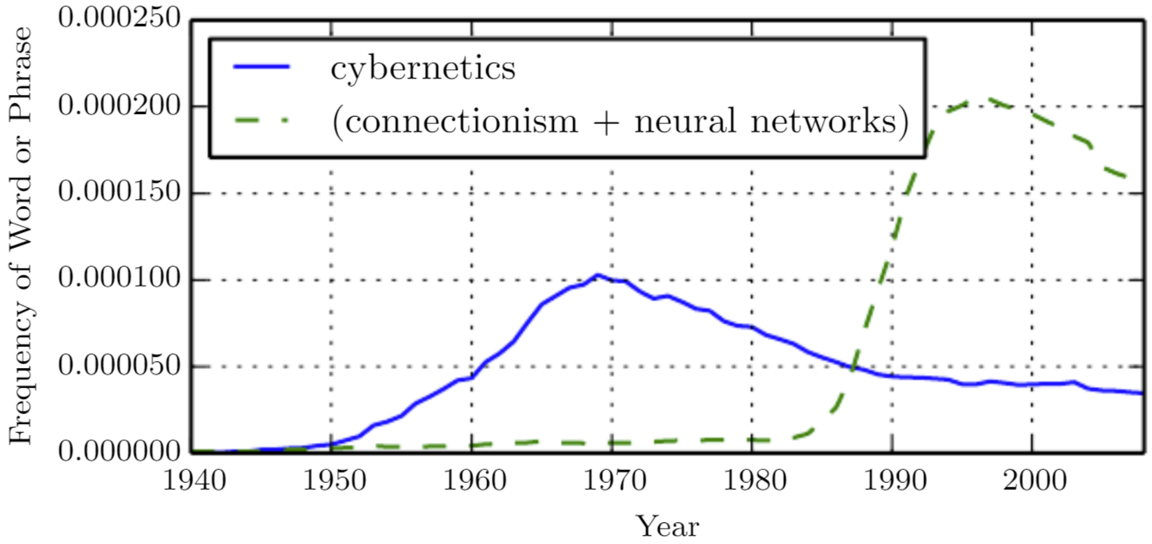
– Yann LeCun, DL researcher

Introduction to artificial intelligence (AI)

- **DNNs** are versatile, powerful, and scalable – can tackle highly-complex tasks such as:
  - ➢ Classification of billions of images
  - ➢ Speech-recognition services (e.g. Apple’s Siri, Amazon’s Alexa, etc)
  - ➢ Recommend products or videos to millions of users
  - ➢ Beat world champions at the game of Go (DeepMind’s AlphaGo)
  - ➢ Development of self-driving cars
  - …

Introduction to artificial intelligence (AI)

- DL has gone with different names in different time periods:
  - 1940s–1960s: **cybernetics** – theories of biological learning
  - 1980s–1990s: **connectionism** – backpropagation (Rumelhart et al., 1986)
  - 2006-now: **deep learning** – advancements in technology

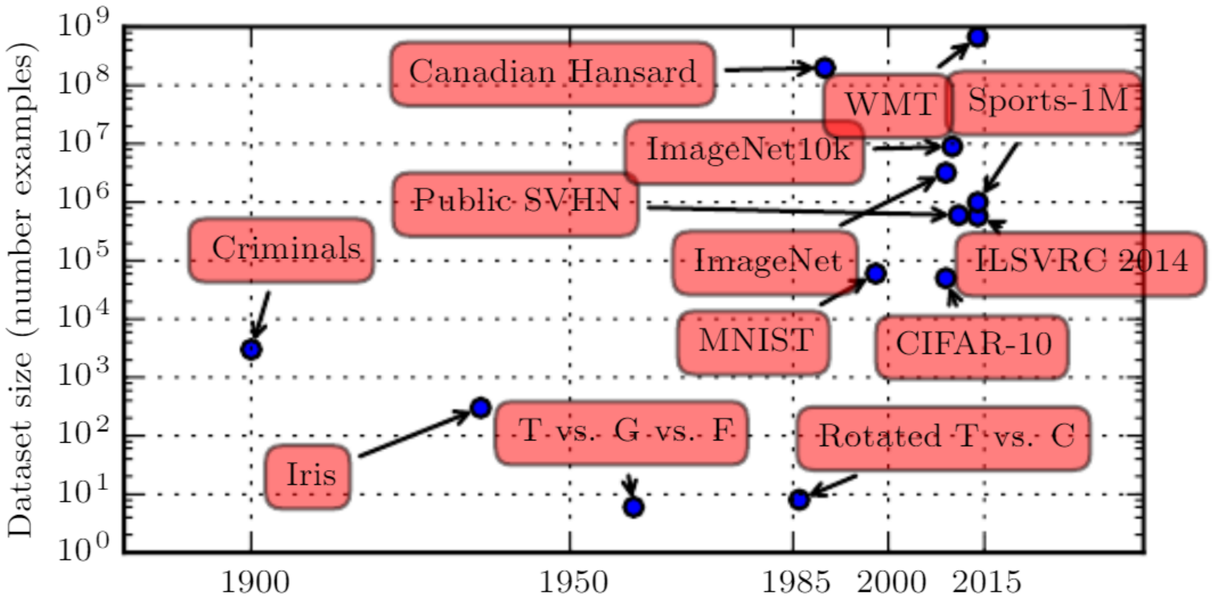


Source: Goodfellow et al., Deep Learning

Introduction to artificial intelligence (AI)

- Causes for rapid developments

➢ amount of available training data has increased

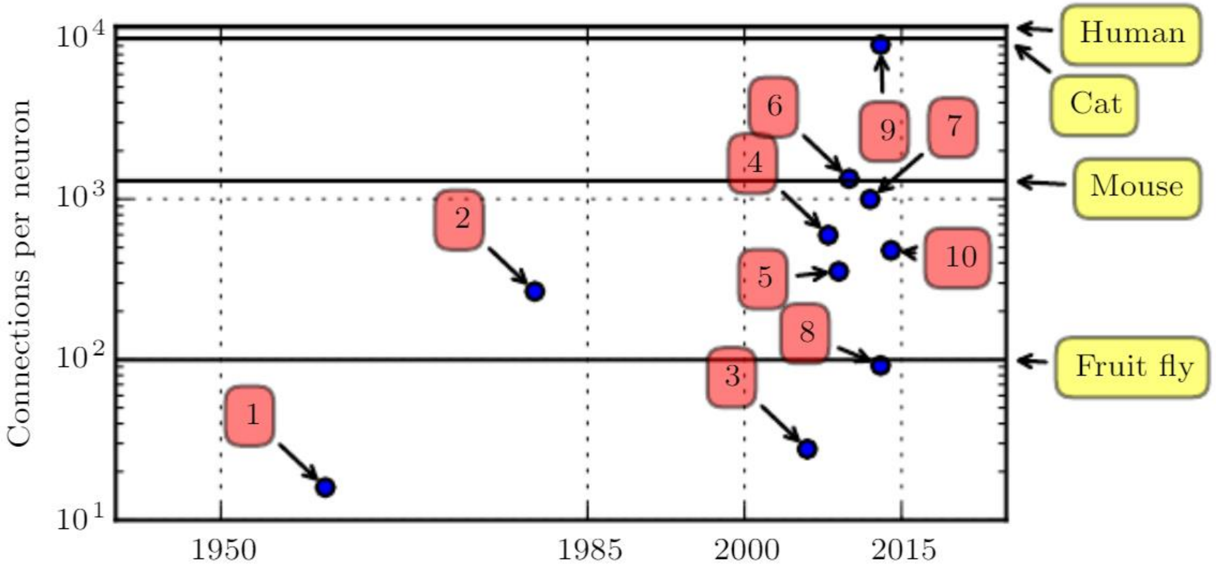


Source: Goodfellow et al., Deep Learning

Introduction to artificial intelligence (AI)

- Causes for rapid developments

➢ Larger and more complex models developed, tailored to application



1. Adaptive linear element (Widrow and Hoff, 1960)
2. Neocognitron (Fukushima, 1980)
3. GPU-accelerated convolutional network (Chellapilla et al., 2006)
4. Deep Boltzmann machine (Salakhutdinov and Hinton, 2009a)
5. Unsupervised convolutional network (Jarrett et al., 2009)
6. GPU-accelerated multilayer perceptron (Ciresan et al., 2010)
7. Distributed autoencoder (Le et al., 2012)
8. Multi-GPU convolutional network (Krizhevsky et al., 2012)
9. COTS HPC unsupervised convolutional network (Coates et al., 2013)
10. GoogLeNet (Szegedy et al., 2014a)

Source: Goodfellow et al., Deep Learning

Introduction to artificial intelligence (AI)

- Causes for rapid developments

  - Hardware advancements such more memory/processing power and access to **Graphics Processing Units** (GPUs) – possibility to train more complex and accurate models

Introduction to artificial intelligence (AI)

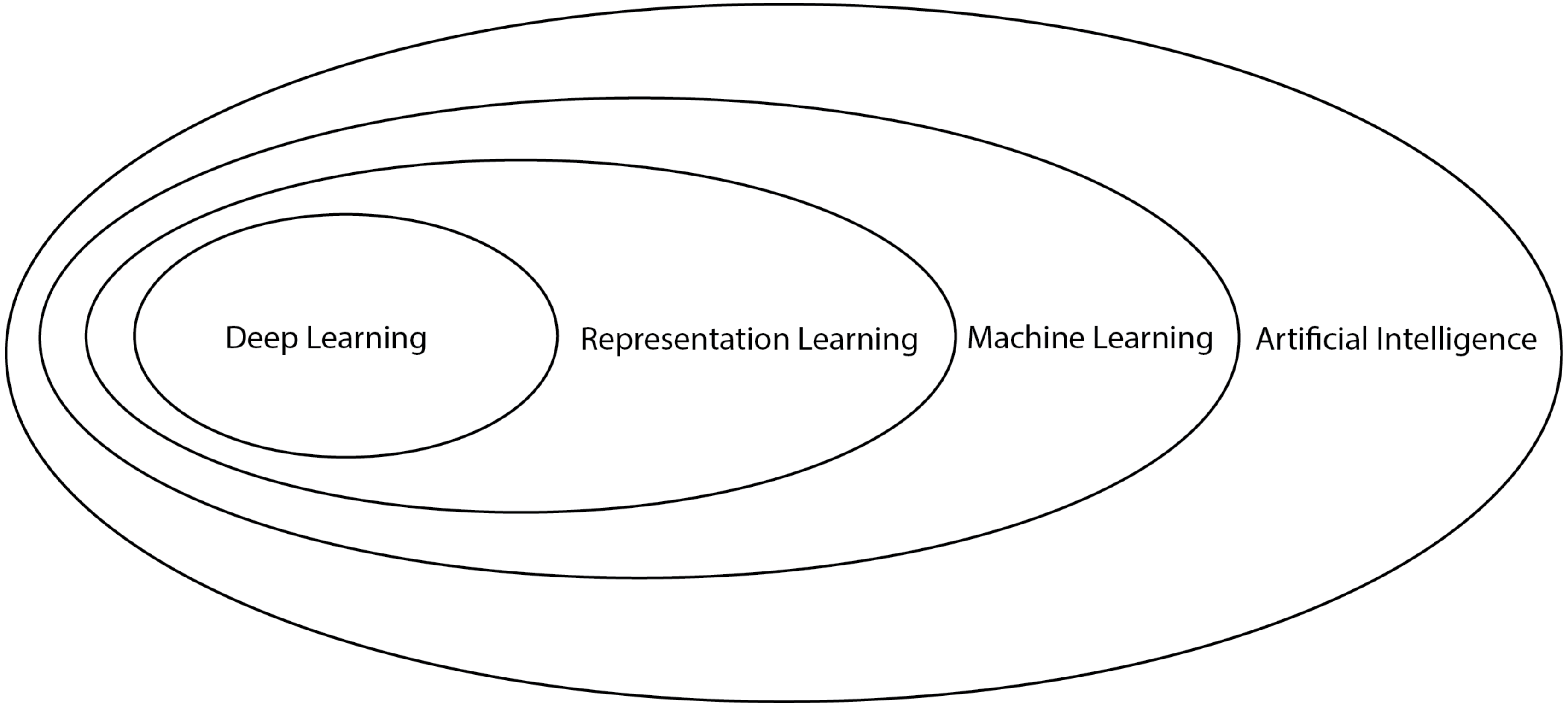
- Causes for rapid developments

  - User-friendly open-source library software for DL:
    - Theano
    - TensorFlow
    - Keras
    - PyTorch

```
from tensorflow import keras
layers = keras.layers

model = keras.models.Sequential()
model.add(layers.Dense(4, activation='relu', input_dim=2))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(loss='MSE', optimizer='SGD', metrics=['accuracy'])
model.fit(xdata, ydata, epochs=200)
```

Introduction to artificial intelligence (AI)



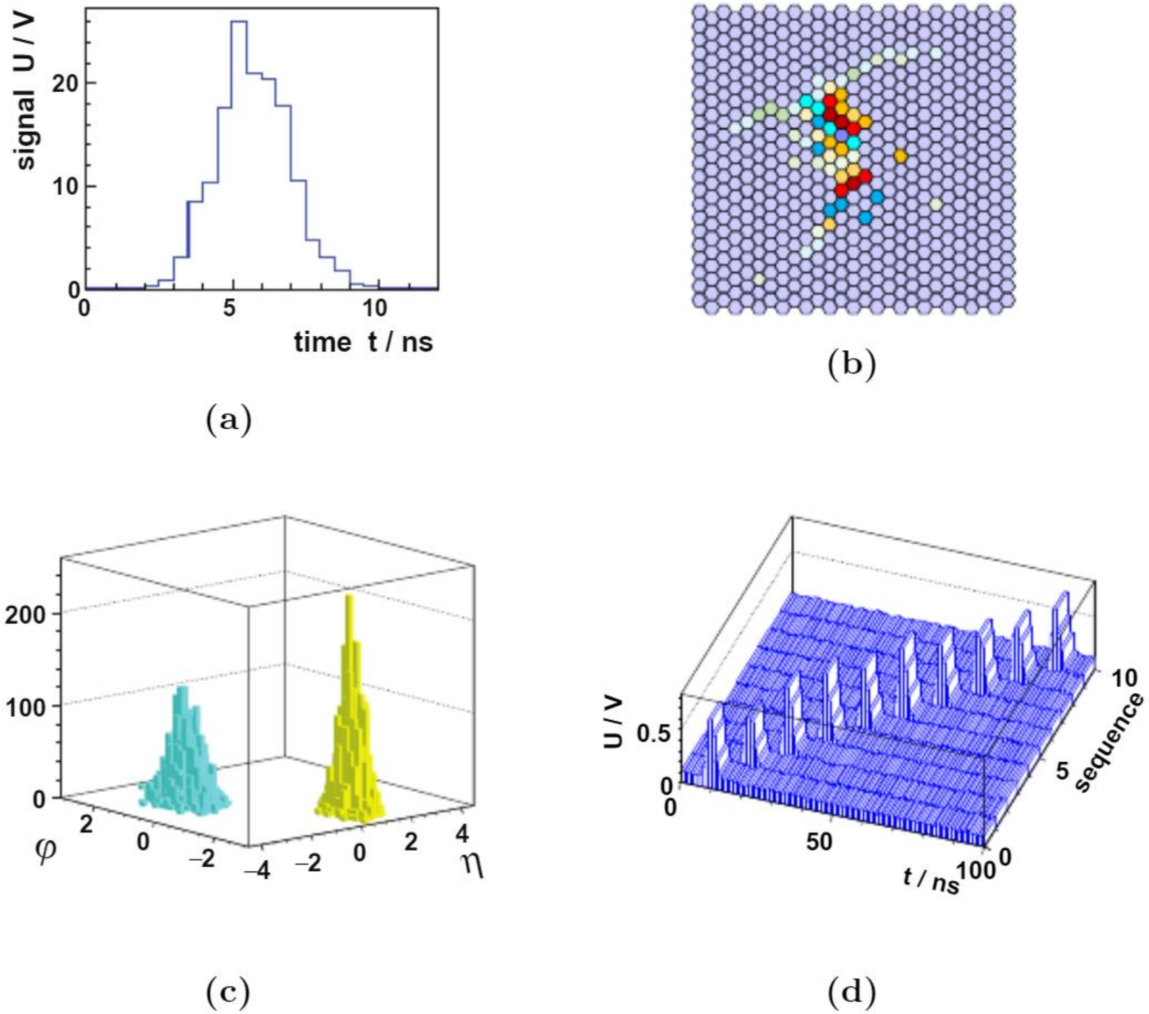
"In AI, system should be understood as including the human engineers. Most of the data-generalization conversion happens during model design."

– Francois Chollet, AI researcher

Models for data analysis

Models for data analysis

- Examples of data in physics
  - (a) amplitude as a function of time
  - (b) camera image of a telescope
  - (c) particles of a collision
  - (d) sequences of sort pulses



Source: M. Erdmann et al., Deep Learning for Physics Research

Models for data analysis

- Data types in physics
  - Data from *k* events
  - 1D sequences
  - Image-like (grid data)
  - Composed: image-like + 1D sequences
  - Point cloud data (3D, spacetime)
  - Heterogeneous data

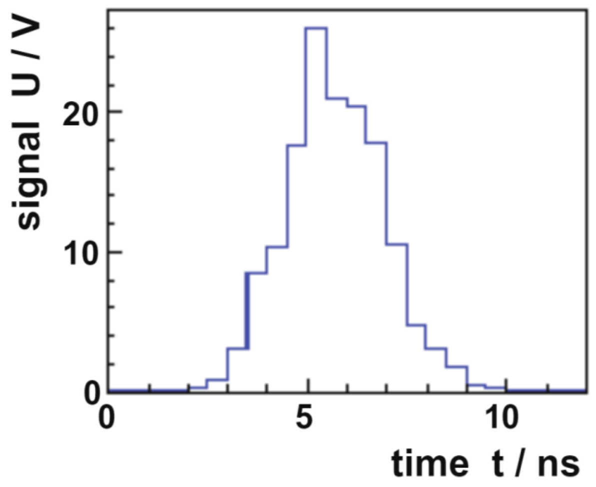
Models for data analysis

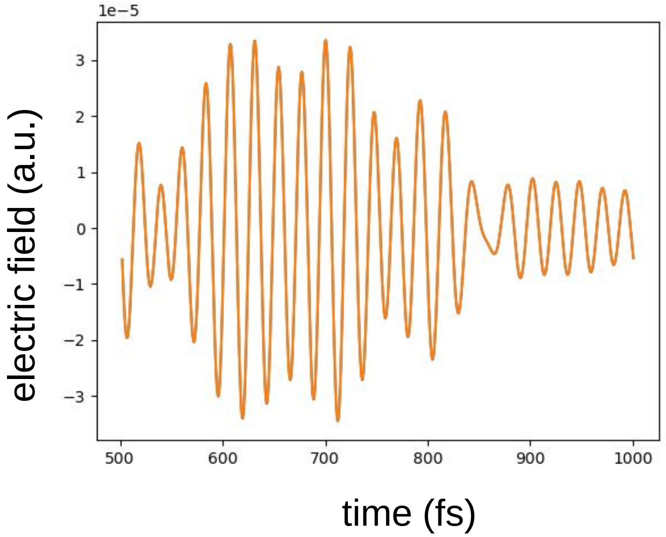
- Data types in physics
  - Data from k events
    - General situation of measuring k times n variables
    - Each event corresponds to an n-dimensional vector: $\vec{x}$
    - This results into an $k \times n$ table (tabular data)

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

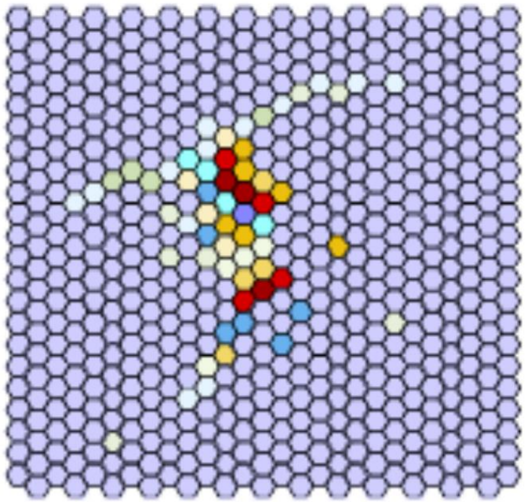
Models for data analysis

- Data types in physics
  - 1D sequences (time series / time traces)
    - Variable x corresponds to an ordered list of numerical values
    - Typically, signal functions $x(t)$
    - But it can be extended to n-dimensional vector $\vec{x}(t)$
    - Examples:
      a. Time-ordered series of electric amplitudes of a microphone or loud-speaker for recording or reproducing audio
      b. Sensors in mobile phones such as a pressure sensor, light sensor, temperature sensor, and three-dimensional acceleration sensors
      c. Electric-field response measured after a laser pulse passes through a liquid medium

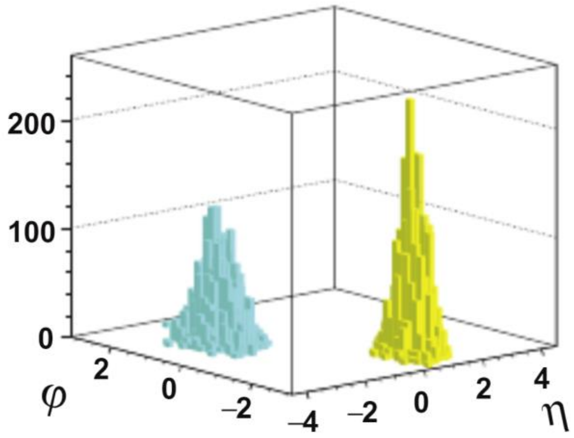




Models for data analysis



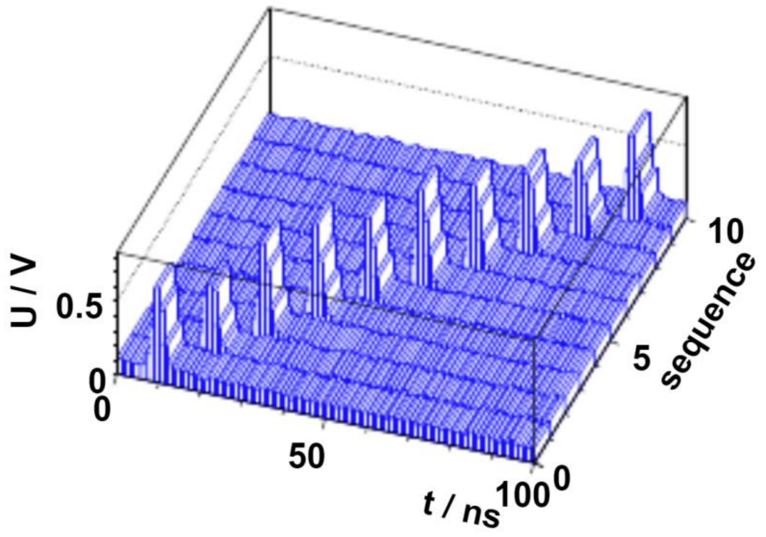
- Data types in physics
  - ➢ Image-like (grid data)
    - Data on a regular 2D (or higher dimensional) grid
    - Examples include regular digital images: pixels arranged on a 2D Cartesian grid
    - In astronomy such data result form instruments such as telescopic cameras
      - Sensors typically arranged in hexagonal structure instead of Cartesian
    - In particle physics: particles of a collision projected onto a physics-motivated coordinate system



Source: M. Erdmann et al., Deep Learning for Physics Research

Models for data analysis

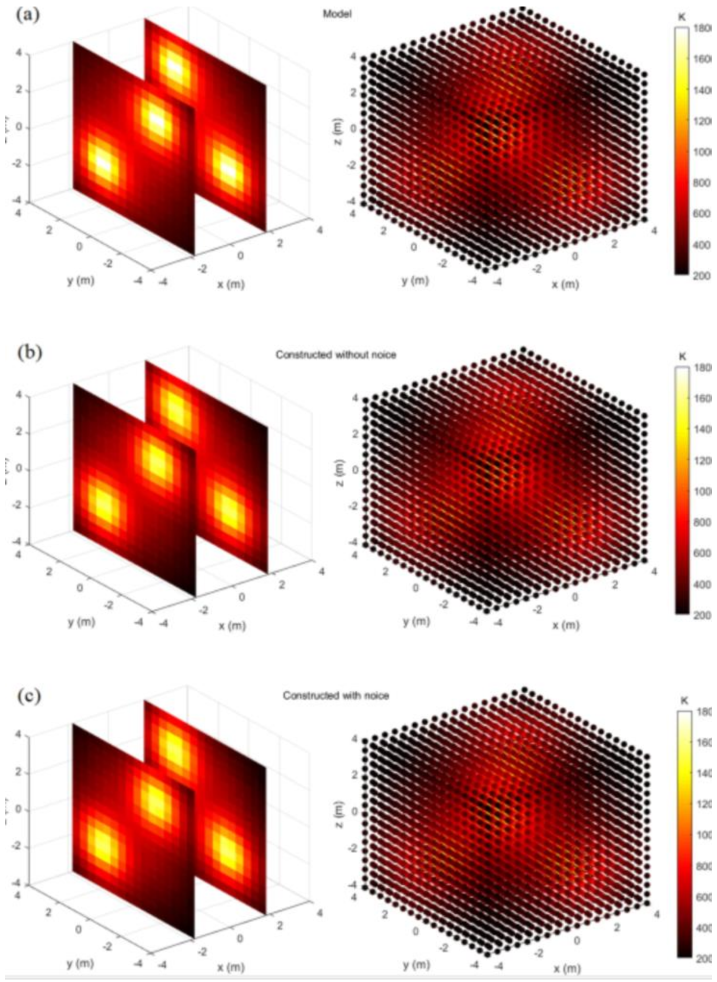
- Data types in physics
    - Composed: image-like + 1D sequences
        - Sequences of image-like data
        - Examples:
            - videos, as sequences of images
            - sequences of short pulses



Source: M. Erdmann et al., Deep Learning for Physics Research

Models for data analysis

- Data types in physics
  - Point cloud data (3D, spacetime)
    - 3D form/object described as a set of data points in space
    - Each point is related to a vector for locating its position
    - The collection of such vectors form the **point cloud**
    - Examples:
      a. Temperature measurements in 3D space
      b. Data for 3D scanners



Source: X. Shen, Construction of three-dimensional temperature distribution using a network of ultrasonic transducers, Scientific Reports 9, 12726 (2019)

Models for data analysis

- Data types in physics
  - Heterogeneous data
    - Sometimes, a measurement is the result of information captured by an *ensemble of devices*
    - Examples:
      a. In seismology, distributed sensors are used for the detection of earthquake epicenter
      b. In particle physics, detectors such as ATLAS and CMS at CERN consist of millions of sensors, which comprise different technologies and all together form a highly heterogeneous measurement system

Models for data analysis

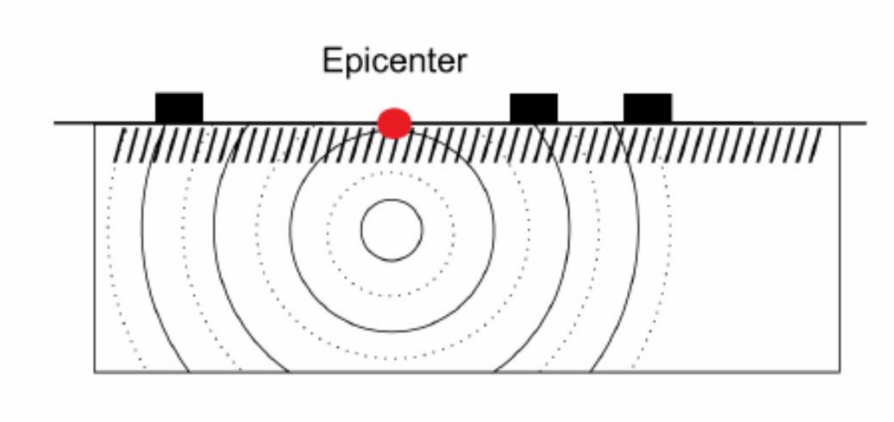
- Model building

    - Scientific question themselves determine which type of information and from how many sensors should be used

    - Intuitive understanding of physics, generates expectations about the aspects of measurements that can be utilized for solving a task

Models for data analysis

- Model building: locating earthquake epicenter



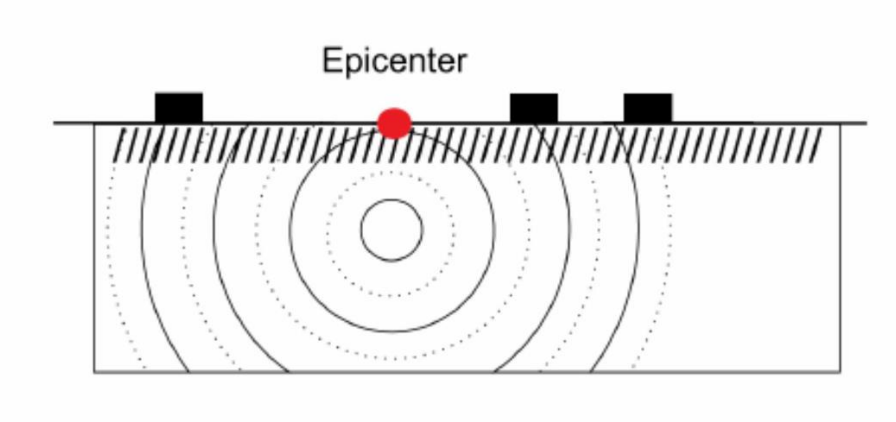
- Physicist's approach:
  - Sheer and compression waves – known velocities
  - Measuring signal times at each station
  - Determine location of epicenters

$$t = x / v_s$$

$$t = x / v_p$$

Models for data analysis

- Model building: locating earthquake epicenter



- Data-driven approach:
  - Network becomes the model – relationship is learned from training data
  - No need to specify positions of stations or velocities
  - Input: arrival times $t_{i,p}$ $t_{i,s}$
  - Output: longitude, latitude of epicenter (l, b)
  - Training: iterative process of comparing with the true values of (l, b)

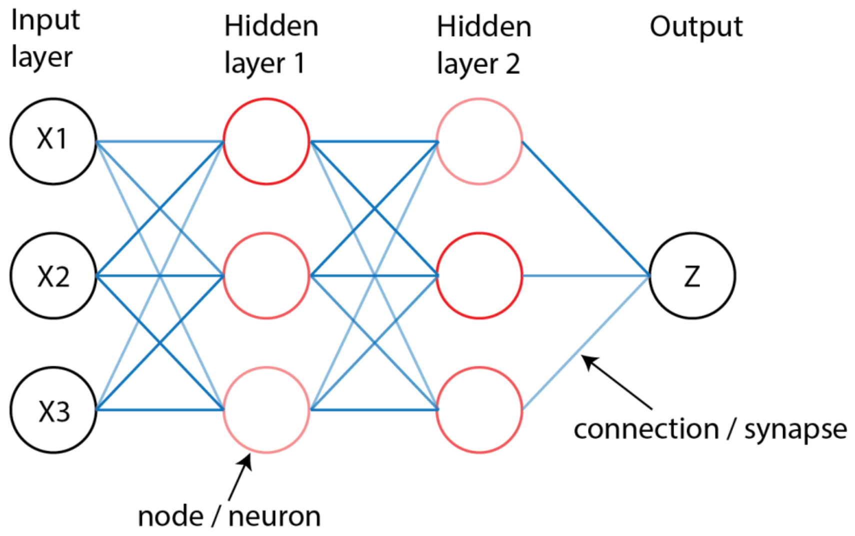
Models for data analysis

- Data-driven model optimization (network training)

  - Network parameters: weights of connections

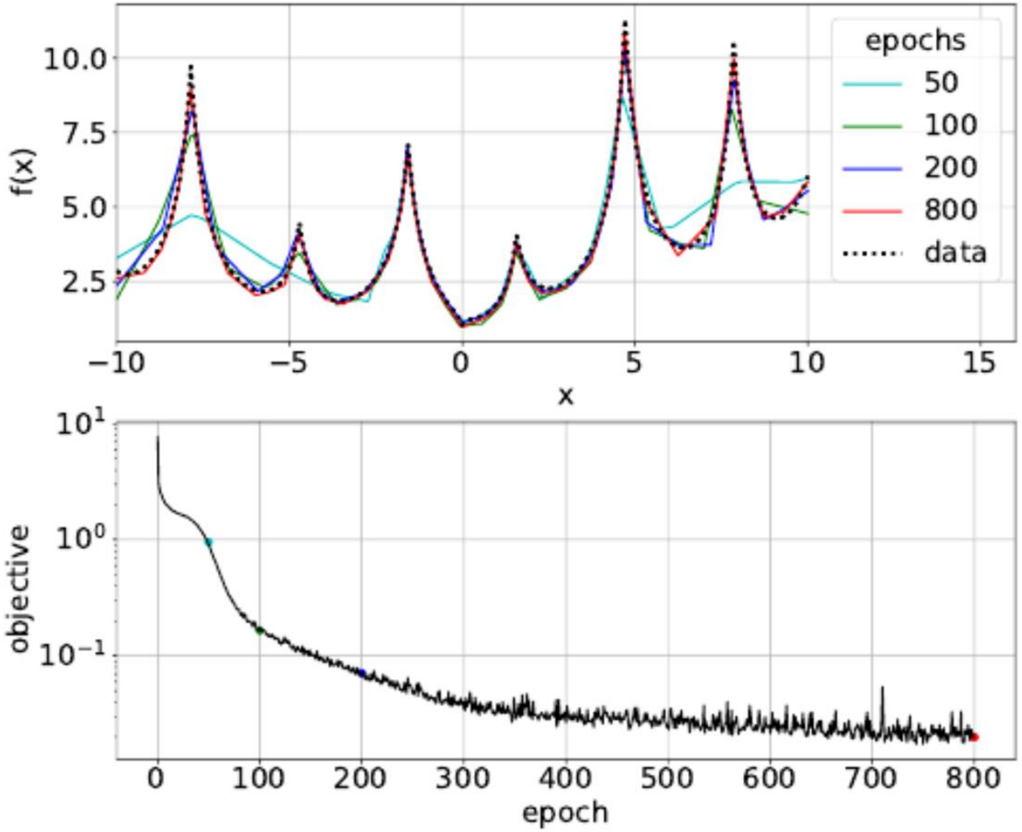
  - Training data set:
    - input values, vector $\vec{x}$
    - Target values y, labels

  - Objective function $\mathcal{L}$:
    - distance between network predictions $f(\vec{x})$ and target y
    - $\mathcal{L}$ depends on scientific question
    - Common way: mean value of squared residuals $\mathcal{L} = \left(\frac{1}{k}\right) \Sigma_{i=1}^{k} \Delta^2$
    - with residuals $\Delta = |f(\vec{x}) - y|$
    - i runs over all data points



Models for data analysis

- Data-driven model optimization (network training)



Source: M. Erdmann et al., Deep Learning for Physics Research

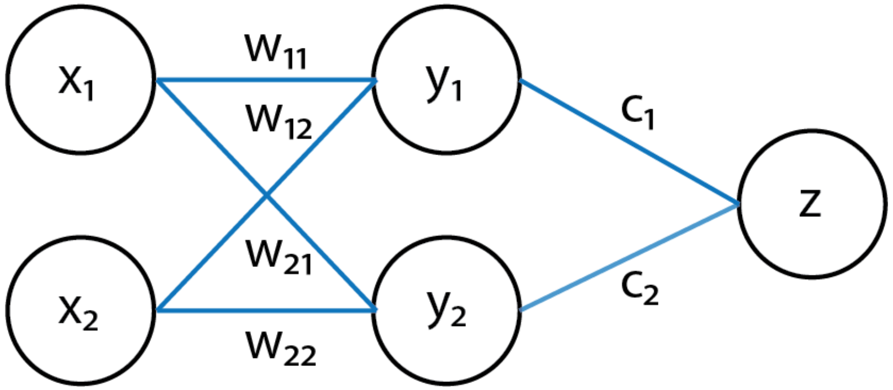
Building blocks of neural networks

Building blocks of neural networks

- Linear mapping and displacement

$$\vec{y} = \boldsymbol{W}\vec{x} + \vec{b}$$

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



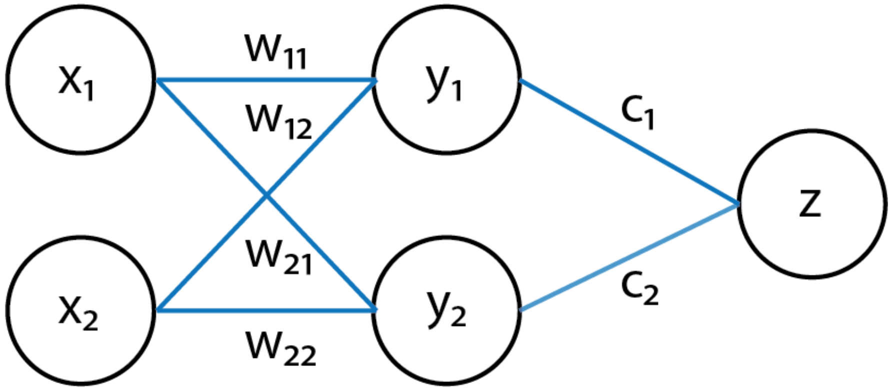
- Affine transformation
  - *Matrix* ***W***: weights
  - Vector *b:* bias

Building blocks of neural networks

- Linear mapping and displacement

$$\vec{y} = \boldsymbol{W}\vec{x} + \vec{b}$$

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



- Another affine transformation

$$z = \boldsymbol{W}'\vec{y} + d = \begin{pmatrix} c_1 & c_2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} + d$$

$$= W_1''x_1 + W_2''x_2 + b''$$

$$W_1'' = c_1W_{11} + c_2W_{21}$$

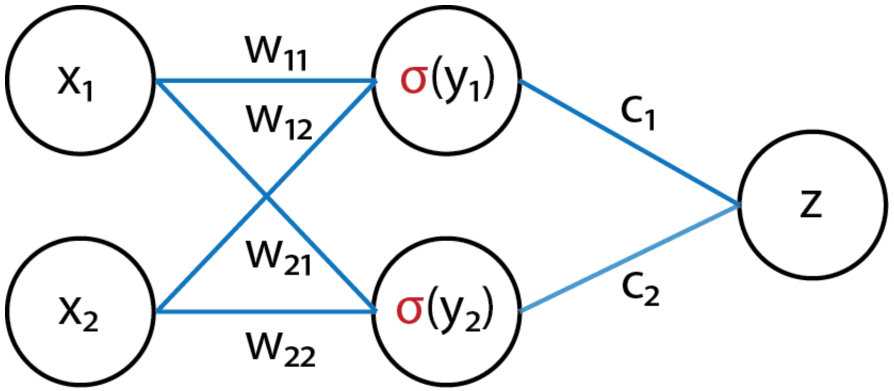
$$W_2'' = c_1W_{12} + c_2W_{22}$$

$$b'' = c_1b_1 + c_2b_2 + d$$

Building blocks of neural networks

- Nonlinear mapping: **activation function**
  - Nonlinearity in the neurons

$$\sigma(\vec{y}) = \begin{pmatrix} \sigma(y_1) \\ \sigma(y_2) \end{pmatrix}$$



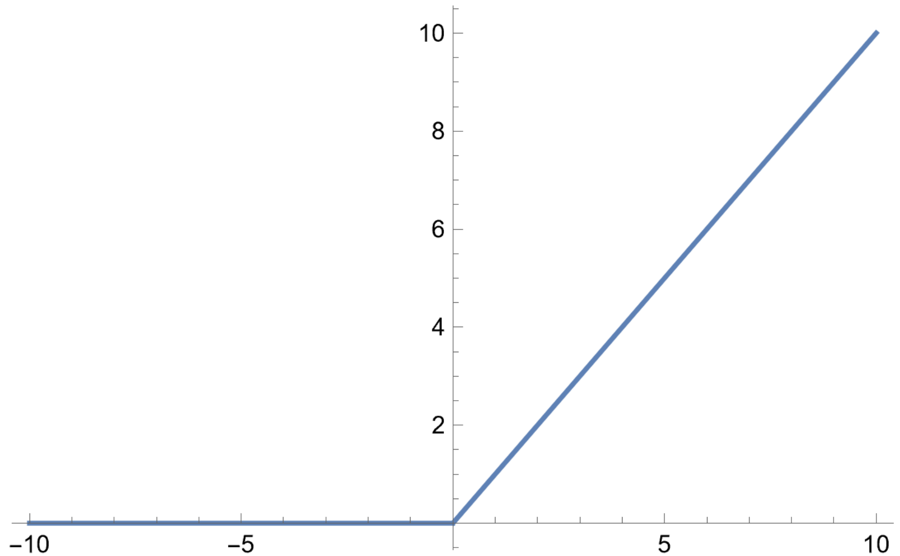
$$z = \boldsymbol{W}' \, \sigma(\vec{y}) + d = \begin{pmatrix} c_1 & c_2 \end{pmatrix} \begin{pmatrix} \sigma(y_1) \\ \sigma(y_2) \end{pmatrix} + d$$

$$= c_1 \, \sigma(W_{11}x_1 + W_{12}x_2 + b_1) + c_2 \, \sigma(W_{21}x_1 + W_{22}x_2 + b2) + d$$

Building blocks of neural networks

- Nonlinear mapping: **activation function**
  - Types of activation functions
    - ➢ Rectified Linear Unit (ReLU)

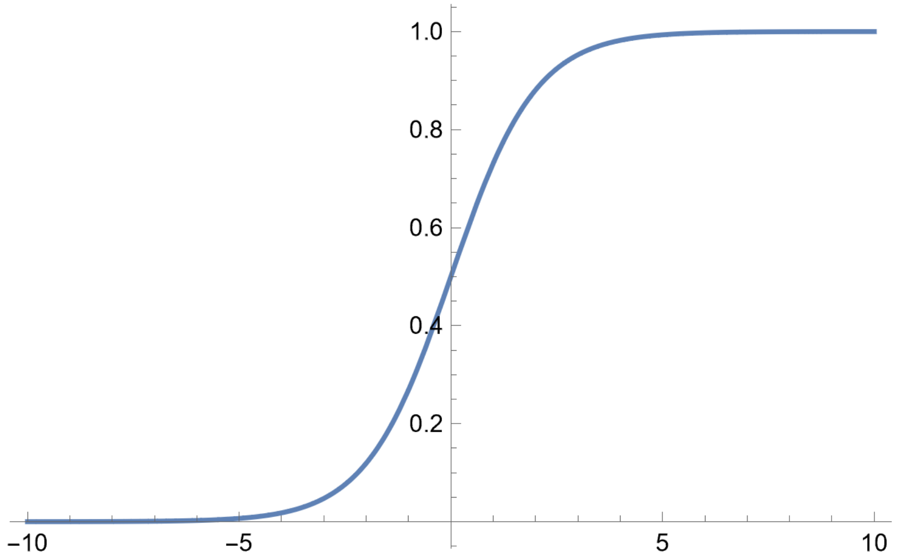
$$\sigma(x) = \max(0, x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$



Building blocks of neural networks

- Nonlinear mapping: **activation function**
  - Types of activation functions
    - ➢ Sigmoid (logistic) function

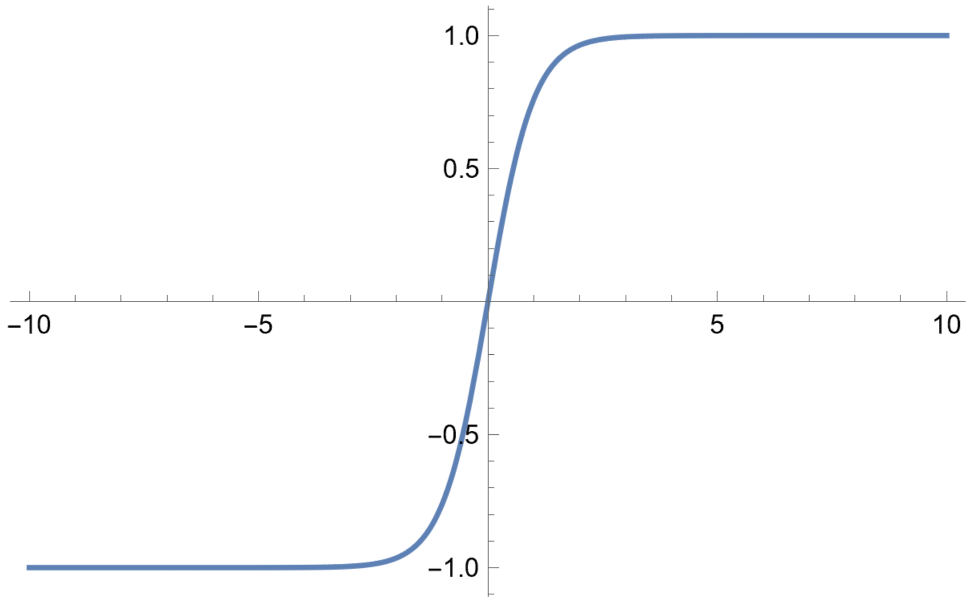
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



Building blocks of neural networks

- Nonlinear mapping: **activation function**
  - Types of activation functions
    - Hyperbolic tangent

$$\sigma(x) = \tanh(x)$$



Building blocks of neural networks

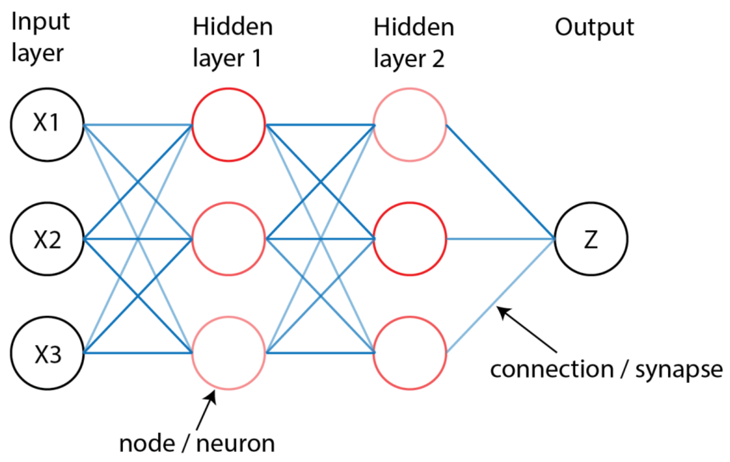
- Network predictions

  - At each node, two transformations
    1. Affine mapping:

       $$\vec{y} = \boldsymbol{W}\vec{x} + \vec{b}$$

    1. Nonlinear activation function:

       $$\vec{z} = \sigma(\boldsymbol{W}\vec{x} + \vec{b})$$

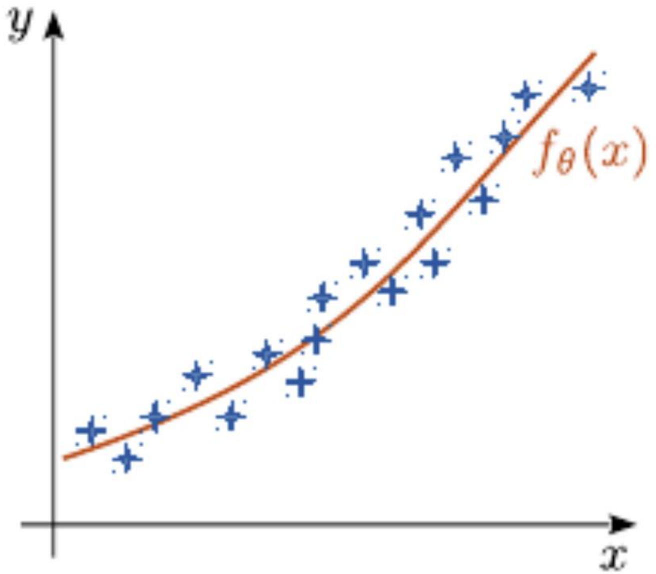


  - Networks have many layers with many nodes
    - In each layer, information from previous layer is used as input
    - This is known as the **forward pass** in a network

Building blocks of neural networks

- Network predictions
  - Regression

$$z = f(x), \qquad \{x, z\} \in R$$

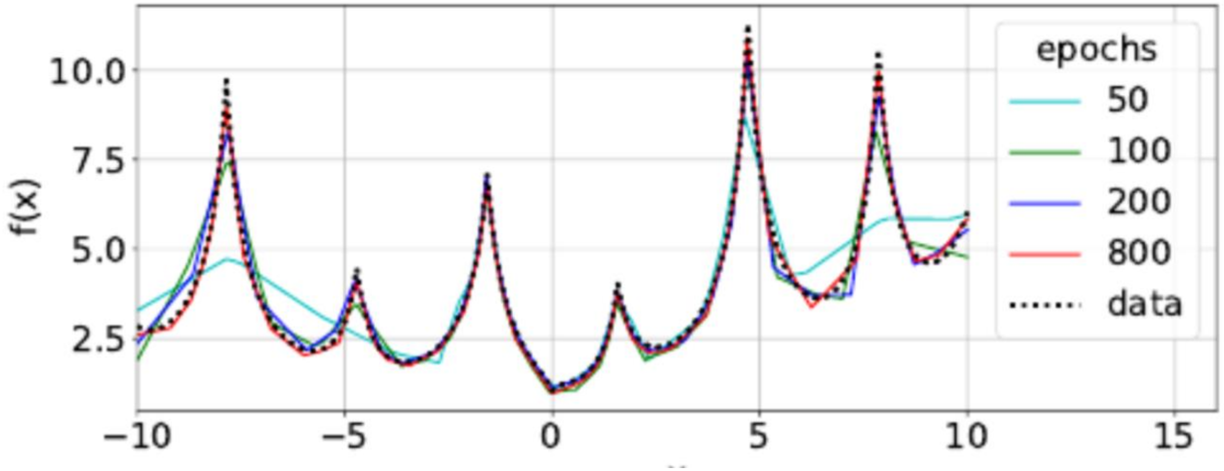


Building blocks of neural networks

- Network predictions
  - Regression

$$z = f(x), \qquad \{x, z\} \in R$$

  - Example: function interpolation



Building blocks of neural networks

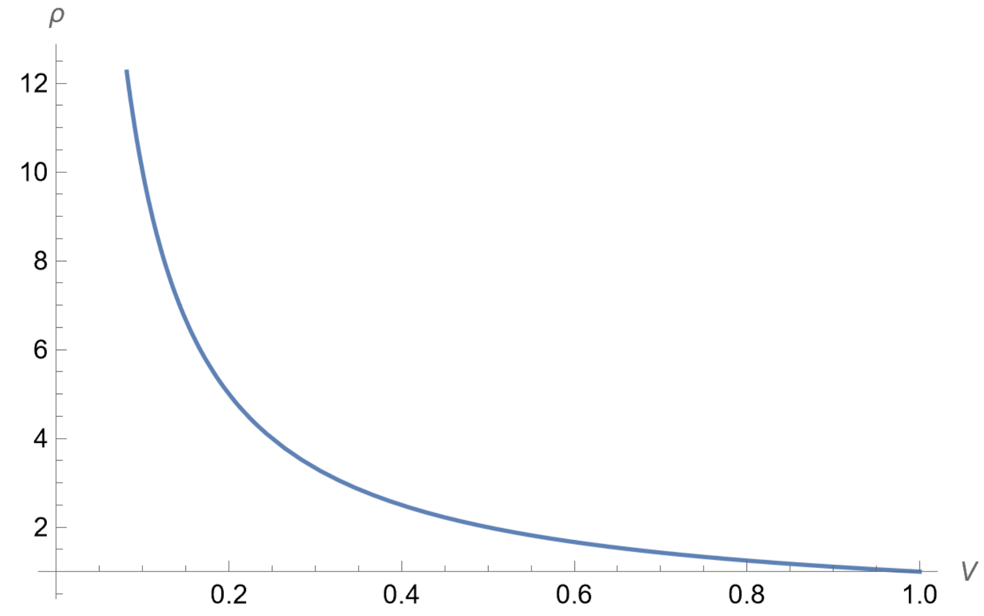
- Network predictions

  - Regression

    $$z = f(x), \qquad \{x, z\} \in R$$

  - Example: mass density

    $$\rho = \frac{m}{V}$$

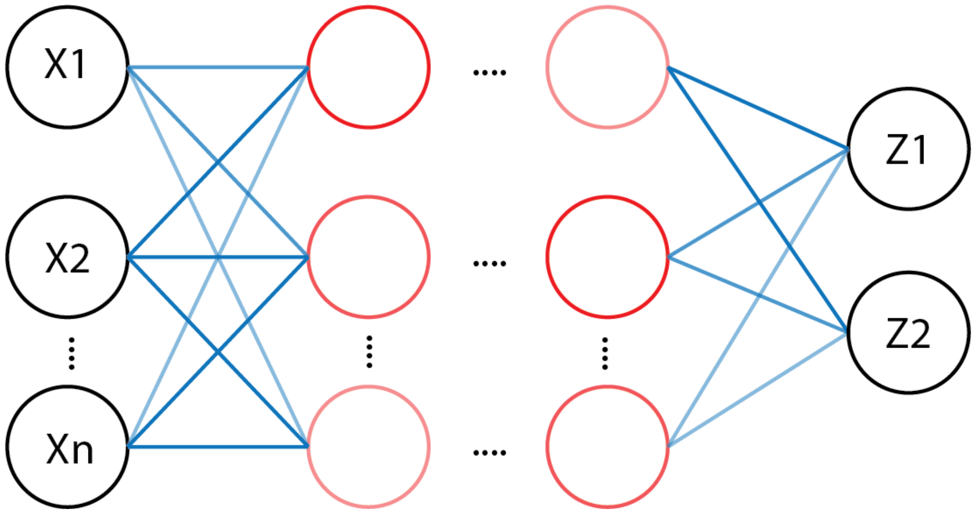


Building blocks of neural networks

- Network predictions
  - Regression

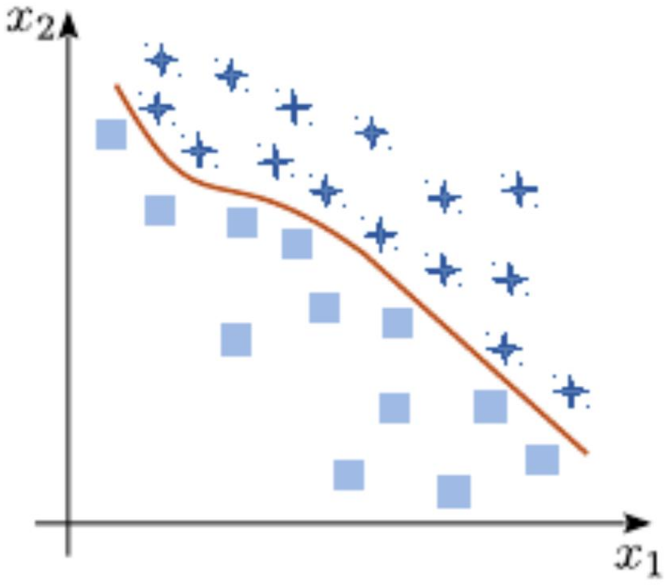
    $\vec{z} = f(\vec{x}), \qquad \{x_i, z_i\} \in R$

  - Example: earthquake epicenter
    - Inputs: time signals from each station
    - Two outputs:
      - ➢ Latitude of epicenter
      - ➢ Longitude of epicenter



Building blocks of neural networks

- Network predictions
  - Classification:
    - Separate data into two or more categories
    - Required: separation line in the feature space

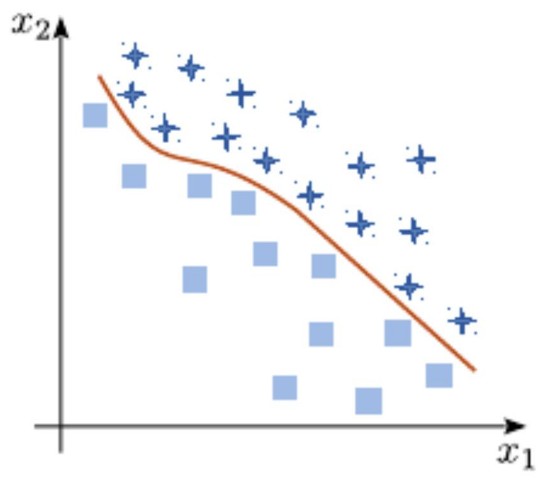


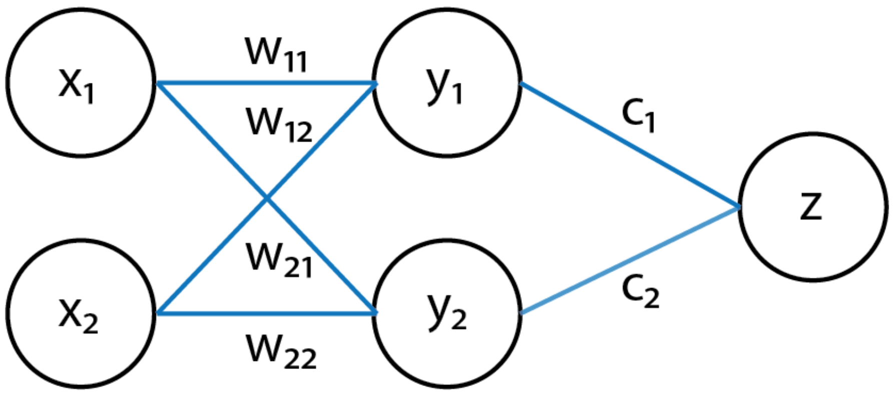
Building blocks of neural networks

- Network predictions
  - Classification:
    - Separate data into two or more categories
    - Required: separation line in the feature space
  - Example:
    - separate signal from background noise
    - Trick: output is subject to a sigmoid function

$$z' = \sigma(z), \qquad \sigma(z) = \frac{1}{1+e^{-z}}$$

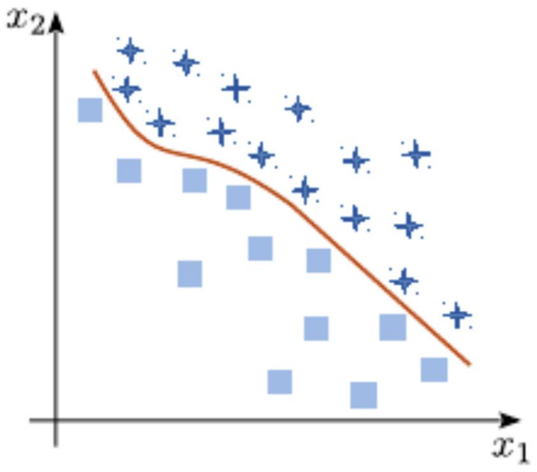
$$z' \epsilon [0,1]$$





Building blocks of neural networks

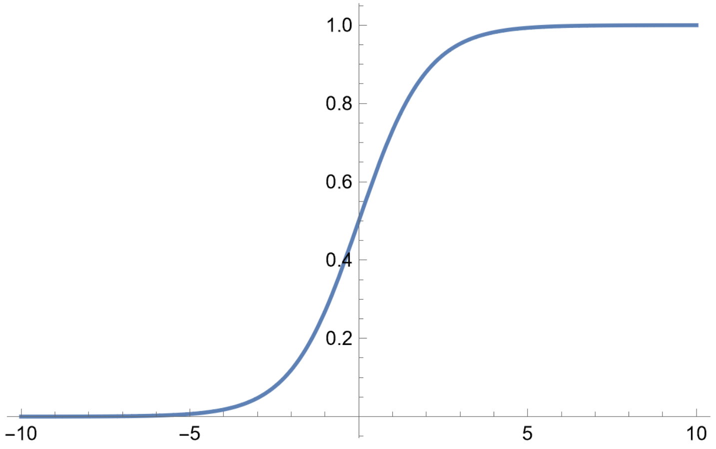
- Network predictions
  - Classification:
    - Separate data into two or more categories
    - Required: separation line in the feature space



  - Example:
    - separate signal from background noise
    - Trick: output is subject to a sigmoid function

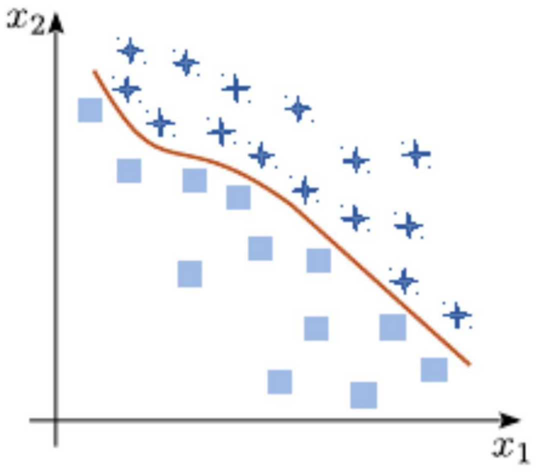
$$z' = \sigma(z), \qquad \sigma(z) = \frac{1}{1+e^{-z}}$$

$$z' \epsilon [0,1]$$



Building blocks of neural networks

- Network predictions
  - Classification:
    - Separate data into two or more categories
    - Required: separation line in the feature space
  - Multi-class classification
    - Network can be extended to multi-variable outputs
    - Example: classification of images into categories such as animals, vehicles, buildings



Building blocks of neural networks

- Universal approximation theorem

  - Question: can a neural network represent any function/mapping in physics?

  - *"A feed-forward network with linear output and at least one hidden layer with a finite number of nodes can approximate any of the above functions to arbitrary precision."*

    - ➢ Theorem developed for continuous functions on closed and bounded subsets of the Euclidean space
    - ➢ Which includes all typical functions relevant to physics!

K. Hornik et al., Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989)
G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control, Signals and Systems (1989)

Building blocks of neural networks

- Universal approximation theorem
  - Question: can a neural network represent any function/mapping in physics?
  - *“A feed-forward network with linear output and at least one hidden layer with a finite number of nodes can approximate any of the above functions to arbitrary precision.”*
    - ➢ The theorem has been confirmed for multilayer networks with a limited number of nodes and various activation functions
    - ➢ Learnability of the function remains an open question

K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Networks 4 (1991)
B. Hanin et al., Approximating continuous functions by ReLU Nets of minimal width, Arxiv (2017)
Z. Lu et al., The expressive power ofneural networks: A view from the width, in I. Guyon et al., Advances in Neural Information Processing Systems (2017)

Summary

- Scientific questions can be answered from data using ML algorithms

- ML algorithms are trained using data by minimizing an objective function

- Physicists' tasks are to develop and improve DNN architectures suitable for solving particular tasks

Summary

- At each node of an DNN, two operations are performed:
  - Linear transformation with displacement (affine mapping)
  - Nonlinear transformation (activation function)

- A DNN can be used for
  - Regression: high-dimensional function approximation
  - Classification: classify objects into *m* categories

- Universal approximation theorem:
  - *"Feed-forward networks with linear output and at least one hidden layer with a finite number of nodes can approximate functions of interest in physics to arbitrary precision"*