Bauhaus-Universität Weimar

1 Computational Mechanics Meets Artificial Intelligence

Leon Herrmann

Stefan Kollmannsberger

Chair of Data Engineering in Construction

Bauhaus-Universität Weimar

Paris, February 2025

Deep Learning in Computational Mechanics – an introductory course, Herrmann et al. 2025

website

book

Bauhaus-Universität Weimar

Contents

What is Artificial Intelligence?

Artificial Intelligence: A Modern Approach, Norvig et al. 2020

Artificial Intelligence

- “Intelligence exhibited by machines/computers“
- (Total) Turing test requires: natural language processing, knowledge representation, automated reasoning, machine learning, (computer vision, robotics)

Intelligence

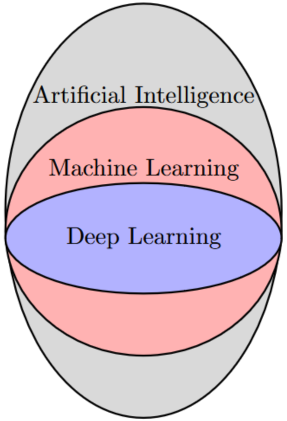
- Human or rational?
- Intelligent thoughts or intelligent behavior?

Machine Learning

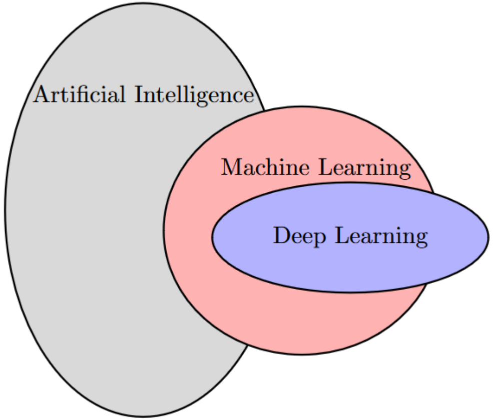
- “Learn from data & generalize to unseen data (without explicit instructions)”

Deep Learning

- “Training (deep) neural networks”



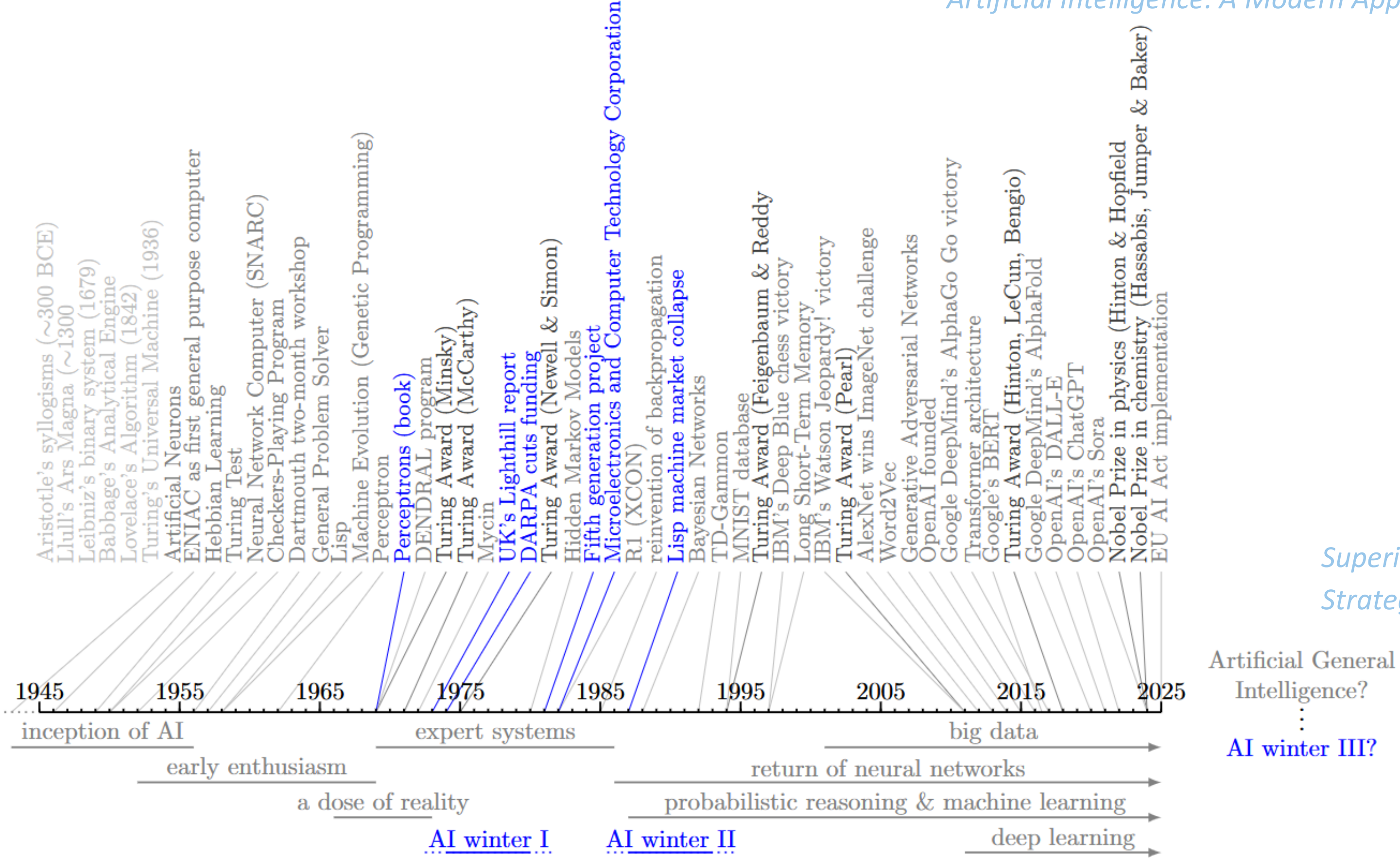
Or?



Inspired by Rebekka Woldseth, author of ”On the use of artificial neural networks in topology optimisation”

History of Artificial Intelligence

Artificial Intelligence: A Modern Approach, Norvig et al. 2020



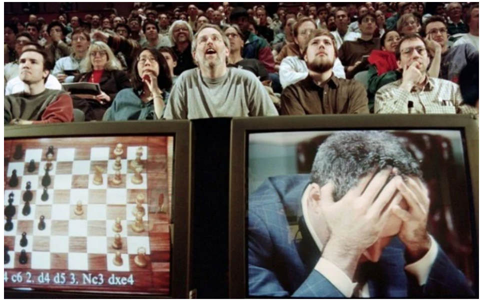
Superintelligence: Paths, Dangers, Strategies, Bostrom 2014

History of Artificial Intelligence

Artificial Intelligence: A Modern Approach, Norvig et al. 2020

- **The inception of artificial intelligence** (1943-1956)
  - Basic physiology of the brain → artificial neurons (on/off); updating rule as Hebbian Learning; SNARC
- **Early enthusiasm, great expectations** (1952-1969)
  - Turing "a machine can never do X"; models were based on logic and symbolic reasoning; (GPS, Lisp, perceptron)
- **A dose of reality** (1966-1973)
  - Overconfidence: models based on "informed introspection" & "intractability of attempted problems"; Lighthill
- **Expert systems** (1969-1986)
  - Instead of general-purpose tools; domain-specific knowledge; (DENDRAL, Mycin, R1); Fifth Generation Project
- **The return of neural networks** (1986-)
  - Reinvention of backpropagation
- **Probabilistic reasoning and machine learning** (1987-)
  - Reaction to failure of expert systems; learn from experience → adaptable & incorporation of uncertainty
  - Hidden Markov Models (Reinforcement Learning); Bayesian Networks; TD-Gammon
- **Big data** (2001-)
  - World Wide Web: large datasets (billions-trillions of samples); ImageNet (challenge), IBM's Watson
- **Deep learning** (2011-)
  - Hardware improvements (GPU: $10^{14} - 10^{17}$ vs CPU: $10^9 - 10^{10}$ Flops); (Deep CNNs in AlexNet); AlphaGo

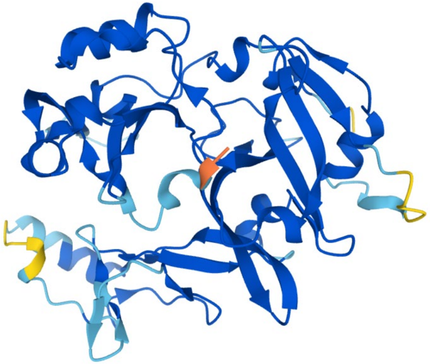
Recent Achievements in Artificial Intelligence



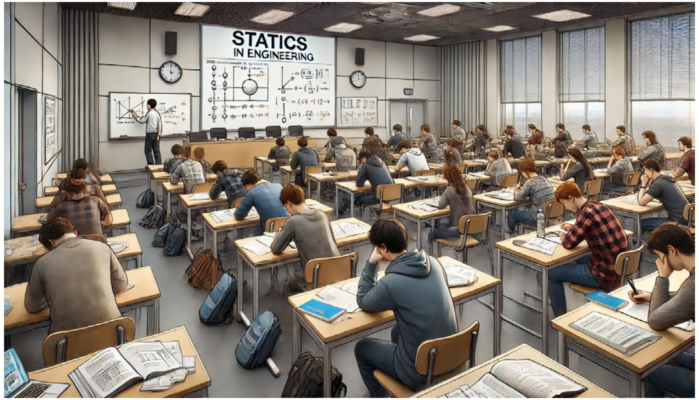
https://media.freemalaysiatoday.com/wp-content/uploads/2022/05/lifestyle-garry-emel-pic-110522.jpg

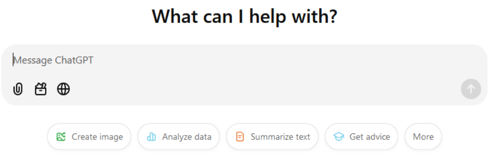
Cats versus dogs

https://media.freemalaysiatoday.com/wp-content/uploads/2016/03/AlphaGo.jpg



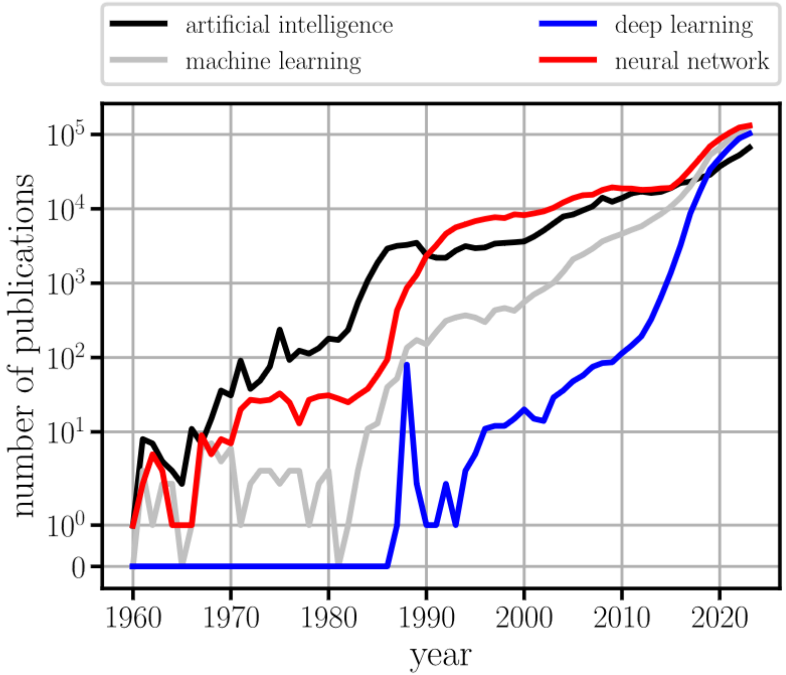
https://commons.wikimedia.org/wiki/File:C12orf29_AlphaFold.png



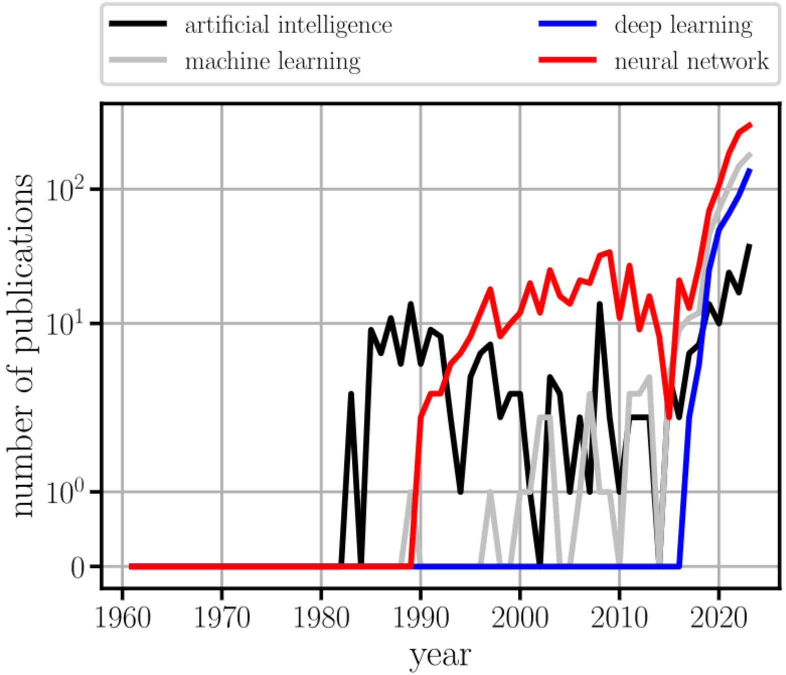


Artificial Intelligence in Science

Check www.aitracker.org for other trends



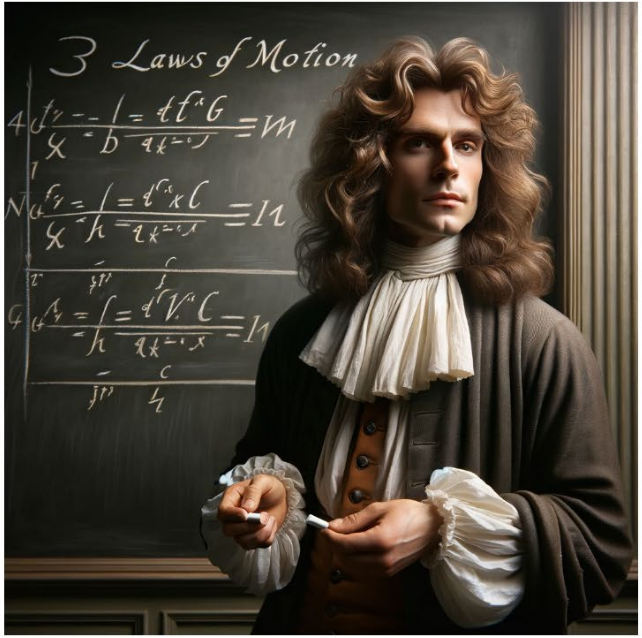
Publications in all fields



Publications in computational mechanics

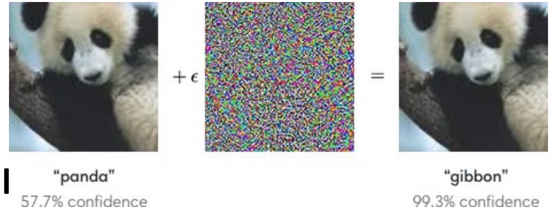
Challenges

- “Generate an image of Isaac Newton in front of a blackboard on which his three laws are written in mathematical notation and chalk.“
- Follow-up: “The laws on the blackboard are incorrect. Please add the correct formulations. If you are unable to do so, simply focus on the second law, which is F=m*a.”



Generated with DALL-E-3

Challenges



https://openai.com/index/attacking-machine-learning-with-adversarial-examples/

Limitations in deep learning in general

- Neural networks break in unpredictable ways → can be consistently fooled
- Deep learning is not robust due to sensitivity to hyperparameters → requires extensive tuning
- Neural networks are uninterpretable, i.e., limited explainability → limits reliability

See chapter 11 for details

Problems in deep learning in computational mechanics

- Reproducibility crisis (bias towards positive results, sensitivity, transparency)
- Fair evaluation metrics are disregarded (breakeven threshold, meaningful metrics, statistical assessments)
- State-of-the-art is not considered

$$\tau = \frac{T_{\text{data}} + T_{\text{train}}}{T_{\text{simulation}} - T_{\text{surrogate}}}$$

Good scientific practice for deep learning in computational mechanics

- Honest assessments & explanations (consider the state-of-the-art & proper metrics)
- Proposed methods should be robust towards hyperparameters (no extensive tuning for a novel problem)
- Careful & narrower selection of problem types (not general-purpose solution)
  - → domain-specific improvements

Towards a meaningful integration of neural networks in computational solid mechanics, Herrmann 2025

Example from topology optimization

The mean squared error

$$MSE = \frac{1}{m}\sum_{i=1}^{m}\left(x_{\text{left}_i} - x_{\text{right}_i}\right)^2$$

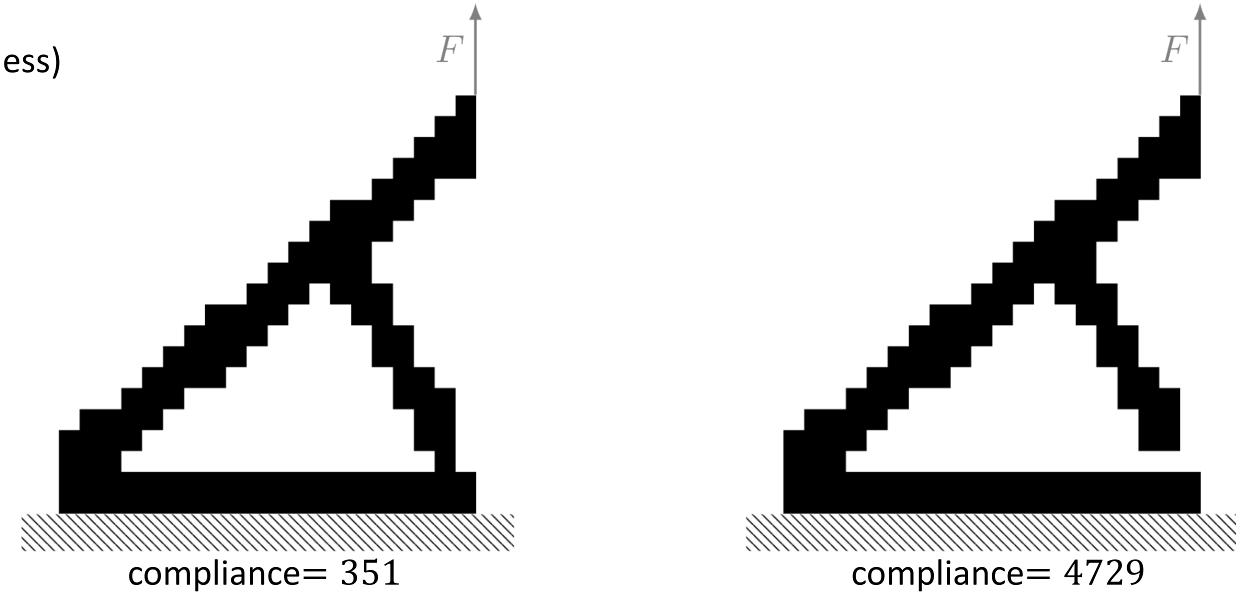
between the two structures is very small ($2.5 \cdot 10^{-3}$), due to one pixel difference.

Structural compliance (inverse of stiffness)

$c = \boldsymbol{F}^T \boldsymbol{u}$

Is different by one order of magnitude

For more details,
see Chapter 9



Computational Mechanics Meets Artificial Intelligence

Computational Mechanics

Abstraction of physical systems (reality) through simplified mathematical models (often differential equations), which are discretized and solved numerically for insight into real-world behavior

Exemplary tasks

- Efficient solutions techniques for forward problems, e.g., finite element, difference, and volume
- Identification tasks (inverse problems), e.g., inferring material distribution/properties from measurements
- Optimization, e.g., finding the optimal material distribution that maximizes stiffness

Machine Learning

Machine Learning, Mitchell 1997

”a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks T, as measured by P, improves with experience E”

Where can machine learning be applied in computational mechanics?

- Identification of mathematical models from data (instead of relying on hand-crafted models)
- Acceleration of forward solvers and optimizers
- Streamlining of pipelines to avoid human experts within the processes

Computational Mechanics Meets Artificial Intelligence

Bauhaus-Universität Weimar

Deep learning in computational mechanics: a review, Herrmann et al. 2024

- **Simulation substitution**
  - Data-driven modelling — Simulation with graph neural networks; DMD; Transfer learning
  - Physics-informed learning — Hamiltonian/Lagrangian neural networks; SINDy; (PINNs)
- **Simulation enhancement** — Input-convex neural networks for material modeling; EUCLID; Neural networks as ansatz function of inverse quantities; Super resolution; Differentiable physics
- **Discretizations as neural networks** — Hardware acceleration with GPUs; (HiDeNN)
- **Generative approaches** — Generative design; Realistic data generation; Anomaly detection; Transformers for natural language processing
- **Deep reinforcement learning** — Control engineering tasks: autonomous flight; robots; Alternative gradient-free optimizer

Bauhaus-Universität Weimar

2 Fundamental Concepts of Machine Learning

Leon Herrmann

Stefan Kollmannsberger

Chair of Data Engineering in Construction

Bauhaus-Universität Weimar

Paris, February 2025

Deep Learning in Computational Mechanics – an introductory course, Herrmann et al. 2025

website

book

Contents

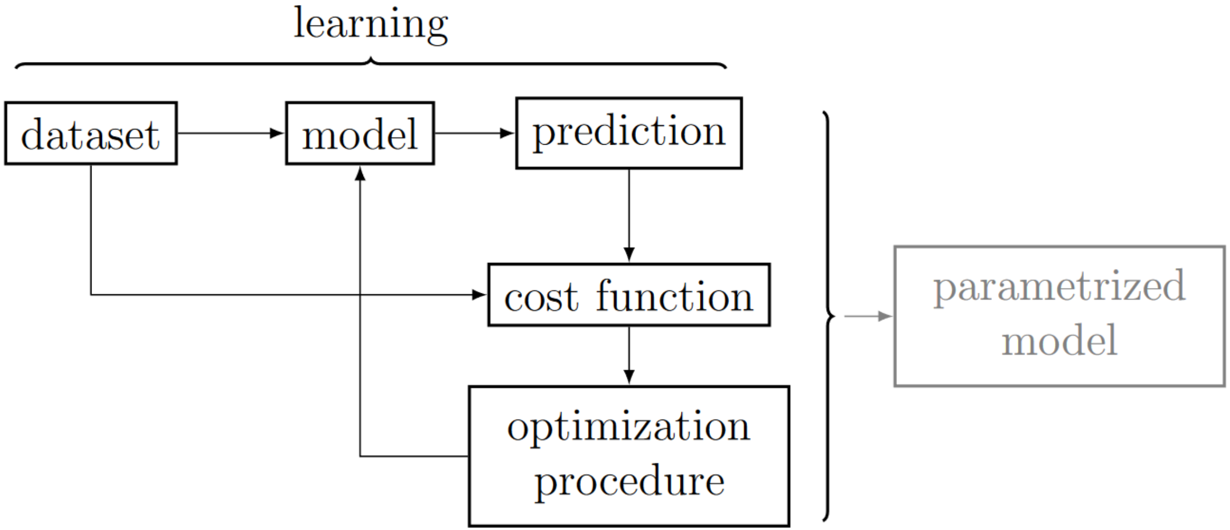
2.1 Definition

"a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks T, as measured by P, improves with experience E"

Machine Learning, Mitchell 1997

Most machine learning algorithms are composed of

- Dataset
- Parametrized model
- Cost function
- Optimization procedure



2.2 Data Structure

Specific dataset (sometimes called design matrix)

$$\boldsymbol{X} = \begin{array}{c c} & \begin{array}{cccc} \text{feature } 1 & \text{feature } 2 & \cdots & \text{feature } n \end{array} \\ \begin{array}{c} \text{example } 1 \\ \text{example } 2 \\ \vdots \\ \text{example } m \end{array} & \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix} \end{array}$$

Examples

- Can be different images, where its features are its pixel values, $n = \text{channels} \times \text{pixels}$
- Can be different houses, where its features are ist properties such as area, number of rooms, age

Notation

- Design matrix $\boldsymbol{X}$
- Design vector of a single example i (1 sample/example) $\boldsymbol{x}_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T$

2.3 Types of Learning

$$\boldsymbol{X} = \begin{array}{r} \\ \text{example } 1 \\ \text{example } 2 \\ \vdots \\ \text{example } m \end{array} \begin{array}{c} \begin{array}{cccc} \text{feature } 1 & \text{feature } 2 & \cdots & \text{feature } n \end{array} \\ \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix} \end{array}$$

Supervised learning

- Algorithm learns from a labeled dataset. Each sample $\boldsymbol{X}_i$ has an accompanying target y_i
- Example: A model learns to distinguish between dogs and cats via annotated images

Unsupervised learning

- Algorithm finds a structure or pattern in the data. This is typically in the form of a probability distribution
- Example: Anomaly detection, i.e., the identification of irregularities in otherwise regular patterns. For example in the detection of tumors in medical imaging

Semi-supervised learning

- Combination of supervised and unsupervised learning, i.e., the data is partly labeled to improve the unsupervised learning.
- Example: The learning of the tumor identification is improved by using some labeled data.

Reinforcement learning

- Interaction between an algorithm and an environment, improving the algorithm to maximize an expected average reward. Common in game-like environments
- Example: The stock market, where more actions with higher rewards are learned

2.4 Machine Learning Tasks

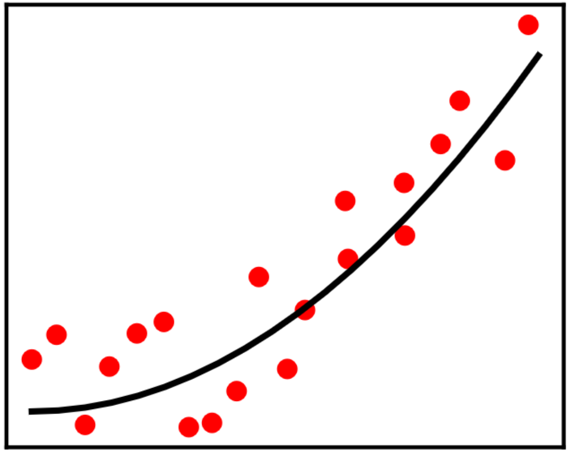
Regression

- Prediction of a numerical value via a (real-valued) mapping between input and output
- Example: Prediction of house prices from criteria like area, number of rooms, age

Classification

- Prediction of a discrete category via a mapping between input and a (discrete) category
- Example: Classification of images in cats and dogs

Classification can be regarded as discrete regression.



Regression

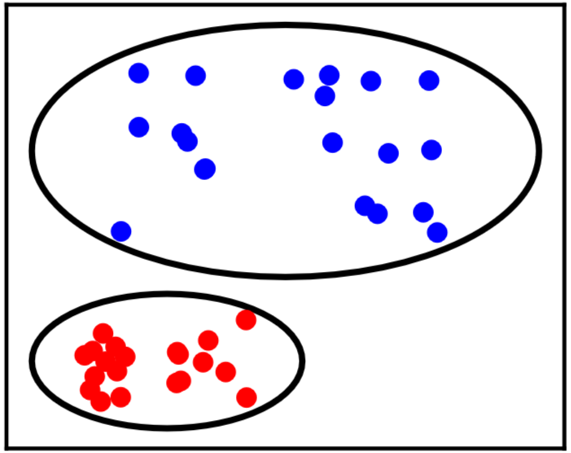
2.4 Machine Learning Tasks

Clustering

- Discovers similarities between data and creates discrete clusters (unsupervised)
- Example: Identification of similar customer groups

Generative modeling

- Generate new data points that resemble a given dataset (without simply reproducing given data points)
- Example: Generate new rim designs given a set of rims



Clustering

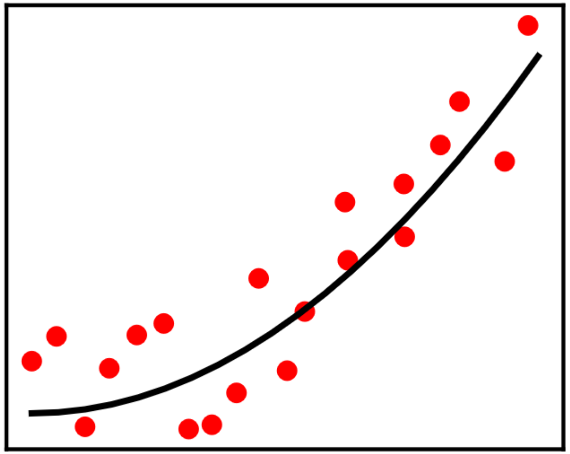
Machine Learning Algorithms

Machine Learning in Additive Manufacturing: State-of-the-Art and Perspectives, Wang et al. 2020

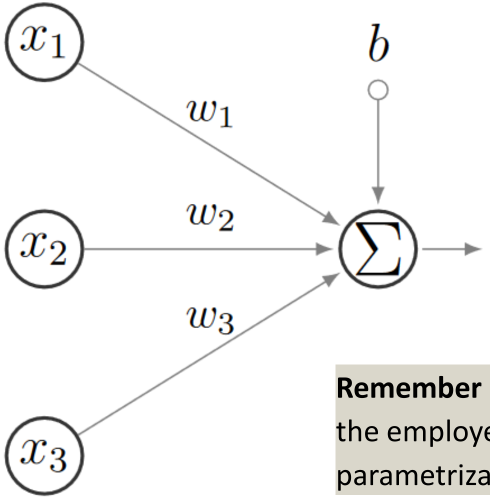
	Classifications	Algorithms	Tasks
	Supervised	Decision trees	Classification
		Random forest	Classfiaciation, regression
covered in Chapter 7		Support vector machines	Classification, regression
		K-nearest neighbours	Classification
		Bayesian network	Classification
		Gaussian process	Regression
		Multi-gene genetic programming	Regression
		Hidden semi-Markov model	Classification
		Multi-layer perceptron	Classification, regression
covered in Chapter 3		Convolutional neural network	Classification
		Recurrent neural network	Time series prediction (regression)
		Adaptive network-based fuzzy inference system	Regression
covered in Chapter 8		Transformers	Regression, classification, generative modeling
	Unsupervised	Self-organizing map	Clustering
		Deep belief network	Classification
covered in Chapter 7		K-means clustering	Clustering
		Reduced order modeling (POD)	Dimensionality reduction
		Autoencoder	Generative modeling, dimensionality reduction
covered in Chapter 8		Generative adversarial networks	Generative modeling, (classification)
		Diffusion model	Generative modeling
	Semi-supervised	Gaussian mixture model	Clustering

2.5 Linear Regression – Prediction

$$\hat{y} = \boldsymbol{w} \cdot \boldsymbol{x} + b = \sum_{j=1}^{n} w_j x_j + b$$

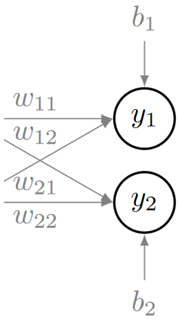


- Target: $\hat{y}$
- Ground truth: $\tilde{y}$ (associated with $\tilde{\boldsymbol{x}}$)
- Example vector: $\boldsymbol{x}$
- Weight vector: $\boldsymbol{w}$
- Bias: b



$$\hat{y} = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

Remember Exercise 3 (stress field from perceptrons): the employed perceptron expresses a similar parametrization



2.5 Linear Regression – Performance Measurement

Prediction in nd for sample i:

$$\hat{y}_i = \boldsymbol{w} \cdot \boldsymbol{x}_i + b = \sum_{j=1}^{n} w_j x_{ij} + b$$

i is an example/sample and j is a feature

- each feature has an associated weight, which is shared across all samples

Remember the design matrix

$$\boldsymbol{X} = \begin{array}{c c} & \begin{array}{cccc} \text{feature } 1 & \text{feature } 2 & \cdots & \text{feature } n \end{array} \\ \begin{array}{c} \text{example } 1 \\ \text{example } 2 \\ \vdots \\ \text{example } m \end{array} & \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix} \end{array}$$

Where each example vector is defined as $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \dots, x_{in})^T$

2.5 Linear Regression – Performance Measurement

Prediction in semi-vector notation for sample i

$$\hat{y}_i = \boldsymbol{w} \cdot \tilde{\boldsymbol{x}}_i + b$$

Squared error for a single sample i (with **ground truth** $\tilde{y}_i$)

$$(\tilde{y}_i - \hat{y}_i)^2$$

Mean squared error (MSE) for a dataset X with m samples (including the design vectors x_i of each sample i)

$$\mathcal{L} = \frac{1}{m}\sum_{i=1}^{m}(\tilde{y}_i - \hat{y}_i)^2 = \frac{1}{m}\sum_{i=1}^{m}(\tilde{y}_i - \boldsymbol{w} \cdot \tilde{\boldsymbol{x}}_i + b)^2$$

Cost function

$$C(\boldsymbol{w}, b) = \mathcal{L} + \cdots$$

Optimization problem

$$\min_{\boldsymbol{w},b} C(\boldsymbol{w}, b) = \min_{\boldsymbol{w},b} \frac{1}{m}\sum_{i=1}^{m}\left(\tilde{y}_i - (\boldsymbol{w} \cdot \tilde{\boldsymbol{x}}_i + b)\right)^2$$

In machine learning optimization is referred to as **learning** when the model is applied to previously unseen problems (i.e., datapoints). This stands in contrast to structural optimization in which one specific design is obtained through optimization.

2.5 Linear Regression – Data Split

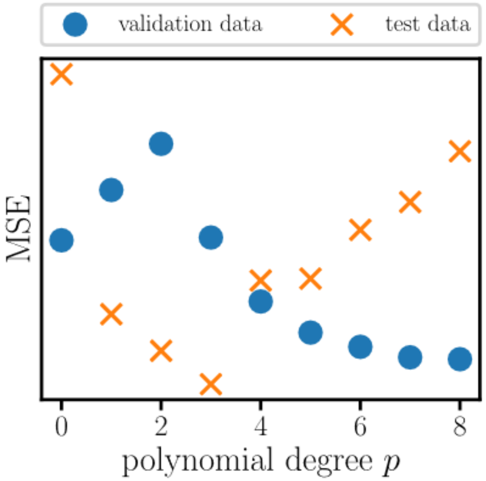
Data is split into

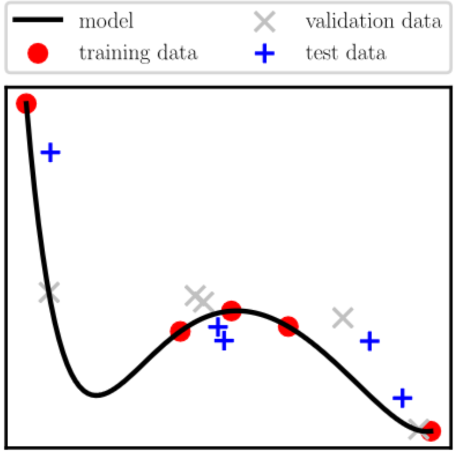
- **Training set** $(\sim 80\%)$
  - to train the model (i.e., find the correct weights and bias)
- **Validation set** $(\sim 10\%)$
  - to validate the training (i.e., evaluate if training is successful, e.g., to detect/avoid **overfitting**)
  - to find the correct (machine learning algorithm) **hyperparameters**
- **Testing set** $(\sim 10\%)$
  - to test/assess the validity of the final model (i.e., weights, bias, and hyperparameters)
  - At this point nothing is allowed to be changed (otherwise testing set becomes validation set)
  - In AI double-blinded challenges are common (test set is released only after handing model to jury)

2.5 Linear Regression – Data Split

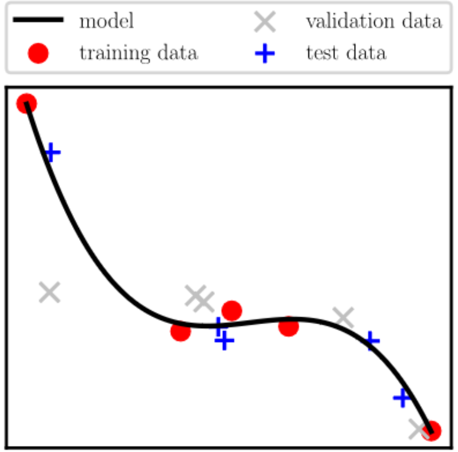
Example

- Fitting of datapoints with a polynomial where the hyperparameter is the polynomial degree p





$p = 3$



$p = 8$

- The validation set shows that $p = 8$ leads to the lowest validation error for the trained model
- The test set shows that some overfitting happened during hyperparameter tuning (this information is not available during/for model development)
- The best model would rely on $p = 3$

2.5 Linear Regression – Optimization

$$\min_{\boldsymbol{w},b} C(\boldsymbol{w},b) = \min_{\boldsymbol{w},b} \frac{1}{m}\sum_{i=1}^{m}\left(\tilde{y}_i - (\boldsymbol{w}\cdot\tilde{\boldsymbol{x}}_i + b)\right)^2$$

Note that $\boldsymbol{X}$ requires a column of ones for the bias b

For a more concise notation let us denote all learnable parameters in a vector $\boldsymbol{\Theta} = (\boldsymbol{w}, b)^T$

All predictions $\hat{y}_i$ are collected in the vector $\hat{\boldsymbol{y}}$.

This allows to write the model function $\hat{y}_i = \boldsymbol{w}^T\tilde{\boldsymbol{x}}_i + b$ as $\hat{\boldsymbol{y}} = \boldsymbol{X\Theta}$ yielding the minimization

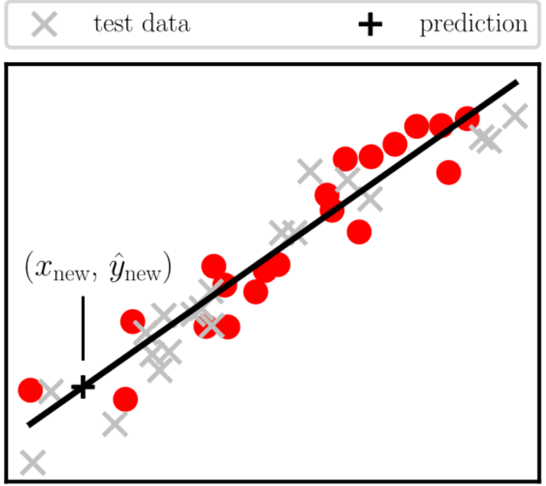
$$\min_{\boldsymbol{\Theta}} C(\boldsymbol{\Theta}) = \min_{\boldsymbol{\Theta}}(\tilde{\boldsymbol{y}} - \boldsymbol{X\Theta})(\tilde{\boldsymbol{y}} - \boldsymbol{X\Theta}) = \min_{\boldsymbol{\Theta}}(\tilde{\boldsymbol{y}}^T\tilde{\boldsymbol{y}} - 2\tilde{\boldsymbol{y}}^T\boldsymbol{X\Theta} + (\boldsymbol{X\Theta})^T\boldsymbol{X\Theta})$$

The minimization is solved by setting the first derivative of C with respect to $\boldsymbol{\Theta}$ to zero (using $\boldsymbol{r} = \tilde{\boldsymbol{y}} - \boldsymbol{X\Theta}$)

$$\frac{1}{2}\frac{\partial \boldsymbol{r}(\boldsymbol{\Theta})^2}{\partial \boldsymbol{\Theta}} = \frac{1}{2}(-2\boldsymbol{X}^T\tilde{\boldsymbol{y}} + 2\boldsymbol{X}^T\boldsymbol{X\Theta}) = -\boldsymbol{X}^T\tilde{\boldsymbol{y}} + \boldsymbol{X}^T\boldsymbol{X\Theta} = 0$$

$$\boldsymbol{X}^T\boldsymbol{X\Theta} = \boldsymbol{X}^T\tilde{\boldsymbol{y}}$$

$$\boldsymbol{\Theta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\tilde{\boldsymbol{y}}$$



<u>Such a closed form solution is only possible if $\hat{\boldsymbol{y}}$ (or rather $\partial C/\partial\boldsymbol{\Theta}$) is **linear** with respect to $\boldsymbol{\Theta}$</u>

2.8.1 Gradient Descent

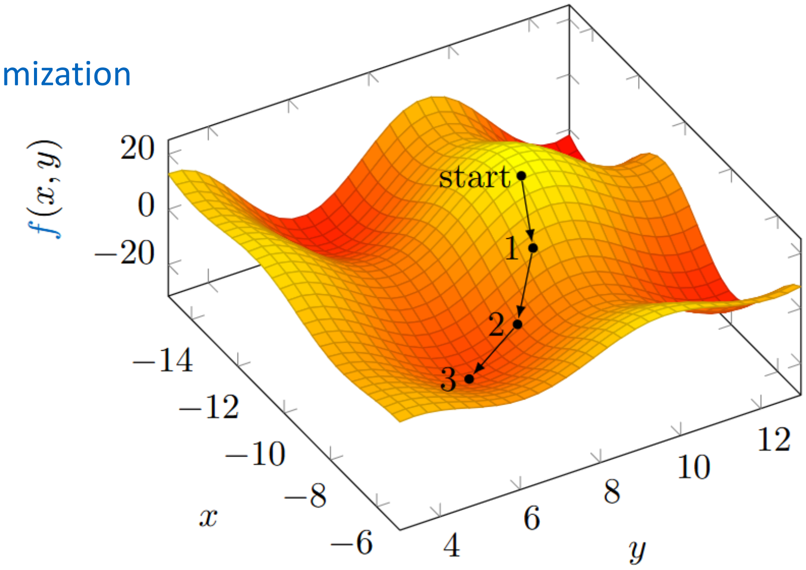
Improve prediction $\hat{y}_i = \boldsymbol{w} \cdot \boldsymbol{x}_i + b$ via (iterative) cost function minimization

$$\min_{\boldsymbol{w},b} C(\boldsymbol{w}, b) = \min_{\boldsymbol{w},b} \frac{1}{m} \sum_{i=1}^{m} \left(\tilde{\boldsymbol{y}}_i - (\boldsymbol{w} \cdot \boldsymbol{x}_i + b)\right)^2$$

Partial derivatives of cost function with respect to each parameter

$$\frac{\partial C}{\partial \boldsymbol{w}} = \frac{1}{m} \sum_{i=1}^{m} -2x_i(\tilde{\boldsymbol{y}}_i - (\boldsymbol{w} \cdot \boldsymbol{x}_i + b))$$

$$\frac{\partial C}{\partial b} = \frac{1}{m} \sum_{i=1}^{m} -2(\tilde{\boldsymbol{y}}_i - (\boldsymbol{w} \cdot \boldsymbol{x}_i + b))$$



Each gradient descent iteration updates the parameters, such that the cost function decreases

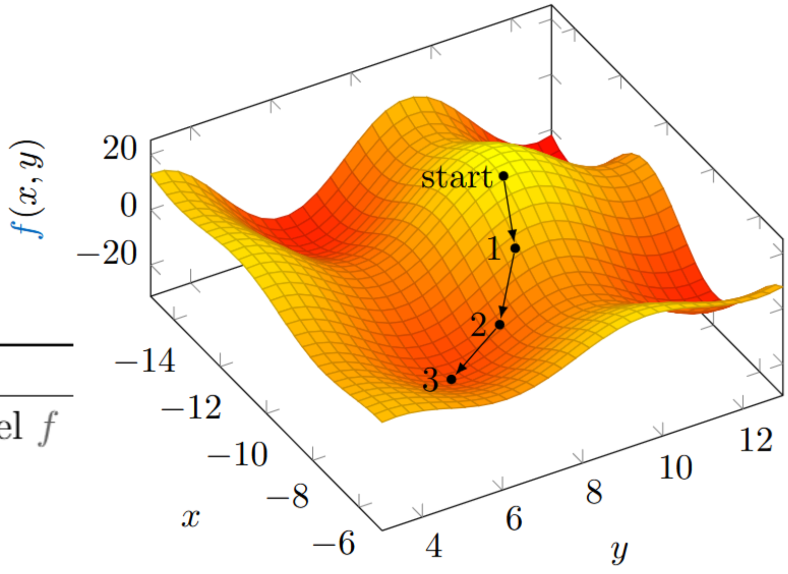
$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \frac{\partial C}{\partial \boldsymbol{w}}$$

$$b \leftarrow b - \alpha \frac{\partial C}{\partial b}$$

α (the **learning rate**) controls the step size

2.8.1 Gradient Descent

- Generalized gradient descent algorithm
- In machine learning:
  - Number of iterations is called **number of epochs**
  - Step size is called **learning rate**



Algorithm 1 Gradient descent

Require: dataset $\tilde{x}, \tilde{y}$, number of epochs n, step size α, model f
initialize the model $f(x; \Theta)$
for all n **do**
Compute the cost function $C(f(\tilde{x}; \Theta), \tilde{y})$
Compute the gradient $\nabla_\Theta C$
Update the model parameters $\Theta \leftarrow \Theta - \alpha \nabla_\Theta C$
end for

Exercises

- E.4 Linear Regression (P & C)
  - Perform a linear regression once by computing the weights directly and once using gradient descent. Do this by hand calculation and with a Python implementation.

Contents

1 Computational Mechanics Meets Artificial Intelligence & 2 Fundamental Concepts of Machine Learning

Leon Herrmann

Stefan Kollmannsberger

Chair of Data Engineering in Construction

Bauhaus-Universität Weimar

Paris, February 2025

Deep Learning in Computational Mechanics – an introductory course, Herrmann et al. 2025

website

book