Workshop at 12th ISNVH Congress, Graz, Austria

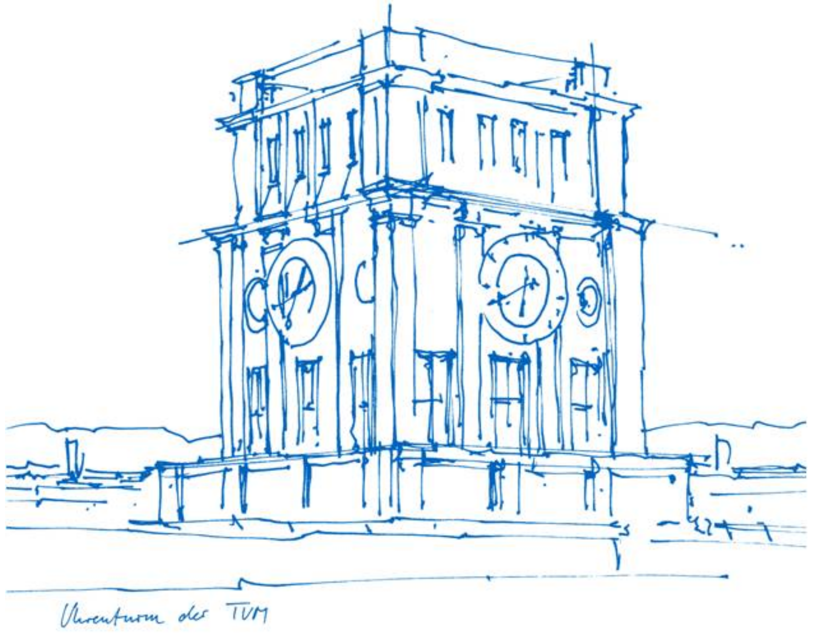
- Data driven methods for NVH -

Marcus Mäder, Johannes Schmid, Steffen Marburg

Technical University of Munich

TUM School of Engineering and Design

Chair of Vibroacoustics of Vehicles and Machines



Content of this workshop

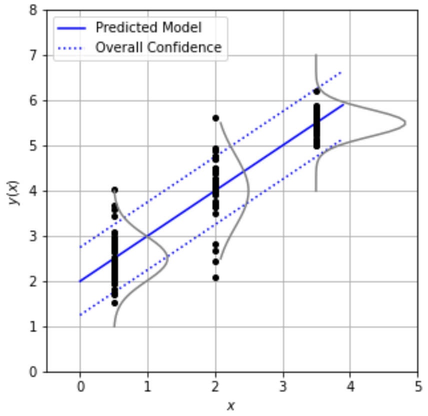
Overview

- Data driven methods need data and methods
- Feeding data in complex algorithms feels similar to learning behavior
- Often referred to Machine Learning or Artificial Intelligence
    - How is this done
    - Why is it useful
    - Can we get smarter

Introduction

Some available learning methods

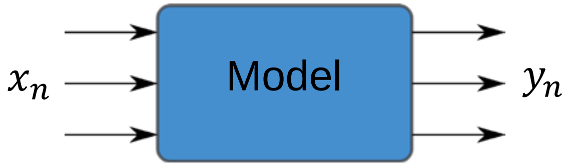
- *Deterministic Learning*:
  An input observation is assigned to deterministic output, e.g. least square methods
- *Probabilistic Learning:*
  An input observation is assigned to an output with its probability
  - *Frequentist Learning:* There exists a true model with fixed parameters
    - Maximum Likelihood Estimation
    - $\theta_i \sim \mathrm{N}\left(m, s_i^2\right)$, with known distribution
    - Mean values are the expected values with $\mathrm{E}[\boldsymbol{y}] = \mathbf{w}^{\mathrm{T}}\mathbf{x}$
    - Predefined distributions of fixed parameters
  - *Bayesian Learning:* There exists a prior model with random parameters



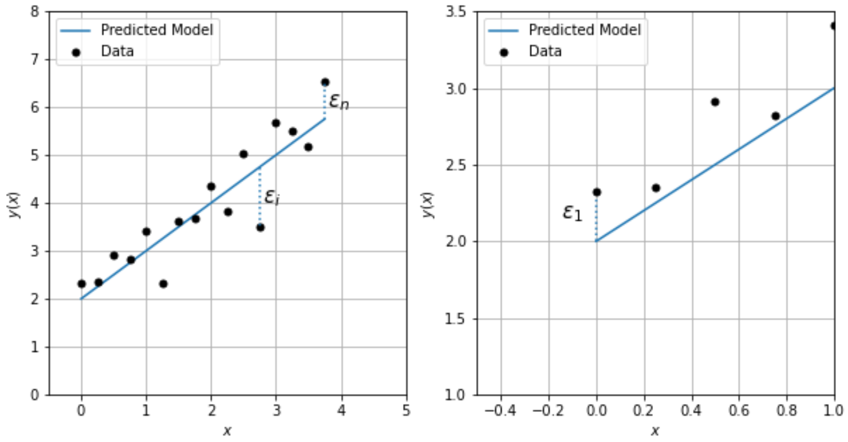
Introduction

From a statistical point of view (linear regression)

- Lets assume a ML algorithm with x_n inputs y_n predictions



$$\boldsymbol{y} = \mathbf{w}^{\mathrm{T}}\mathbf{x} + \boldsymbol{\varepsilon}$$



- $\boldsymbol{y}$: Predictions
- $\mathbf{x}$: Observed input
- $\mathbf{w}$: Coefficients
- $\boldsymbol{\varepsilon} \sim \mathbf{N}(0, \boldsymbol{\Sigma}^2)$: Random noise
- $\boldsymbol{\Sigma}^2 = \{\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2\}$

- Task: Find coefficients $\mathbf{w}$ and $\boldsymbol{\Sigma}^2$ in the frame of a parameter identification

Introduction – A few words to intelligence

The meaning behind

- Intelligence (Latin *intellegere* “understanding”, *inter* “between”, and *legere* “reading, choosing”)
  - Collective term in psychology for human cognitive performance
  - Standardized tests measure underlying cognitive ability
  - No general valid definition
- Machine Learning mimicking how the brain works
  - Helping to understand complex systems
  - Data based for storing experiences
  - Simplifying difficult tasks
  - Supporting with solving problems

Source: https://en.wikipedia.org/wiki/Intelligence, http://www.educationalneuroscience.org.uk

Introduction – Machine Learning

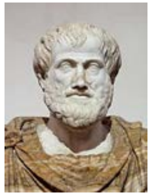
A definition (Tom Mitchell)

- Well-posed Learning Problem: A computer is said to **learn** from experience *E* with respect to some task T and some performance measure *P*, if its performance on *T*, as measured by *P*, improves with experience *E*.
- At the beginning of each ML-project specify
  - Task *T*
  - Experience *E*
  - Performance measure *P*

Source: Mitchell, T. (1997). Machine Learning. McGraw-Hill, New York.

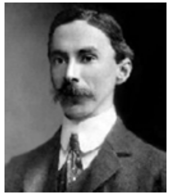
Introduction – How machines started learning

A brief history of AI – TUM lecture FTM

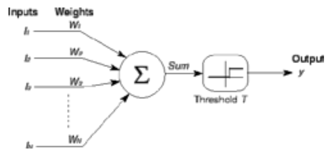

300 BC: Aristoteles – Describing syllogism

Induction
1. Premise
↓
n. Premise
Conclusion

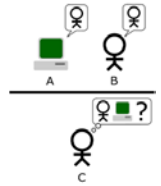
1739: Hume – Empiricism, Induction


1913: Russel – Formal Logic

1931: Gödel – Untestable Logic



1943: McCulloch & Pitts – Basics ANN



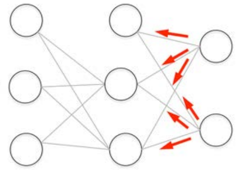
1951: Turing – Machine Intelligence

1956: McCarthy – Artificial Intelligence

```
Welcome to
                  EEEEEE  LL      IIII  ZZZZZZZ   AAAAA
                  EE      LL       II        ZZ  AA   AA
                  EEEEE   LL       II      ZZZ   AAAAAAA
                  EE      LL       II    ZZ      AA   AA
                  EEEEEE  LLLLLL  IIII  ZZZZZZZ  AA   AA

Eliza is a mock Rogerian psychotherapist.
The original program was described by Joseph Weizenbaum in 1966.
This implementation by Norbert Landsteiner 2005.

ELIZA: Is something troubling you ?
YOU:   Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU:   They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU:   Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU:   He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU:   It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:
```
1966: Weizenbaum – NLP Eliza


1986: Hinton – ANN Backprop.

2009: Google – Self Driving Car

2011: IBM Watson – Won Jeopardy

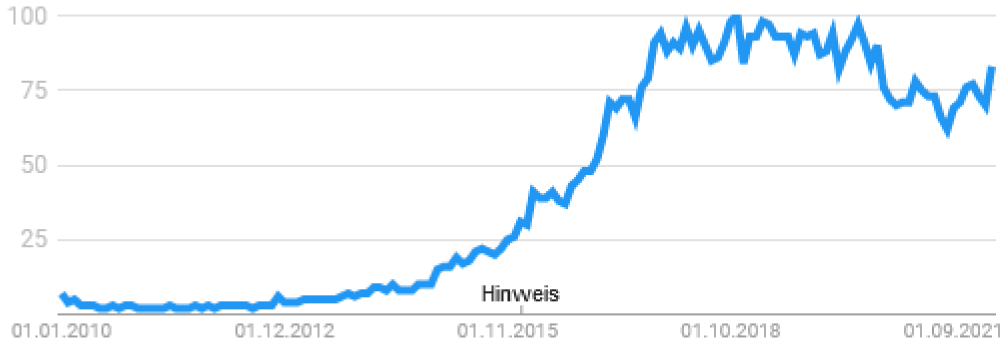
2018: Google Duplex – Personal Assistant

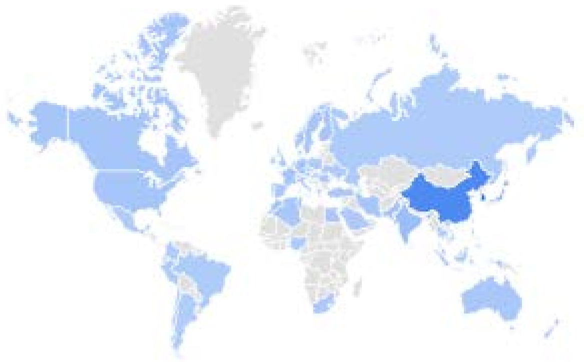
Source: Adopted from TUM lecture: Prof. Lienkamp "Artificial Intelligence in Automotive Technology"

Introduction – How machines started learning

Trends of Deep Learning Search

(1) South Korea 100%, (2) China 57%, (3) Singapore 26%, …,
(9) India 13%, (15) USA 7%, (16) Germany 7%



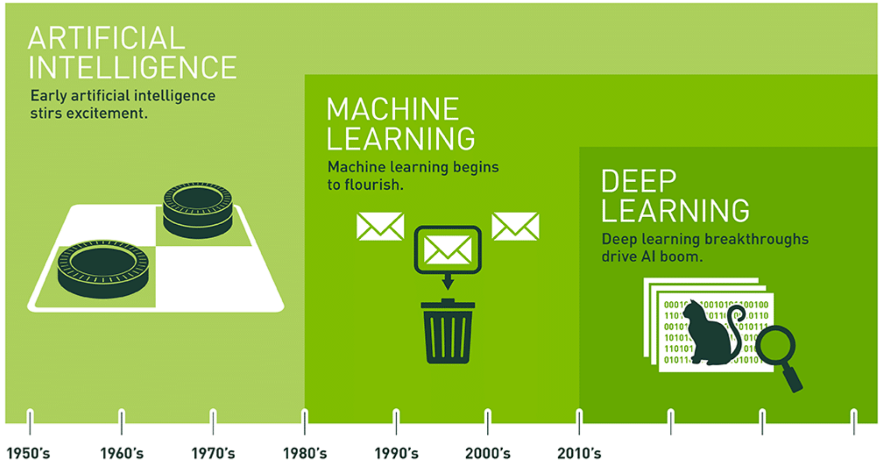


Source: https://trends.google.com: Search ”Deep Learning“, April 2022

Introduction – What are the differences

Difference between AI, ML, and DL

- General AI – Human senses, reasoning etc.
- Narrow AI – Classification tasks (Voice, Image)
- AI development exploded since 2015
  - GPUs used for parallel processing
  - Availability of big data
- ML algorithms use data for training
  - Clustering, Computer Vision
  - Reinforced Learning, Bayesian Networks
- DL networks inspired by human brain
  - Neurons and weights give probability vector


ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's

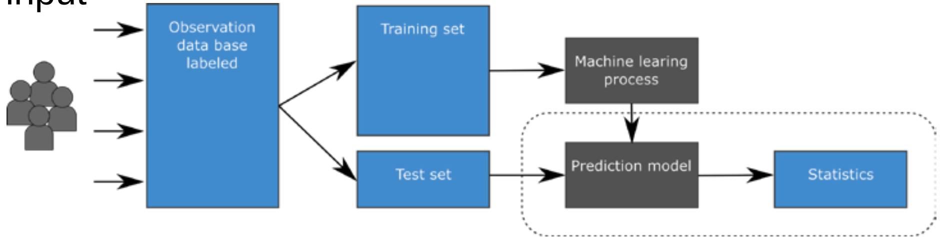
Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Source: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

Introduction – What are the differences

Supervised and unsupervised learning

- Supervised Learning
  - Full set of labeled data is available when training the algorithm
  - Algorithms' task is to predict the given label after training
  - Useful for classification and regression problems
    - Classification: Predict discrete value w.r.t. input data
    - Regression: Predict outcome of continuous input
  - Best suited when ground truth exists or set of reference points



Source: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

Introduction – What are the differences

Supervised and unsupervised learning

- Unsupervised learning
  - Data set that is given to the algorithm is unlabeled
  - Algorithm extracts common properties (features)
  - Organization of data

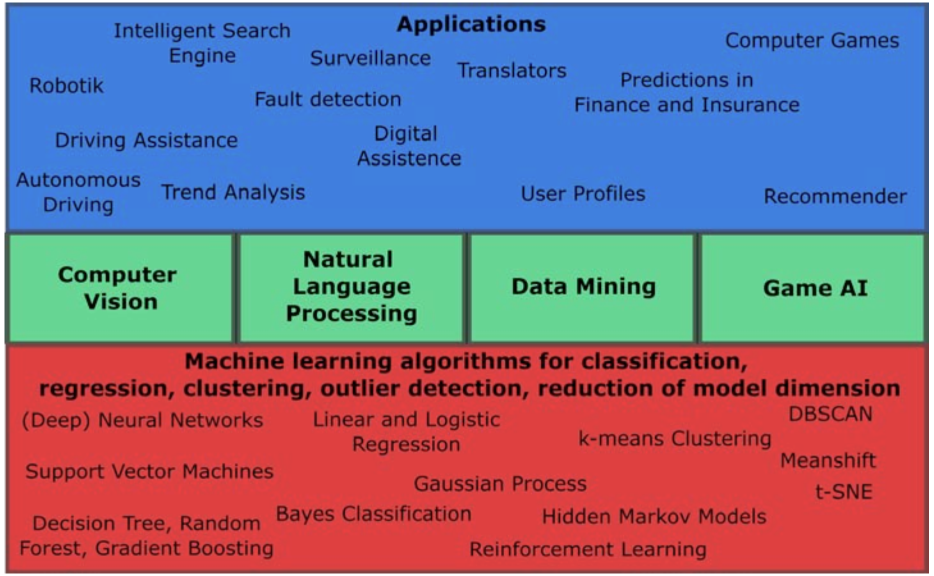
• Clustering:	Group training data w.r.t. similarities
• Anomaly detection:	Find outliers in dataset
• Association:	Find common features of data samples (Product recommendation)
• Autoencoders:	De-noising of data, e.g. pictures, video, and audio

Source: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

Introduction – What can I do with it

Algorithms and Applications

- Variety of algorithms and applications
- Applications in mechanical engineering
  - Material parameter estimation
  - Fault detection
  - Structural health monitoring
  - Digital twins
  - Robotics
  - Geometry optimization
  - Visualization

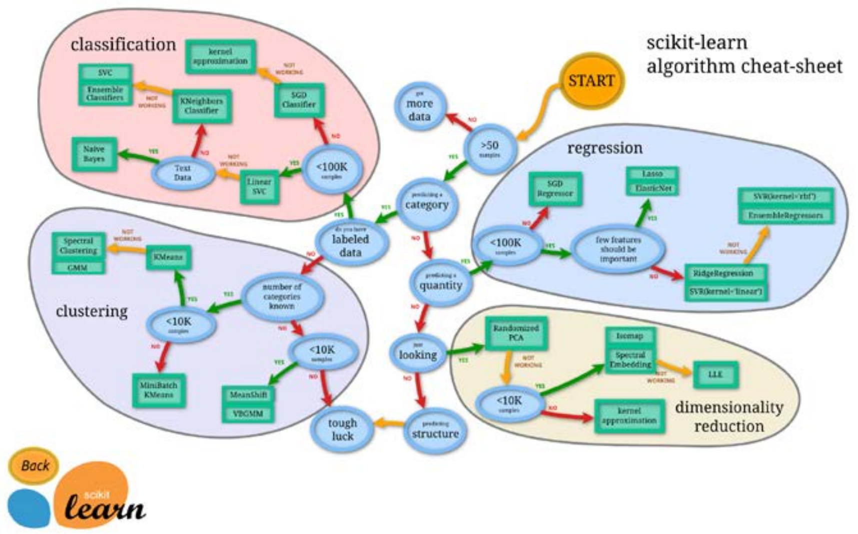


Source: Lecture notes Prof. J. Maucher, “Machine Learning”, HDM-Stuttgart

Introduction – How to navigate

Algorithm selection of machine learning

- Understand the problem or task
- Specify
  - Task
  - Experience
  - Performance Measure
- Use predefined structures and algorithms
- Use the internet
  - Many problems have been solved already
  - Understand structures
  - Apply them to specific needs

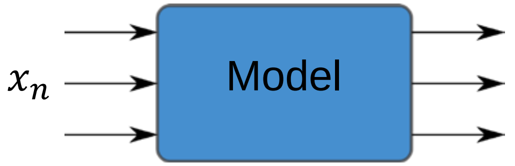


Source: https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

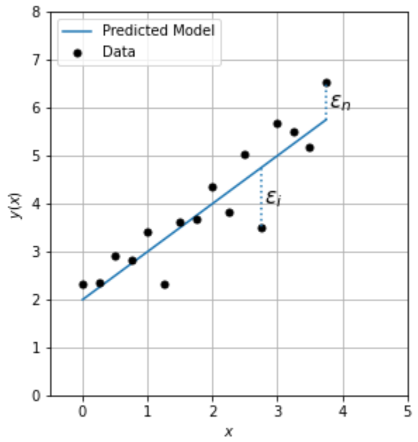
First step – The learning problem

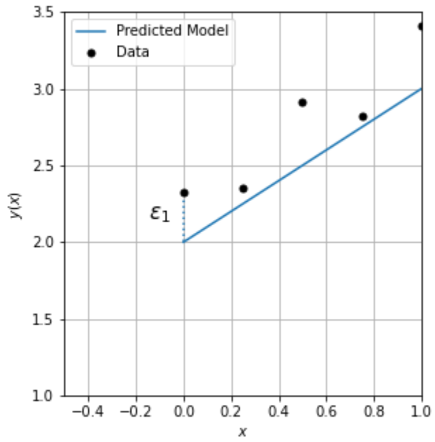
Linear regression – how to

- Lets assume a ML algorithm with x_n inputs y_n predictions



$$\boldsymbol{y} = f(\mathbf{w}^{\mathrm{T}}\mathbf{x} + \boldsymbol{\varepsilon})$$





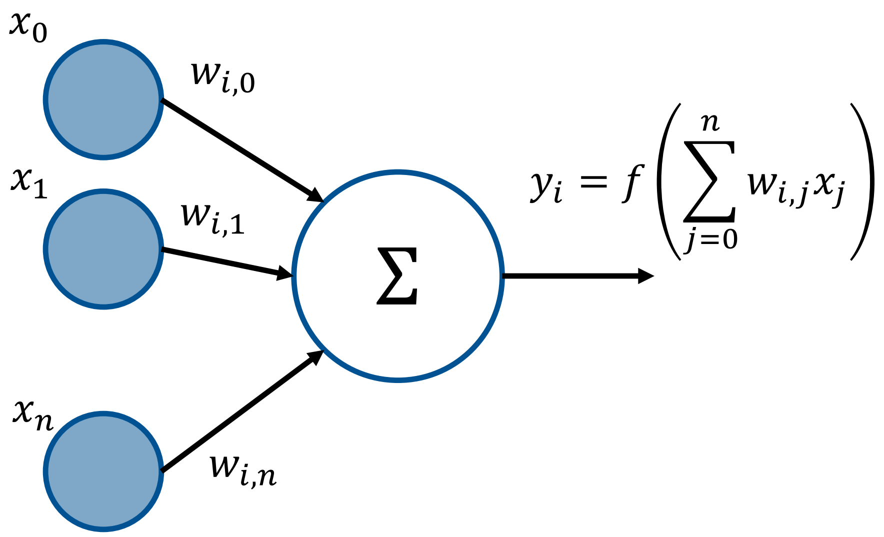
- $\boldsymbol{y}$: Predictions
- $\mathbf{x}$: Observed input
- $\mathbf{w}$: Coefficients/weights
- $\boldsymbol{\varepsilon} \sim \mathbf{N}(0, \boldsymbol{\Sigma}^2)$: Random noise
- $f(z)$: Linear or non-linear function

- Task: Find coefficients $\mathbf{w}$ for a suitable model

Mathematical model of a single neuron

Mimicking the natural behavior of a neuron

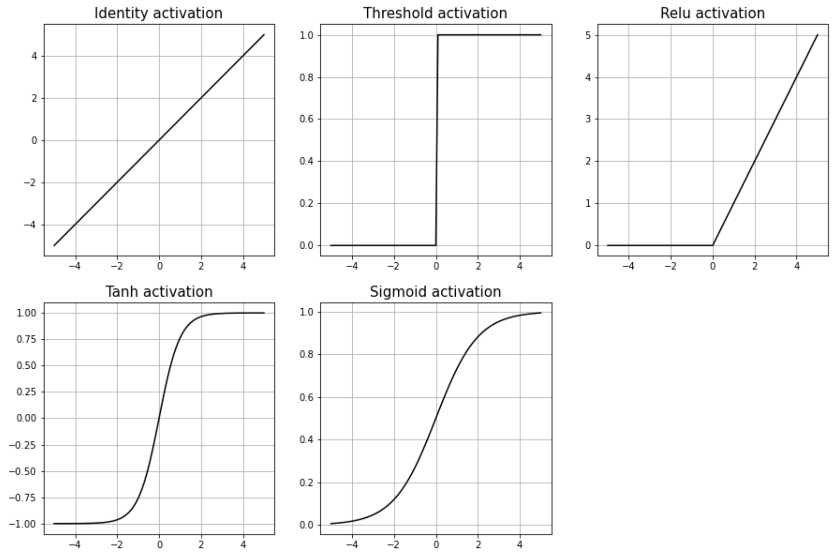
- Weights $w_{i,j}$ represent connection strengths between neuron j in previous layer and neuron i in current layer.
- The activation function $f(z)$ transforms the weighted sum of $z = \sum_{j=0}^{n} w_{i,j} x_j$ of the input of the neuron into an output value.
- Activation function is chosen depending on the task (classification, regression), the learning algorithm, and the network topology.



Mathematical model of a single neuron

Different activation functions

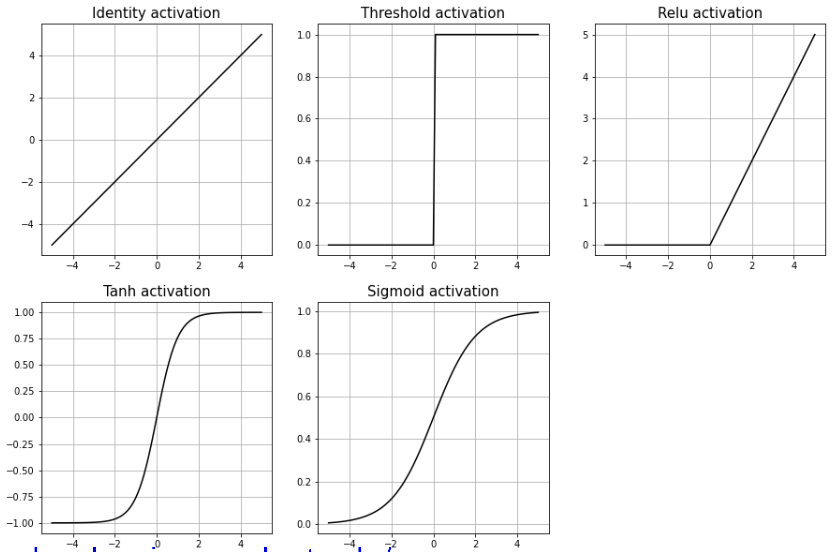
- Identity $f(z) = z$
  - Regression problems
- Threshold $f(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}$
  - Not continuous differentiable
- ReLU $f(z) = \max(0, z)$
  - Exit of convolution layer in convolutional neural network
  - Not continuous differentiable
    - Set value of derivative at $z = 0$, e.g. $(0,0.5,1)$
    - Softplus function $f(z) = \ln(1 + e^{z})$



Mathematical model of a single neuron

Different activation functions

- Tanh $f(z) = \tanh z$
  - Nowadays preferred over sigmoid
  - Results in better training and better prediction
- Sigmoid $f(z) = \frac{1}{1+e^{-cz}}$
  - Classification problems $K = 2$
- Both functions saturate and are only sensitive around $z = 0$
- Very deep networks suffer from vanishing gradient problem (then use ReLU)

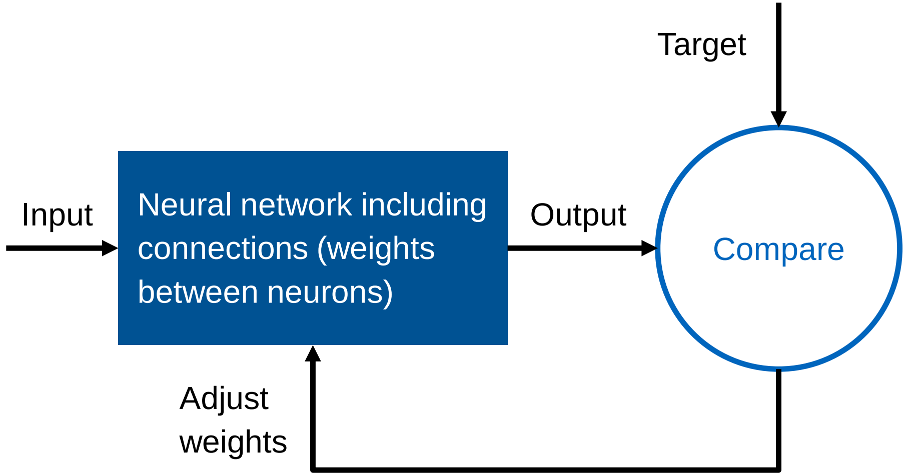


Source: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

Mathematical model of a single neuron

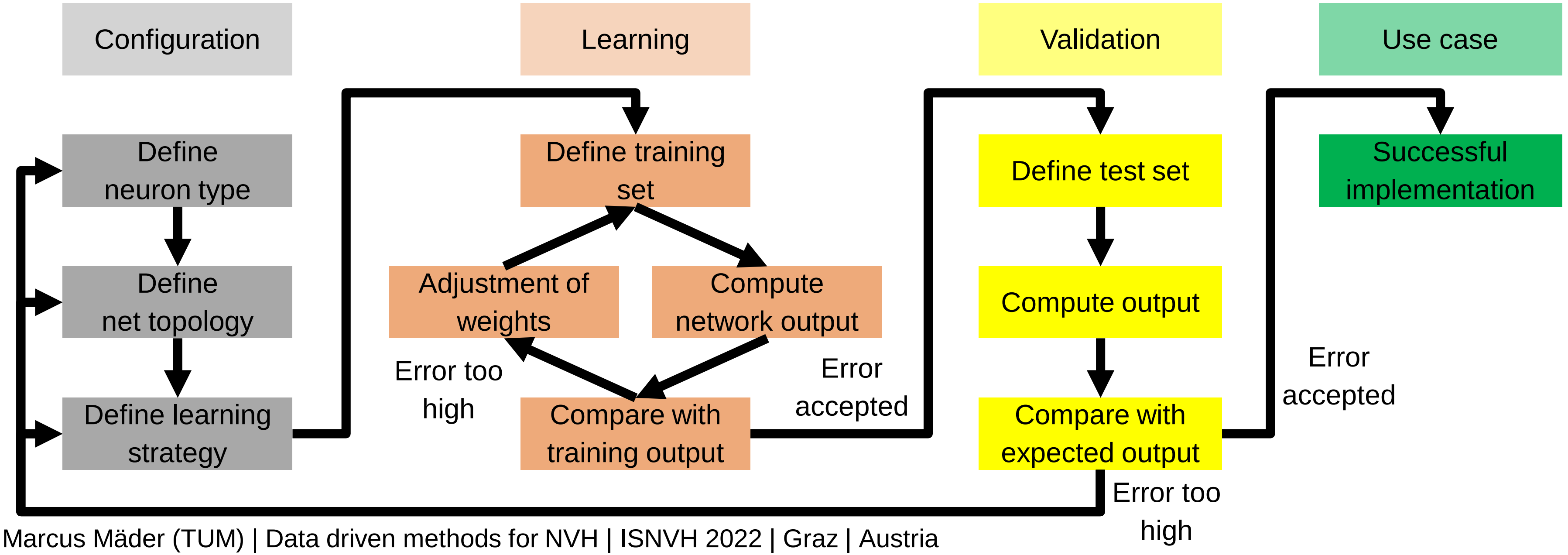
Principle of artificial neural network

- In each iteration, the result (output) is compared with performance standard (target).
- Depending on this comparison, the weights of the connections in the neural network are adjusted.
- This adaptation represents the learning


Target
Input
Neural network including connections (weights between neurons)
Output
Compare
Adjust weights

Neural network design

The different phases



Single Perceptron

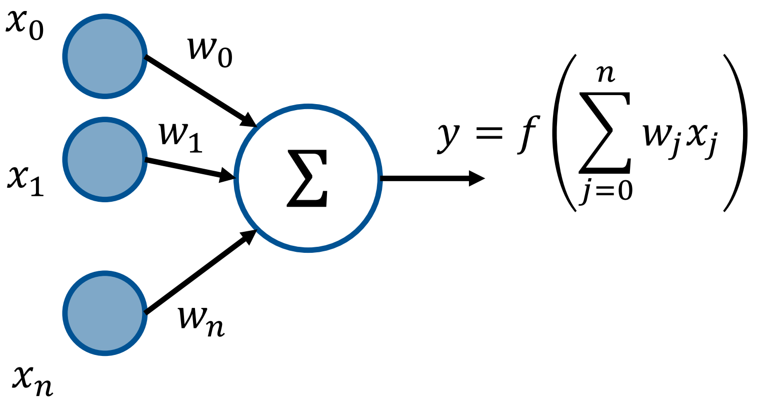
Definition

- **Single Layer Perceptron:** A neural feedforward network in which there are only input and output neurons and all neurons of the input layer are fully connected to the neurons of the output layer is called a single layer perceptron (SLP).
  - For Boolean classifiers ($K = 2$) usually one output neuron is enough
  - For ($K > 2$) classification, K output neurons are usually used
  - For regression with scalar function value, one output neuron is enough.

Single Perceptron

Perceptron with single output

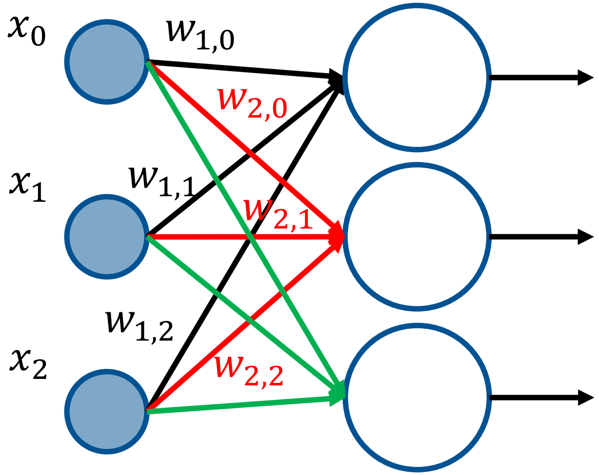
- Feature vector $(x_1, \dots, x_n)$
- Bias is usually constant $x_0 = 0$
- Weighted sum at the entrance of the perceptron
  - $\sum_{j=0}^{n} w_j x_j = \boldsymbol{w}^{\mathrm{T}} \cdot \boldsymbol{x}$
- Bias $w_0 x_0$ often written as b
  - $y = f\left(\sum_{j=1}^{n} w_j x_j + b\right)$
- In classification where there are two outputs the threshold activation or sigmoid activation function can be used



Single Perceptron

SLP for classification within more than two classes

- The single-layer perceptron contains exactly K neurons in the output layer
- For each neuron there is one row in the weight matrix
- $\mathbf{w}^{\mathrm{T}} = \begin{pmatrix} w_{1,0} & w_{1,1} & w_{1,2} \\ w_{2,0} & w_{2,1} & w_{2,2} \\ w_{3,0} & w_{3,1} & w_{3,2} \end{pmatrix}$



$$y_1 = f\left(\sum_{j=0}^{n} w_{1,j} x_j\right)$$

$$y_2 = f\left(\sum_{j=0}^{n} w_{2,j} x_j\right)$$

$$y_3 = f\left(\sum_{j=0}^{n} w_{3,j} x_j\right)$$

Single Perceptron

SLP for classification within more than two classes

- A simple example for three classes $(K = 3)$ and two features $(n = 2)$
- $\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = f\left(\begin{pmatrix} w_{1,0} & w_{1,1} & w_{1,2} \\ w_{2,0} & w_{2,1} & w_{2,2} \\ w_{3,0} & w_{3,1} & w_{3,2} \end{pmatrix}\begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}\right) = \boldsymbol{y} = f\left(\mathbf{w}^{\mathrm{T}} \cdot \boldsymbol{x}\right)$
- $f()$ is the activation function
- For $K = 2, y$ is a scalar; for $K > 2, y$ is a K- dimensional vector
- Each matrix element $w_{i,j}$ is the weight between the Input neuron j and the output neuron i
- Alternatively, $\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = f\left(\begin{pmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}\right) = \boldsymbol{y} = f\left(\mathbf{w}^{\mathrm{T}} \cdot \boldsymbol{x} + \boldsymbol{b}\right)$

Single Perceptron

SLP for classification within more than two classes

- So far, we considered single layer perceptron
  - Contains only input and output layer
  - Input and output neurons are completely connected with each other
  - Suitable for
    - Regression of linear functions
    - Classification with linear discriminants

Multilayer Perceptron

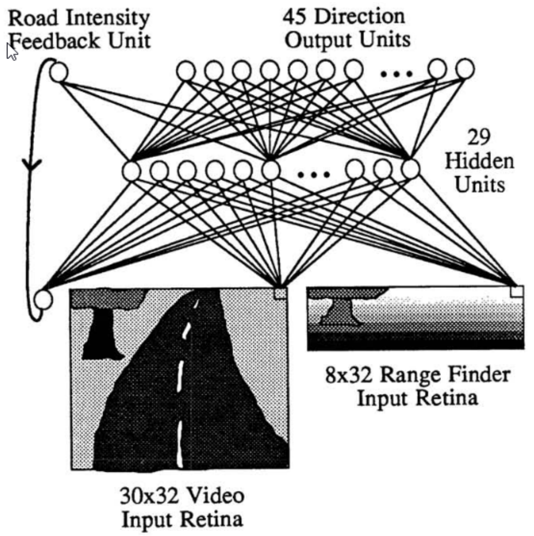
Comparing MLP with SLP

- Now, we consider multilayer perceptron
  - Contains hidden layers between the input and output layers
  - The outputs of each layer are connected to inputs of all neurons of the respective next layer
  - Suitable for
    - Regression of all continuous functions with only one hidden layer
    - Regression also of non-continuous functions with at least two hidden layers
    - Classification with non-linear discriminants
  - Until 2007: In practice, one rarely uses more than one hidden layer, because the analysis of a network with many hidden layers turns out to be quite complicated

Multilayer Perceptron

Applications

- D. Pomerleau, ALVINN: An Autonomous Land Vehicle In a Neural Network (1988), DOI:10.1184/R1/6603146.V1
- An input neuron is assigned to each of the camera's 960 pixels
- Training by driving with real driver
- Successful driving with up to 70 mph over a distance of 90 miles on highway


Road Intensity Feedback Unit
45 Direction Output Units
29 Hidden Units
8x32 Range Finder Input Retina
30x32 Video Input Retina

Source: https://proceedings.neurips.cc/paper/1988/file/812b4ba287f5ee0bc9d43bbf5bbe87fb-Paper.pdf

Multilayer Perceptron

Applications

- Identification of handwriting using MNIST database
  https://python-course.eu/machine-learning/
- Calibration of Nonlinear Dynamic Models of Structures
  https://doi.org/10.3389/fbuil.2022.873546
- Machine learning for metabolic engineering
  https://www.sciencedirect.com/science/article/pii/S109671762030166X
- Machine learning for structural engineering
  https://www.sciencedirect.com/science/article/pii/S2352012422000947
- Data-Driven Aerospace Engineering
  https://arc.aiaa.org/doi/full/10.2514/1.J060131

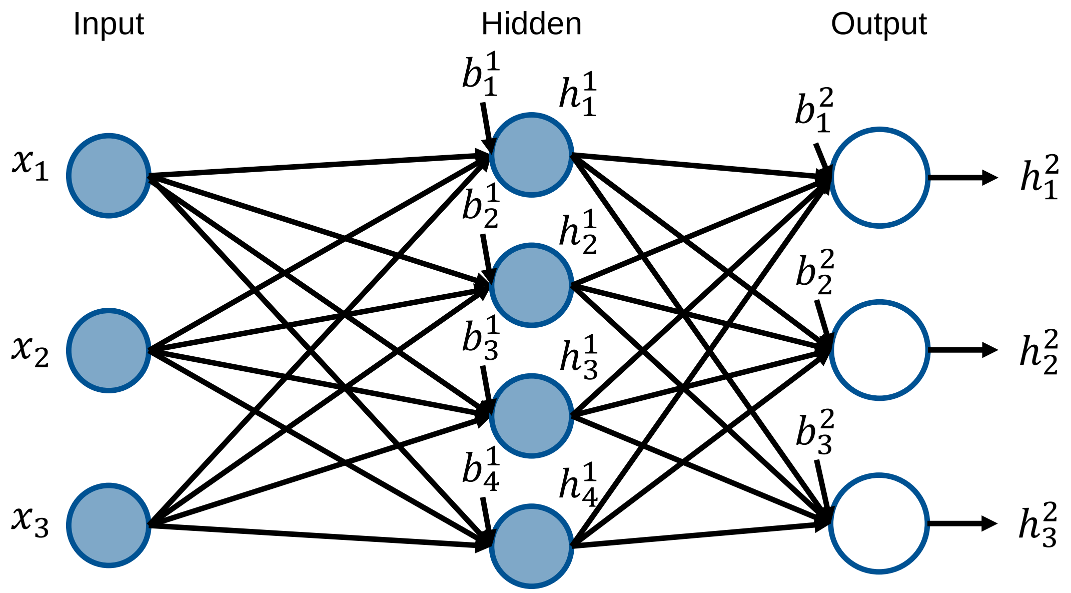
Multilayer Perceptron

Generalized framework

- An L-layer MLP consists of $L-1$ hidden layers and the output layer in depth L
- Input to the network: $\boldsymbol{h}^0 = \boldsymbol{x}$
- Output of the network: $\boldsymbol{h}^L = \boldsymbol{y}$
- Output of z^k neurons in layer k with $k \in (0, \dots, L)$: $\boldsymbol{h}^k = \left(h_1^k, h_2^k, \dots, h_{z_k}^k\right)$
- In the $(z_k \times z_{k-1})$-matrix $\boldsymbol{w}^k$, the entry w_{ij}^k is the weight of the connection from the j-th neuron in layer $k-1$ to the i-th neuron in layer k
- The bias inputs to the $\boldsymbol{z}_k$ neurons in layer k are $\boldsymbol{b}_k = \left(b_1^k, b_2^k, \dots, b_{z_k}^k\right)$
- Calculation of the output in layer k for $k \in (1, \dots, L)$: $\boldsymbol{h}^k = f\left(\left(\mathbf{w}^k\right)^{\mathrm{T}} \boldsymbol{h}^{k-1} + \boldsymbol{b}^k\right)$, where $f()$ denotes the activation function

Multilayer Perceptron

Topology of MLP with $K = 3$ and $L = 2$



Multilayer Perceptron

Some remarks to topology of MLP

- Each neuron h_i^k in a hidden layer k computes its output according to the activation function, e.g. sigmoid activation function $h_i^k = \frac{1}{1+\mathrm{e}^{-\left(\left(\mathbf{w}_{i,j}^k\right)^{\mathrm{T}} \boldsymbol{h}_j^{k-1} + \boldsymbol{b}_i^k\right)}}$
- Neurons in the output layer
  - Neurons h_i^L in the output layer form an SLP that takes the outputs of the last hidden layer as inputs
  - $\boldsymbol{y} = \boldsymbol{h}^L = f\left((\boldsymbol{\Theta}^L)^{\mathrm{T}} \cdot \boldsymbol{h}^{L-1} + \boldsymbol{b}^L\right)$
  - Regression: activation of output neuron is identity function and $\boldsymbol{y}$ contains one element
  - $K = 2$ Classier: activation of the output neuron is sigmoid function and $\boldsymbol{y}$ contains only one element
  - $K > 2$ Classier: activations of the output neurons are softmax functions and $\boldsymbol{y}$ contains K element.

Backpropagation

Basic principle of backpropagation learning algorithm

- Formulate **error function $\boldsymbol{E}$** on the output of the MLP
- Determine dependence of the error function on the parameters and weights $E(\mathbf{w}, \ldots)$
- Calculate partial derivatives $\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}$ of the error function $E(\mathbf{w})$
- Weight adjustment proportional to negative partial derivative

$$\Delta \mathrm{w}_{ij} = -\alpha \frac{\partial E(\mathbf{w})}{\partial w_{ij}}$$

Backpropagation

Backpropagation of neurons

- The output layer is a SLP taking the outputs of the last hidden layer as inputs
- The output y and the prediction can be used for the error function

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}\left(a_i^L - y_i\right)^2 \text{ with } a_i^L = f\left(z_i^L\right) \text{ and } z_i^L = \left(\mathbf{w}_i^L\right)^{\mathrm{T}} \cdot \boldsymbol{h}_i^{L-1} + b_i^L$$

$$\frac{\partial E(\mathbf{w})}{\partial w_i} = \frac{\partial E(\mathbf{a})}{\partial a_i^L}\frac{\partial a_i^L}{\partial z_i^L}\frac{\partial z_i^L}{\partial w_i^L}$$

- Weight adjustment as

$$\Delta \mathrm{w}_{ij} = -\alpha\frac{\partial E(\mathbf{w})}{\partial w_{ij}} \text{ and } \Delta\boldsymbol{h}_i^{L-1} = -\alpha\frac{\partial E(\mathbf{w})}{\partial \boldsymbol{h}_i^{L-1}} \text{ and } \Delta b_i^L = -\alpha\frac{\partial E(\mathbf{w})}{\partial b_i^L}$$

- https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd
- https://mlfromscratch.com/neural-networks-explained/#/
- https://brilliant.org/wiki/backpropagation/#the-backpropagation-algorithm

Workshop at 12th ISNVH Congress, Graz, Austria

- Data driven methods for NVH -

… so good so far