Applications of Machine Learning in Mechanics of Materials

A. Hartmaier

Interdisciplinary Centre for Advanced Materials Simulation (ICAMS)

Ruhr-Universität Bochum

alexander.hartmaier@rub.de

INTERDISCIPLINARY CENTRE FOR ADVANCED MATERIALS SIMULATION

Documentation

Course material including numerical examples:

[Link to Binder]

[GitHub repository]

PDF's of lecture slides under docs.

Outline

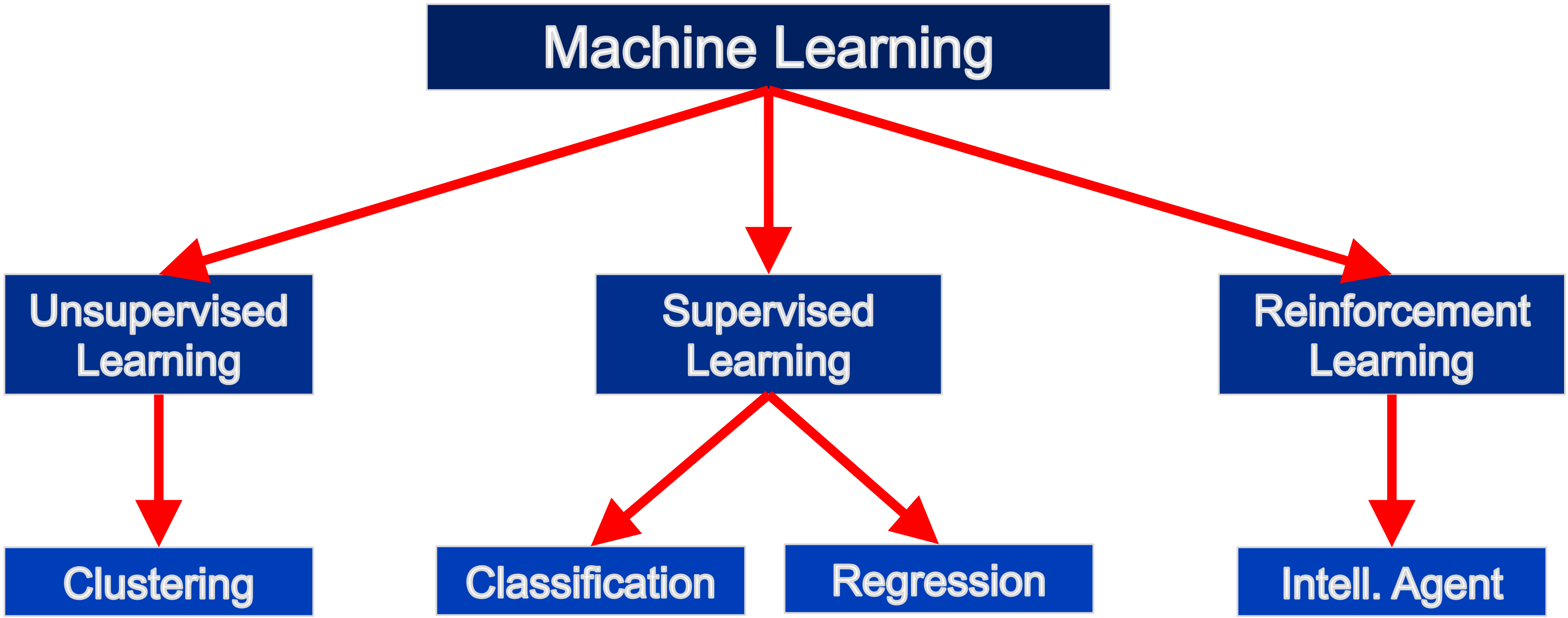
Applications of Machine Learning in Mechanics of Materials

Part I
- Theoretical homogenization rules
- Micromechanical modeling

Part II
- Theory of Finite Element Analysis (FEA)
- Data generation

Part III
- Training of machine learning models
- Analysis of results

Machine Learning (ML)



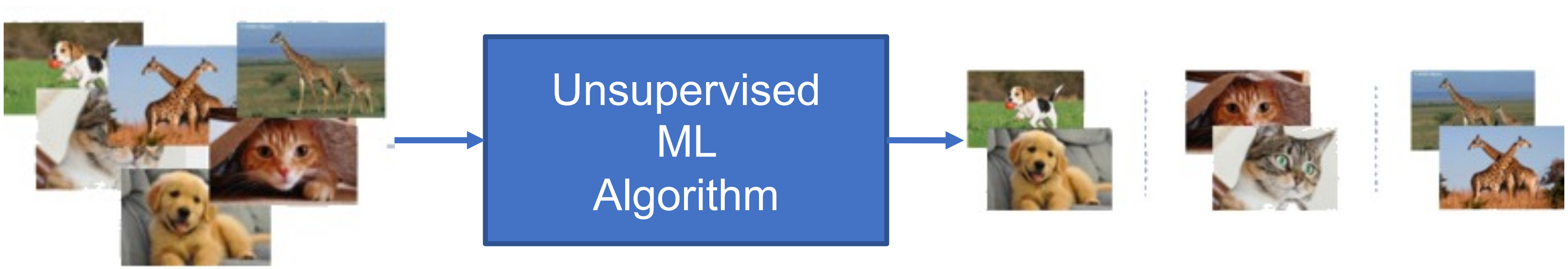
All examples of this lecture have been performed with scikit-learn (https://scikit-learn.org/stable/)

Task: Create a computer code that can play "Go"

Source: Wikipedia
Wikimedia Commons, CC BY-SA 3.0

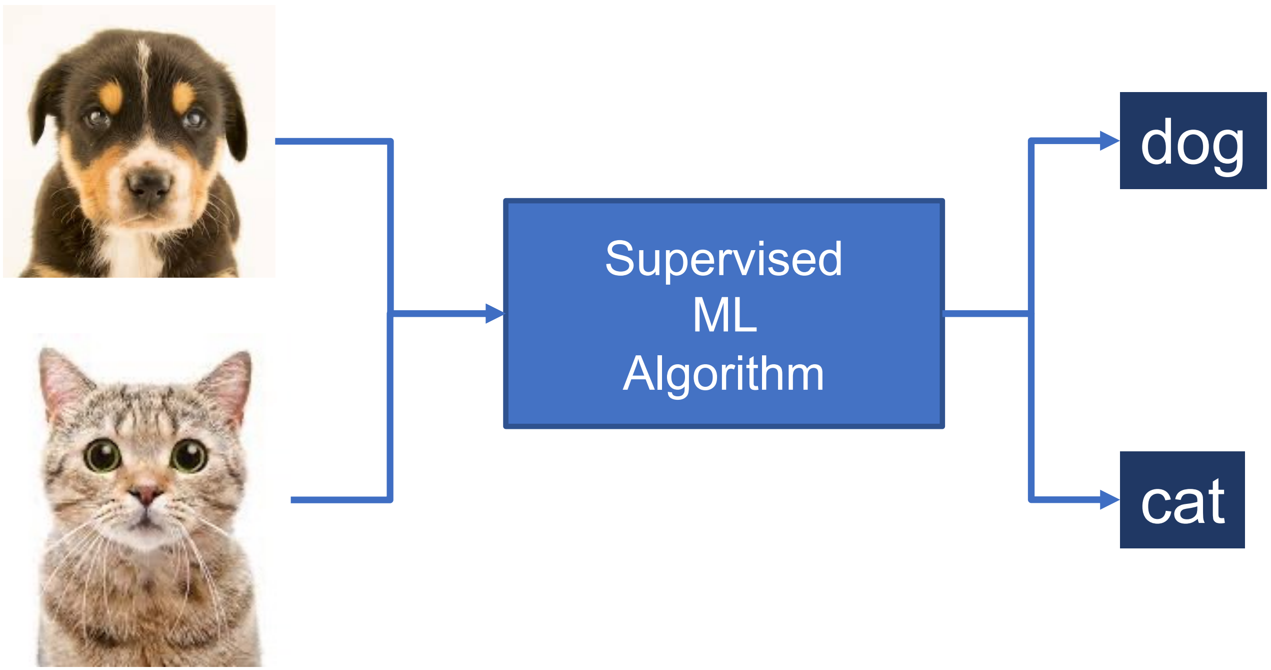
Unsupervised Learning: Clustering

Task: Sort pictures of same animals into groups (clustering)



Supervised Learning: Classification

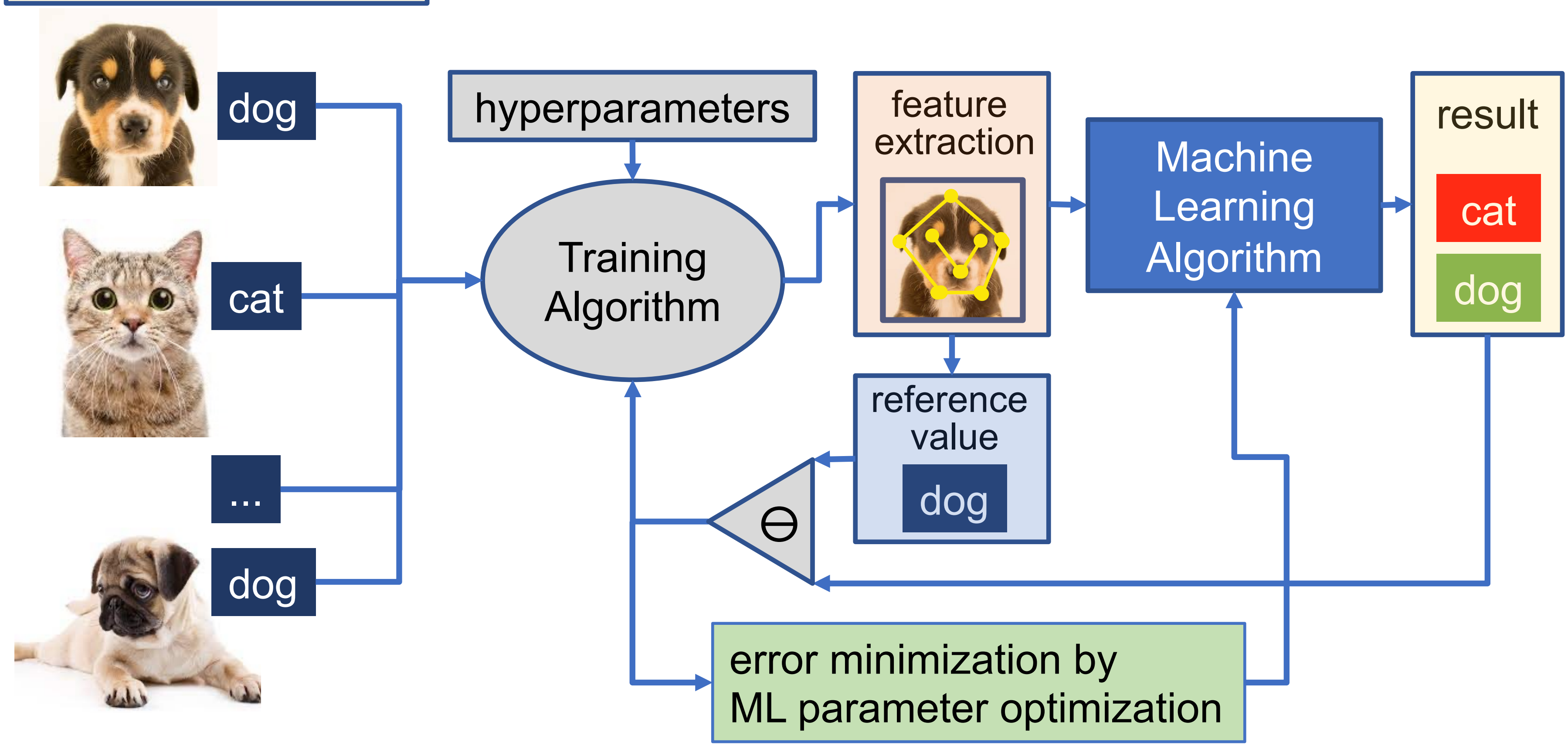
Task: Identify pictures of cats and dogs (classification)



Supervised Learning: Training of ML Model

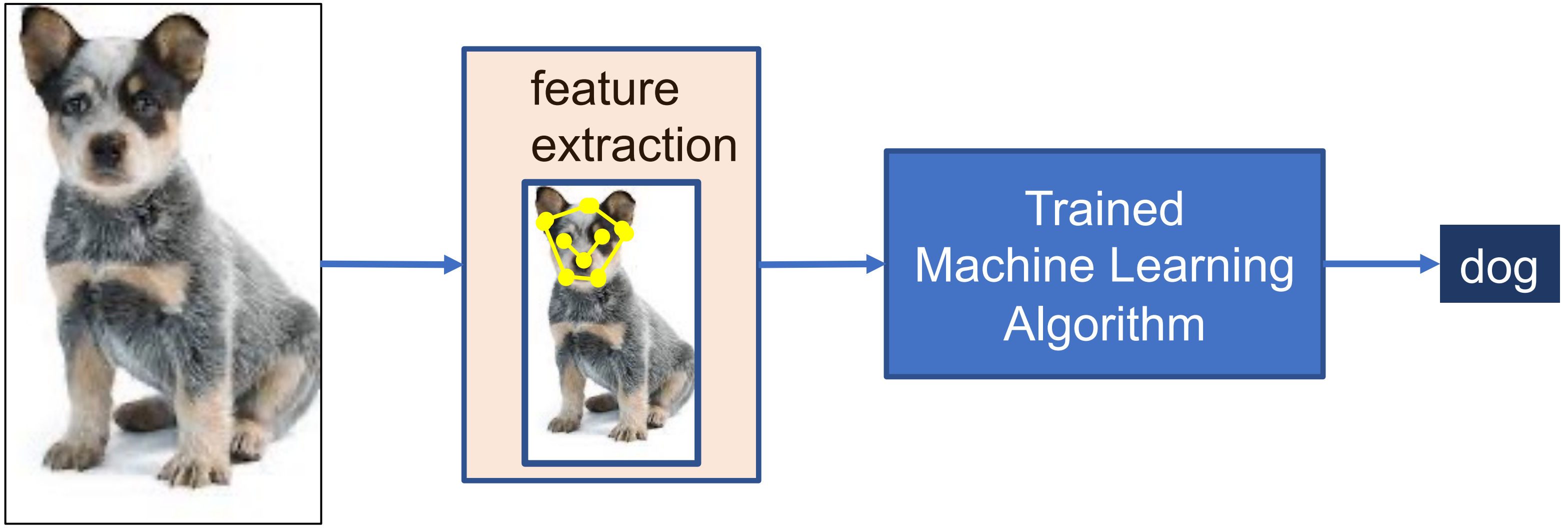
- choice of ML algorithm
- feature design
- hyperparameter optimization

Raw Data with Label



Supervised Learning: Validation

Validation with unseen data



Supervised Learning: Feature Extraction

feature extraction

Automated feature extraction from raw data:
- convolutions of raw data (CNN)
- autoencoder
- Fast Fourier Transform
- N-point-statistics/auto correlation
- Principle Component Analysis
- Singular Value Decomposition

Feature extraction based on domain knowledge:
- extraction of physical quantities
- correlation analysis
- experience with similar tasks

Supervised Learning: Regression

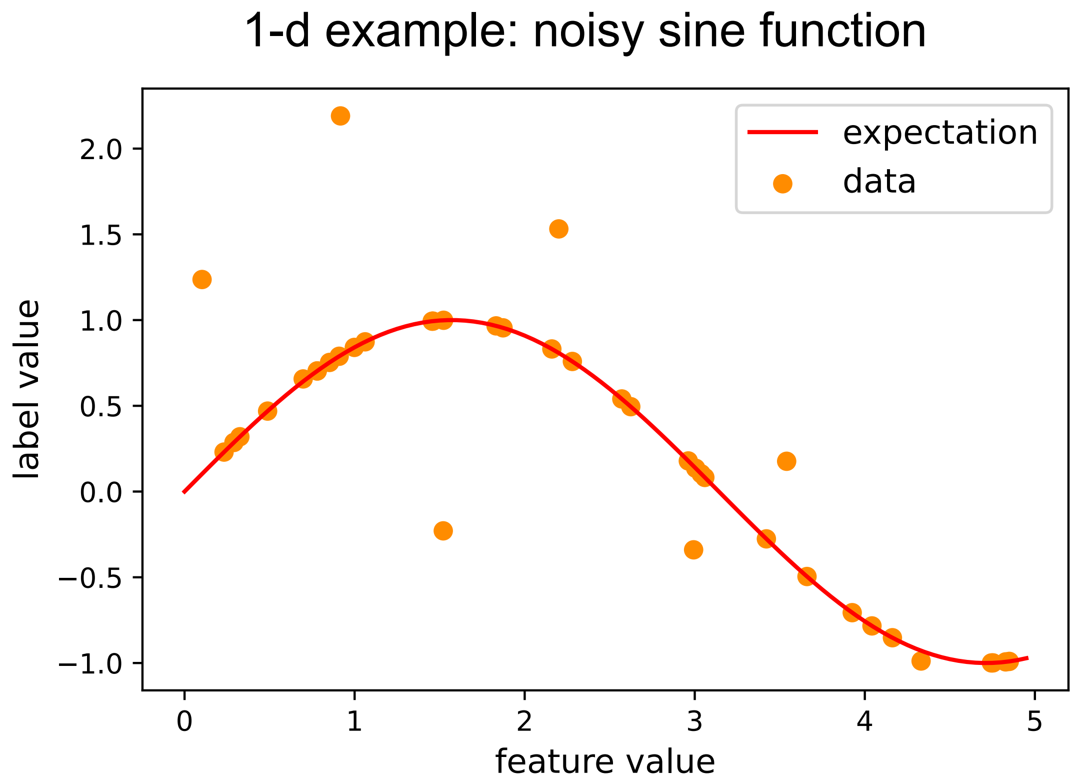
input vector "features"

Selection of features (or descriptors) determines the physics of the ML model

output vector "label" / result

Training Procedure
Find ML parameters that minimize deviation between result of ML model and known data point (ground truth).

Supervised Learning: Regression

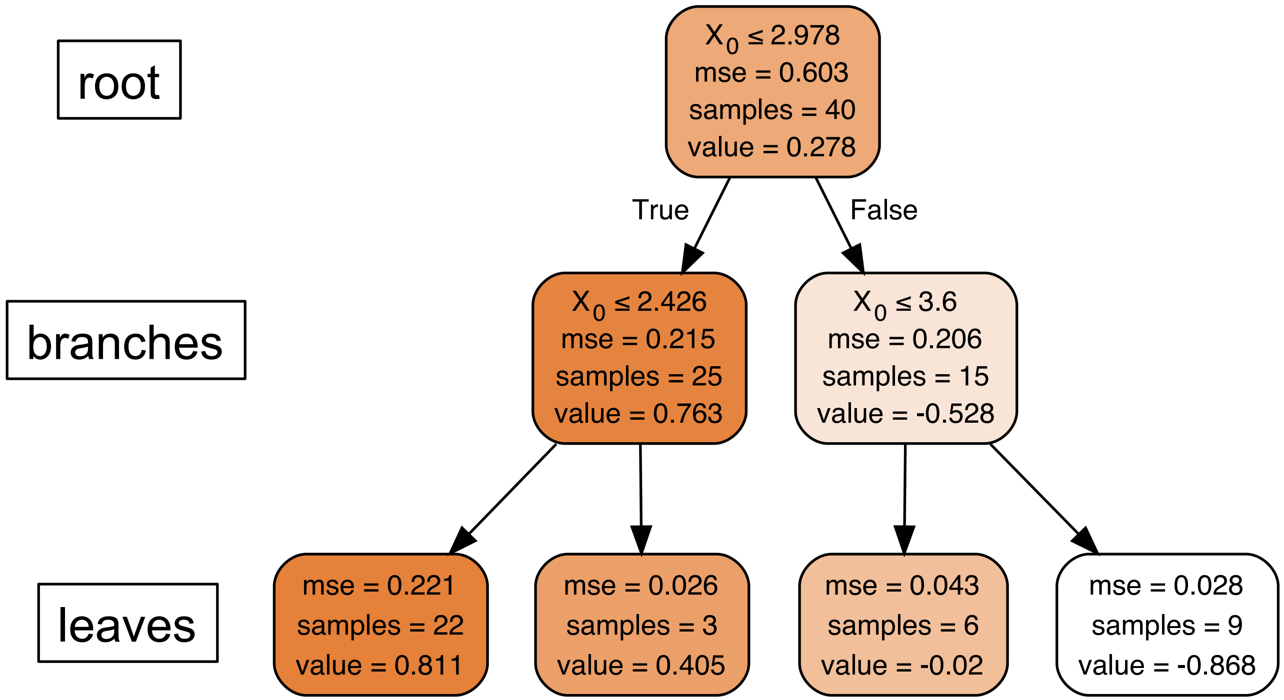


$$y = f(x) = \sin x$$

$$0 \leq x \leq 5$$

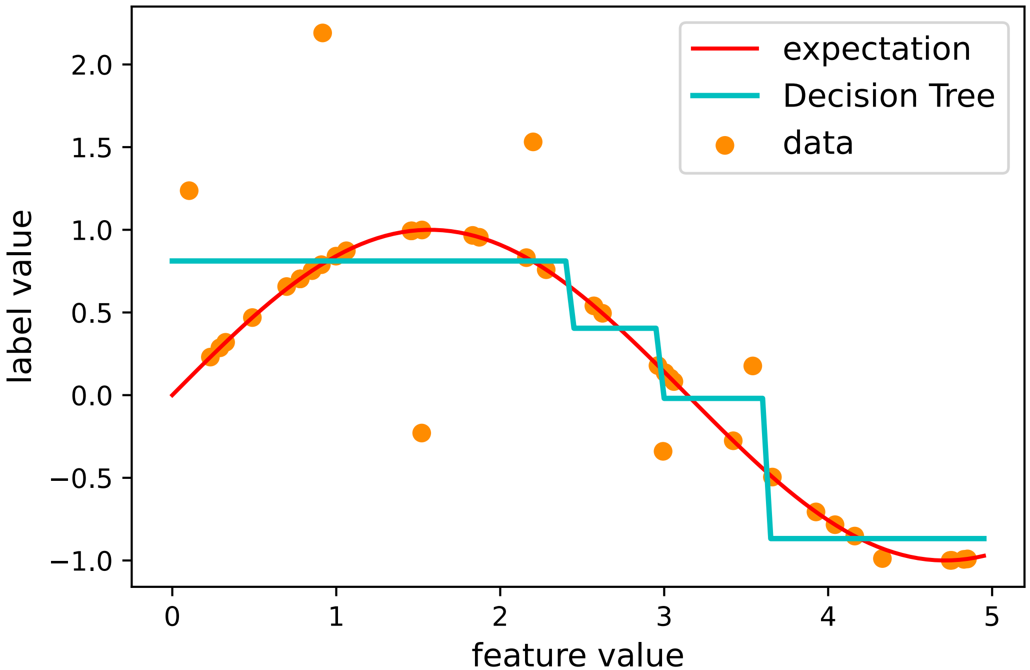
Decision Tree Regression

Succession of if-clauses leads to final result in “leaves”



Supervised Learning: Decision Tree Regression

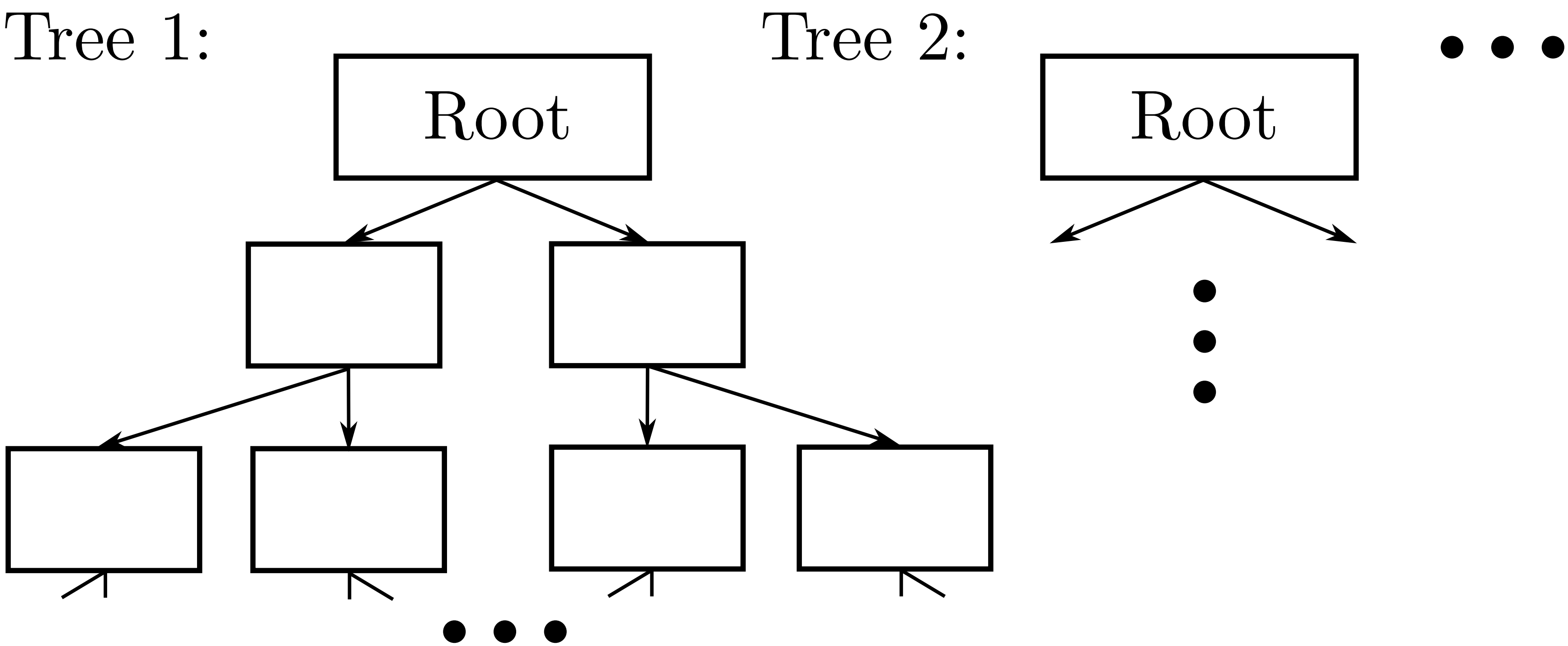
1-d example: noisy sine function



Decision Tree
hyperparameters:
depth = 2

$$y = f(x) = \begin{cases} 0.811 & \text{if} \quad x \leq 2.426 \\ 0.405 & \text{if} \quad 2.426 < x \leq 2.978 \\ -0.02 & \text{if} \quad 2.978 < x \leq 3.6 \\ -0.868 & \text{if} \quad x > 3.6 \end{cases}$$

Random Forest Regression

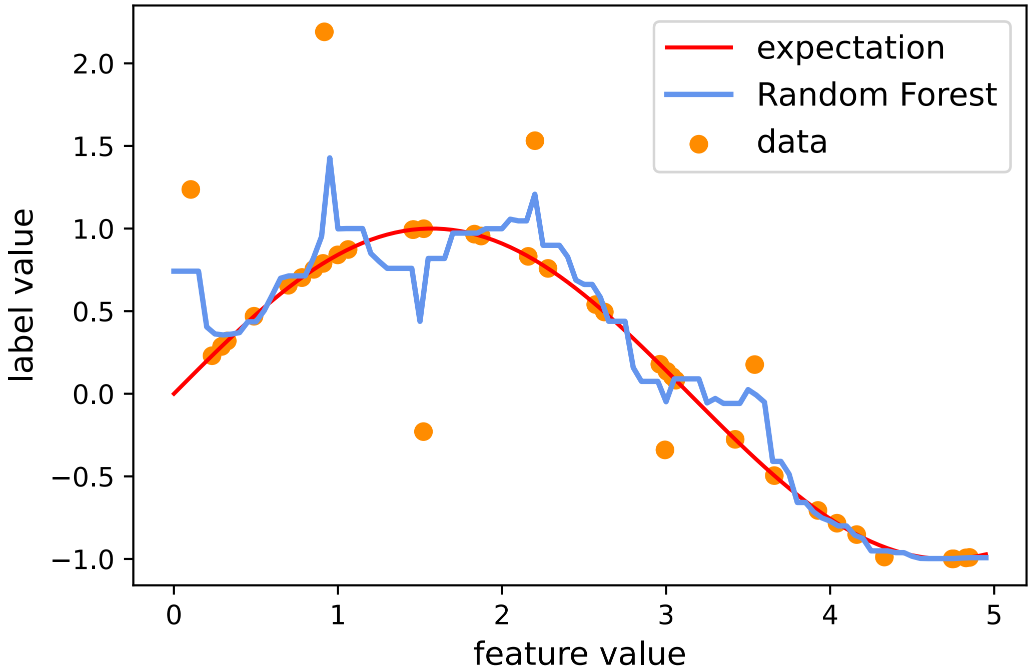


Goal: Create model that predicts output value for given input data by learning simple decision rules

- Number of trees = 100 … 500
- Leaves contain either 1 or 0 samples
- Final result is average of the leave values obtained from all trees

Supervised Learning: Random Forest Regression

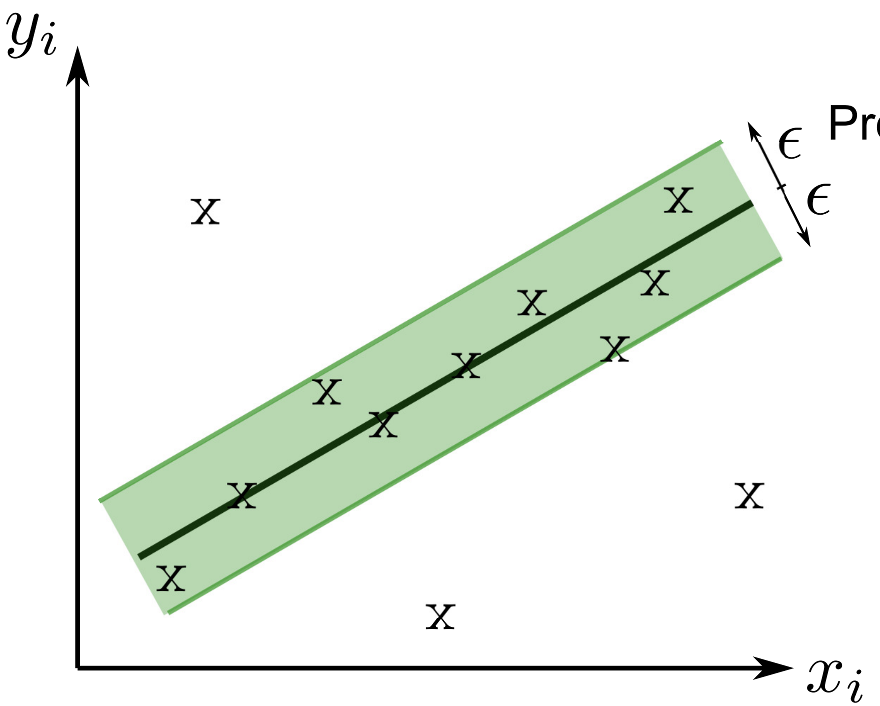
1-d example: noisy sine function



Random Forest hyperparameters:
max. depth = 5
N_{tree} = 20

$$y = f(x) = \frac{1}{N}\sum_{i=1}^{N} f_{DT}^{(i)}(x)$$

Support Vector Machine (Regression/Classification)



Precision ϵ = 1%

SVM function:

$$f(x) = \sum_{k=1}^{n} y_k a_k K\left(x_k^{(SV)}, x\right) + \rho$$

RBF kernel:

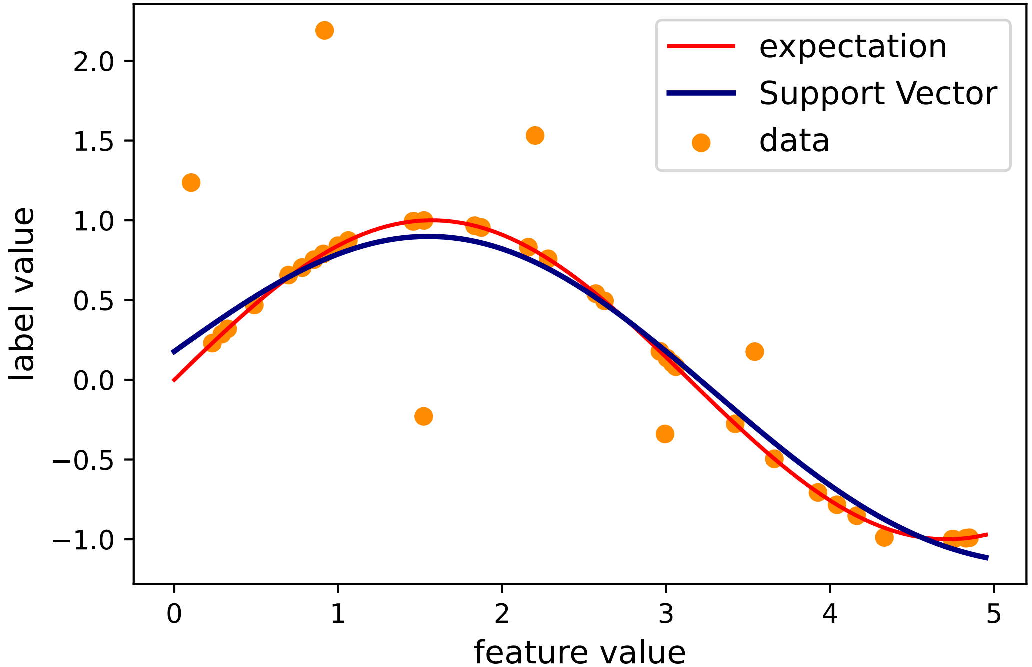
$$K\left(x_k^{(SV)}, x\right) = \exp\left(-\gamma \left\| x - x_k^{(SV)} \right\|^2\right)$$

Goal: Find a function such that data points lie within a corridor of $\pm\epsilon$ (function as flat as possible, actual error unimportant, penalty for outliers)

- Linear or Gaussian kernel for interpolation between support vectors
- Support vectors determined during training function (data points closest to target function)

Supervised Learning: Support Vector Regression

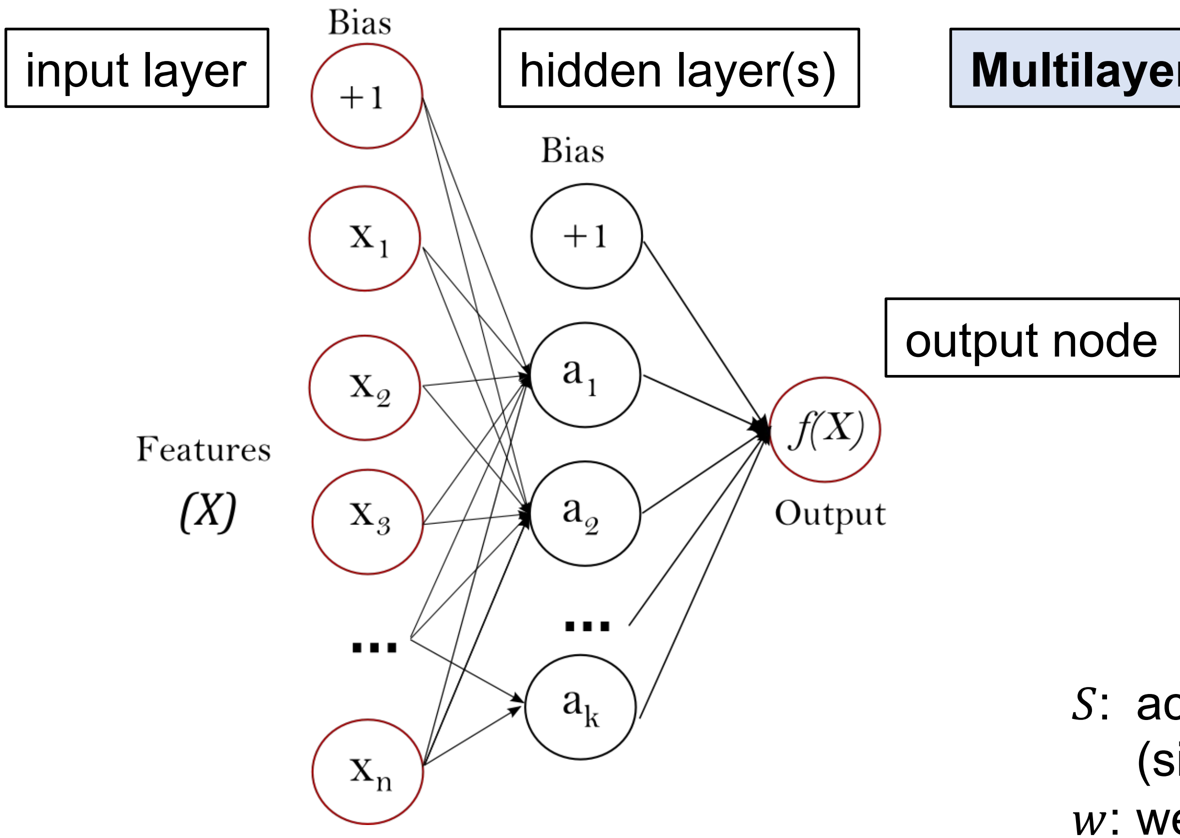
1-d example: noisy sine function



Support Vector Mach.
hyperparameters:
C = 10
gamma = 0.1
kernel: radial basis fct.

$$f(x) = \sum_{k=1}^{n} y_k a_k K\left(x_k^{(SV)}, x\right) + \rho$$

Neural Networks (Regression/Classification)



Multilayer Perceptron (MLP)

$$f_{\text{neuron}}(x) = S(w \cdot x - U)$$

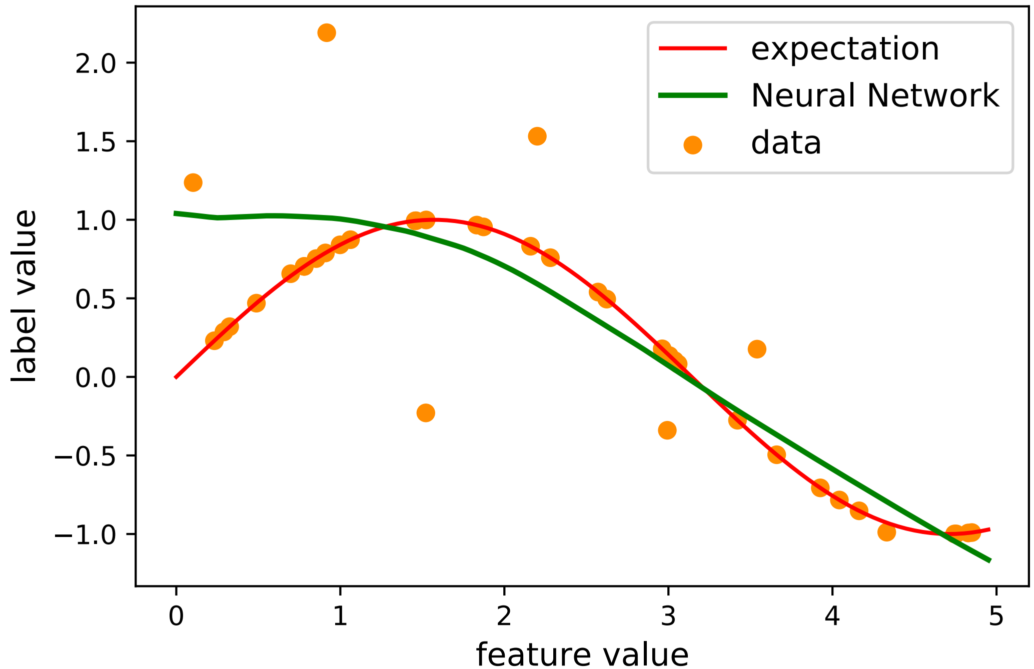
S: activation function (sigmoid, hyperb. tan, ...)
w: weight for input signals
U: bias

Goal: Find bias values and weights for activation functions that describe training data best. – *Deep learning*: multiple hidden layers.

Source: scikit-learn

Supervised Learning: Neural Network

1-d example: noisy sine function



Neural Network
hyperparameters:
hidden layers: 3
neurons: 100
activation function: relu
init. learning rate: 0.001
…

$$f(x) = \sum_{i=1}^{N_{\mathrm{neuron}}} w^{(i)} f_{\mathrm{neuron}}^{(i)}$$

Summary Machine Learning Methods

Common supervised Machine Learning (ML) methods

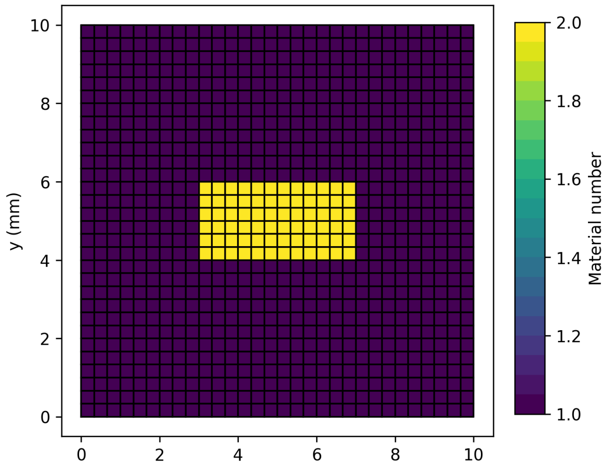
- Random Forrest
  (simple and robust reference method, prone to overfitting, only 2 hyperparameters)
- Support Vector Machines
  (robust and moderate data requirements, only 2 hyperparameters)
- Neuronal Networks
  (very flexible, prone to overfitting, large volumes of training data required, many hyperparameters)

Feature Selection determines physics of the trained model.

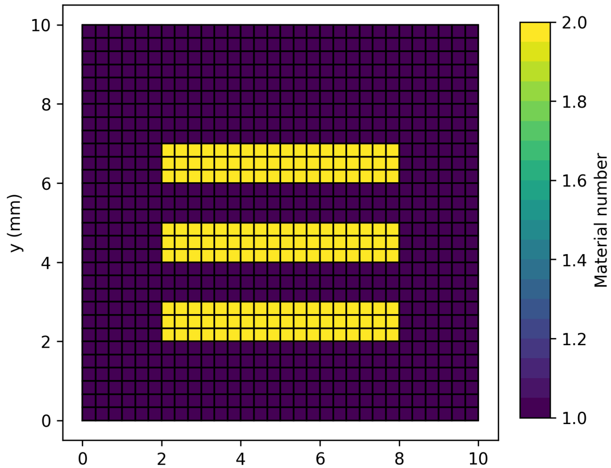
Hyperparameter tuning decisive for success of training.

Independent Validation necessary to ensure validity of trained model.

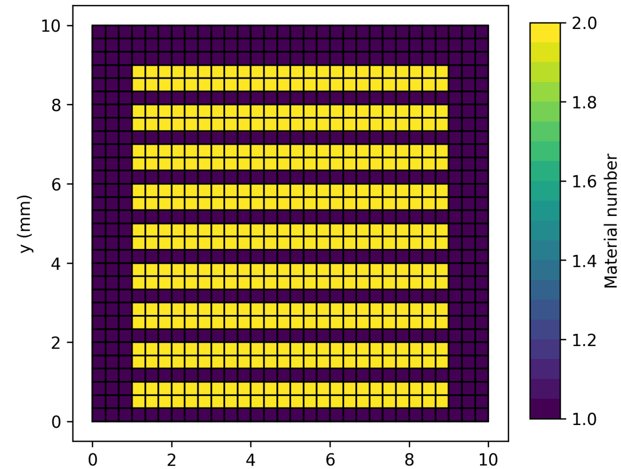
Composite Microstructures for FEM Simulation



Low Fraction Filler
(8%)



Medium Fraction Filler
(18%)

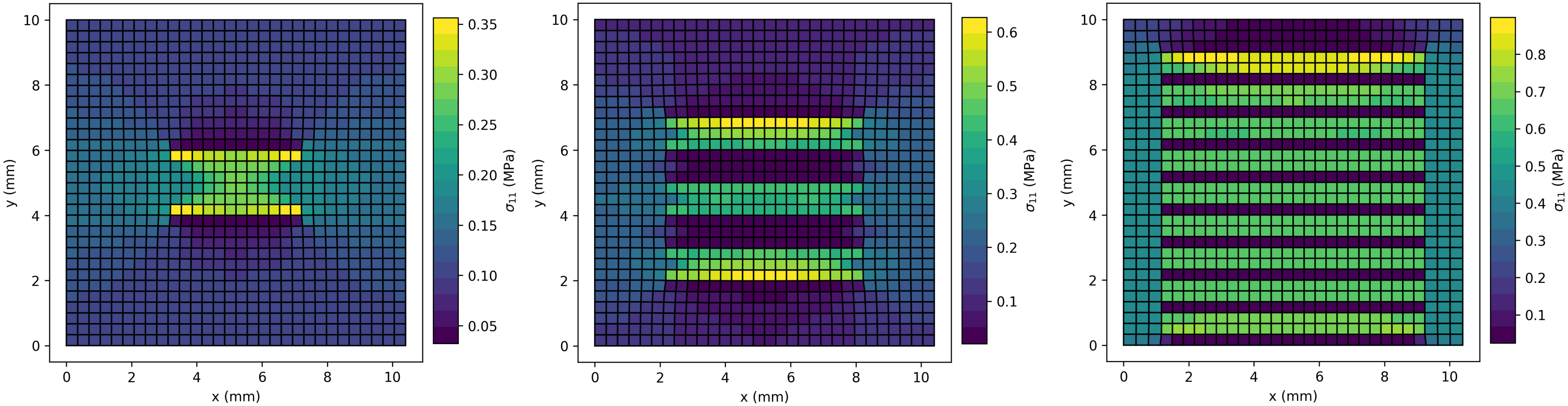


High Fraction Filler
(48%)

Domain: 10 mm × 10 mm
Mesh: 30 × 30 → 900 elements

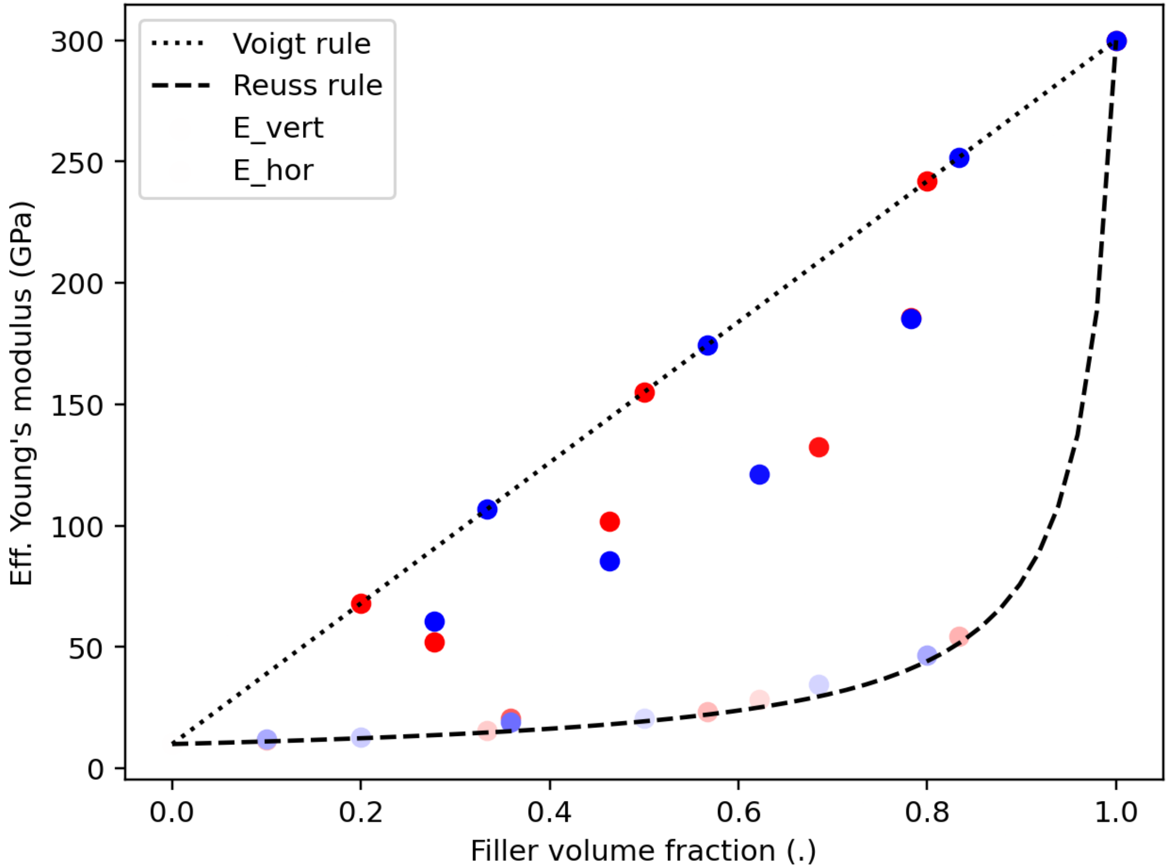
- Three filler configurations with increasing volume fraction
- Filler shape and layout affect stiffness response

Stress Distribution in Composite Under Load



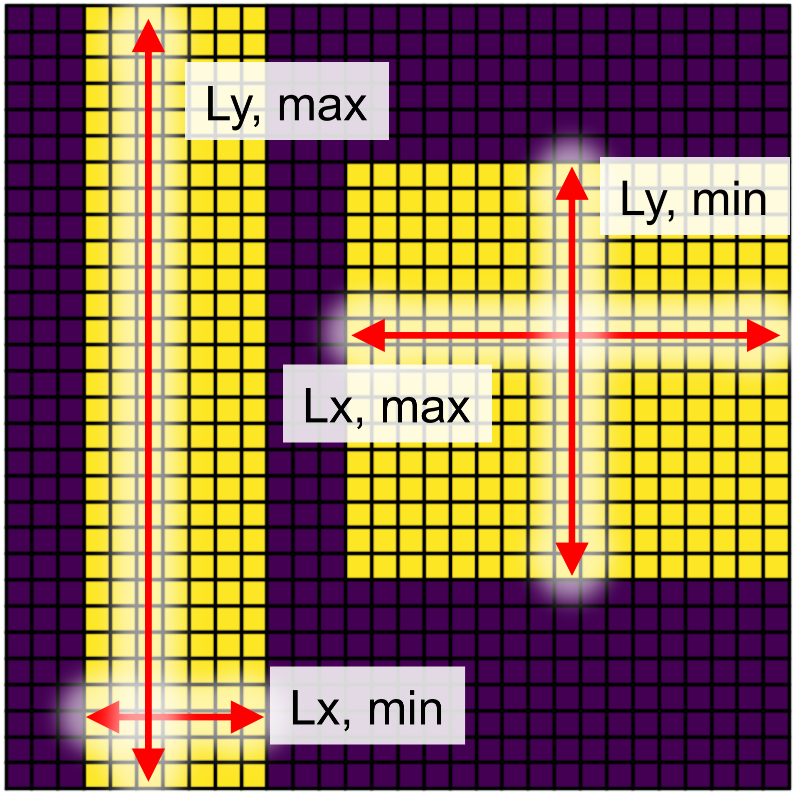
- ➢ High-modulus filler zones concentrate stress
- ➢ Stress localization intensifies with increased filler content

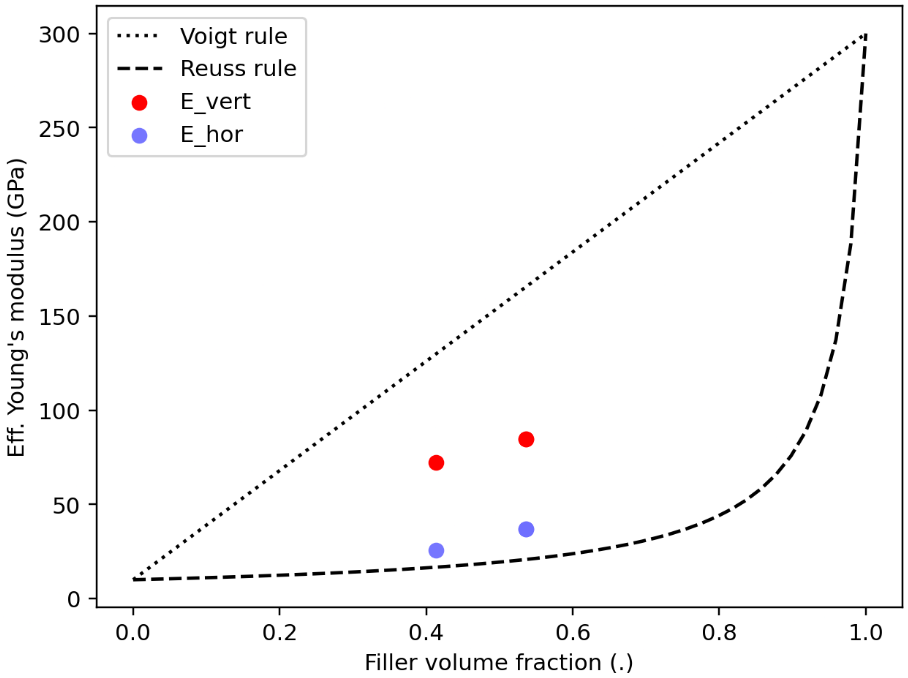
Effective Modulus vs. Filler Volume Fraction



- ➢ FEM results lie between Voigt (upper) and Reuss (lower) bounds
- ➢ Higher filler fraction increases stiffness anisotropy
- ➢ **Problem:** Non-unique relationship b/w stiffness and volume fraction.

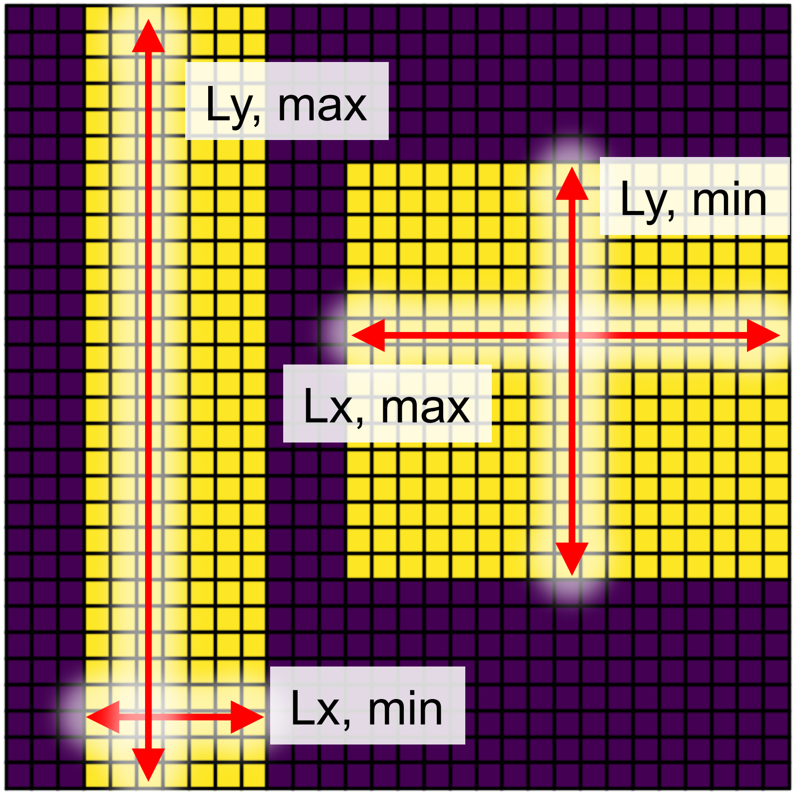
Effective Modulus: Microstructure descriptors





- **Solution:** Introduce further descriptors for filler arrangement
- Here: Min and max of x- and y-dimension of all fillers

Data generation

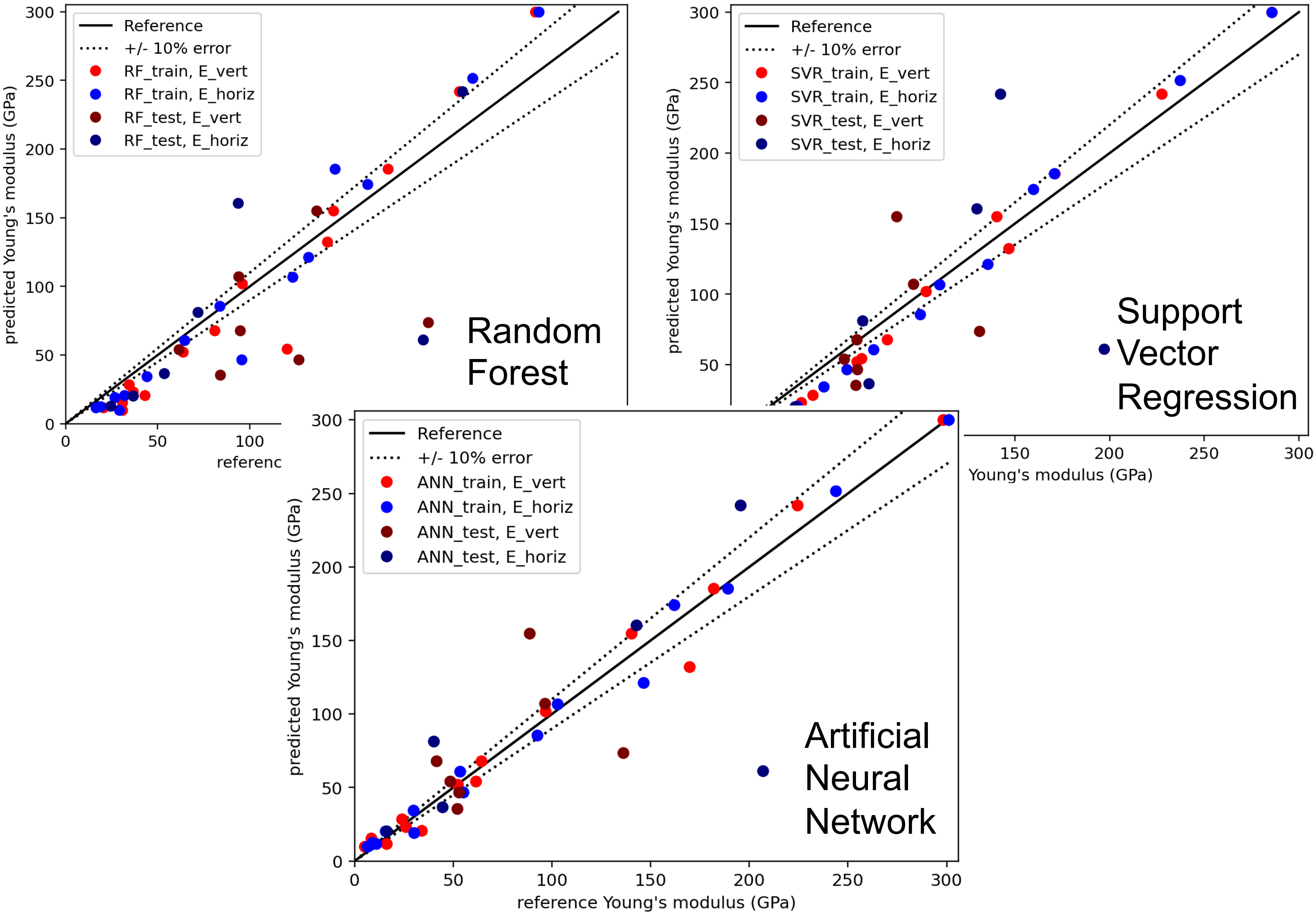


JSON file:

▼ root
▼ Phase_Prop
E1 10
E2 300
▼ VF [] 2 items
0 0.41333333333333333
1 0.5355555555555556
▼ SH_x0 [] 2 items
0 0.2
1 0.23333333333333334
▼ SH_x1 [] 2 items
0 0.5333333333333333
1 0.5666666666666667
▼ SH_y0 [] 2 items
0 0.4
1 0.5333333333333333
▼ SH_y1 [] 2 items
0 1
1 1
▼ Corners [] 18 items
0 8
1 0.1
2 0.3
3 0
4 1
5 0.43333333333333335
6 0.9666666666666667
7 0.36666666666666664
8 0.7666666666666667
9 8
10 0.1
11 0.3333333333333333
12 0
13 1
14 0.43333333333333335
15 1
16 0.26666666666666666
17 0.8
▼ E_vert [] 2 items
0 72.1003823640159
1 84.82489295704768
▼ E_hor [] 2 items
0 25.607091170525926
1 36.9686678620007

Dataset should always contain complete information about microstructure, model parameters, descriptors and results.

Machine learning analysis



Summary

- Machine learning functions map a domain (features) onto results (labels)
- Feature design is important to describe system completely and uniquely
- Choice of proper machine learning model strongly influences the quality