# Machine Learning for Systems and Control

5SC28

Lecture 3B

dr. ir. Maarten Schoukens & dr. ir. Roland Tóth

Control Systems Group

Department of Electrical Engineering

Eindhoven University of Technology

Academic Year: 2020-2021 (version 1.0)

EINDHOVEN UNIVERSITY OF TECHNOLOGY

# Learning Outcomes

Overfitting and how to prevent it.

How to use artificial neural networks for dynamic models?

What is the link between Gaussian processes and artificial neural networks?

# Artificial Neural Networks

Overfitting and Regularization   (or Training a Neural Network Cont’d)

Simple Neural Networks to Model Dynamics

Artificial Neural Networks and Gaussian Processes

# Artificial Neural Networks

Overfitting and Regularization (or Training a Neural Network Cont'd)

Simple Neural Networks to Model Dynamics

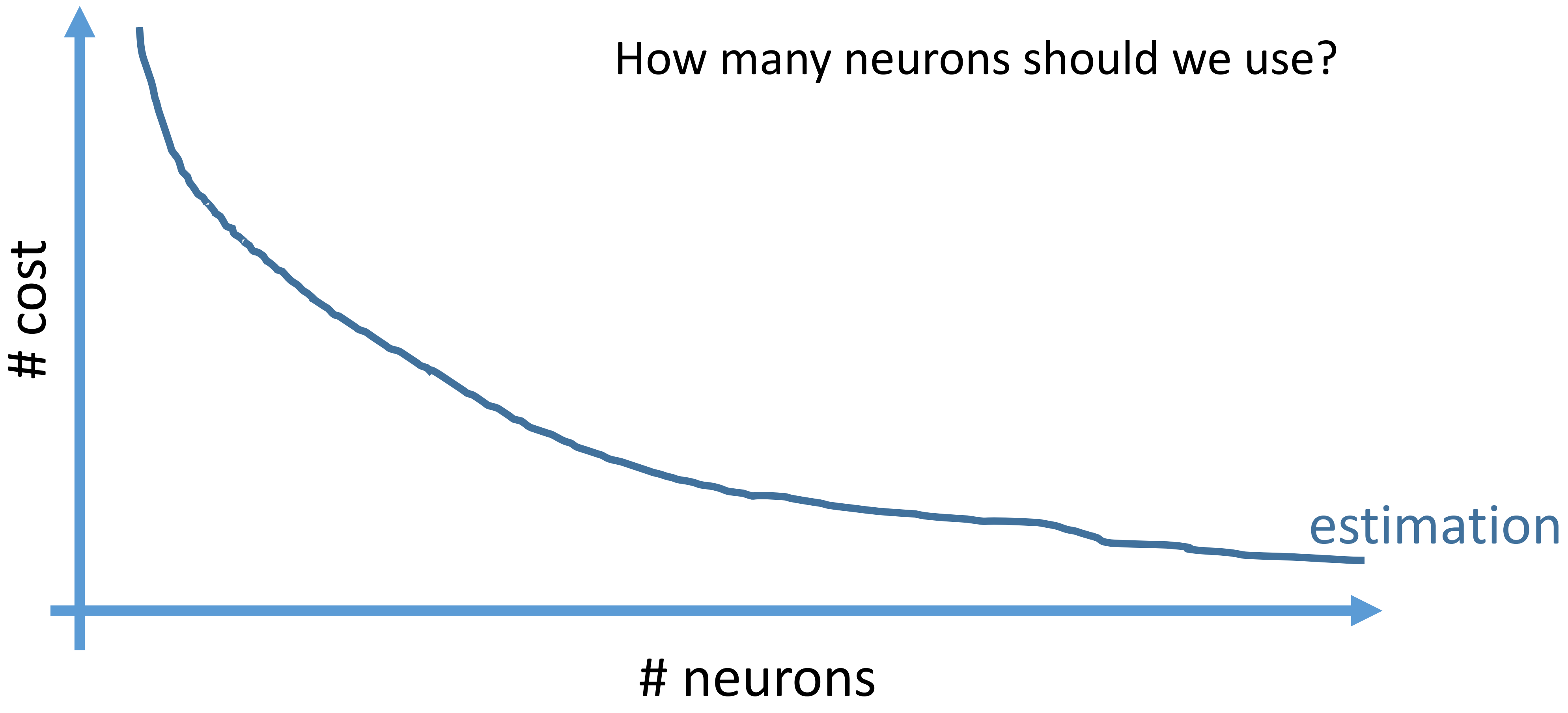
Artificial Neural Networks and Gaussian Processes

# Overfitting

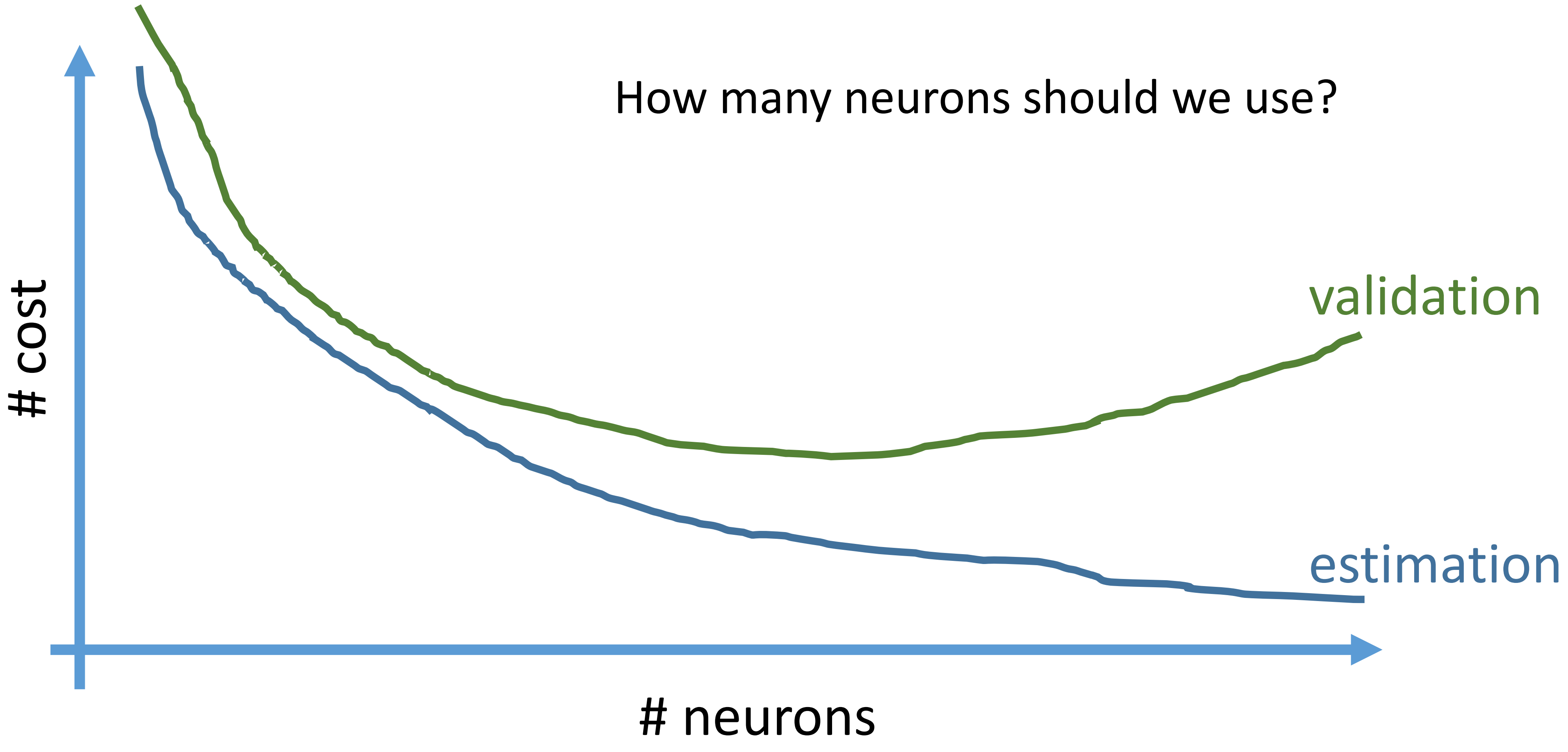
The model tries to reproduce too exactly the dataset used for estimation, due to which the model fails to predict or simulate additional data.

# Overfitting

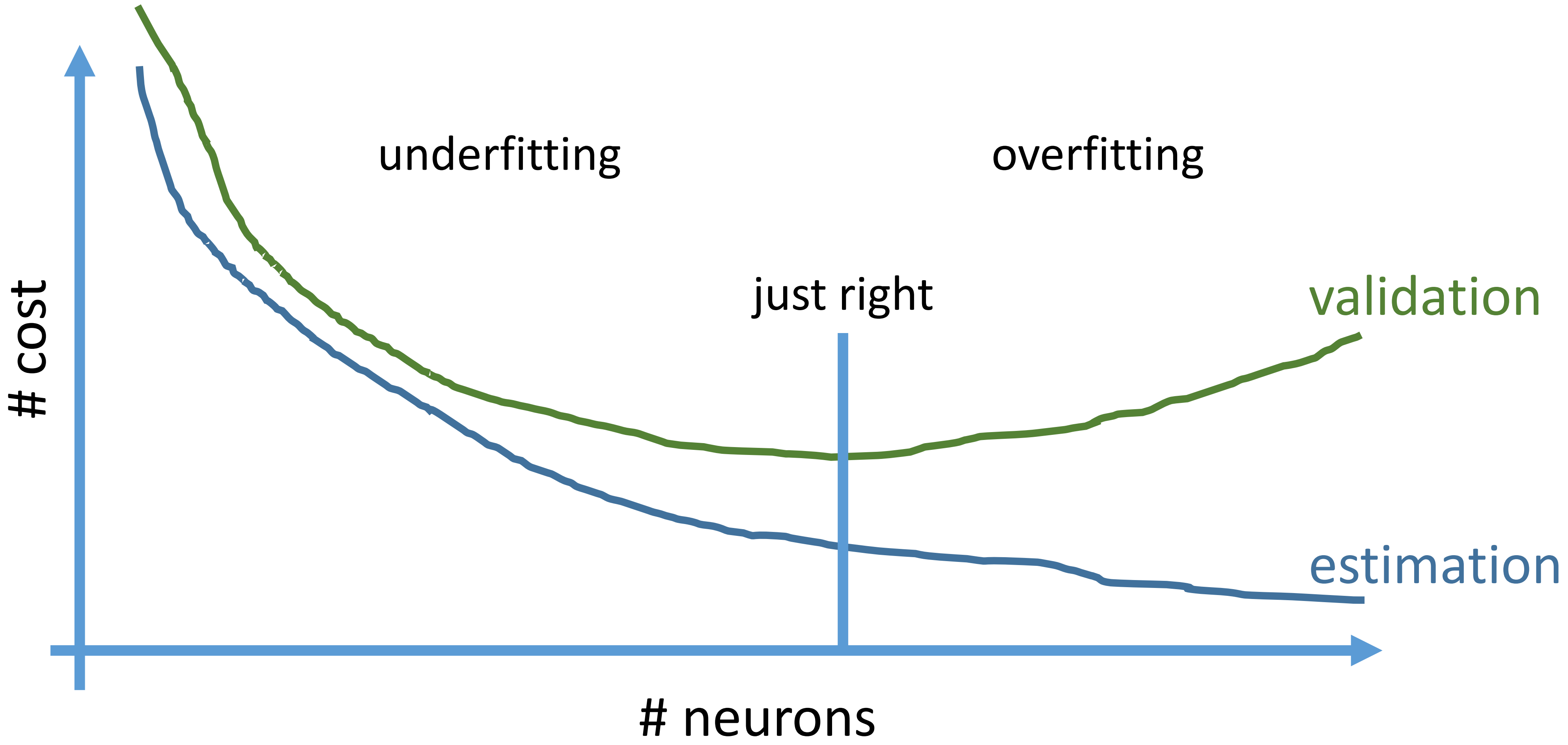
How many neurons should we use?



# Overfitting



# Overfitting

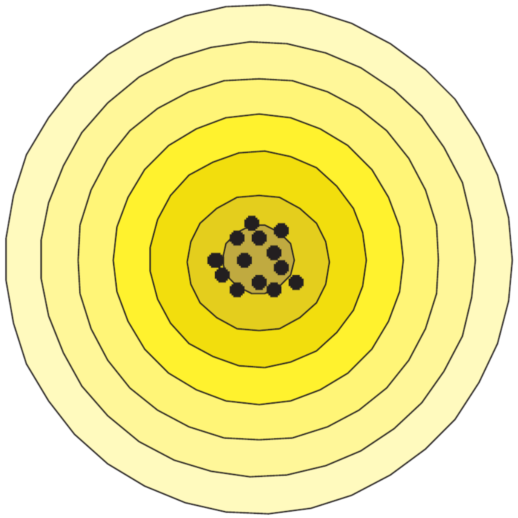


# Bias – Variance Tradeoff

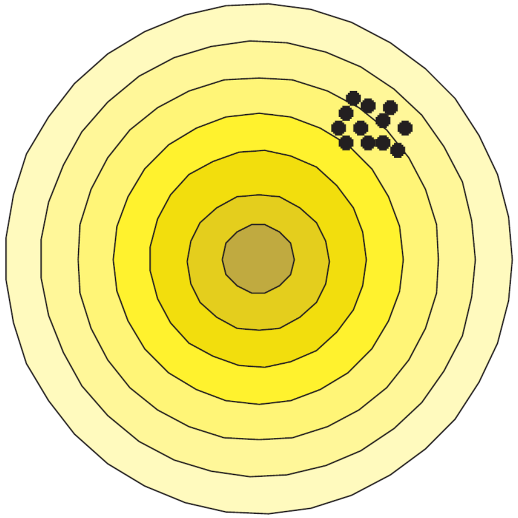
An **estimator** $\hat{\theta}_N$ of $\theta_\text{o}$ is a mapping from the (measured) data $\mathcal{D}_N$ to $\hat{\theta}_N$. If the data contains random variables, then $\hat{\theta}_N$ is a random variable.

# Bias – Variance Tradeoff

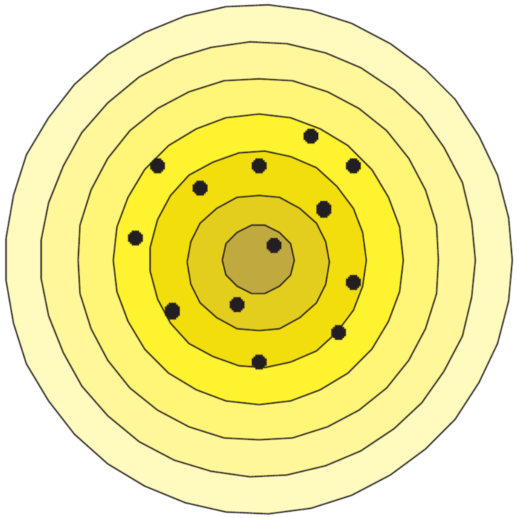
An **estimator** $\hat{\theta}_N$ of $\theta_o$ is a mapping from the (measured) data $\mathcal{D}_N$ to $\hat{\theta}_N$. If the data contains random variables, then $\hat{\theta}_N$ is a random variable.



Unbiased
Small variance



Biased
Small variance

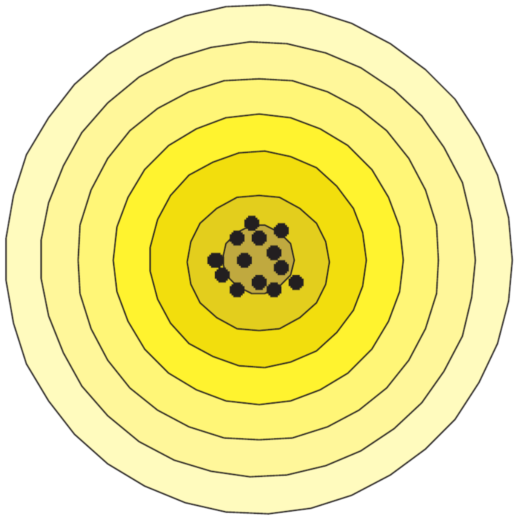


Unbiased
Large variance

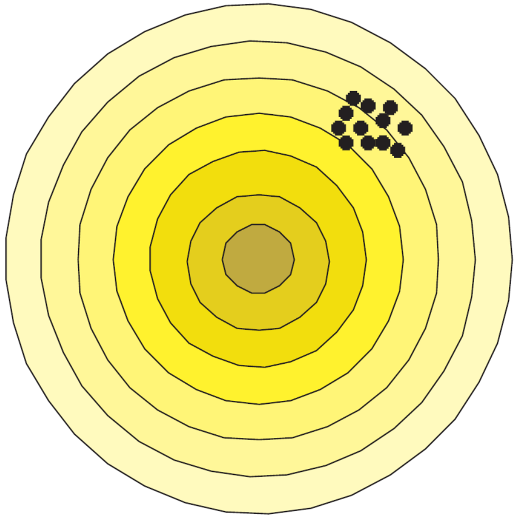
# Bias – Variance Tradeoff

$$V_N(\theta) = \frac{1}{N}\sum_{k=0}^{N-1}(y_k - \hat{y}_k)^2$$

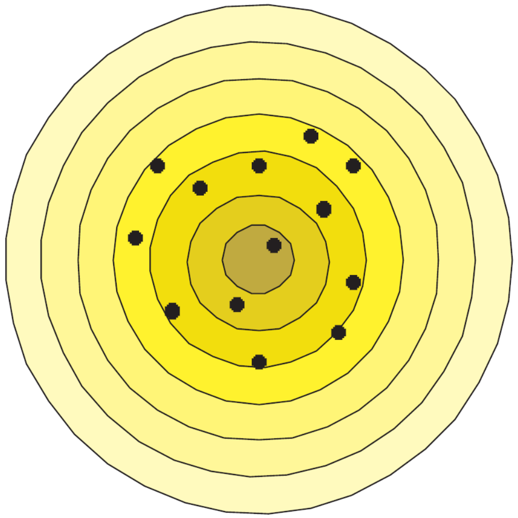
Can be split in a contribution due to parameter bias and parameter variance



Unbiased
Small variance



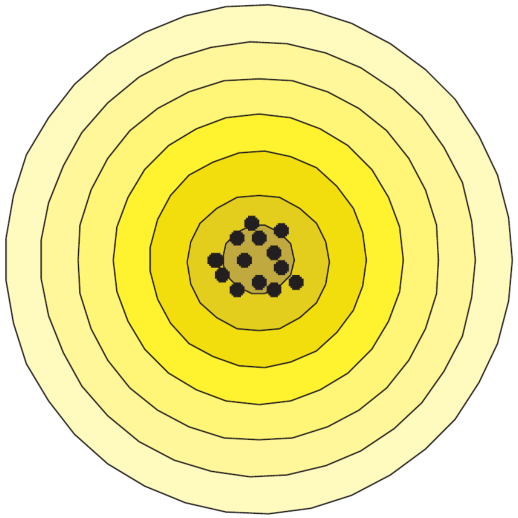
Biased
Small variance



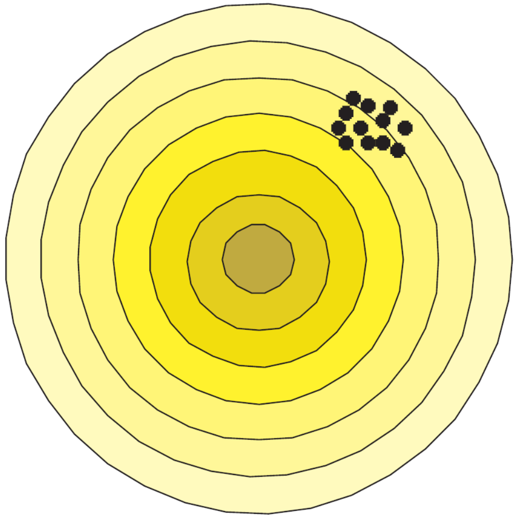
Unbiased
Large variance

# Bias – Variance Tradeoff

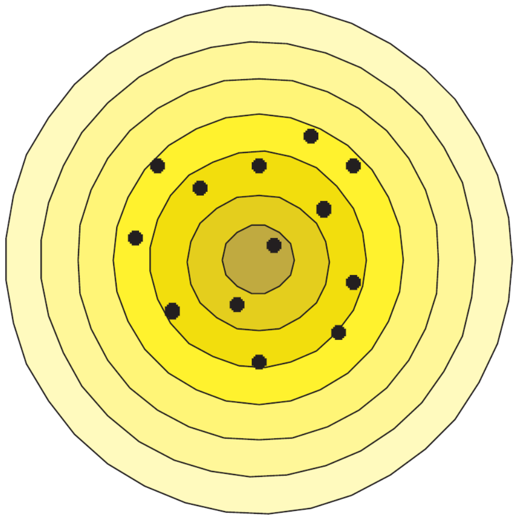
What combination of parameter bias and variance leads to the smallest cost?



Unbiased
Small variance



Biased
Small variance



Unbiased
Large variance

# Estimation – Validation – Test

**Estimation / Training dataset**:

dataset used to estimate the parameters

**Validation dataset**:

dataset used to validate the model structure, tune the hyperparameters (e.g. # neurons, # training iterations, network structure, …). The dataset is independent from the estimation data.

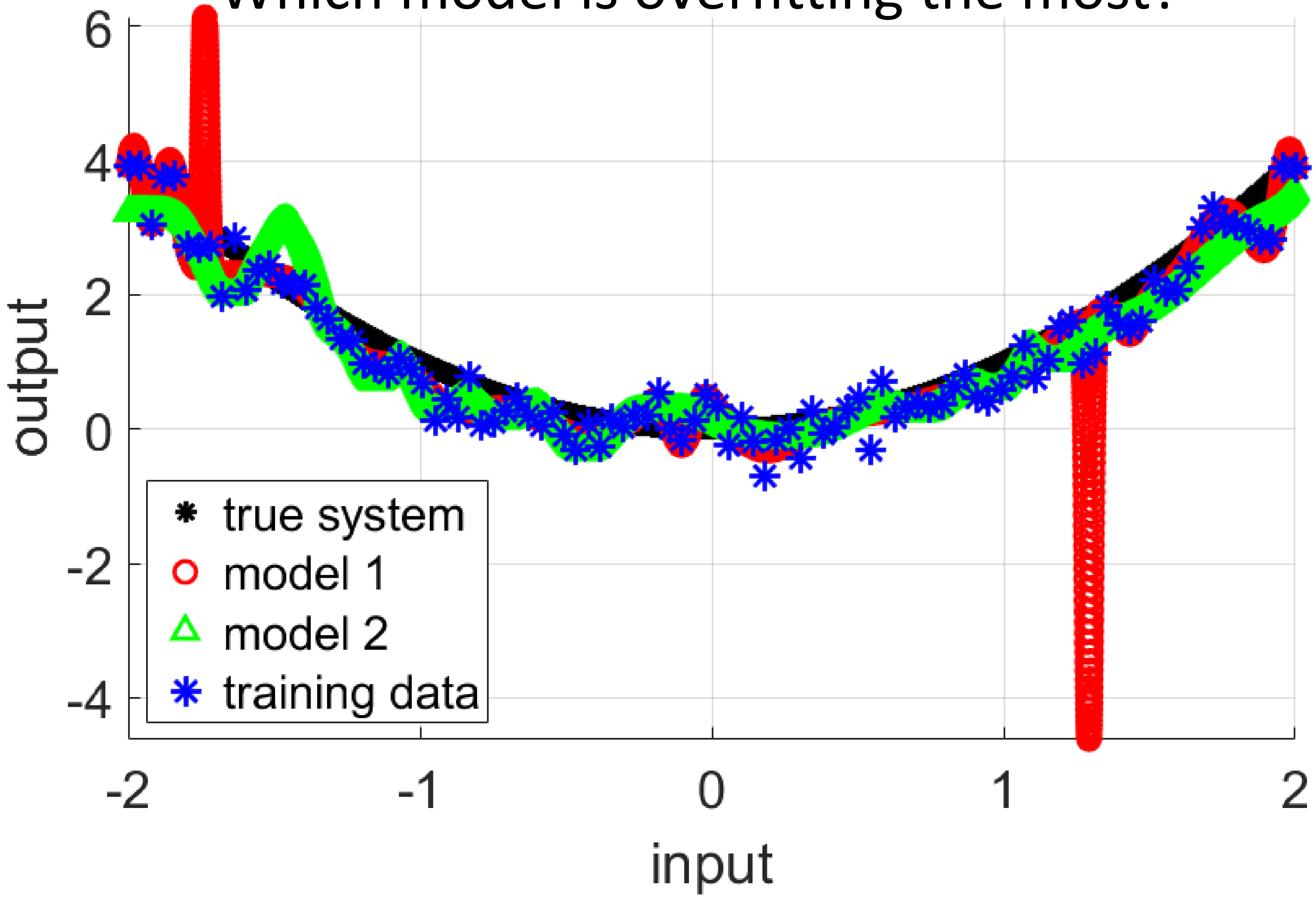
**Test dataset:**

dataset to test the quality of the final model. The dataset is independent from the estimation and validation dataset.

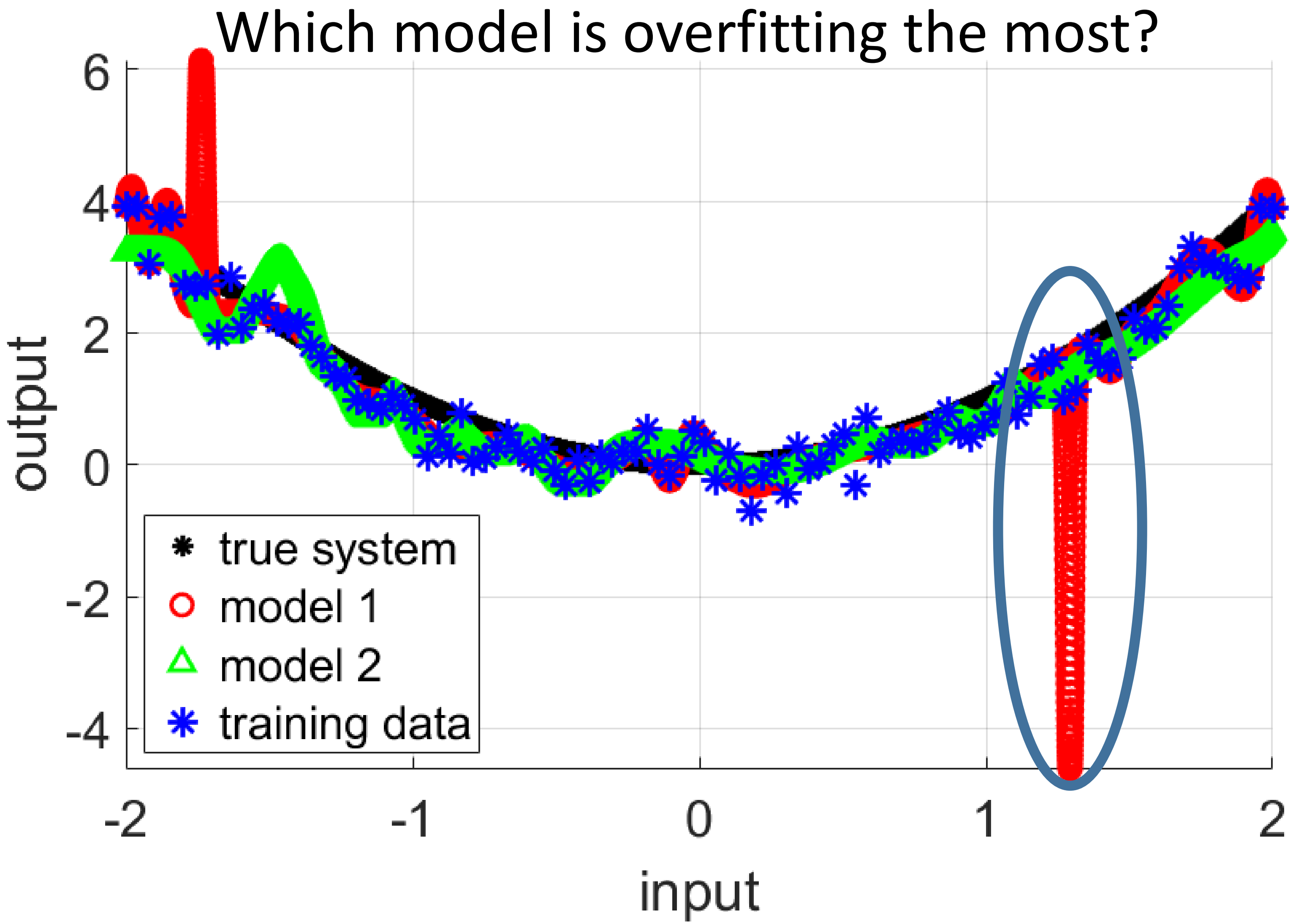
Typically all datasets are chosen such that they follow the same probability density function and other statistics (e.g. power spectral density for time series).

# Overfitting

Which model is overfitting the most?

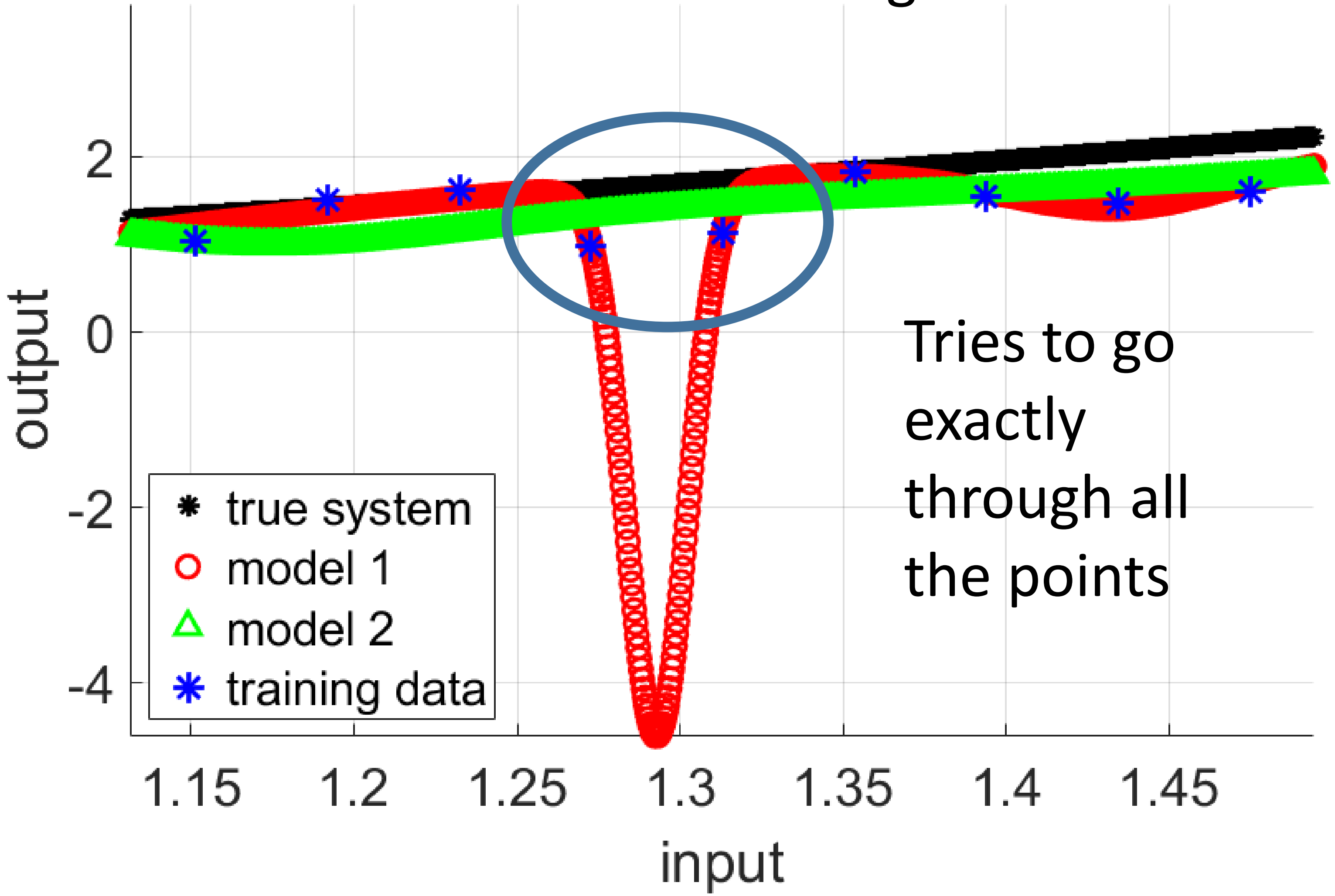


# Overfitting



# Overfitting

Which model is overfitting the most?




Tries to go exactly through all the points

# Early Stopping

At every iteration of the nonlinear optimization problem, evaluate the cost on the validation dataset.

Two general use cases:

1. If validation cost does not decrease for m consecutive iterations: stop the optimization algorithm (shorter optimization time)
2. After optimization, select the iteration for which the lowest validation cost is obtained (longer optimization time)

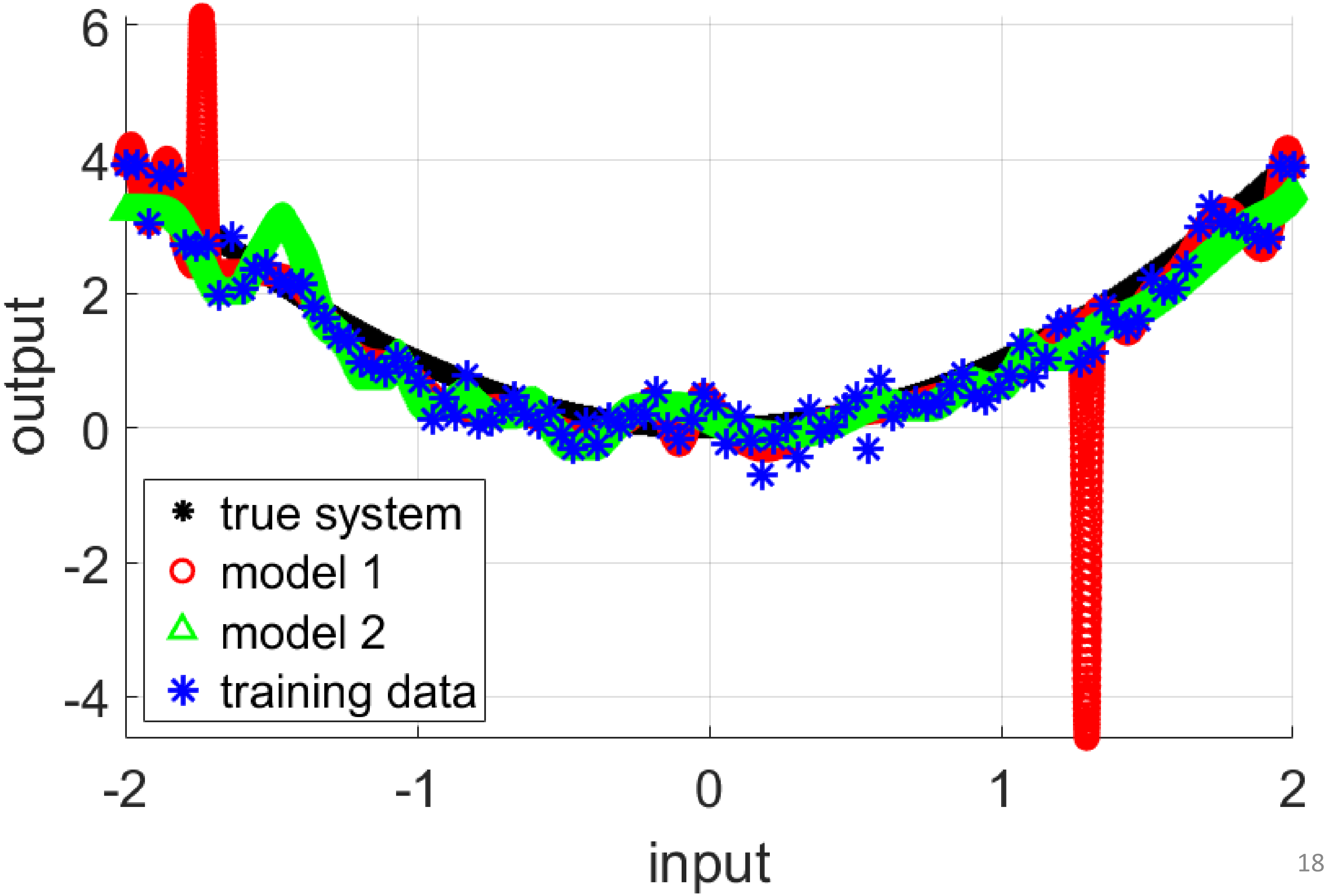
# Early Stopping

Red:

No validation dataset is used.

Green:

20% of the total available data is used as validation dataset.



# Splitting a Dataset

Standard approach:

Random assignment of datapoint to estimation / validation / test

Underlying assumption:

Neighboring datapoints are independent

# Splitting a Dataset: Time-Series

Neighboring datapoints are often NOT independent

Correlated input-output data

Correlated disturbances

➔ Work with 3 blocks of data or 3 separate time-series

# Parameter Norm Penalization

$L^2$ Regularization, ridge regression or Thikonov regularization:

$$V_N(\theta) = \frac{1}{N}\sum_{k=0}^{N-1}(y_k - \hat{y}_k)^2 = \frac{1}{N}\mathbf{e}^T\mathbf{e}$$

$$V_N(\theta) = (1-\lambda)\frac{1}{N}\mathbf{e}^T\mathbf{e} + \lambda\theta^T\theta$$

Parameter vector

**Hyperparameter**: trade-off between regularization and data fit

**Typically only the network weights are penalized**, not the biases
Since the network weights tell how multiple neuron inputs are combined, more data is required to estimate them well

# Parameter Norm Penalization

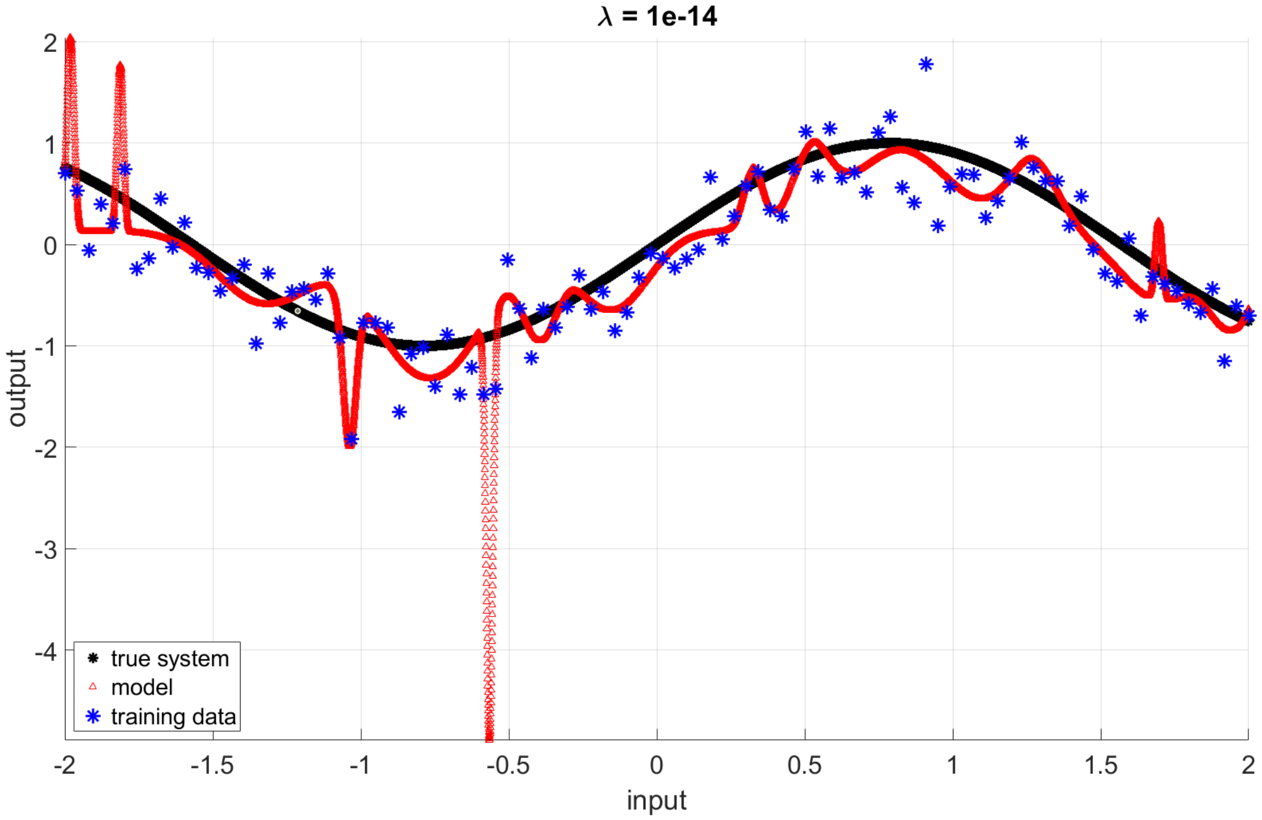
## $L^2$ Regularization, ridge regression or Thikonov regularization:

Forces changes in the parameter vector that do not contribute significantly in reducing the cost to decay away during training
(more info in: I. Goodfellow et al., *Deep Learning*, The MIT Press, pp. 223-227)

Also used for RKHS estimation: see lecture 2

# Parameter Norm Penalization

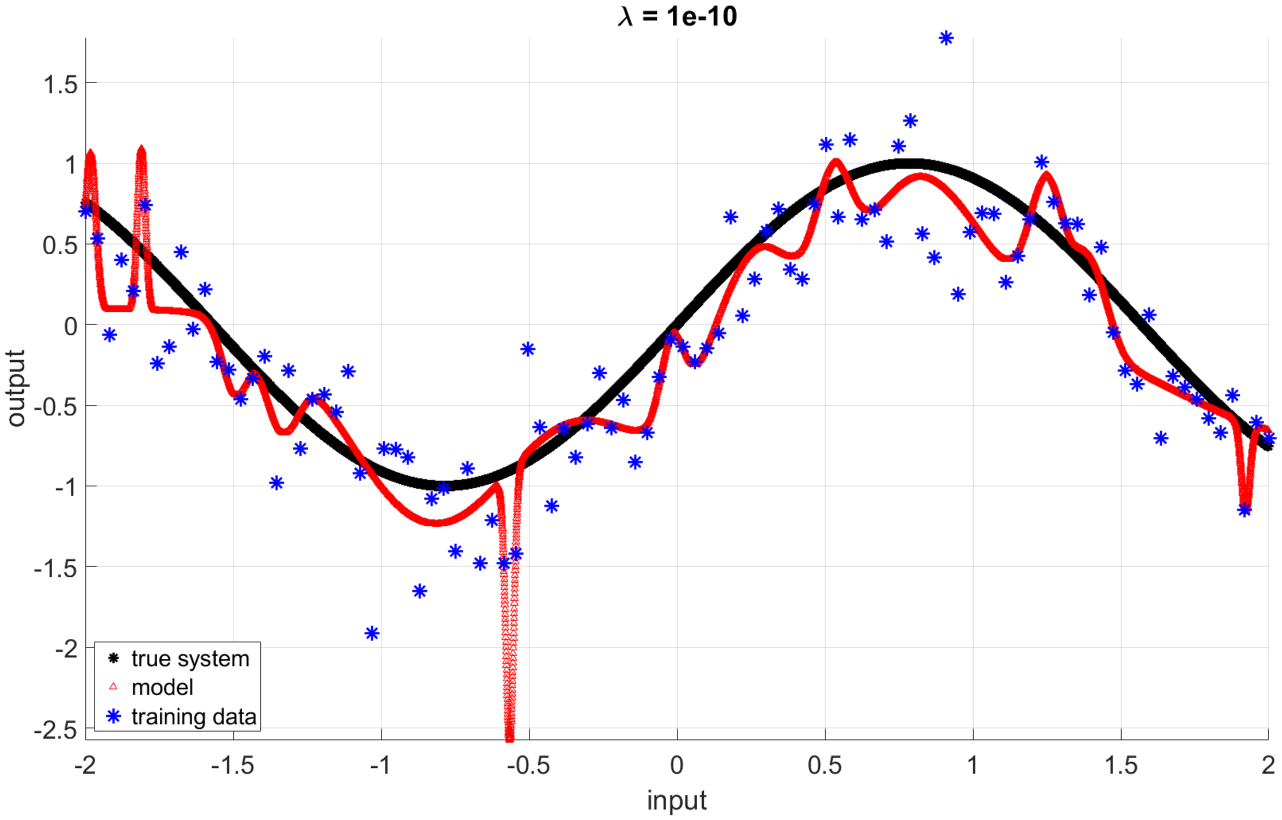
# neurons = 15




$$V_N(\theta) = (1-\lambda)\frac{1}{N}\mathbf{e}^T\mathbf{e} + \lambda\theta^T\theta$$

# Parameter Norm Penalization

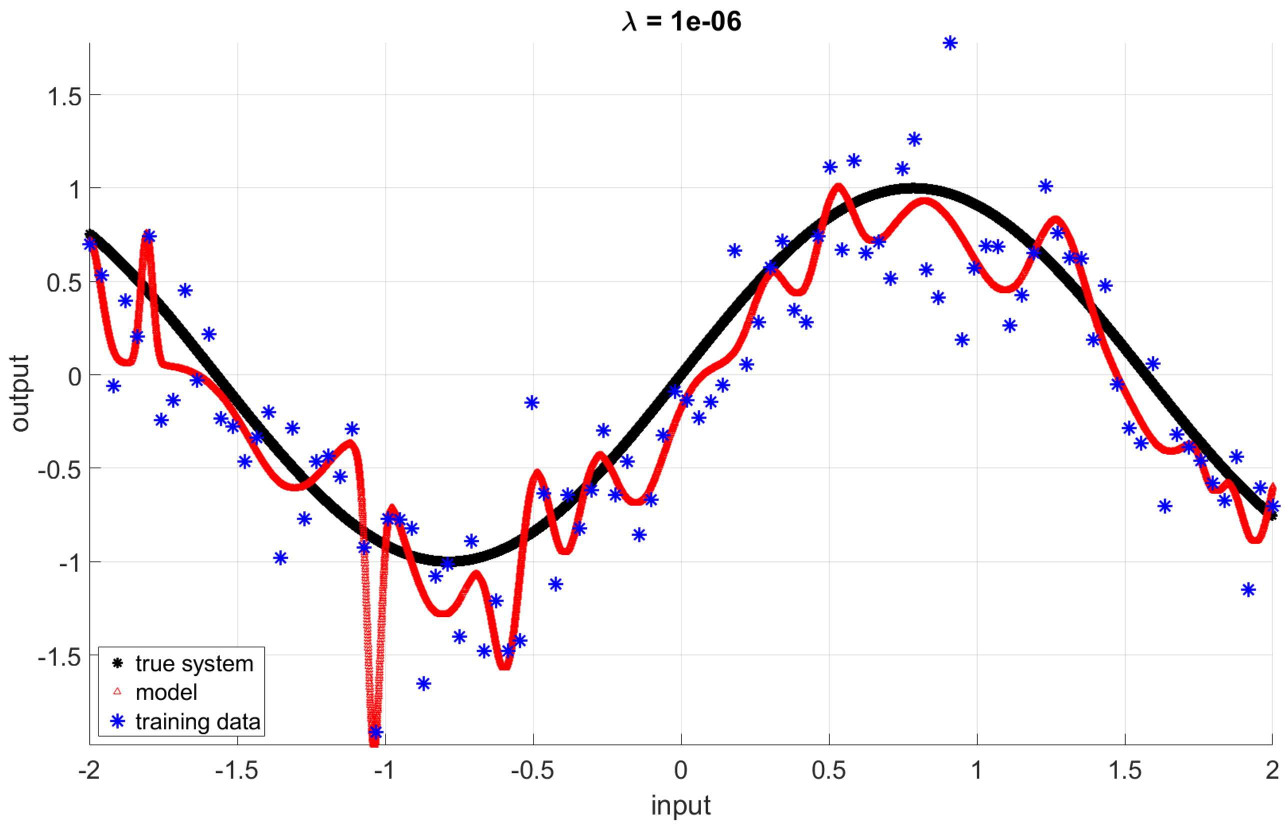
\# neurons = 15




$$V_N(\theta) = (1-\lambda)\frac{1}{N}\mathbf{e}^T\mathbf{e} + \lambda\theta^T\theta$$

# Parameter Norm Penalization

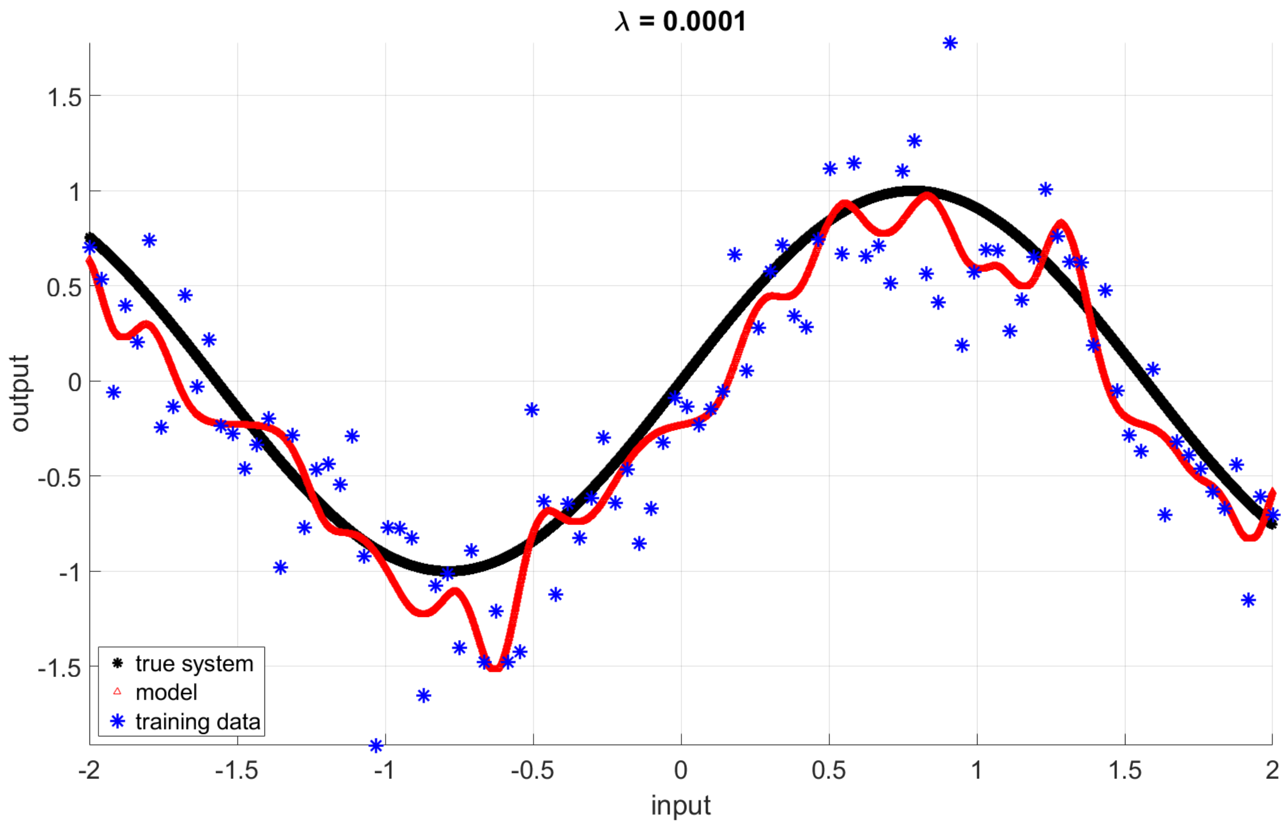
# neurons = 15




$$V_N(\theta) = (1-\lambda)\frac{1}{N}\mathbf{e}^T\mathbf{e} + \lambda\theta^T\theta$$

# Parameter Norm Penalization

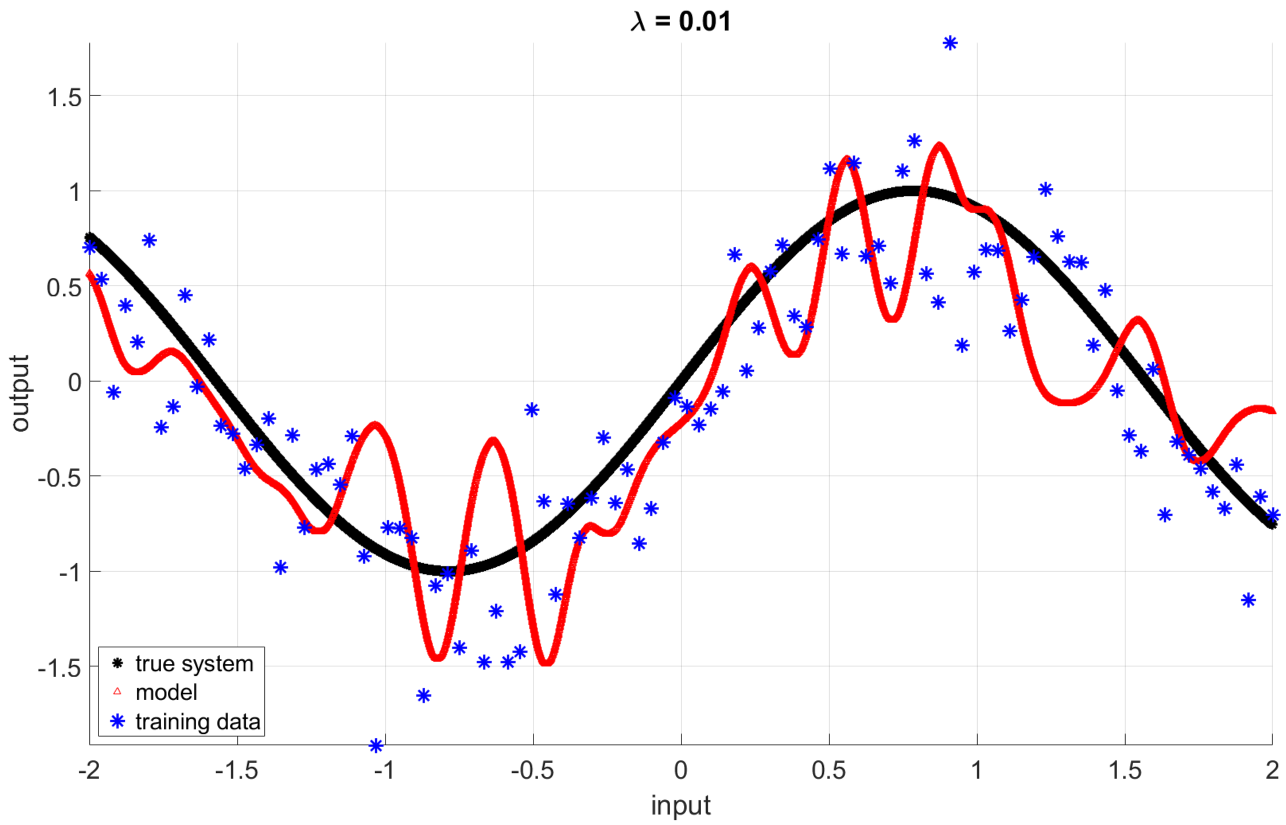
# neurons = 15




$$V_N(\theta) = (1-\lambda)\frac{1}{N}\mathbf{e}^T\mathbf{e} + \lambda\theta^T\theta$$

# Parameter Norm Penalization

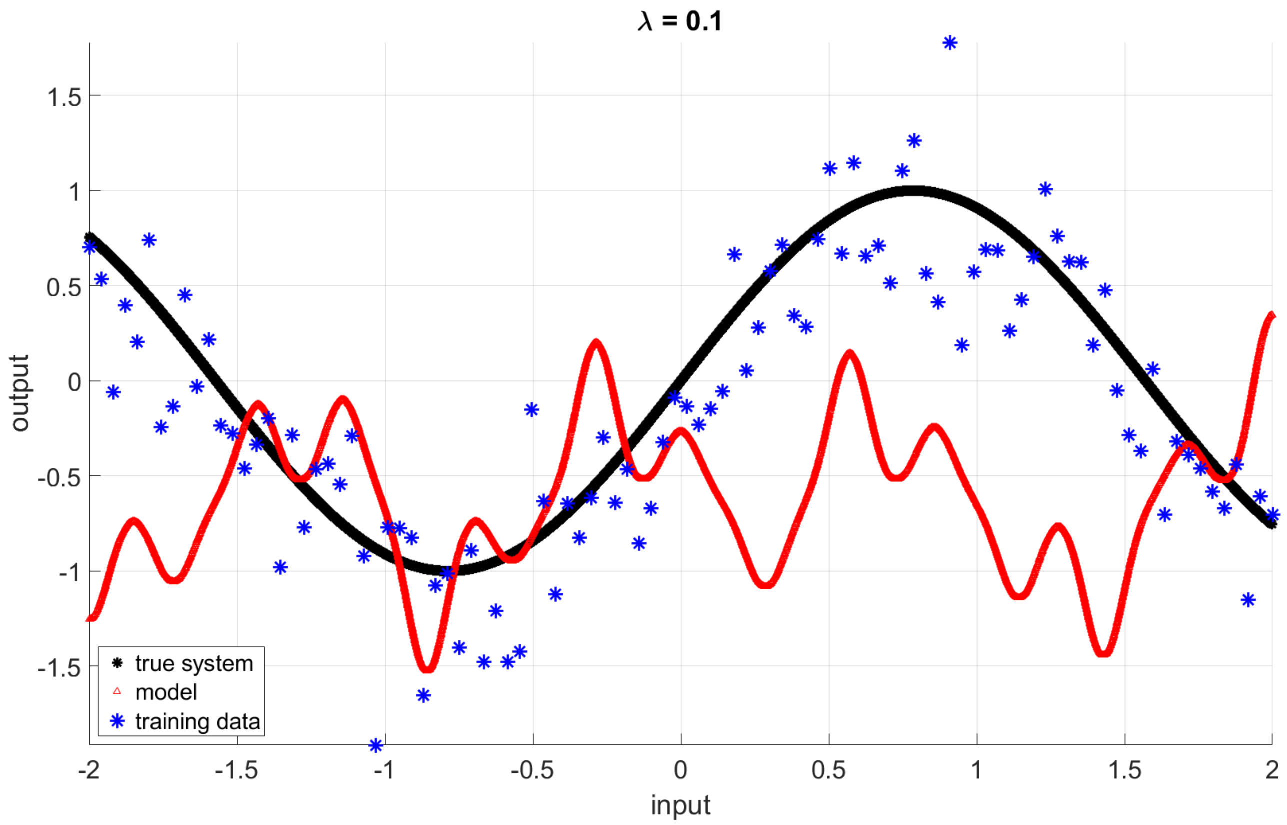
# neurons = 15




$$V_N(\theta) = (1-\lambda)\frac{1}{N}\mathbf{e}^T\mathbf{e} + \lambda\theta^T\theta$$

# Parameter Norm Penalization

# neurons = 15




$$V_N(\theta) = (1-\lambda)\frac{1}{N}\mathbf{e}^T\mathbf{e} + \lambda\theta^T\theta$$

# Parameter Norm Penalization

$L^2$ Regularization, ridge regression or Thikonov regularization:

$$V_N(\theta) = (1-\lambda)\frac{1}{N}\mathbf{e}^T\mathbf{e} + \lambda\theta^T\theta$$

Select hyperparameter through validation

(for other approaches see GP lecture)

# Artificial Neural Networks

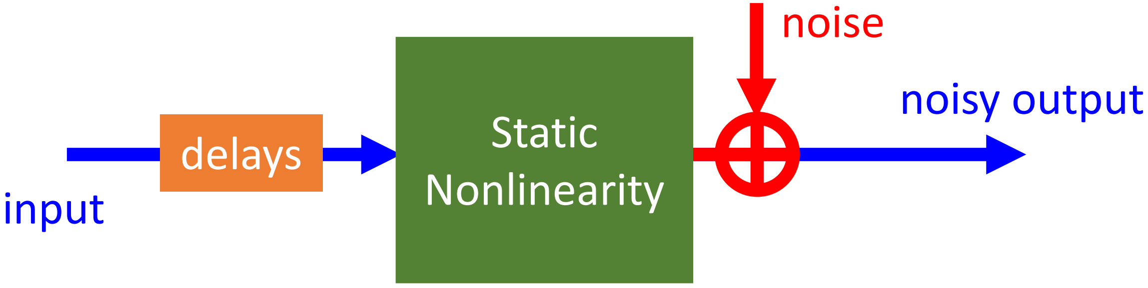
Training, Validation and Test (or Training a Neural Network Cont'd)

Overfitting and Regularization

Simple Neural Networks to Model Dynamics

Artificial Neural Networks and Gaussian Processes

# Nonlinear Finite Impulse Response (NFIR)

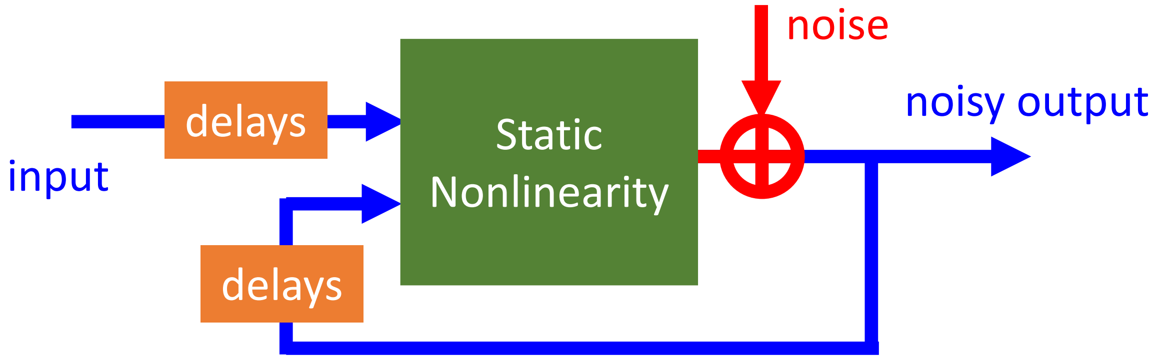



$$\boxed{y_k} = \underbrace{\boxed{f}}_{\text{To be estimated}}\underbrace{\boxed{(u_k, u_{k-1}, \ldots, u_{k-n_b})}}_{\text{known}}\underbrace{\boxed{+\, e_k}}_{\text{unknown}}$$

The nonlinear function can be represented by a simple feedforward network with 1 or more hidden layer

Network inputs: delayed system inputs and outputs

# Nonlinear Autoregressive with eXogenous Input (NARX)




$$\boxed{y_k} = \boxed{f}\boxed{(u_k, u_{k-1}, \ldots, u_{k-n_b}, y_k, \ldots, y_{k-n_a})}\boxed{+\, e_k}$$

To be estimated known unknown

The nonlinear function can be represented by a simple feedforward network with 1 or more hidden layer

Network inputs: delayed system inputs

# Nonlinear State Space (Measured States)

known states

known input

$$x_{k+1} = f\left(x_k, u_k\right) + w_k$$

process noise

$$y_k = h\left(x_k, u_k\right) + e_k$$

output noise

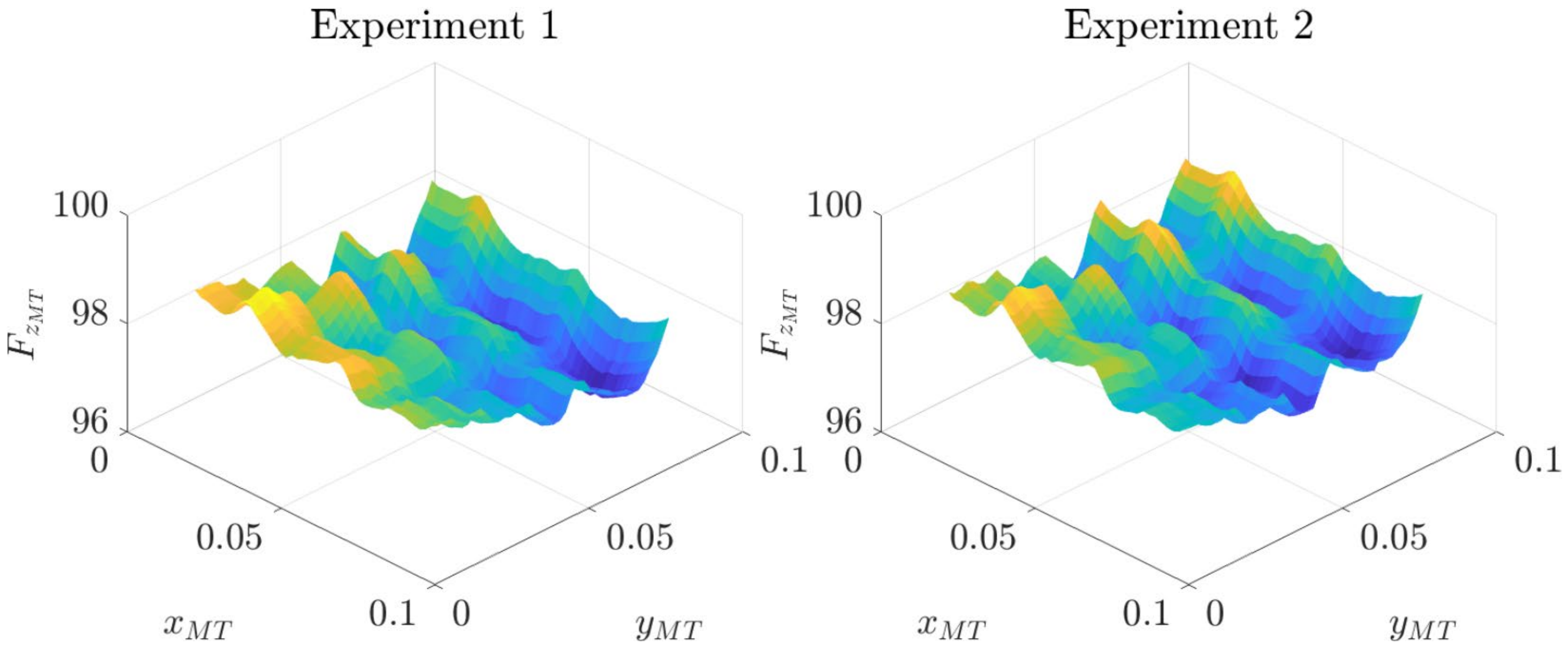
known output

multivariate static nonlinearities

The nonlinear function can be represented by a simple feedforward network with 1 or more hidden layer

Network inputs: system inputs and states

# Capturing Residual Dynamics



# Artificial Neural Networks

Training, Validation and Test (or Training a Neural Network Cont'd)

Overfitting and Regularization

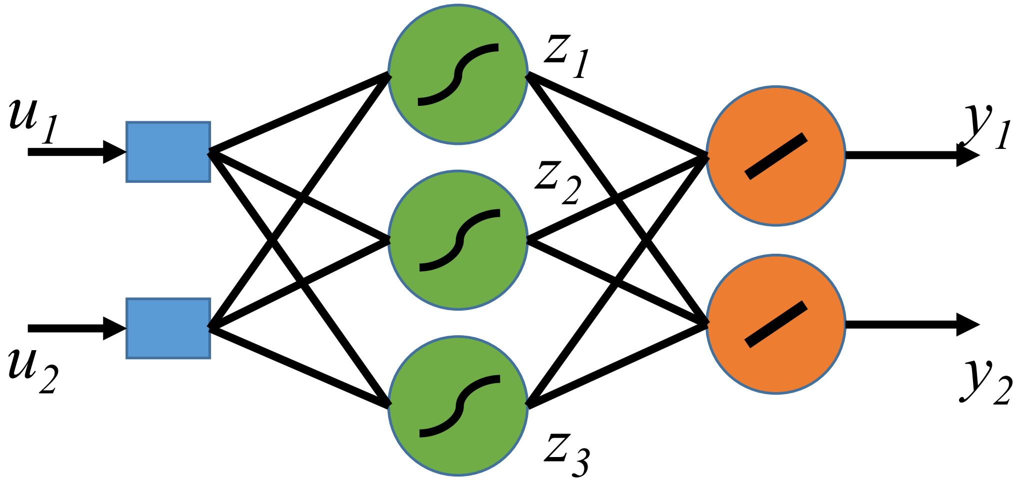
Simple Neural Networks to Model Dynamics

Artificial Neural Networks and Gaussian Processes

# From ANN to Gaussian Process

Artificial Neural Networks with 1 hidden layer can be interpreted as a Gaussian Process[1]

Recall: ANN with 1 hidden layer equation




$$\boldsymbol{z} = g\left(\boldsymbol{W}_1 \boldsymbol{u} + \boldsymbol{b}_1\right)$$

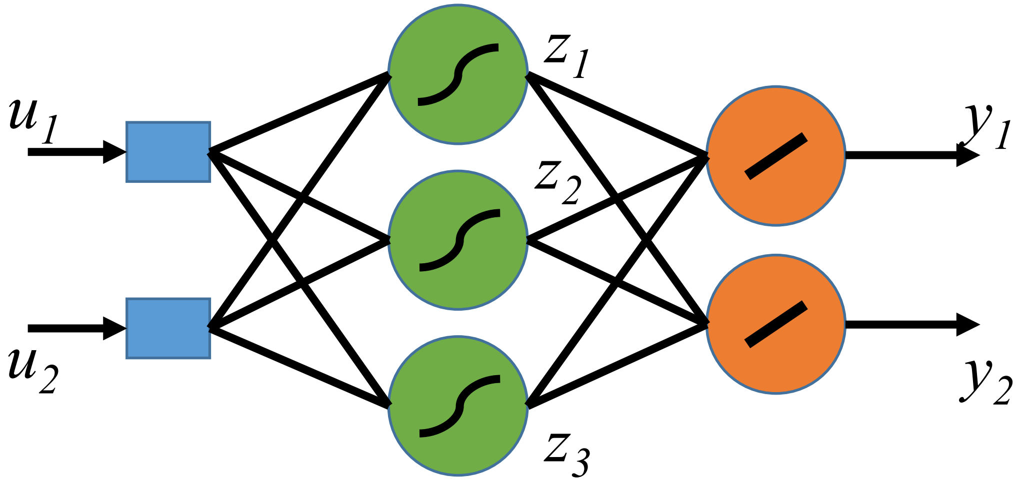
$$\boldsymbol{y} = \boldsymbol{W}_2 \boldsymbol{z} + \boldsymbol{b}_2$$

$$\boldsymbol{y} = f(\boldsymbol{u}) = \boldsymbol{W}_2\, g\left(\boldsymbol{W}_1 \boldsymbol{u} + \boldsymbol{b}_1\right) + \boldsymbol{b}_2$$

[1] C.E. Rasmussen and C.K.I Williams, Gaussian Processes for Machine Learning, The MIT Press, 2006, pp. 90-91

# From ANN to Gaussian Process

$$\boldsymbol{y} = f(\boldsymbol{u}) = \boldsymbol{W}_2\, g\left(\boldsymbol{W}_1 \boldsymbol{u} + \boldsymbol{b}_1\right) + \boldsymbol{b}_2$$




Assign independent zero-mean distributions with variance $\sigma_W$ and $\sigma_b$ to $\boldsymbol{W}_2$ and $\boldsymbol{b}_2$ respectively.

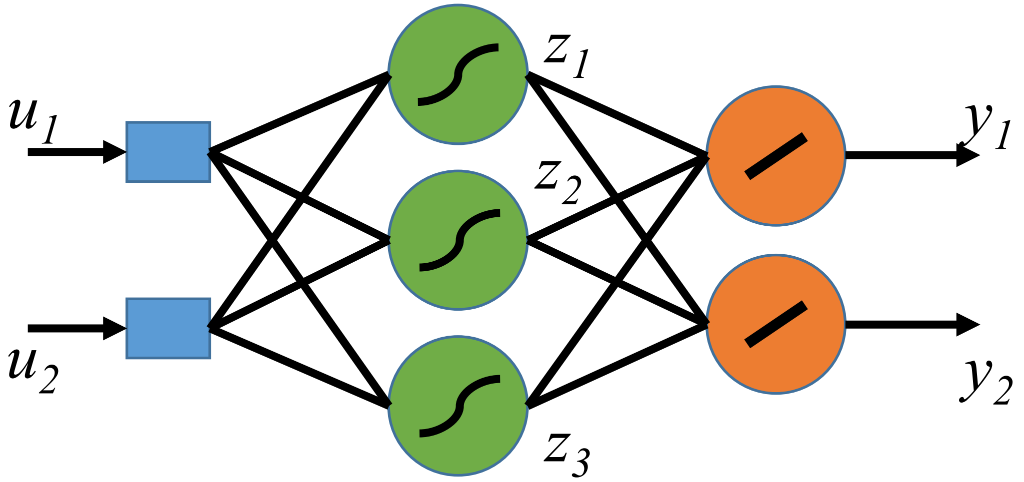
Let the hidden weights and biases ($\boldsymbol{W}_1$ and $\boldsymbol{b}_1$) be independently and identically distributed.

Denote the concatenation of all the weights and biases, $\boldsymbol{W}_1$ , $\boldsymbol{W}_2$, and $\boldsymbol{b}_1$, $\boldsymbol{b}_2$ as $\boldsymbol{\omega}$.

Activation function $g$ is bounded (e.g. sigmoid).

# From ANN to Gaussian Process

$$\boldsymbol{y} = f(\boldsymbol{u}) = \boldsymbol{W}_2\, g\left(\boldsymbol{W}_1 \boldsymbol{u} + \boldsymbol{b}_1\right) + \boldsymbol{b}_2$$




$$\mathbb{E}_{\boldsymbol{\omega}}\{f(\boldsymbol{u})\} = 0$$

$$\mathbb{E}_{\boldsymbol{\omega}}\{f(\boldsymbol{u}) f(\tilde{\boldsymbol{u}})\} = \sigma_b^2 + \sum_j \sigma_W^2 \mathbb{E}_{\boldsymbol{W}_1, \boldsymbol{b}_1}\{g\left(\boldsymbol{W}_1 \boldsymbol{u} + \boldsymbol{b}_1\right) g\left(\boldsymbol{W}_1 \tilde{\boldsymbol{u}} + \boldsymbol{b}_1\right)\}$$

$$= \sigma_b^2 + n\sigma_W^2 \mathbb{E}_{\boldsymbol{W}_1, \boldsymbol{b}_1}\{g\left(\boldsymbol{W}_1 \boldsymbol{u} + \boldsymbol{b}_1\right) g\left(\boldsymbol{W}_1 \tilde{\boldsymbol{u}} + \boldsymbol{b}_1\right)\}$$

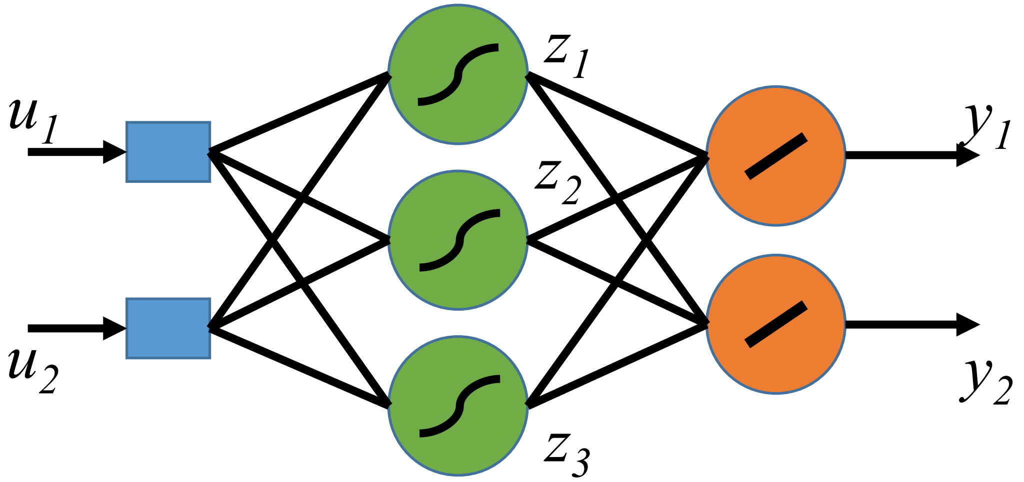
Number of neurons

# From ANN to Gaussian Process

Sum over $n$ identically and independently distributed variables.
The activation functions are bounded.

→ Central limit theorem applies

→ Stochastic process converges to a Gaussian process for $n$ growing to infinity




$$\mathbb{E}_{\boldsymbol{\omega}}\{f(\boldsymbol{u})f(\tilde{\boldsymbol{u}})\} = \sigma_b^2 + \sum_j \sigma_W^2 \mathbb{E}_{\boldsymbol{W}_1,\boldsymbol{b}_1}\{g\left(\boldsymbol{W}_1\boldsymbol{u}+\boldsymbol{b}_1\right)g\left(\boldsymbol{W}_1\tilde{\boldsymbol{u}}+\boldsymbol{b}_1\right)\}$$

$$= \sigma_b^2 + \underbrace{n}_{\text{Number of neurons}}\sigma_W^2 \mathbb{E}_{\boldsymbol{W}_1,\boldsymbol{b}_1}\{g\left(\boldsymbol{W}_1\boldsymbol{u}+\boldsymbol{b}_1\right)g\left(\boldsymbol{W}_1\tilde{\boldsymbol{u}}+\boldsymbol{b}_1\right)\}$$

# From ANN to Gaussian Process

Neural network covariance function:

$$K(\boldsymbol{u}, \boldsymbol{u}') = \mathbb{E}_{\boldsymbol{W}_1, \boldsymbol{b}_1} \{g(\boldsymbol{W}_1 \boldsymbol{u} + \boldsymbol{b}_1)\, g(\boldsymbol{W}_1 \boldsymbol{u}' + \boldsymbol{b}_1)\}$$

$$\Uparrow$$

$$\mathbb{E}_{\boldsymbol{\omega}}\{f(\boldsymbol{u}) f(\tilde{\boldsymbol{u}})\} = \sigma_b^2 + \sum_j \sigma_W^2 \mathbb{E}_{\boldsymbol{W}_1, \boldsymbol{b}_1} \{g(\boldsymbol{W}_1 \boldsymbol{u} + \boldsymbol{b}_1)\, g(\boldsymbol{W}_1 \tilde{\boldsymbol{u}} + \boldsymbol{b}_1)\}$$

$$= \sigma_b^2 + n \sigma_W^2 \mathbb{E}_{\boldsymbol{W}_1, \boldsymbol{b}_1} \{g(\boldsymbol{W}_1 \boldsymbol{u} + \boldsymbol{b}_1)\, g(\boldsymbol{W}_1 \tilde{\boldsymbol{u}} + \boldsymbol{b}_1)\}$$

Number of neurons

# From ANN to Gaussian Process

Neural network covariance function:

$$K(\boldsymbol{u}, \boldsymbol{u}') = \mathbb{E}_{\boldsymbol{W}_1, \boldsymbol{b}_1} \left\{ g\left(\boldsymbol{W}_1 \boldsymbol{u} + \boldsymbol{b}_1\right) g\left(\boldsymbol{W}_1 \boldsymbol{u}' + \boldsymbol{b}_1\right) \right\}$$

$$\tilde{\boldsymbol{W}} = \begin{bmatrix} \boldsymbol{b}_1 & \boldsymbol{W}_1 \end{bmatrix}$$

$$\tilde{\boldsymbol{u}} = \begin{bmatrix} 1 & \boldsymbol{u} \end{bmatrix}$$

$$K(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{u}}') = \mathbb{E}_{\tilde{\boldsymbol{W}}} \left\{ g\left(\tilde{\boldsymbol{W}} \tilde{\boldsymbol{u}}\right) g\left(\tilde{\boldsymbol{W}} \tilde{\boldsymbol{u}}'\right) \right\}$$

# From ANN to Gaussian Process

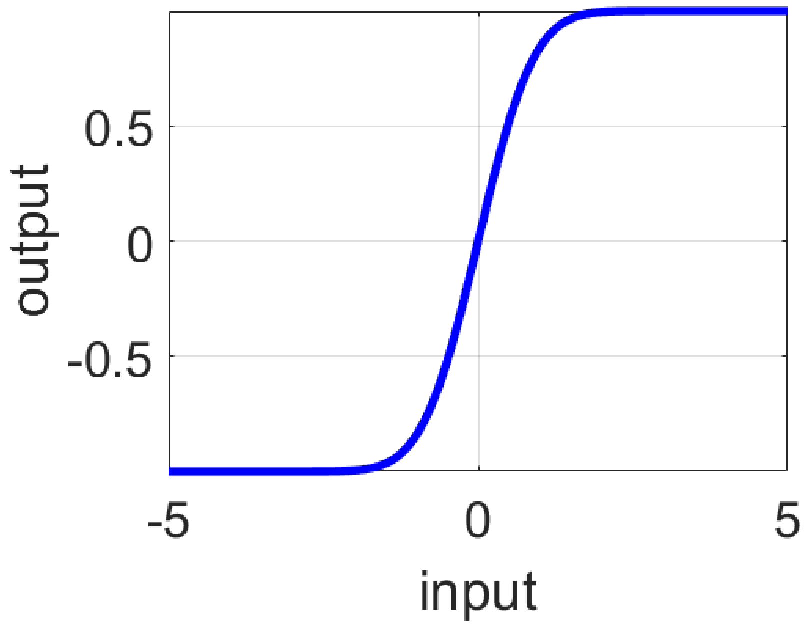
Neural network covariance function:

$$K(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{u}}') = \mathbb{E}_{\tilde{\boldsymbol{W}}} \left\{ g\left(\tilde{\boldsymbol{W}} \tilde{\boldsymbol{u}}\right) g\left(\tilde{\boldsymbol{W}} \tilde{\boldsymbol{u}}'\right) \right\}$$

Take: $\tilde{\boldsymbol{W}} \sim \mathcal{N}(0, \Sigma)$

Activation function:

$$g(x) = \mathrm{erf}\,(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-x^2} dx$$

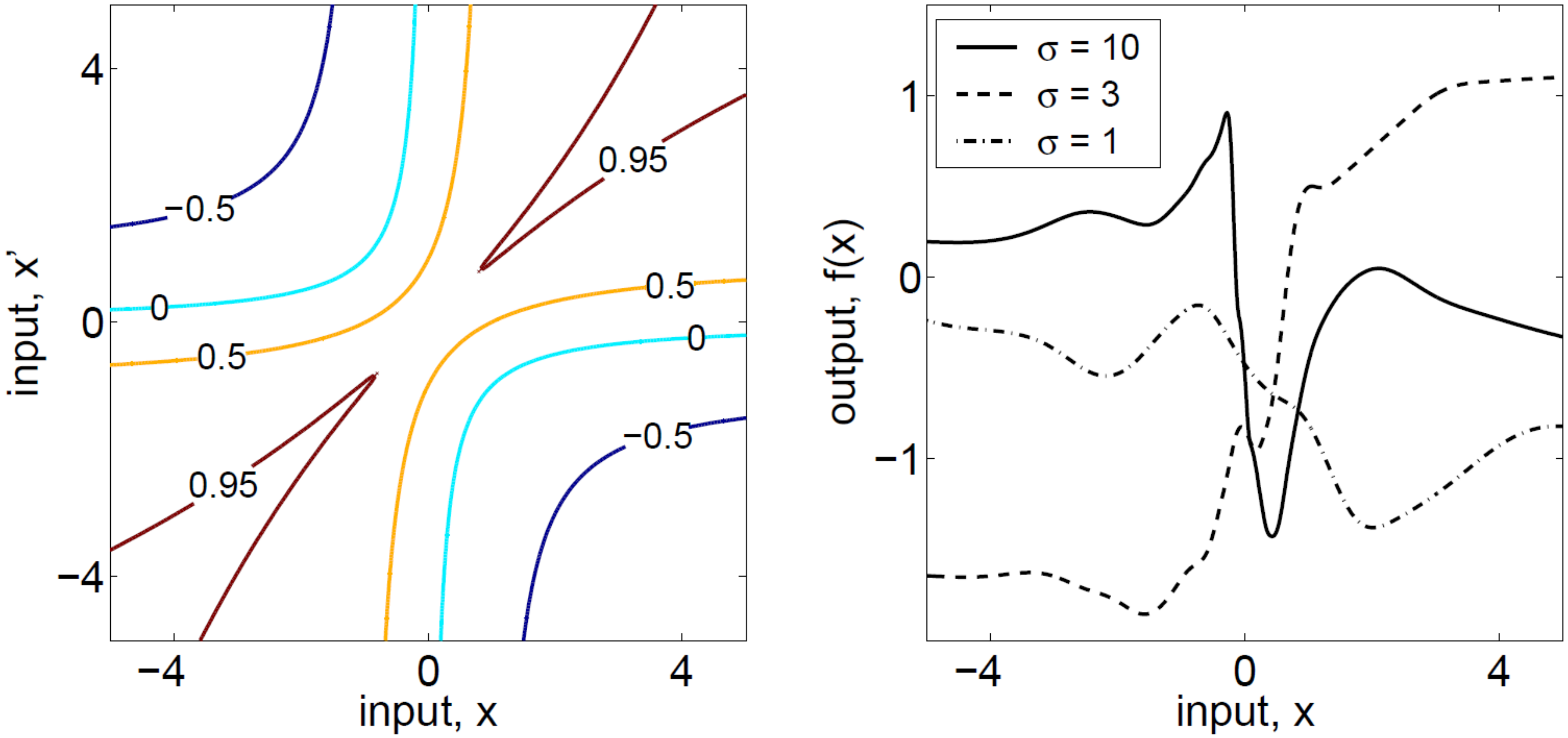


# From ANN to Gaussian Process

Neural network covariance function:

$$K(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{u}}') = \frac{2}{\pi} \sin^{-1} \left( \frac{2\tilde{\boldsymbol{u}}^T \Sigma \tilde{\boldsymbol{u}}'}{\sqrt{\left(1 + 2\tilde{\boldsymbol{u}}^T \Sigma \tilde{\boldsymbol{u}}'\right)\left(1 + 2\tilde{\boldsymbol{u}}^T \Sigma \tilde{\boldsymbol{u}}'\right)}} \right)$$

# From ANN to Gaussian Process




(a), covariance (b), sample functions

Figure 4.5: Panel (a): a plot of the covariance function $k_{NN}(x, x')$ for $\sigma_0 = 10$, $\sigma = 10$. Panel (b): samples drawn from the neural network covariance function with $\sigma_0 = 2$ and $\sigma$ as shown in the legend. The samples were obtained using a discretization of the $x$-axis of 500 equally-spaced points.



Source: C.E. Rasmussen and C.K.I Williams, Gaussian Processes for Machine Learning, The MIT Press, 2006, pp. 92

# Conclusion

What is an artificial neural network?

Simple feedforward networks

Approximation properties

Training an artificial neural network

Training, Validation and Test (or Training a Neural Network Cont'd)

Artificial Neural Networks and Gaussian Processes

# What is Missing?

What is deep learning and what are deep neural networks?

How to train deep neural networks?

Deep neural networks for dynamical systems?