Artificial Intelligence and Machine Learning – Application in Computer Aided Engineering

Thomas Most
Jonas Rotermund
Johannes Will
Lars Gräning

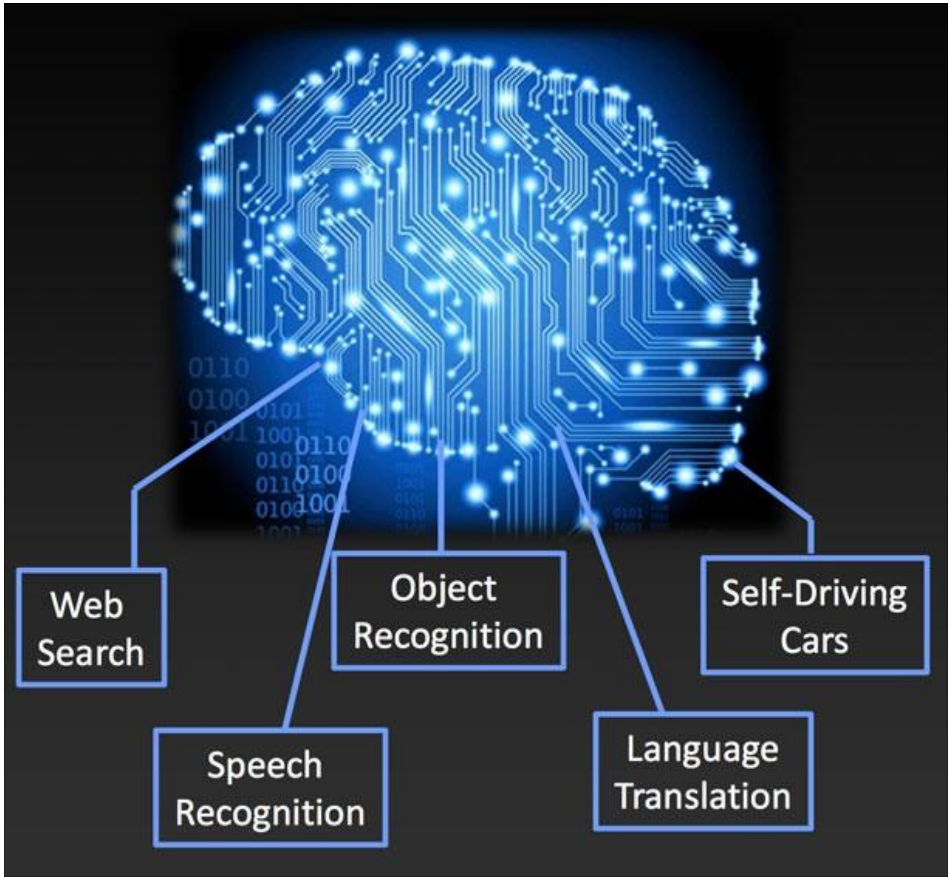
Dynardo GmbH

16. Weimarer Optimierungs- & Stochastiktage 2019,
06.-07. Juni 2019, Weimar

Artificial Intelligence

In computer science, artificial intelligence (AI), sometimes called machine intelligence, is intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans and animals. Colloquially, the term "artificial intelligence" is used to describe machines that mimic "cognitive" functions that humans associate with other human minds, such as "learning" and "problem solving‟

Russell & Norvig (2009). Artificial Intelligence: A Modern Approach. Prentice Hall

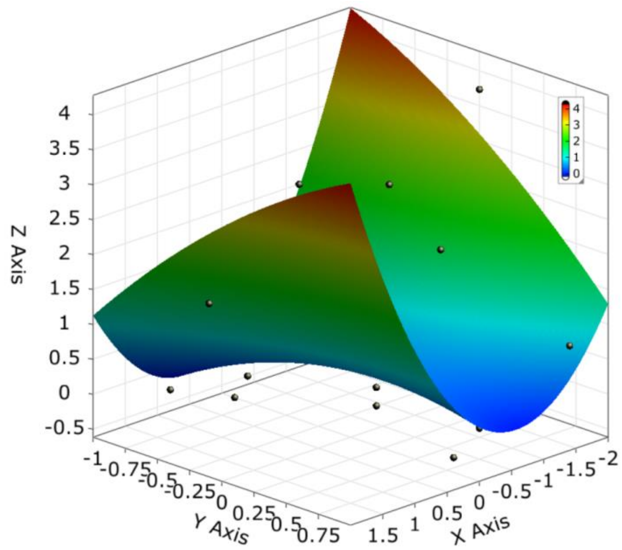


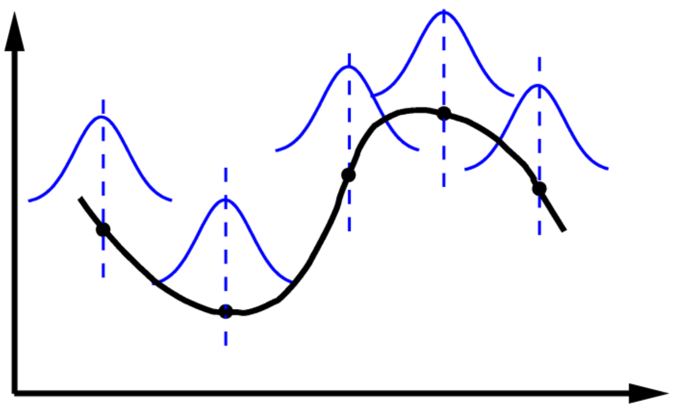
- Pattern recognition
- Nature Inspired Optimization
- Machine learning

Machine Learning

Response Surface Method

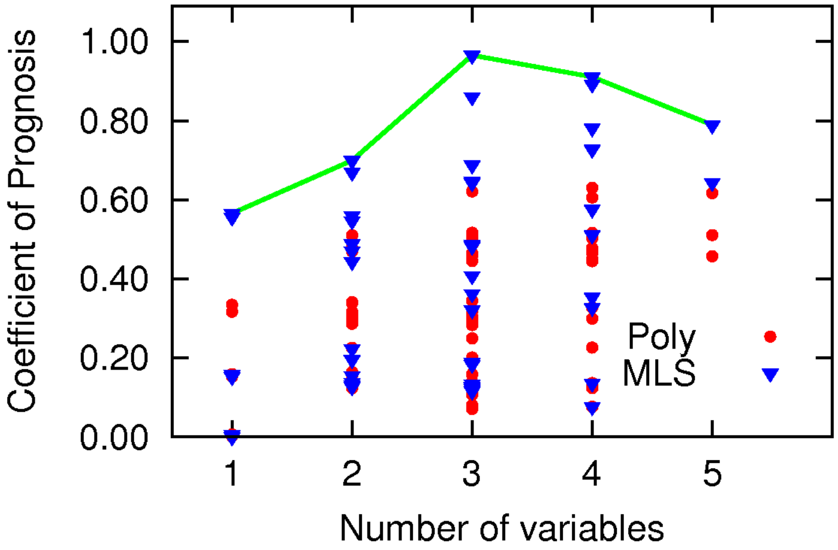
- Approximation of response variables as explicit function of all input variables
- Approximation function can be used for sensitivity analysis
- Global methods (**Polynomial regression, Neural Networks**, …)
- Local methods (Spline interpolation, **Moving Least Squares**, Radial Basis Functions, **Kriging**, …)
- **Approximation quality decreases with increasing input dimension!**
- Successful application requires **objective** measures of the **prognosis quality!**

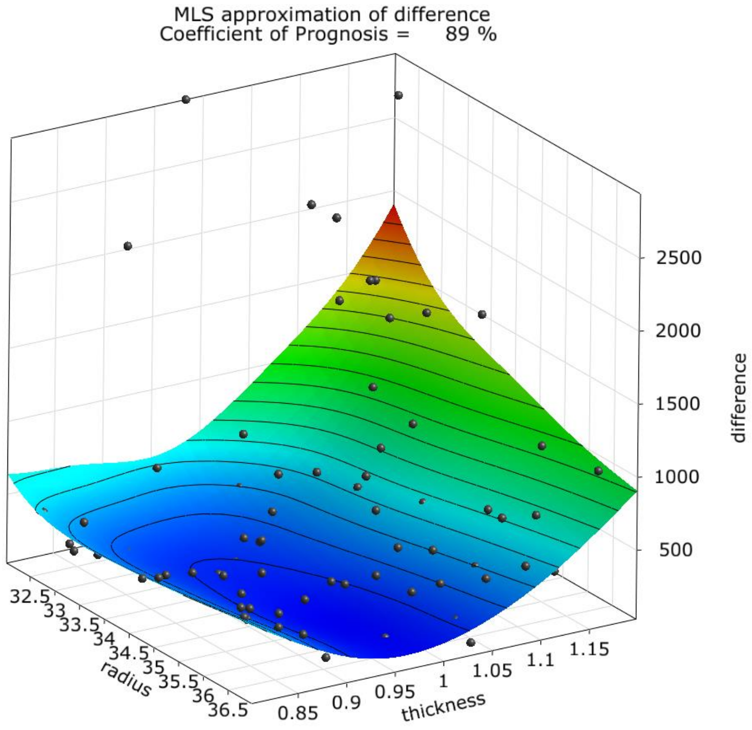




Metamodel of Optimal Prognosis (MOP)

- Objective measure of **prognosis quality**
- Determination of **relevant parameter subspace**
- Determination of **optimal approximation model**
- Approximation of solver output by fast surrogate model **without over-fitting**
- Evaluation of **variable sensitivities**

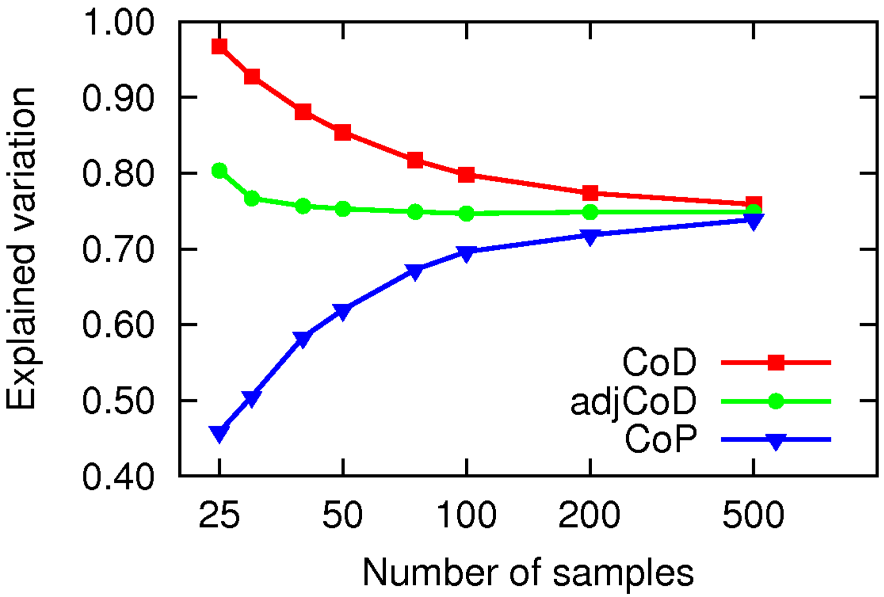




Coefficient of Prognosis (CoP)

- The CoP measures how good the regression generalizes for unknown data points
- **Fraction of explained variation of the prediction of a response**

$$CoP = 1 - \frac{SS_E^{Prediction}}{SS_T}$$

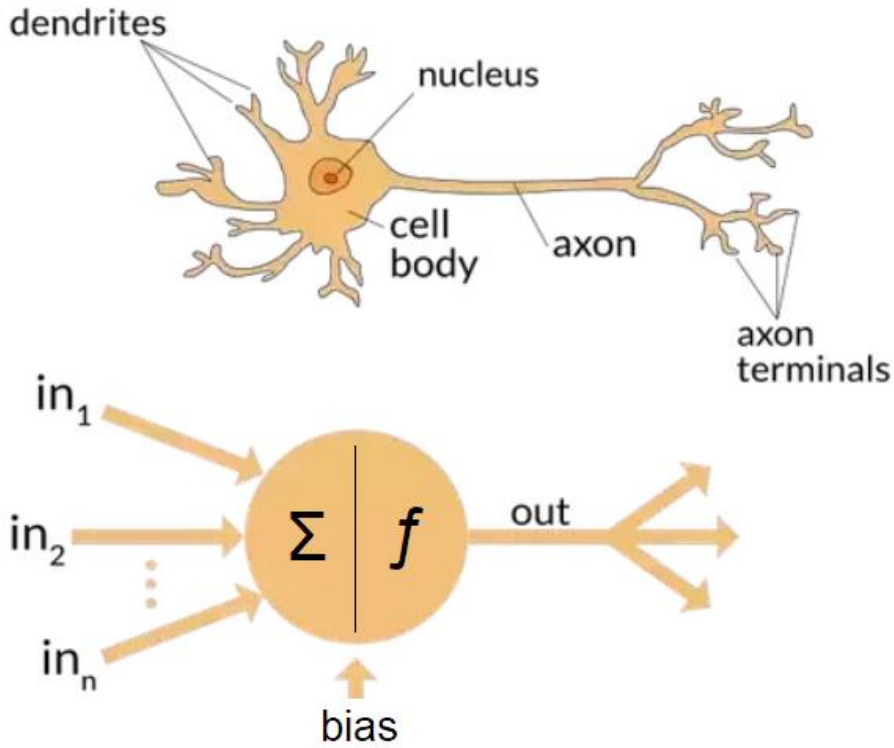


- CoP increases with increasing number of samples
- ➢ It assesses the approximation quality much more reliable than the CoD
- CoP is model-independent and suitable for other meta-models

- Estimation of CoP by additional (new) data set causes additional effort
- ➢ Cross validation using a partitioning of the available samples is applied

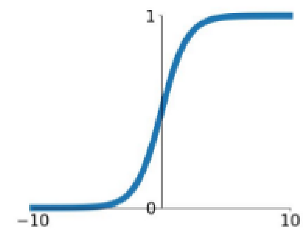
Artificial Neural Networks

Artificial Neurons – Activation Functions



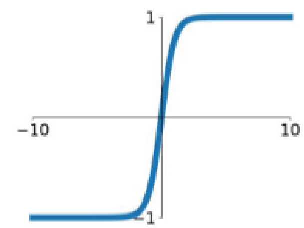
Sigmoid

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



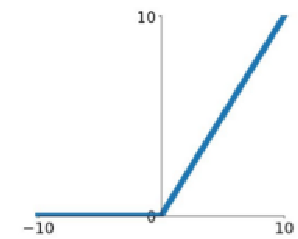
tanh

$$\tanh(x)$$



ReLU

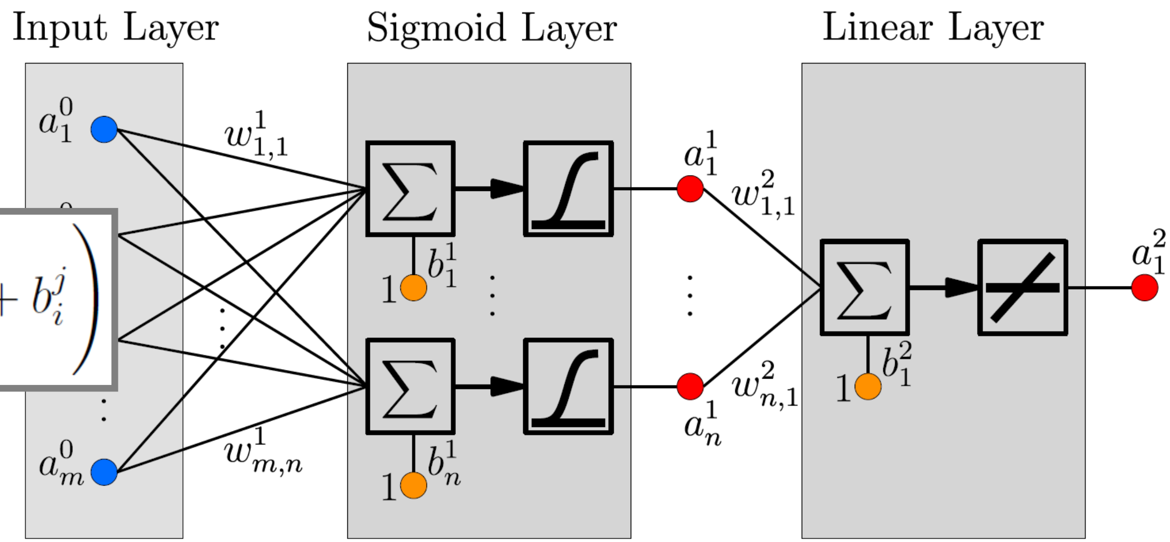
$$\max(0, x)$$



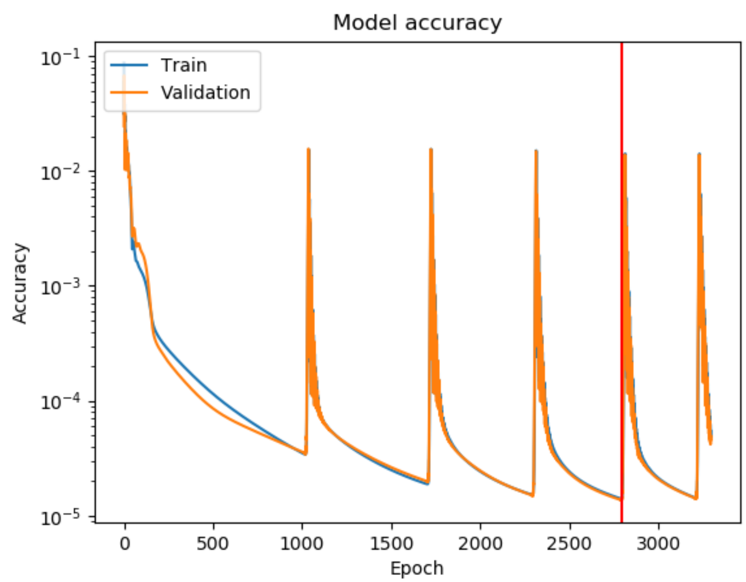
- Neurons operate linearly or nonlinearly on weighted sum of inputs

Feedforward Network

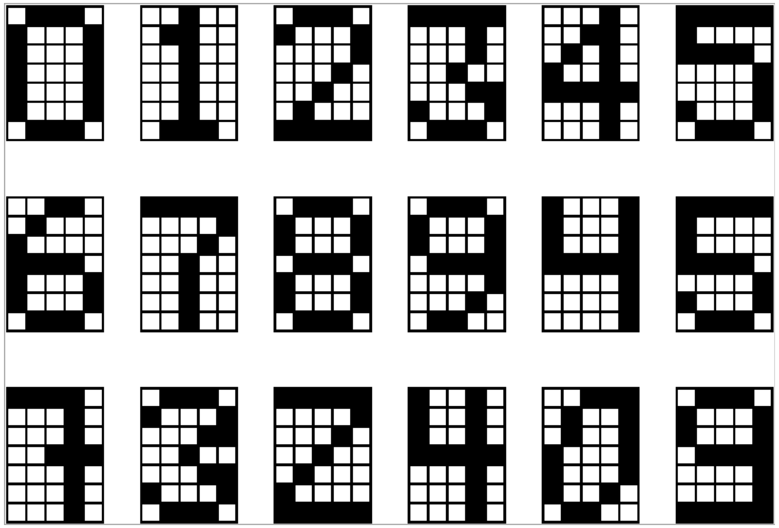
$$a_i^j = f_i^j(x) = f\left(\sum_{k=1}^{m} w_{k,i}^j a_k^{j-1} + b_i^j\right)$$



- Output and hidden layer consist of linear or nonlinear neurons
- Feedforward network with only linear neurons is a linear regression!
- Training by minimizing the sum of squared errors in a training data set
- Early stopping to avoid overfitting

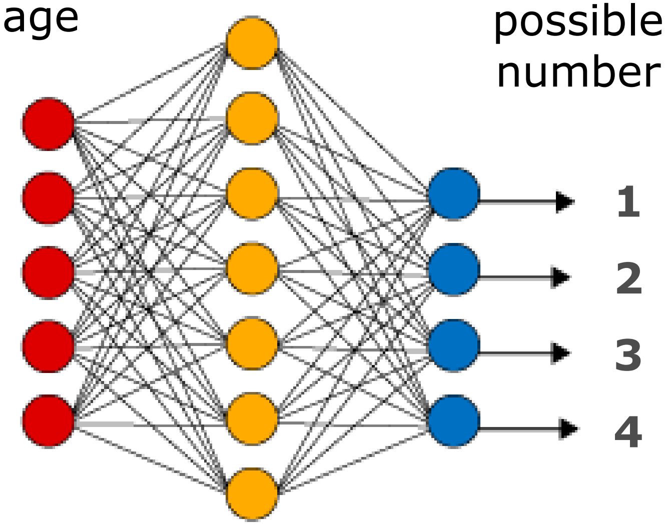


Feedforward Network as Classifier



Input:
Each pixel
in image

Output:
Each
possible
number



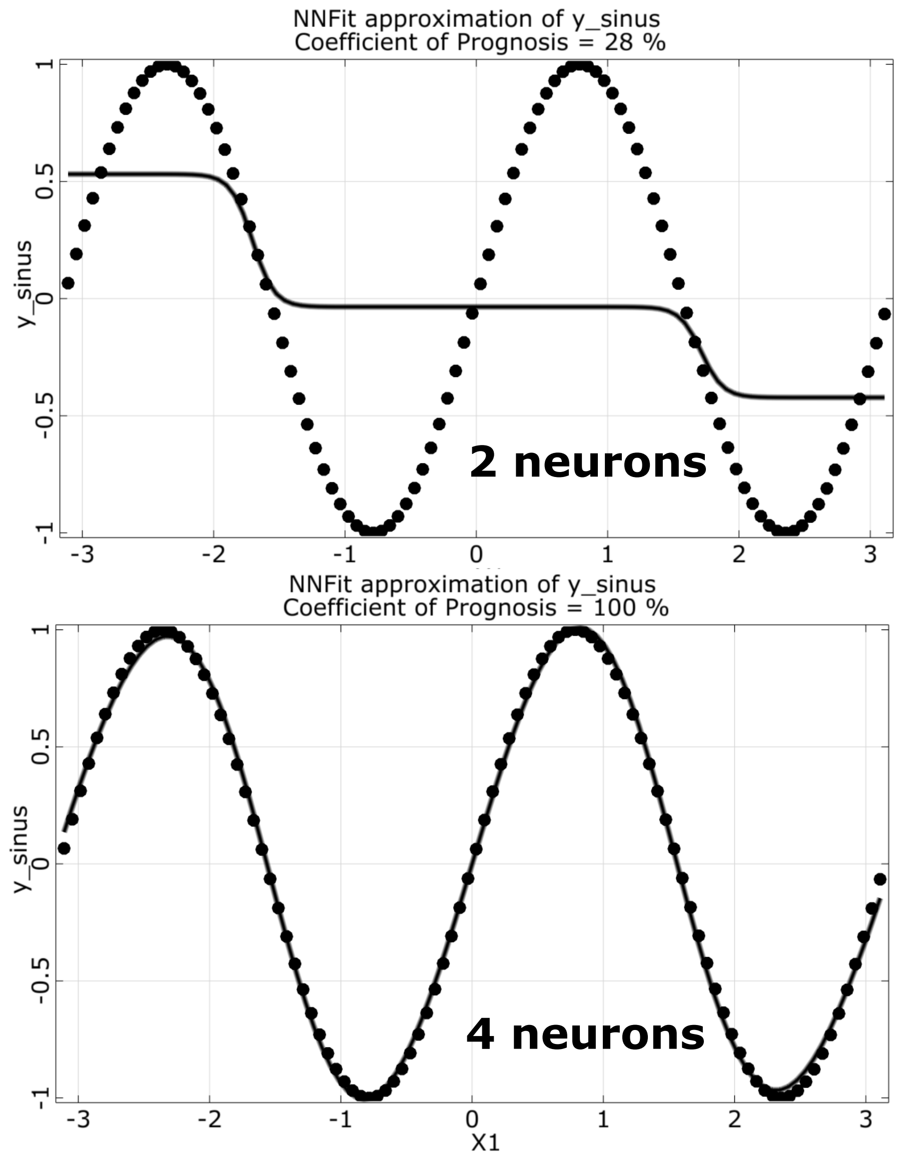
- Mainly used in image/pattern recognition
- Can be applied for each discrete output
- Each output state is one output neuron
- Output classification privides probability!
- ➢ Can handle nominal discrete inputs & outputs

Feedforward Network as Regression Model

- Network output is a global approximation function similar to polynomials
- ➢ Very fast output approximation (after model was trained)

- Higher flexibility with larger number of neurons and layers
- ➢ Risk of overfitting increases

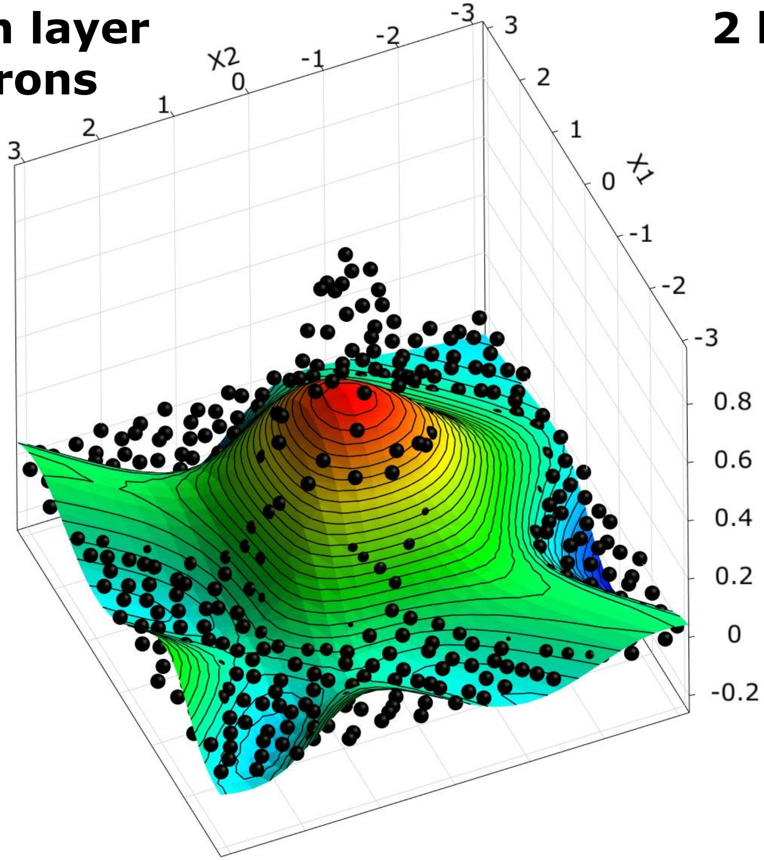
- ➢ Best compromise between available data and model complexity is needed



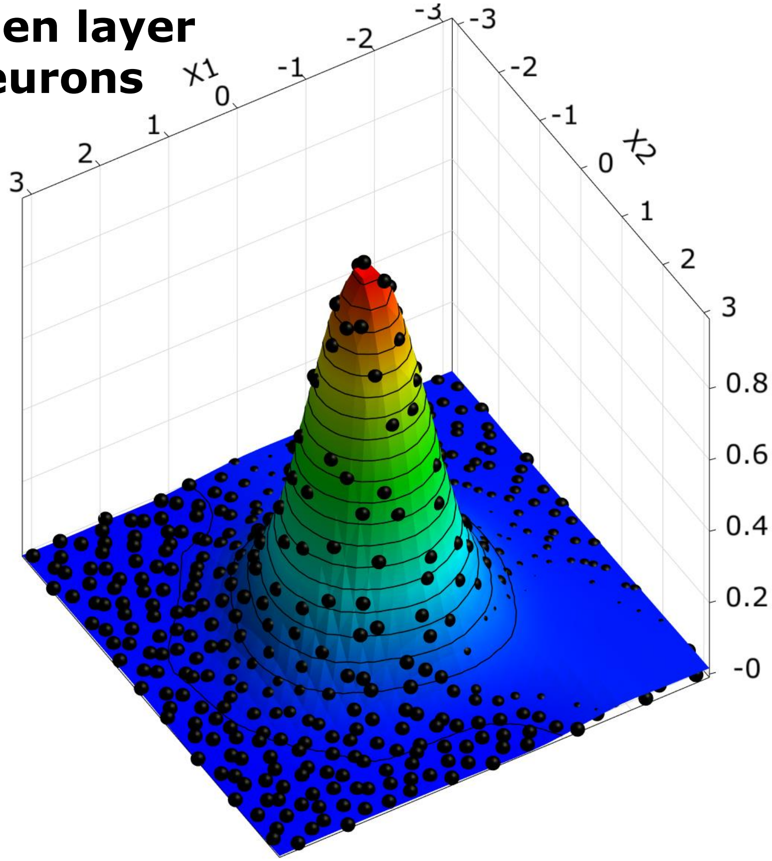
Feedforward Network as Regression Model

- Interactions can be represented better with more hidden layers
- ➢ Deep network architecture may improve regression

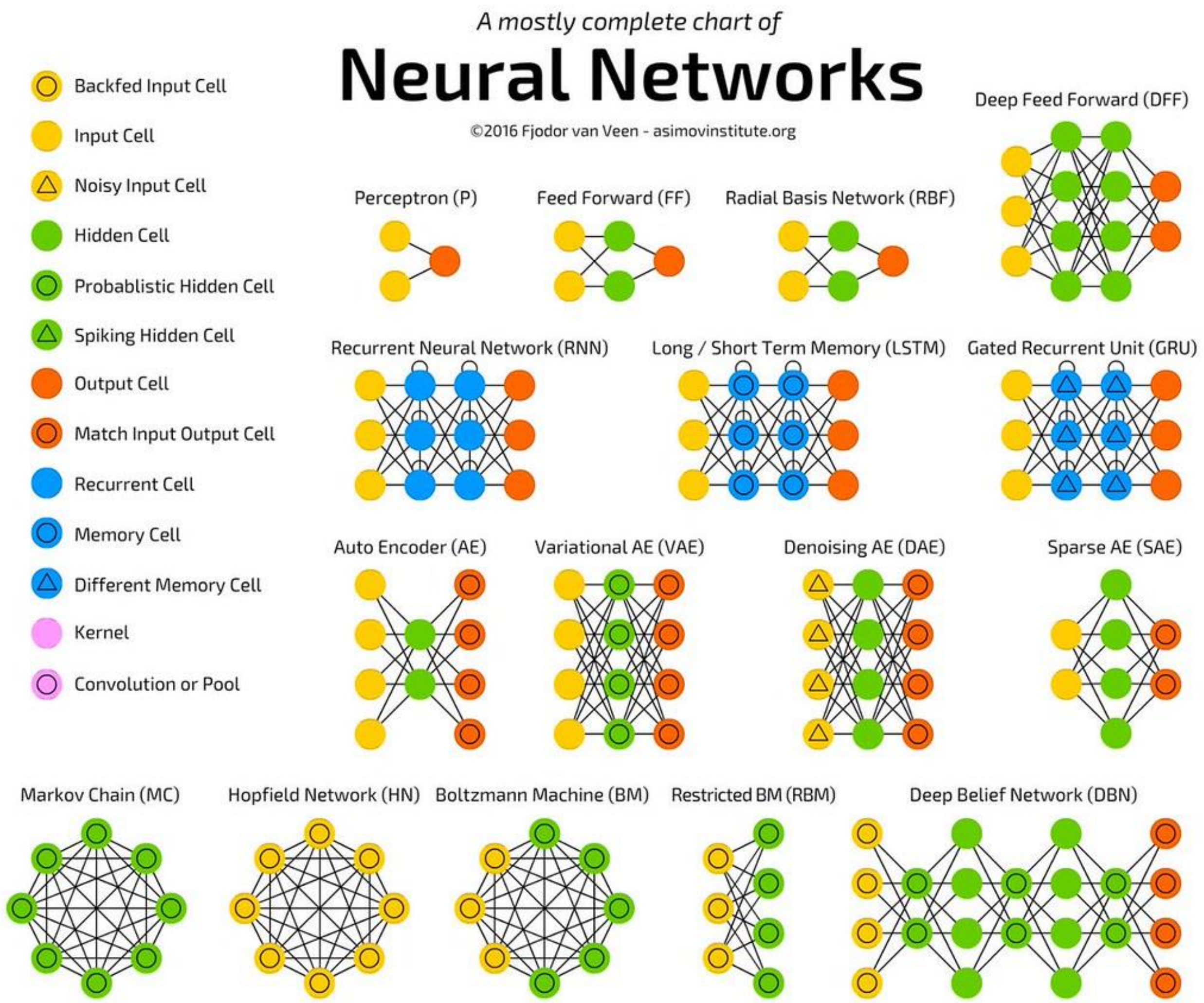
**1 hidden layer
4 neurons**



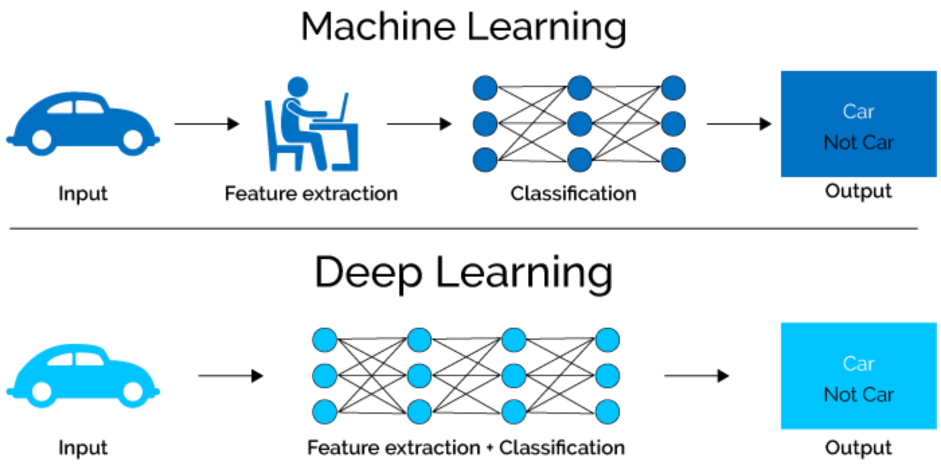
**2 hidden layer
4 neurons**

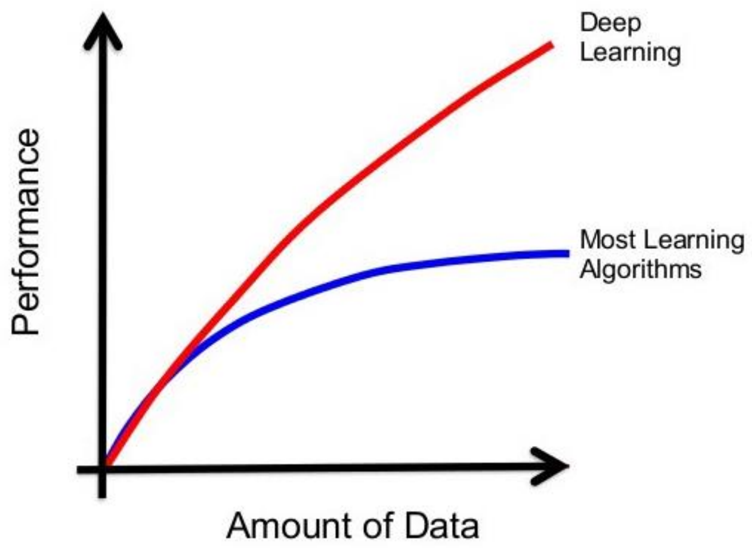


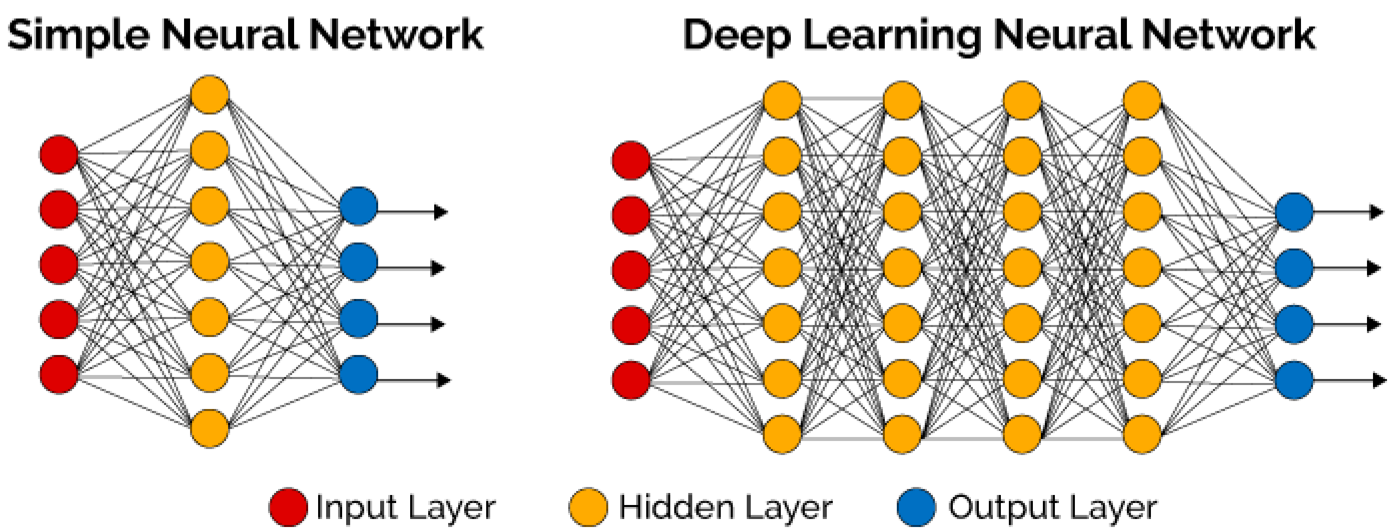
Network Architectures



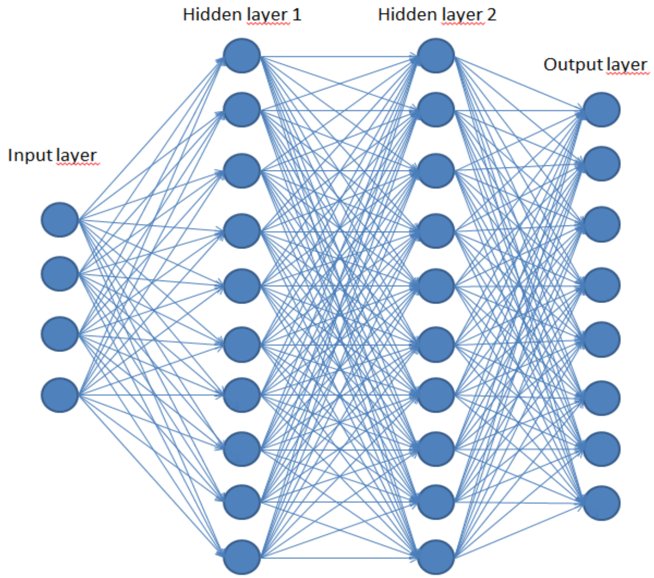
Deep Learning







Keras & Tensorflow Libraries



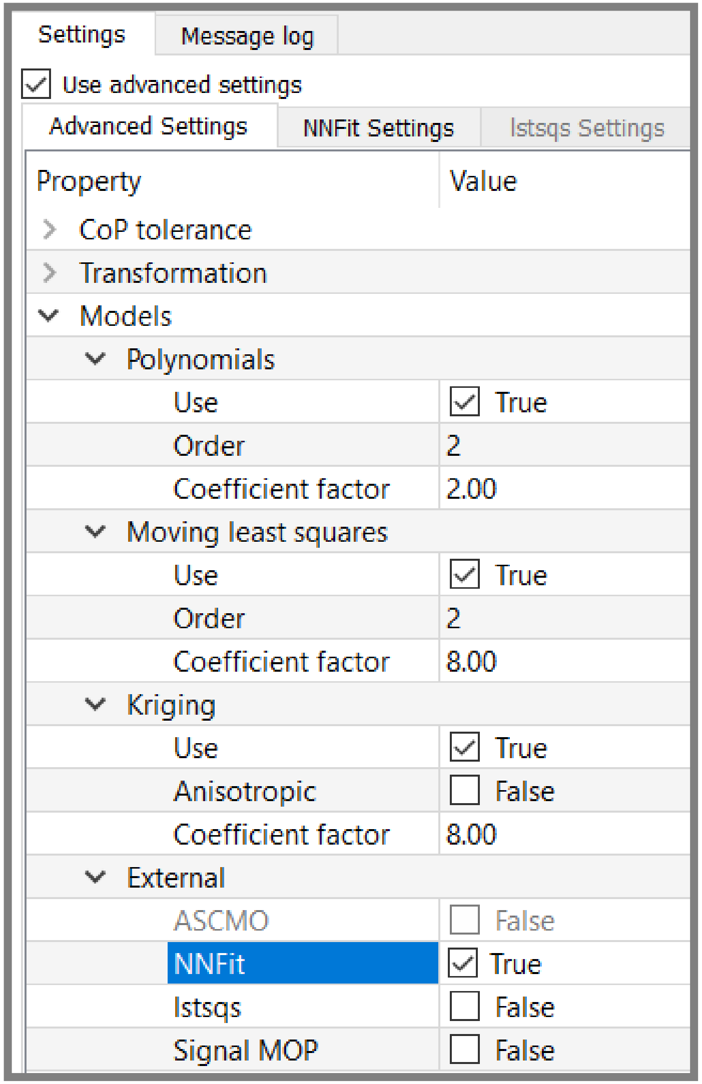
Implementation in custom surrogate interface of optiSLang:

- ✓ **Automatic configuration** of **neurons** and **layers**
- ✓ Cross validation to estimate **Coefficient of Prognosis**
- ✓ Available as **external python** environment
- ✓ Neural networks are treated as one of a library of approximation models
- ✓ Competition is done in the MOP framework based on the CoP

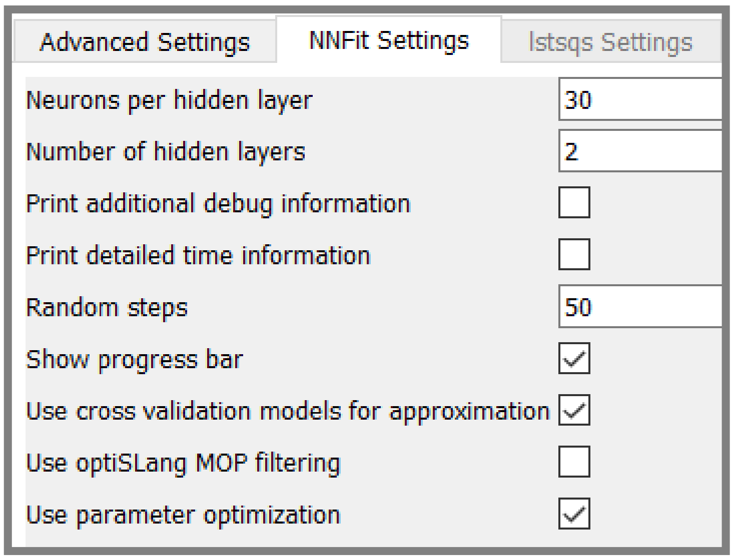
MOP with Custom Surrogates

Python Interface with defined API

- Initialization of model and data
- Automatic build of the surrogate model
- Considering MOP subspaces (optional)
- Return cross validation estimate
- Approximation call

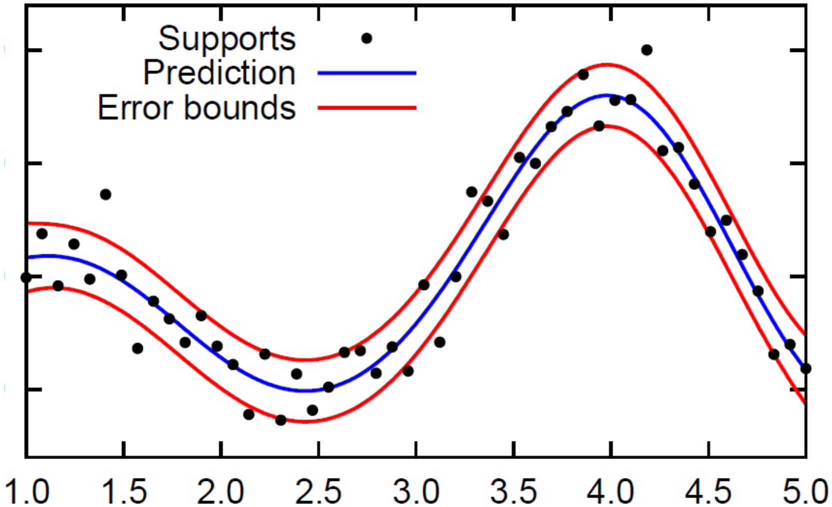


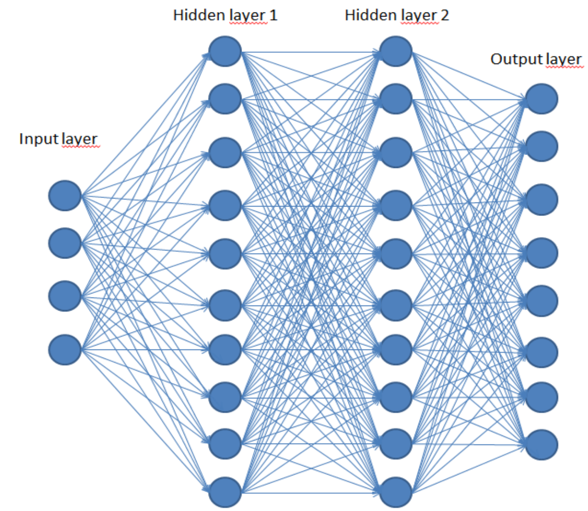
- Custom settings:



Deep Gaussian Covariance Network (DGCN)

Gaussian Process (Kriging) + ANN



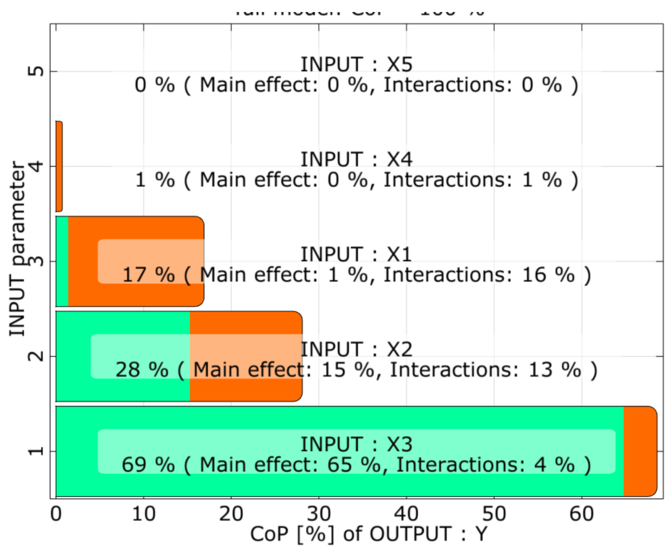


- Kriging basis functions
- Local covariance matrix approximated with neural network
- ➢ More flexibility to represent local effects

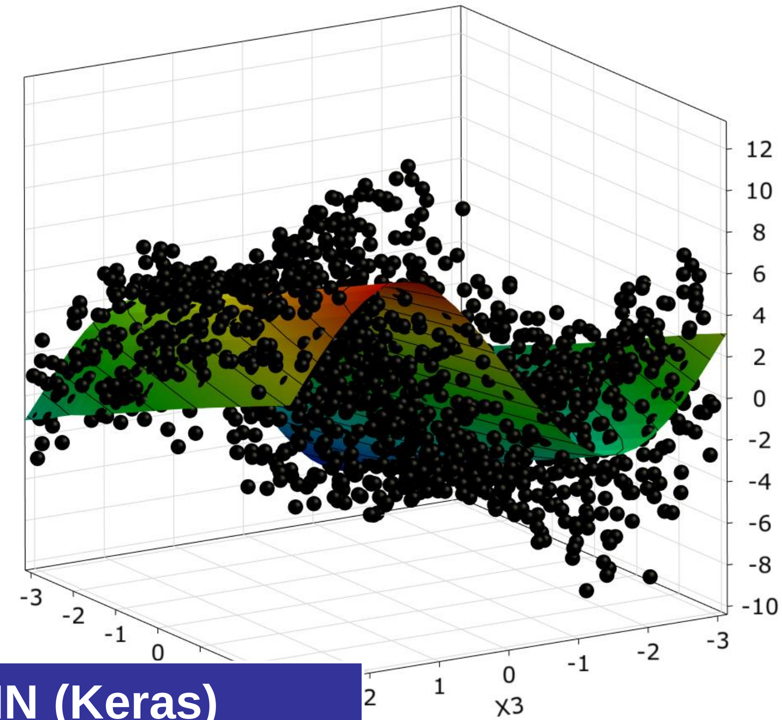
- Kevin Cremanns & Dirk Roos, PI Probaligence GmbH

Examples

Analytical Test Function



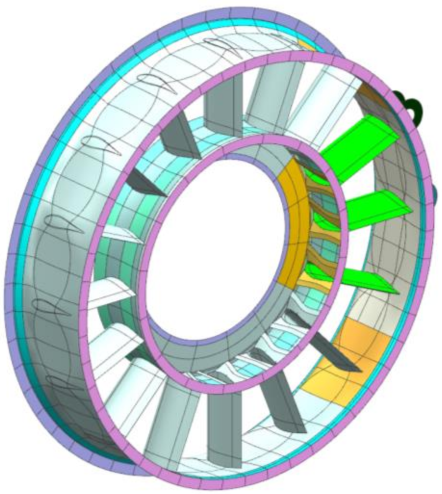
5 inputs,
3 important



Samples	Kriging		ANN (Keras)	
	Training	CoP	Training	CoP
100	2 sec	99.99 %	4,5 min	99.86 %
500	3 min	99.99 %	8 min	99.93 %
2000	330 min	99.99 %	14 min	99.94 %
10000	?		68 min	99.96 %

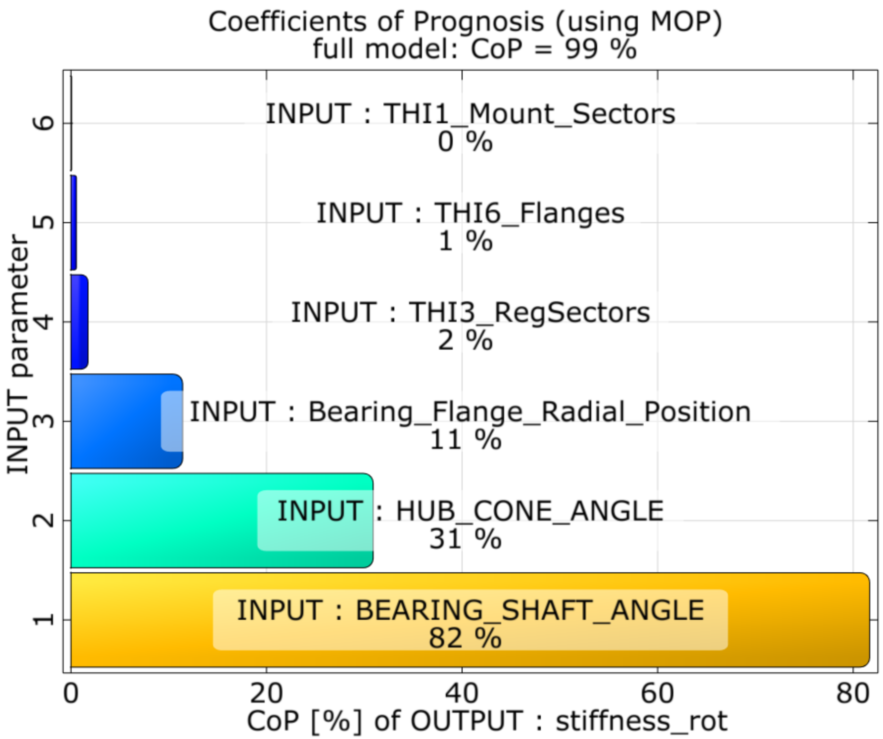
Turbine Data

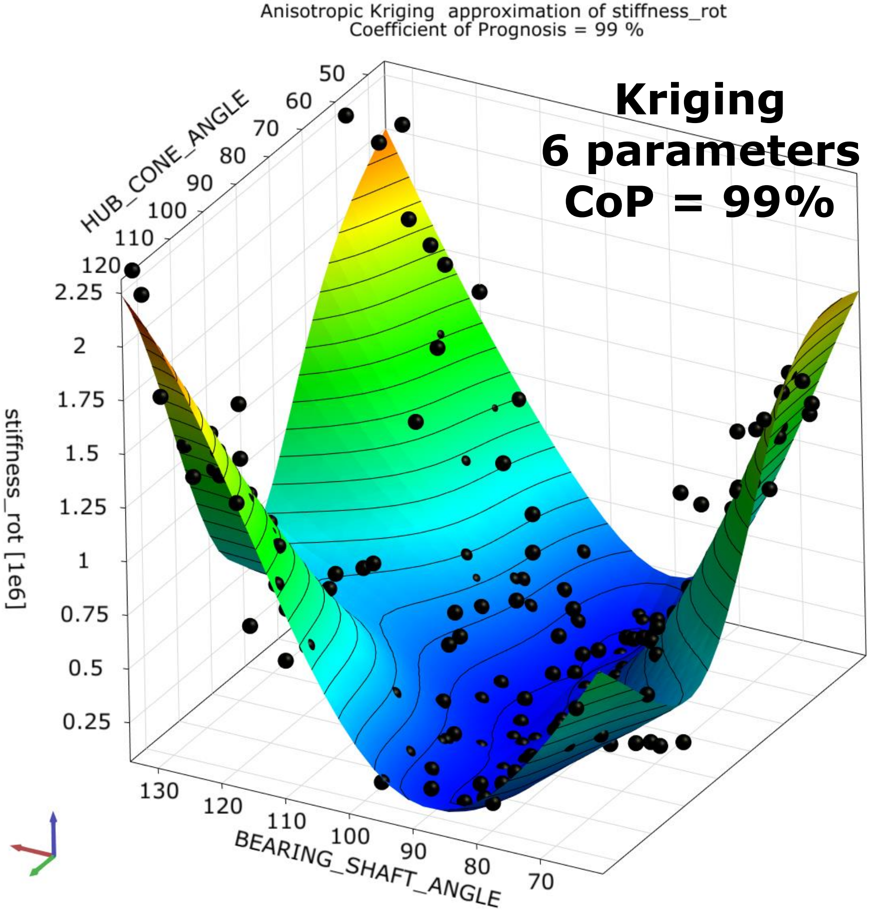
	BEARING_SHAFT_ANGLE	Bearing_Flange_Radial_Position	HUB_CONE_ANGLE	EM_Ridge_Radial_Deviation	OC_TYPE	EM_Tangential_Offset	Tmax	THI1_Mount_Sectors	THI2RadialFlange	THI3_RegSectors	THI4_HubSurface	THI5_HubStiffnessRing	THI6_Flanges	Total
log_min_life_all	10.5 %	2.9 %	30.5 %	2.0 %	17.0 %		3.9 %			9.8 %	9.1 %		1.2 %	61.4 %
log_min_life_vane		7.6 %	32.6 %		33.2 %	0.7 %	0.2 %	0.0 %	0.6 %		4.8 %			72.9 %
min_life_all	0.0 %	7.8 %	24.6 %	0.0 %	30.6 %	0.0 %	0.9 %	1.1 %	0.0 %	15.0 %	8.7 %	0.0 %	0.7 %	58.0 %
min_life_vane		5.7 %	30.4 %		29.7 %			8.8 %		0.5 %	5.1 %			65.0 %
pressure_loss							100.0 %							100.0 %
fan_blade_out			0.0 %	6.4 %	0.6 %	34.3 %		63.5 %	0.0 %	0.0 %				96.8 %
stiffness_axial	58.6 %	7.0 %	35.2 %		16.9 %	4.8 %	0.1 %	5.3 %	0.0 %	9.4 %			0.1 %	90.7 %
stiffness_rot	81.7 %	11.4 %	30.9 %					0.1 %		1.7 %			0.6 %	98.7 %
mass_total		3.7 %	0.1 %	1.1 %		0.3 %		14.2 %	2.6 %	74.5 %	0.3 %	0.1 %	3.3 %	99.8 %



- Mass, stiffness, pressure loss and life time are given with respect to geometry parameters
- 13 parameters are considered, 176 data sets were available

Turbine Data – Rotational Stiffness



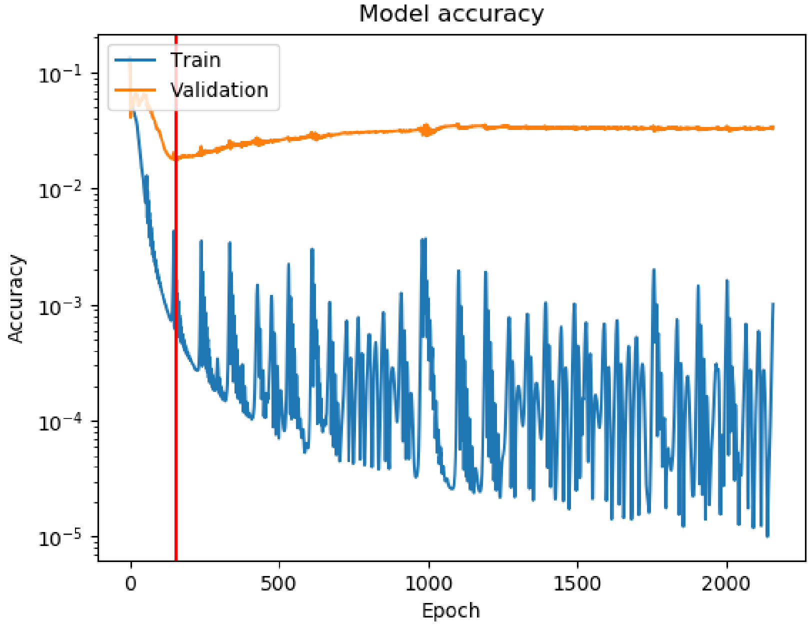


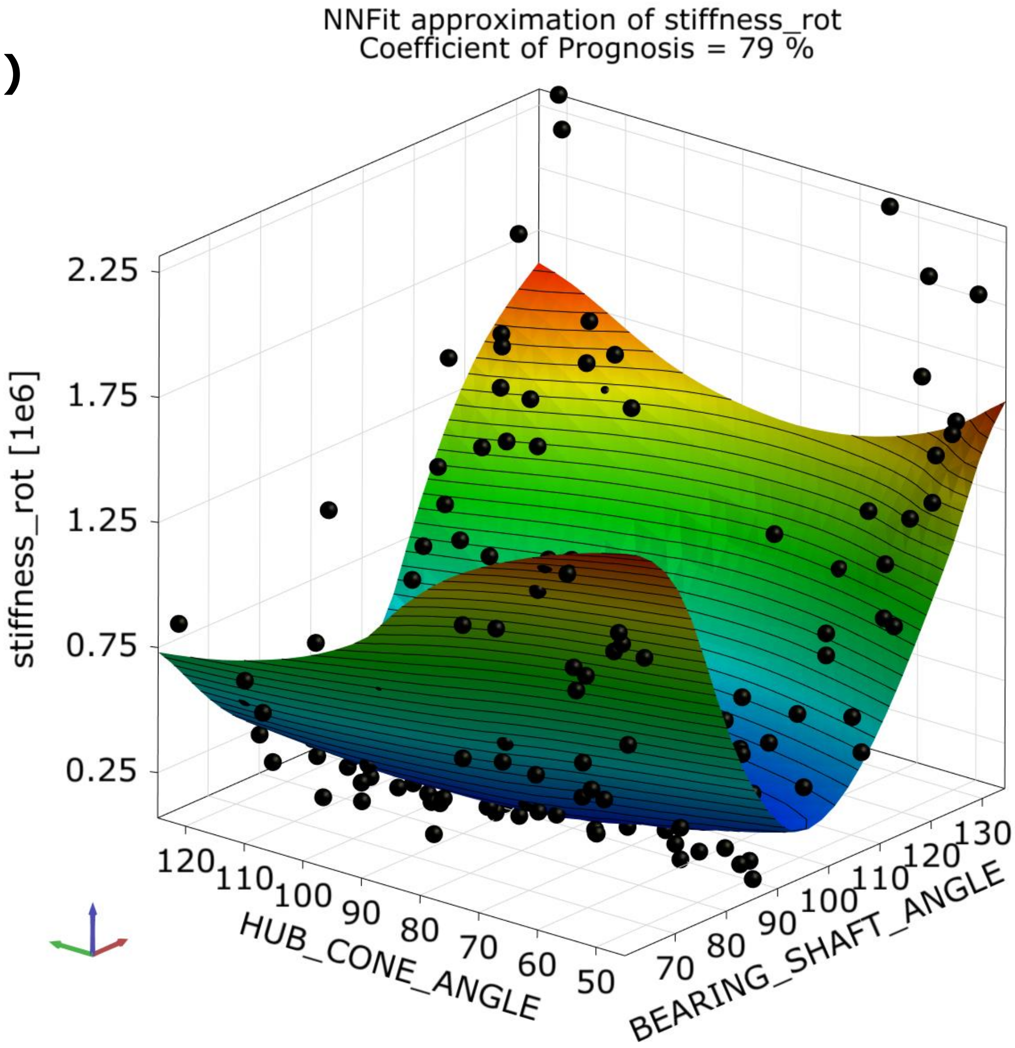
- Anisotropic kriging (Gaussian process) is optimal approximation model type

Turbine Data – Rotational Stiffness

13 parameters (full space)
3 layer/ 7 neurons
CoD = 98 %
CoP = 79 %

- Significant larger error in validation data indicates overfitting

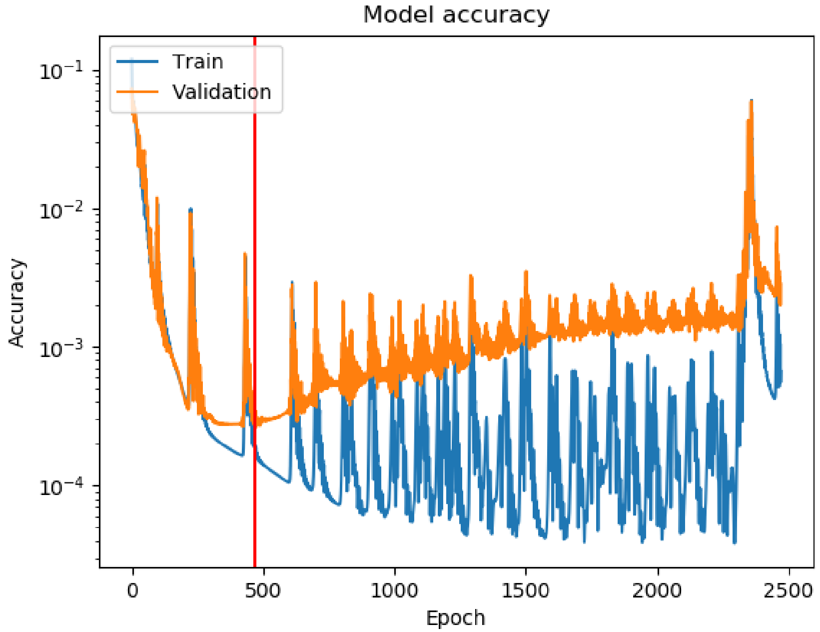


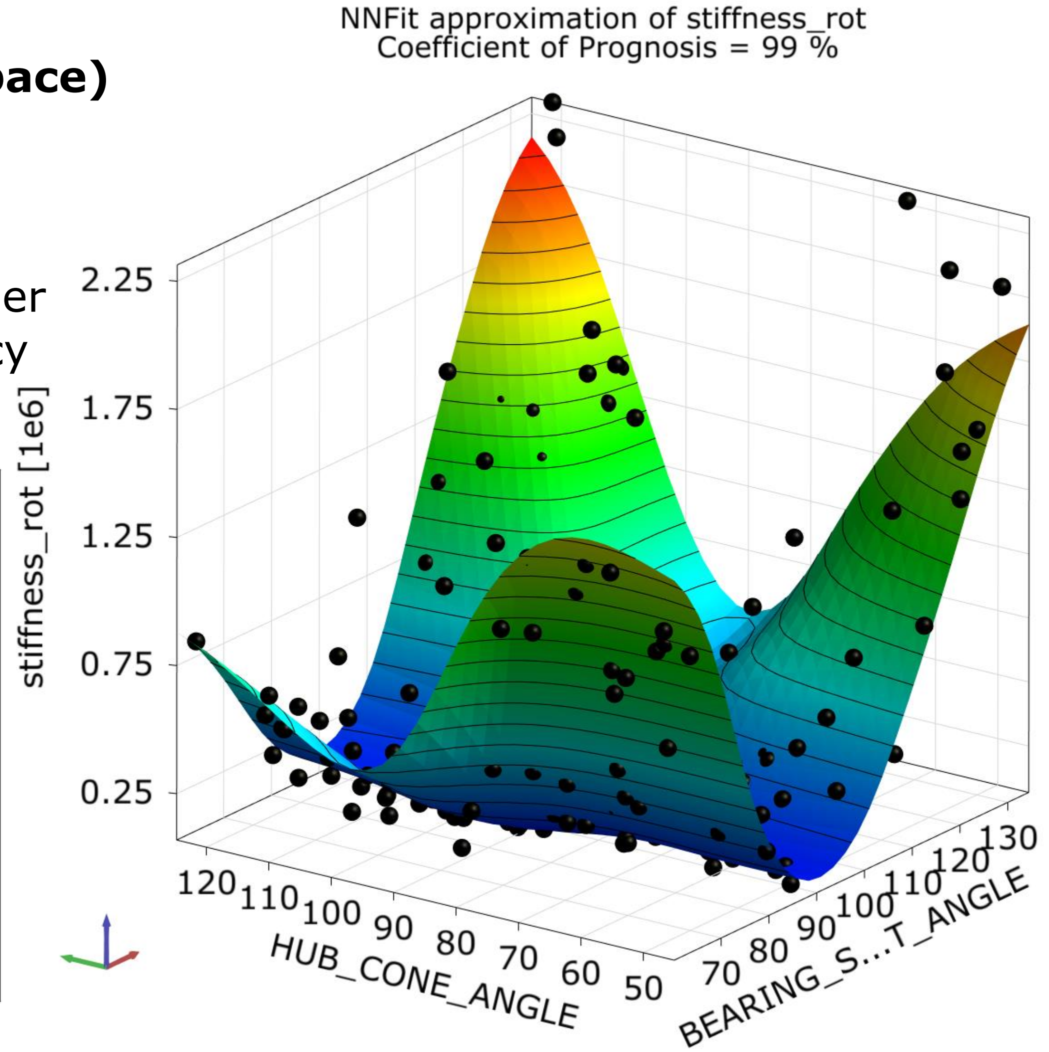


Turbine Data – Rotational Stiffness

6 parameters (optimal subspace)
3 layer/ 7 neurons
CoD = 99 %
CoP = 98 %

- Reduction of parameter number significantly improves accuracy

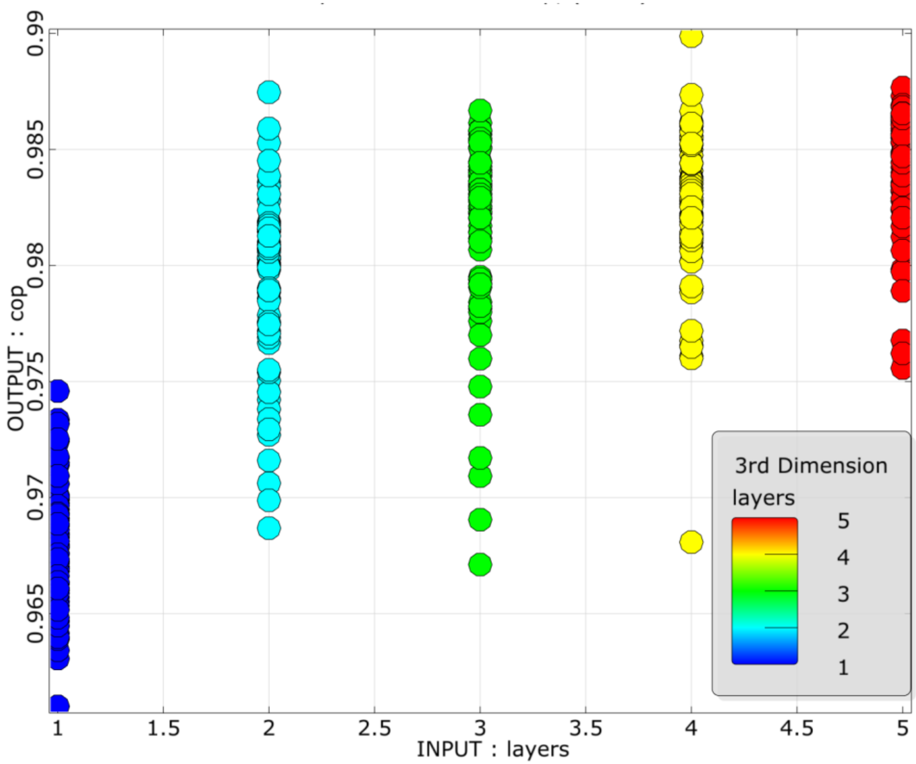


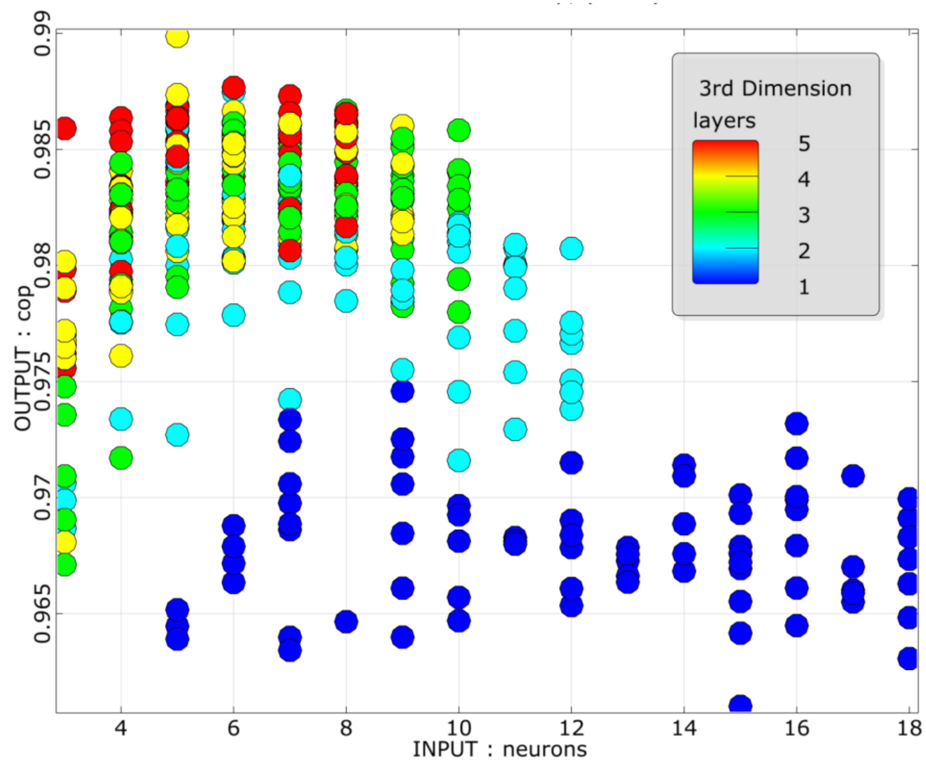


Turbine Data – Rotational Stiffness

Influence of number of neurons and layers

- Two or more layers show similar behavior
- 5 to 7 neurons are most efficient
- Adjustment of optimal network architecture is problem/data dependent

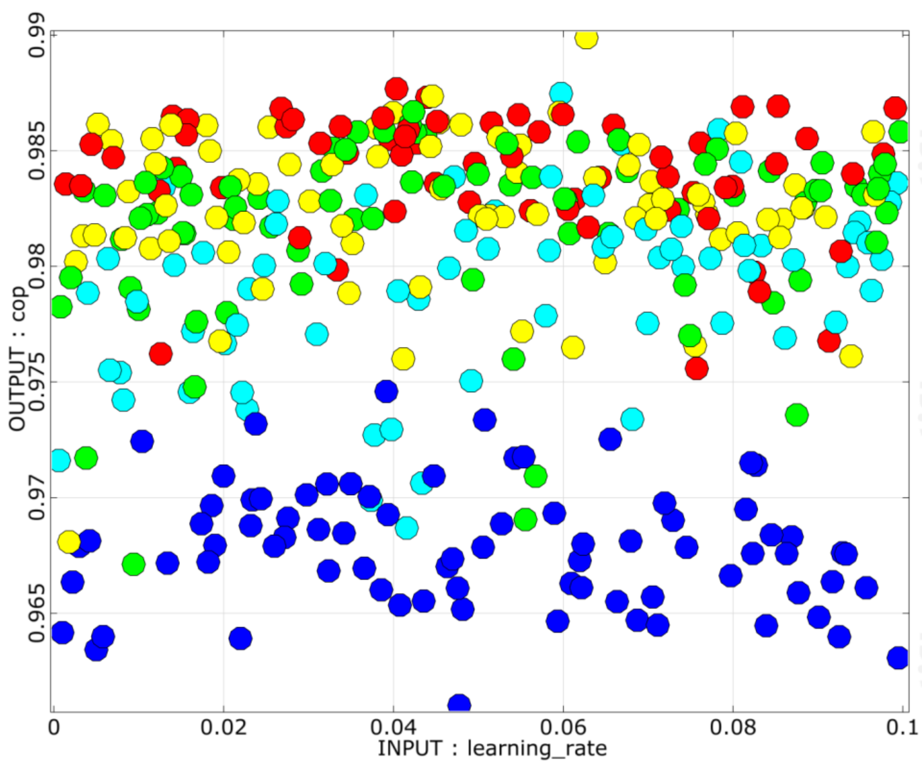


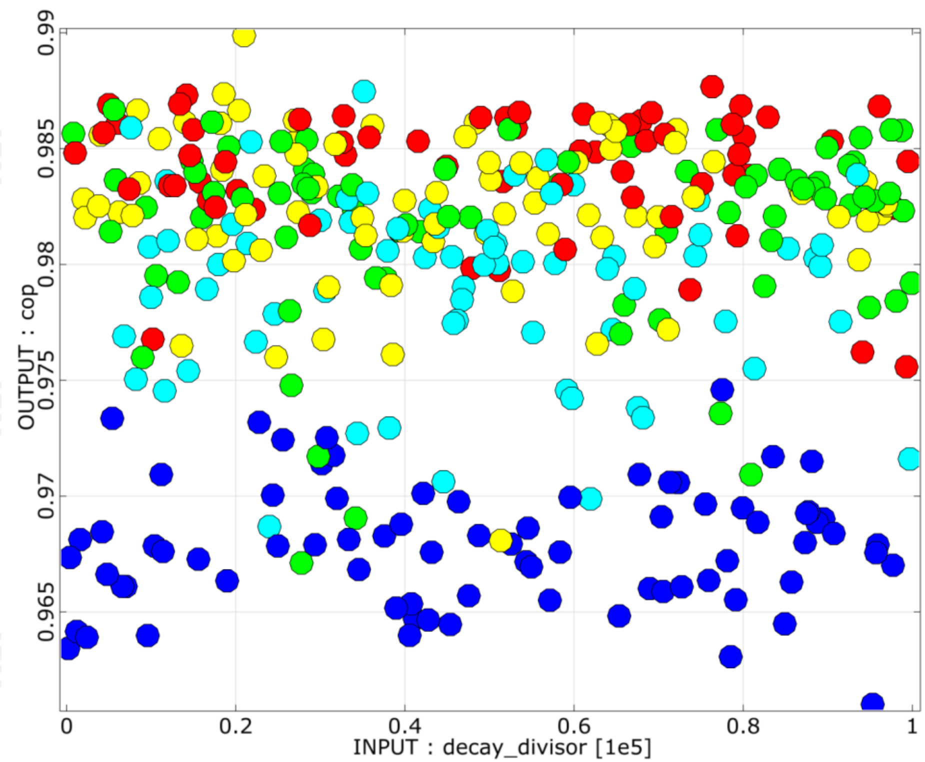


Turbine Data – Rotational Stiffness

Influence of learning rate and decay

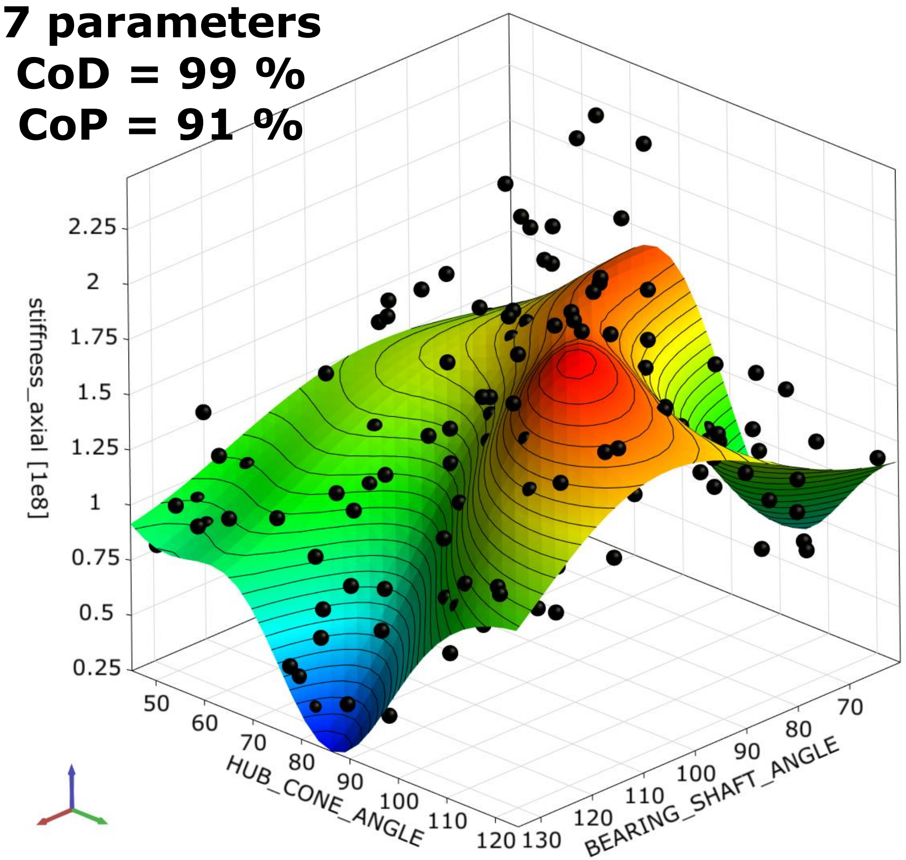
- Almost no influence
- Problem independent set up



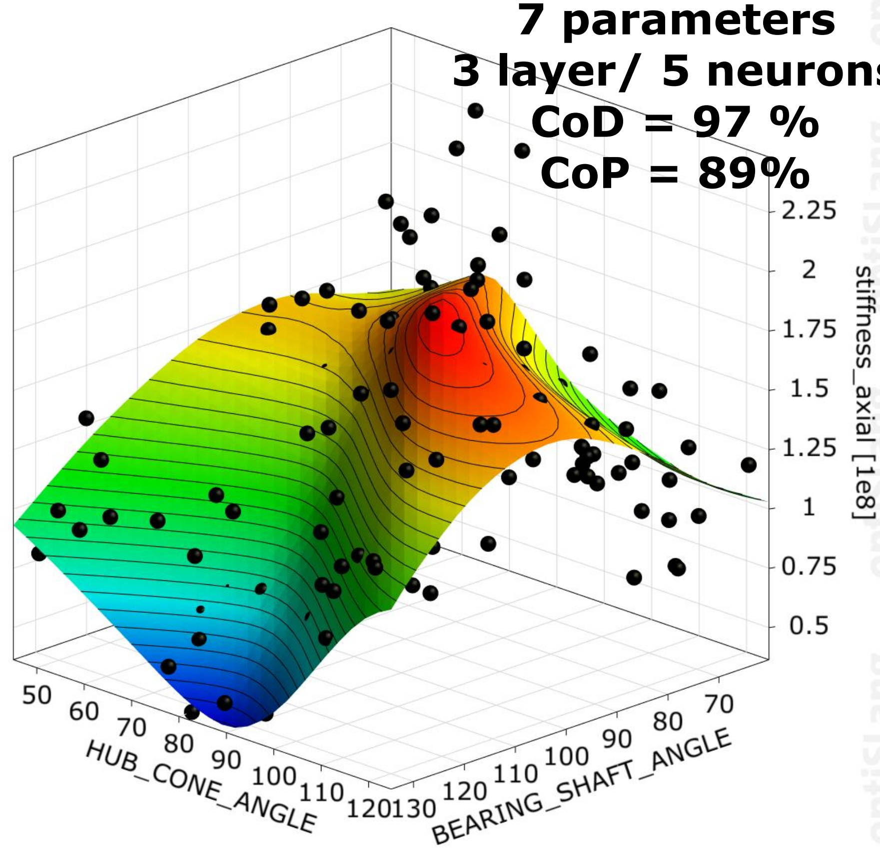


Turbine Data – Axial Stiffness

Kriging
7 parameters
CoD = 99 %
CoP = 91 %



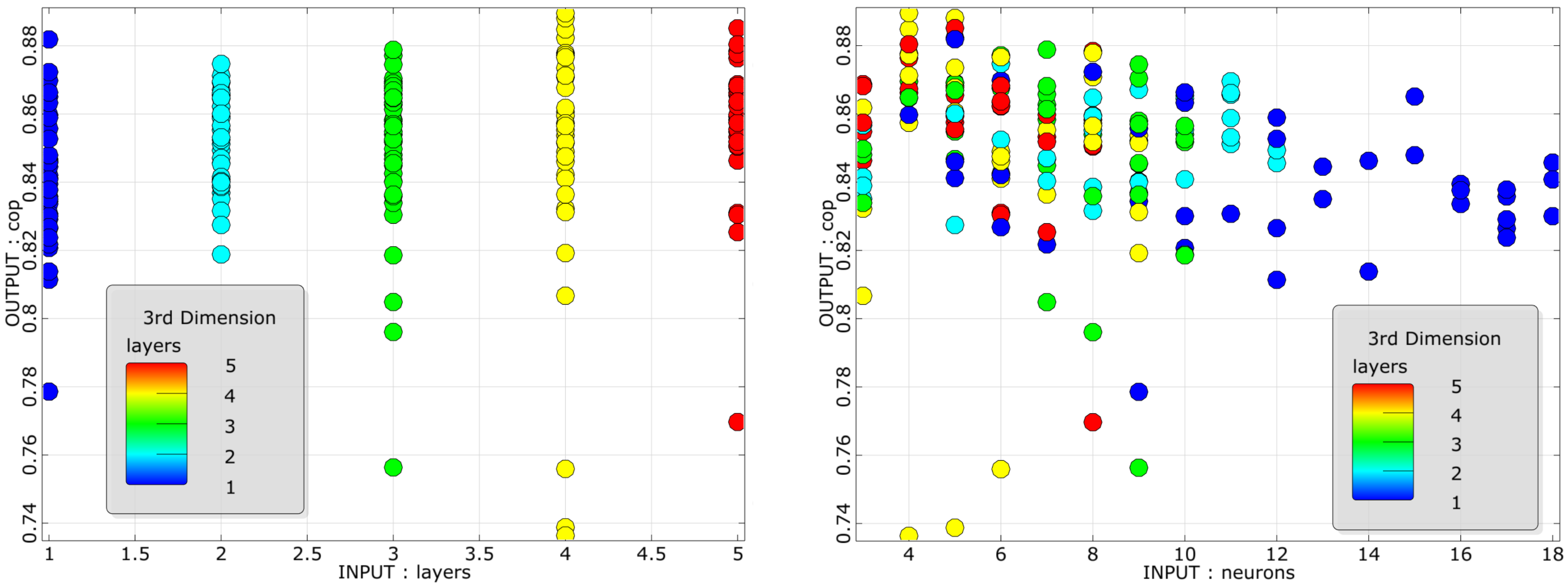
ANN
7 parameters
3 layer/ 5 neurons
CoD = 97 %
CoP = 89%



Turbine Data – Axial Stiffness

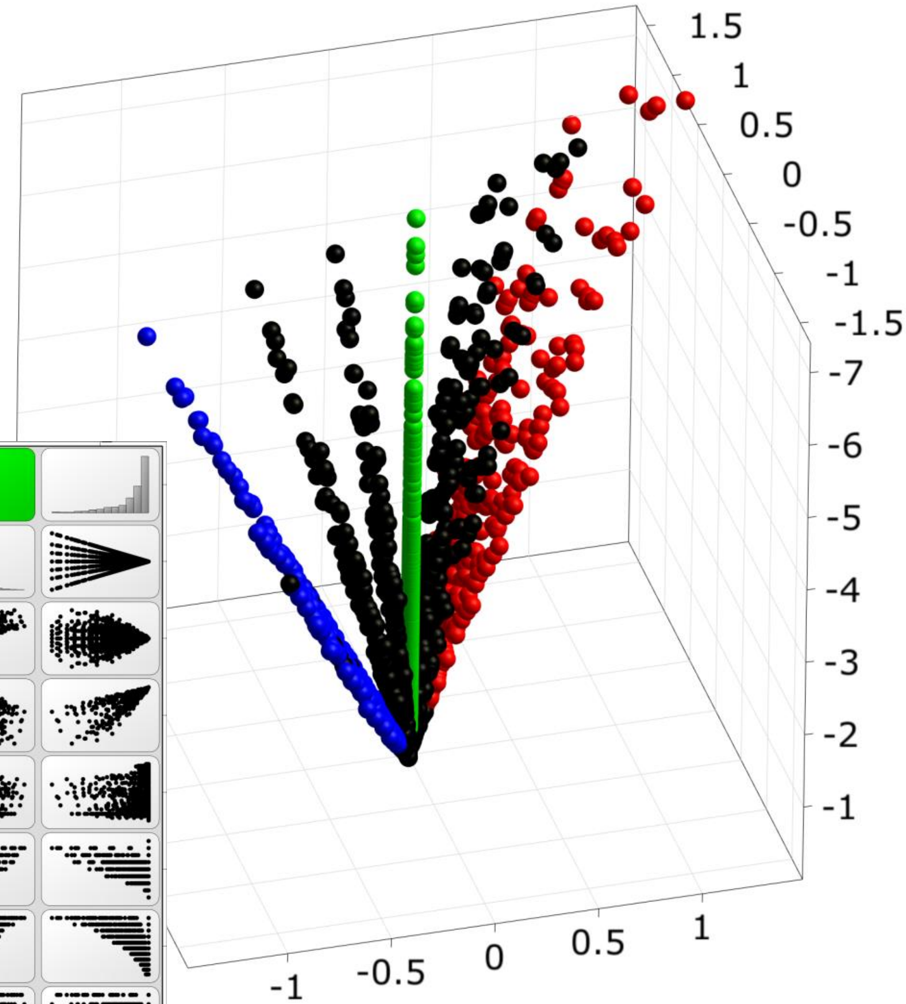
Influence of number of neurons and layers

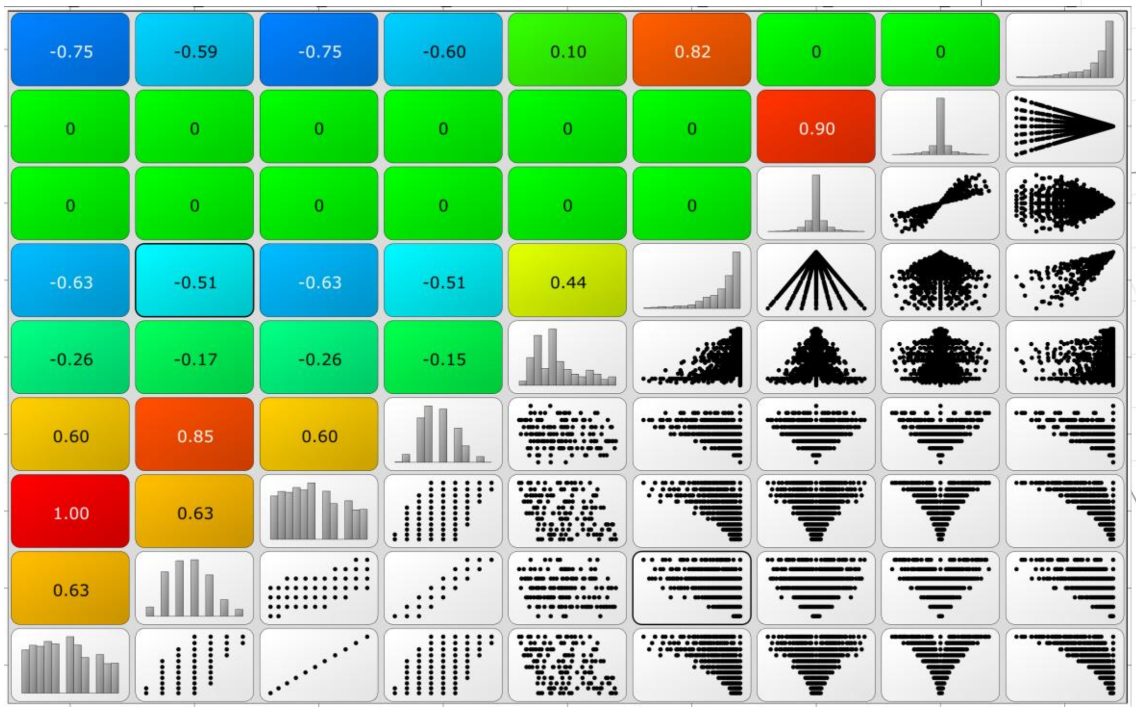
- Influence of layer number is smaller
- 4 to 8 neurons are most efficient
- Learning rate and decay is minor important



Load Bearing Capacity

- 6000 designs
- 9 input parameters with linear and nonlinear dependencies

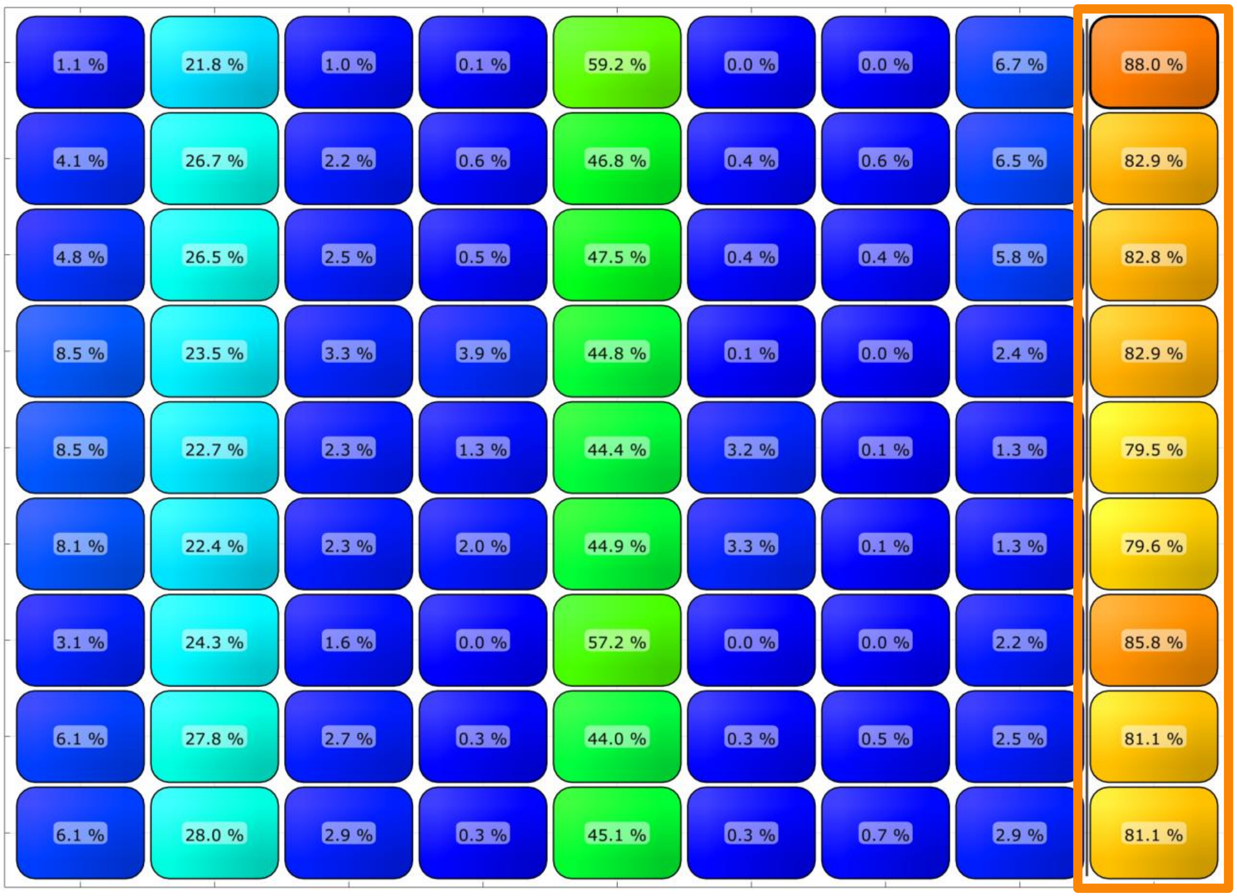




Load Bearing Capacity

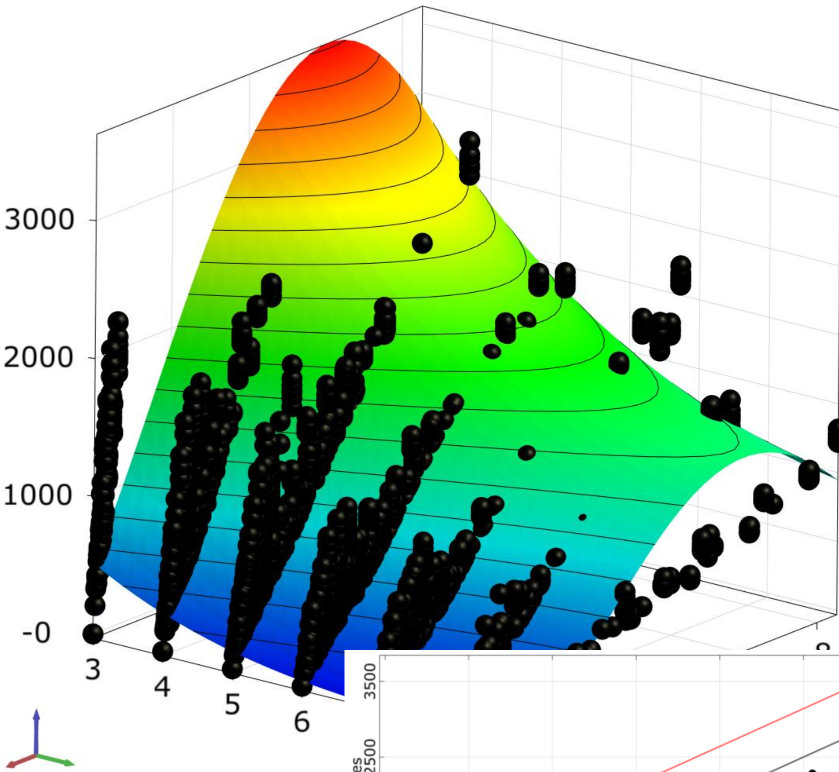
- Neural network with 3 hidden layers and 10 neurons can represent all responses very well

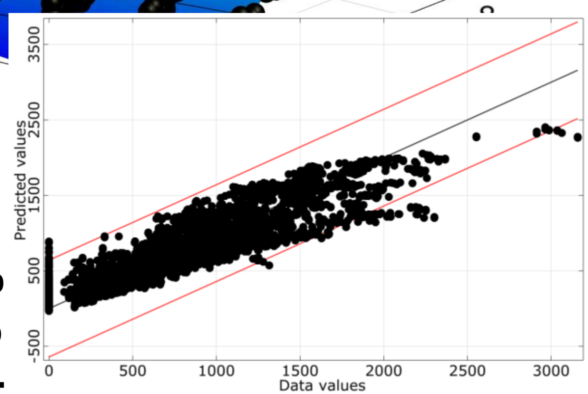
Polynomials



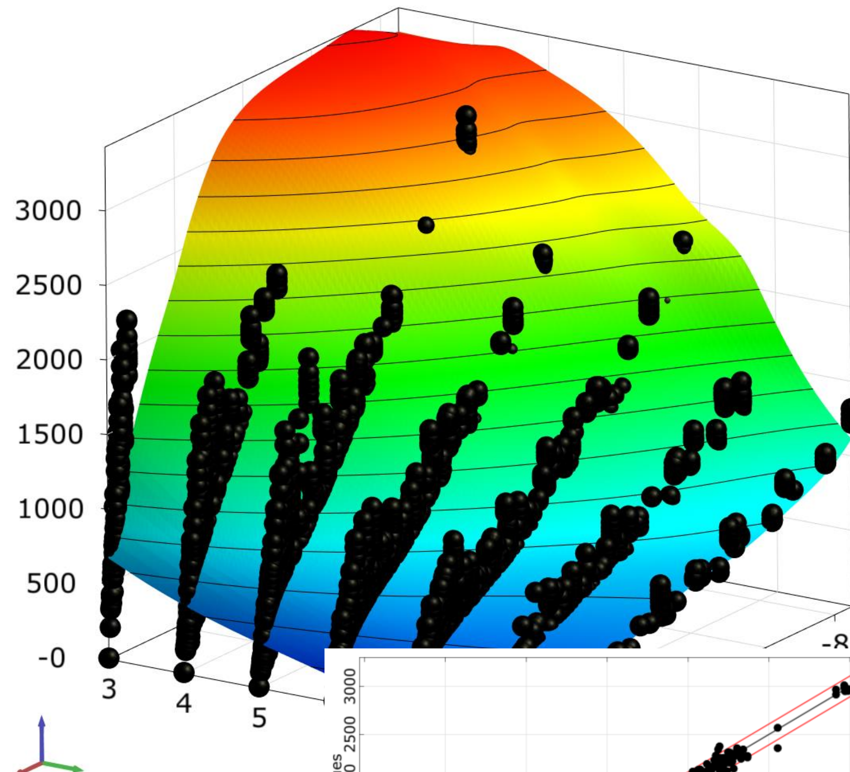
Neural Network

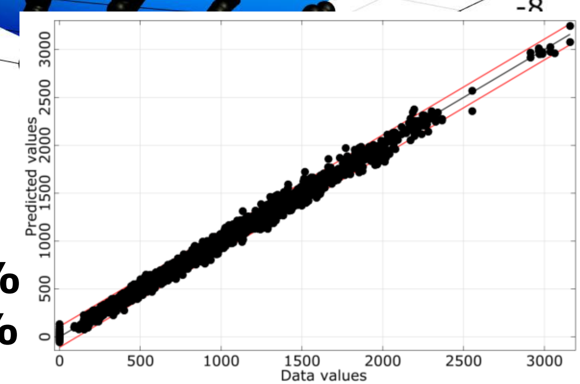
Load Bearing Capacity





Polynomial
CoD = 81 %
CoP = 81 %
RMSE = 214



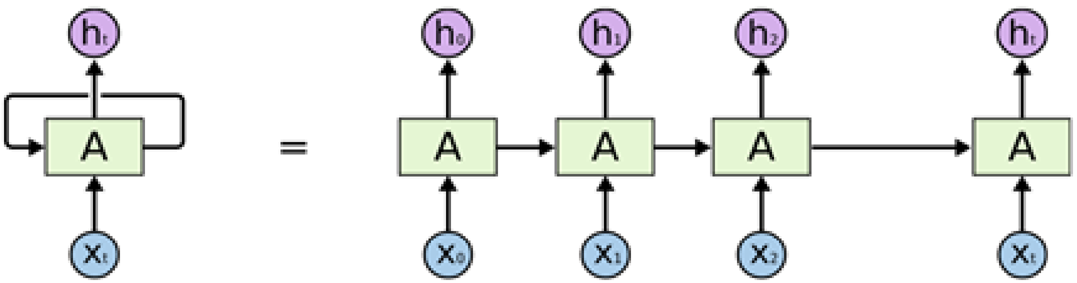


ANN
CoD = 99,7 %
CoP = 99,5 %
RMSE = 36

Time Series Prediction

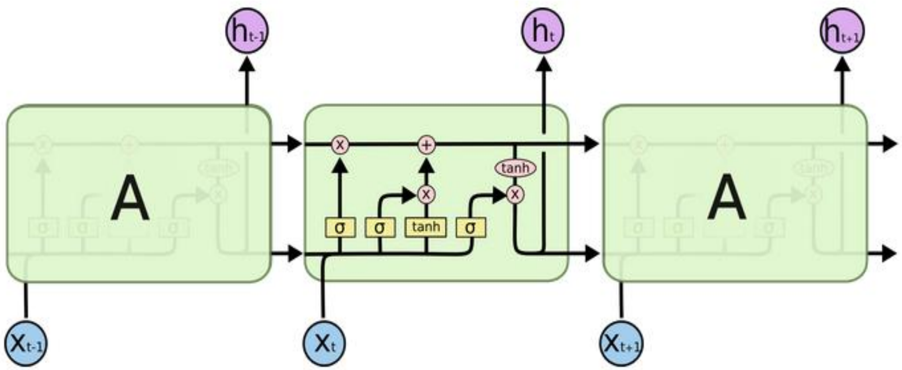
Recurrent Neural Network

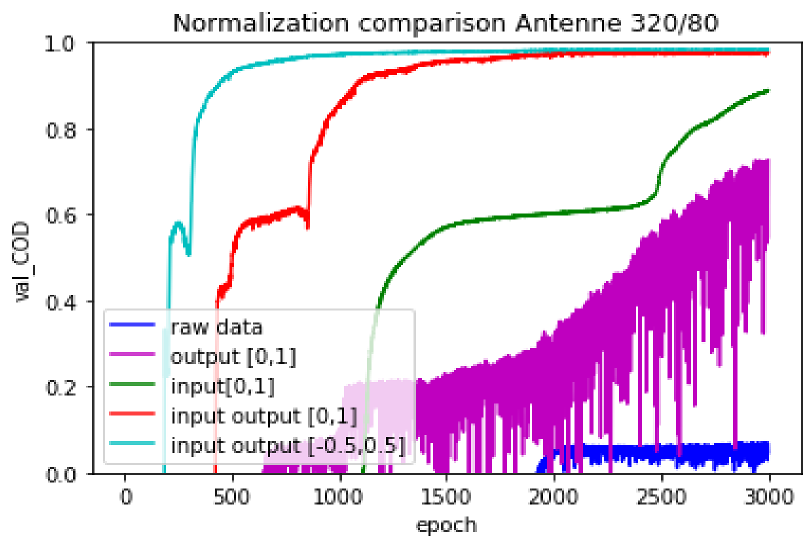
Simple RNN:



Loops to keep information from the outputs

LSTM (Long Short Term Memory):





Signal Approximation with Recurrent Networks

Example Lotka-Volterra

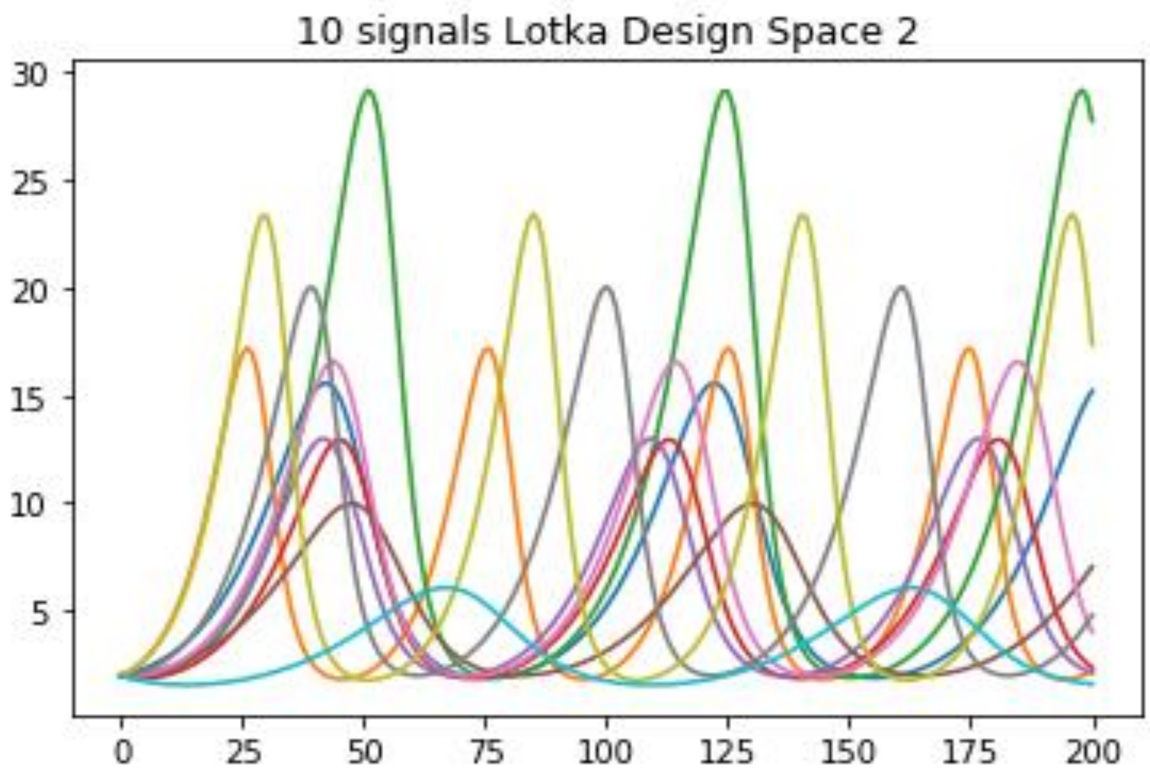
Num_input: 4

Num_points: 201

Type_layer: LSTM

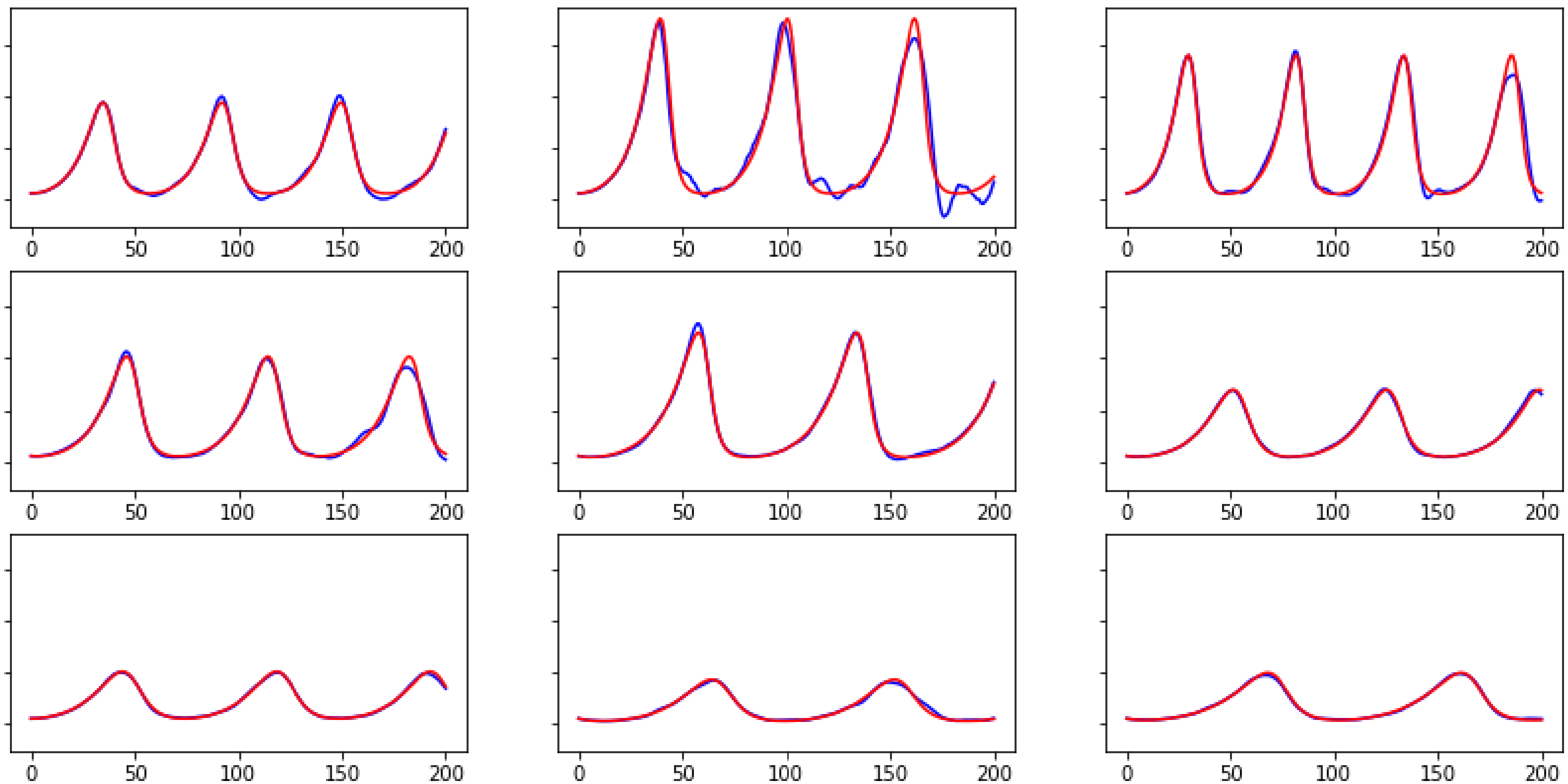
Hiden_layers: [4, 5, 6]

Nb_units: [20, 40, 60]



Signal Approximation with Recurrent Networks

Example Lotka-Volterra



Summary

- Deep learning library is available as additional external surrogate model in the MOP framework
- Significant speed up for large data sets compared to local models, but more flexibility and higher accuracy than polynomials
- MOP parameter reduction and optimal network design is the key for success with small data sets

For more information, please visit the Dynardo booth:

Deep Learning libraries	- Lars Gräning, Jonas Rotermund
Custom surrogate interface	- Katrin Kühn
MOP framework	- Ulrike Adam, Thomas Most
MOP/AMOP Post Processing	- Torsten Deckner, Ulrike Adam