

Introduction to machine learning

General about

- What is machine learning?
- Capacity, overfitting and underfitting
- Hyper parameters and model selection

What is Machine Learning?

“Learning is any process by which a system improves performance from experience.”

“Machine Learning is concerned with computer programs that automatically improve their performance through experience. “

- Herbert Simon

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 in T , as measured by P , improves with experience E .

- Tom Mitchell

According to Tom Mitchell

Learning =

- Improve over task T
- With respect to performance P
- Based on experience E

Tasks (T)

Some common tasks:

Classification: The algorithm is asked to give one of k classes for which the input belongs to

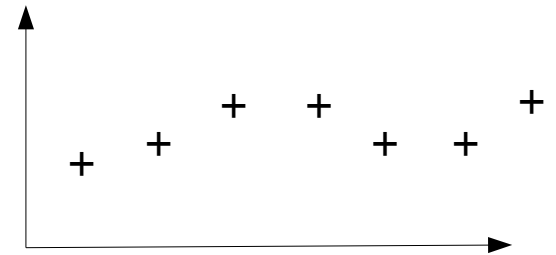
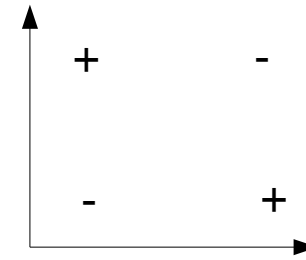
Regression: Predict a numerical output given an input

Translation: sequence in \rightarrow sequence out

*It's raining cats and dogs \rightarrow
Det regnar katter och hundar*

Synthesis and sampling: : Generate new data "similar" to the given data

6 \rightarrow 6
6
6
6



Performance measures (P)

Accuracy can be used for classification problems. Easy to understand, but there plenty of others. **Area under ROC** curve is sometimes better.

Instead of performance one can also talk about errors. **Mean squared error** can be used for regression problems, but others exist!

Other tasks have other performance or error measures. The choice is very application dependent and may be difficult to formulate.

Note 1: It is important to that performances or error should be measured on independent data

Note 2: Error measures used to communicate results are typically not the same as error used during model training

Experience (E)

Based on the type of experience there are broadly two kinds of learning algorithms (excluding reinforcement learning) :

Supervised learning and **Unsupervised learning**

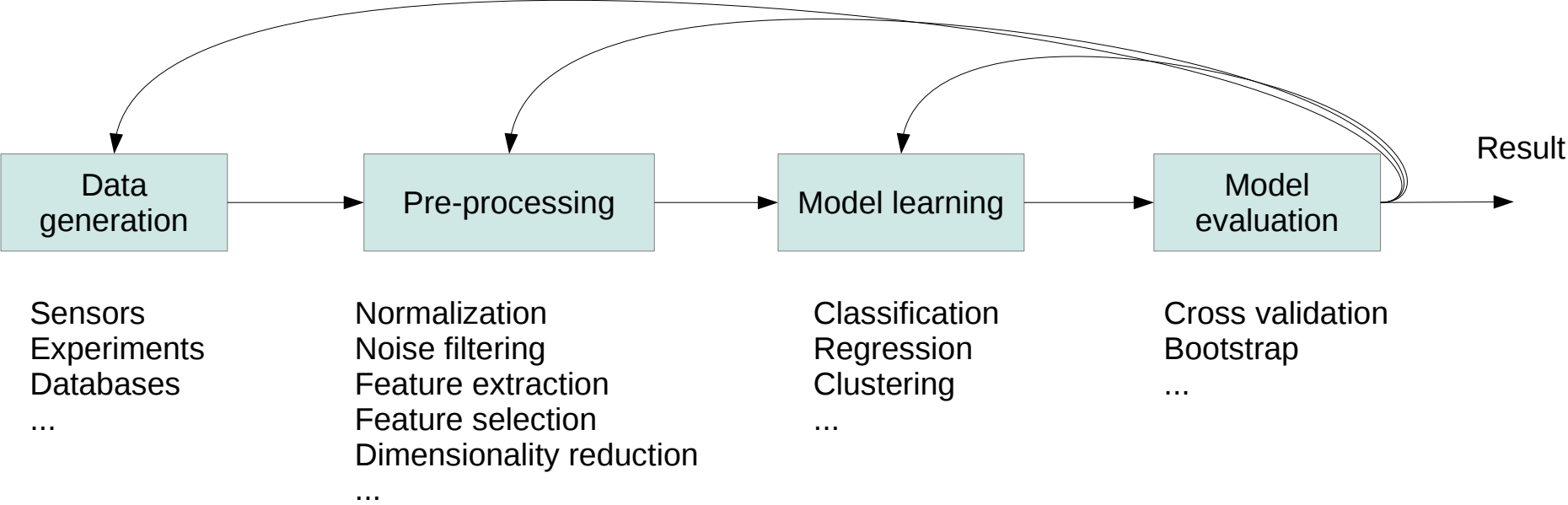
Generally, a data set without labels means unsupervised learning

$\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ \longrightarrow **Unsupervised learning**

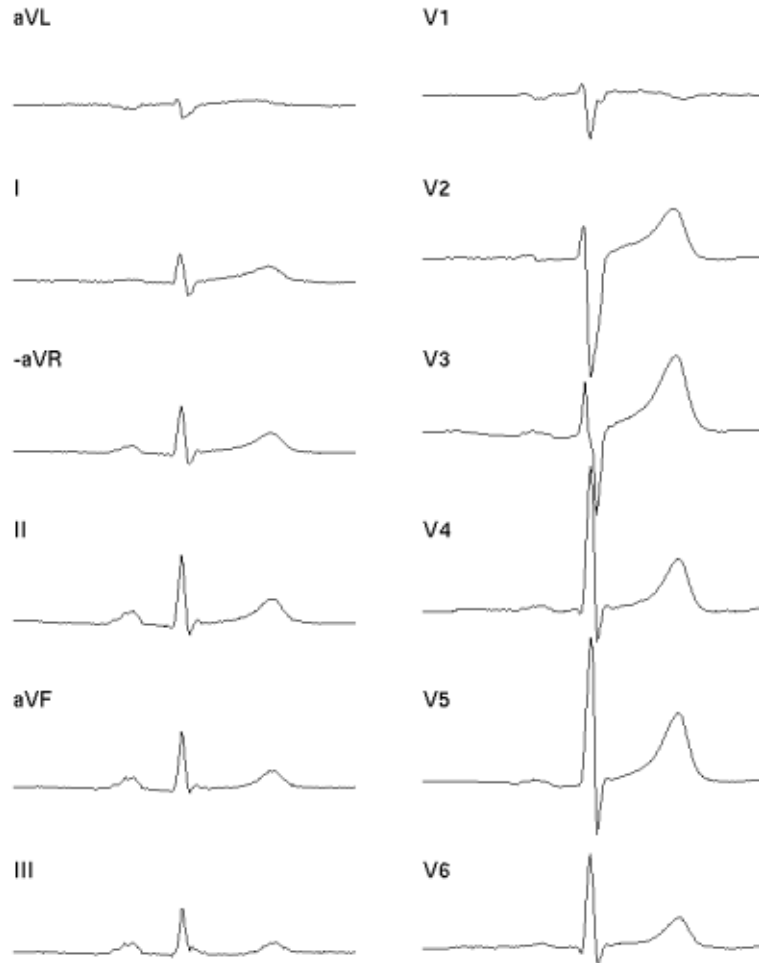
If we add labels

$\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} + \{\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_N\}$ \longrightarrow **Supervised learning**

Typical learning process



An example



Classification of ECGs

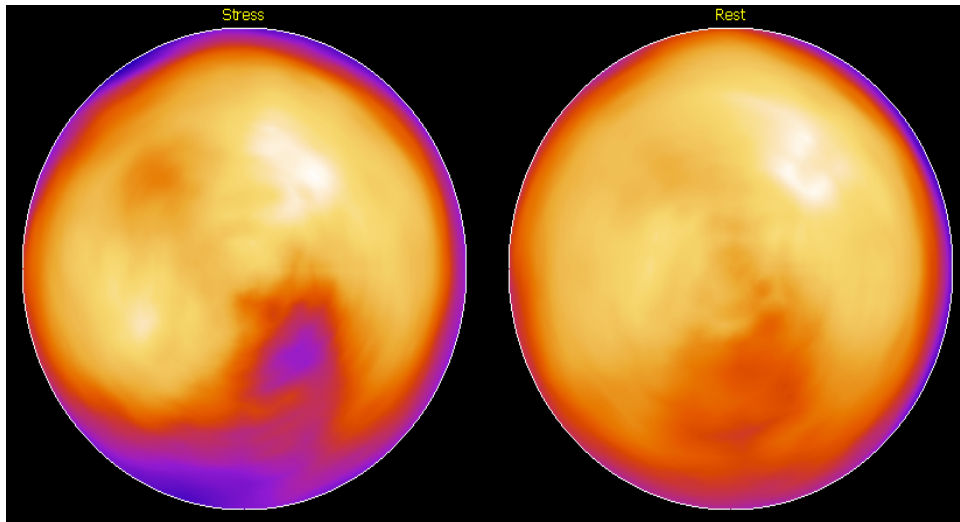
Input data $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$

Extracted features
Raw signal

Labels $\{\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_N\}$

AMI or not AMI
Ischemia or not ischemia

Another example



Classification of “heart” images

Input data $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$

Extracted features
Image (RGB pixel values)

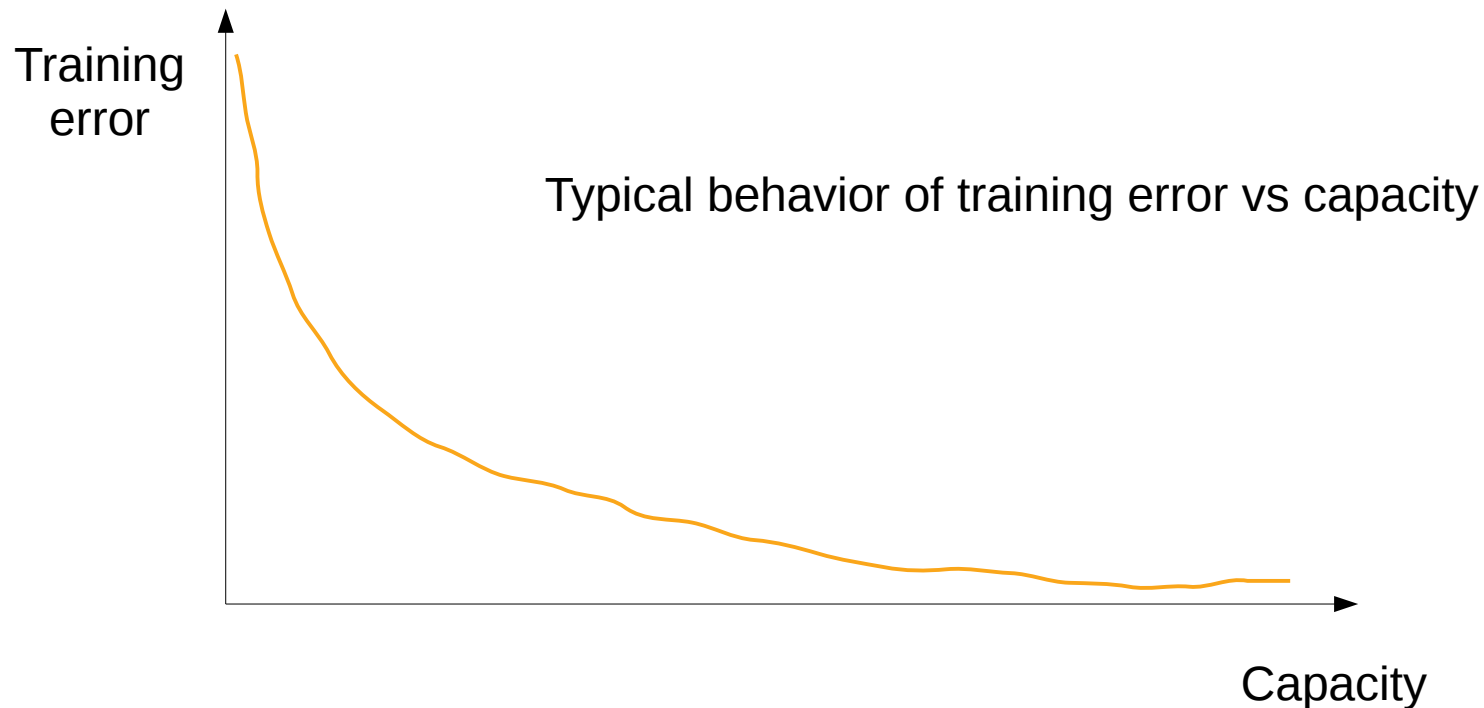
Labels $\{\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_N\}$

AMI or not AMI
Ischemia or not ischemia

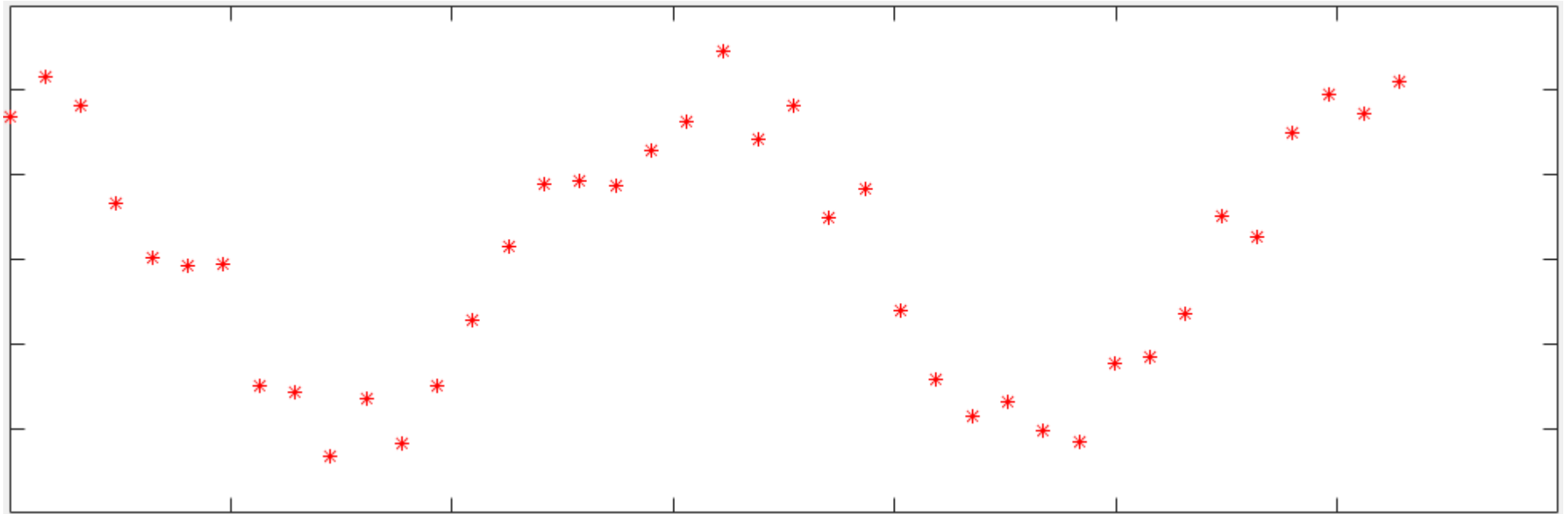
Capacity, overfitting and underfitting

We have a data set D of cases (data points) used to train a model.

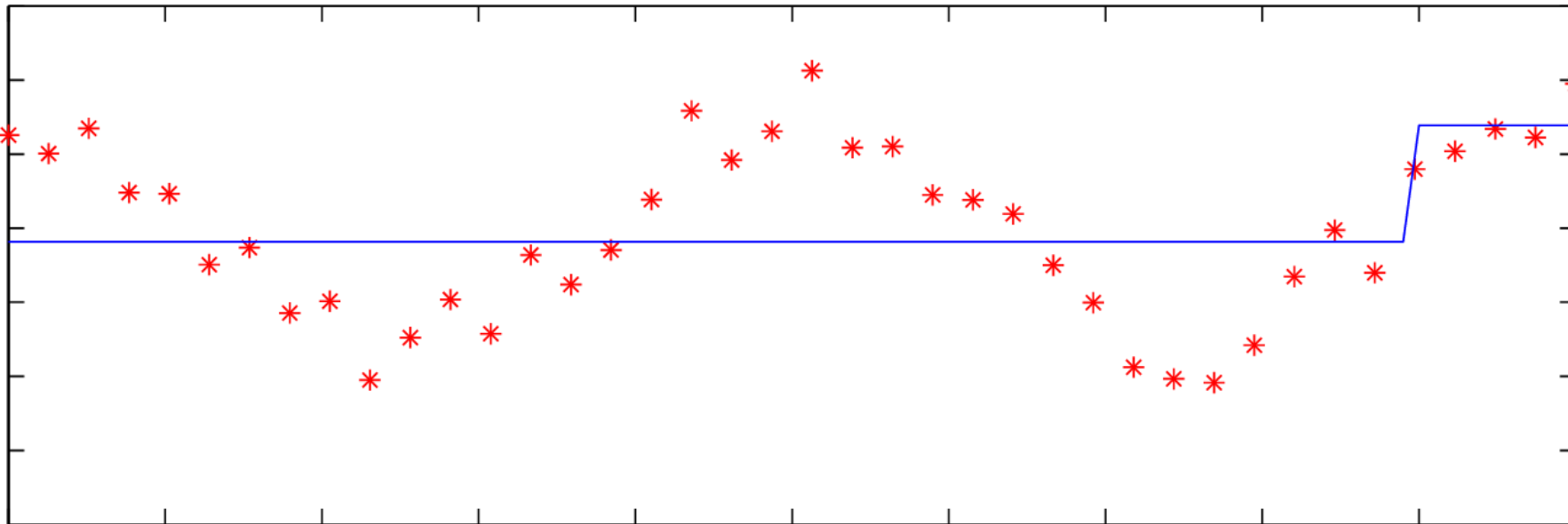
A model can be said to possess a certain capacity (= the ability to fit a range of different functions. With small capacity we can only fit a limited number of functions and increasing the capacity means fitting a larger set of functions).



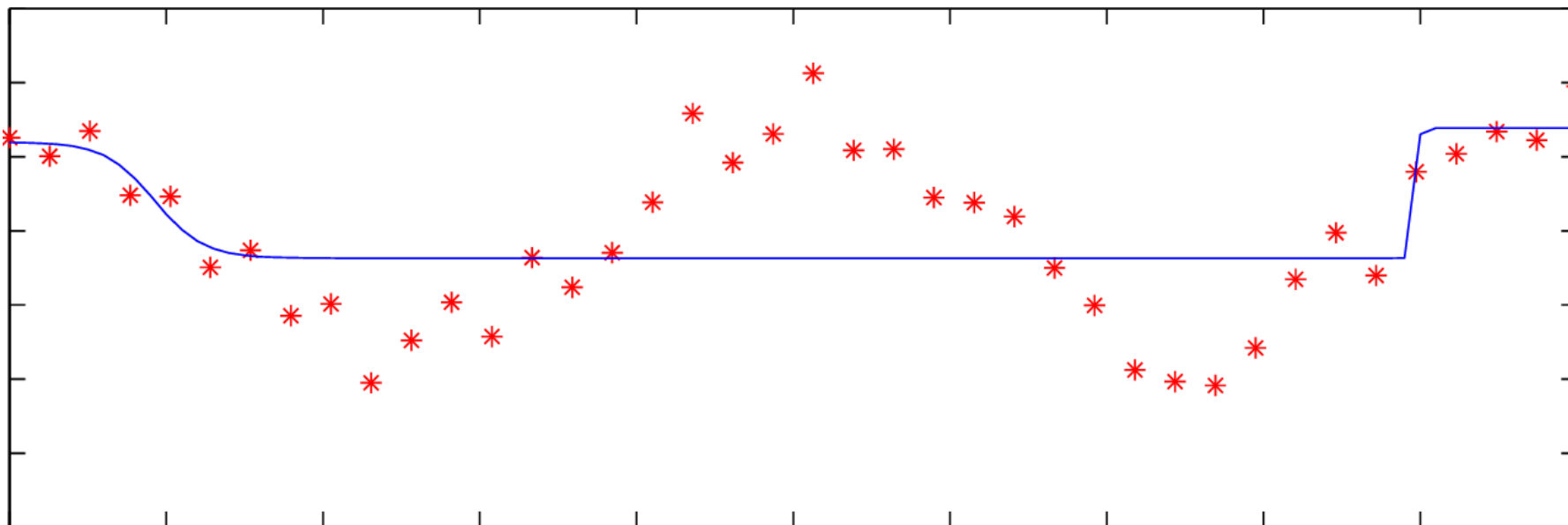
Simple curvefitting task!



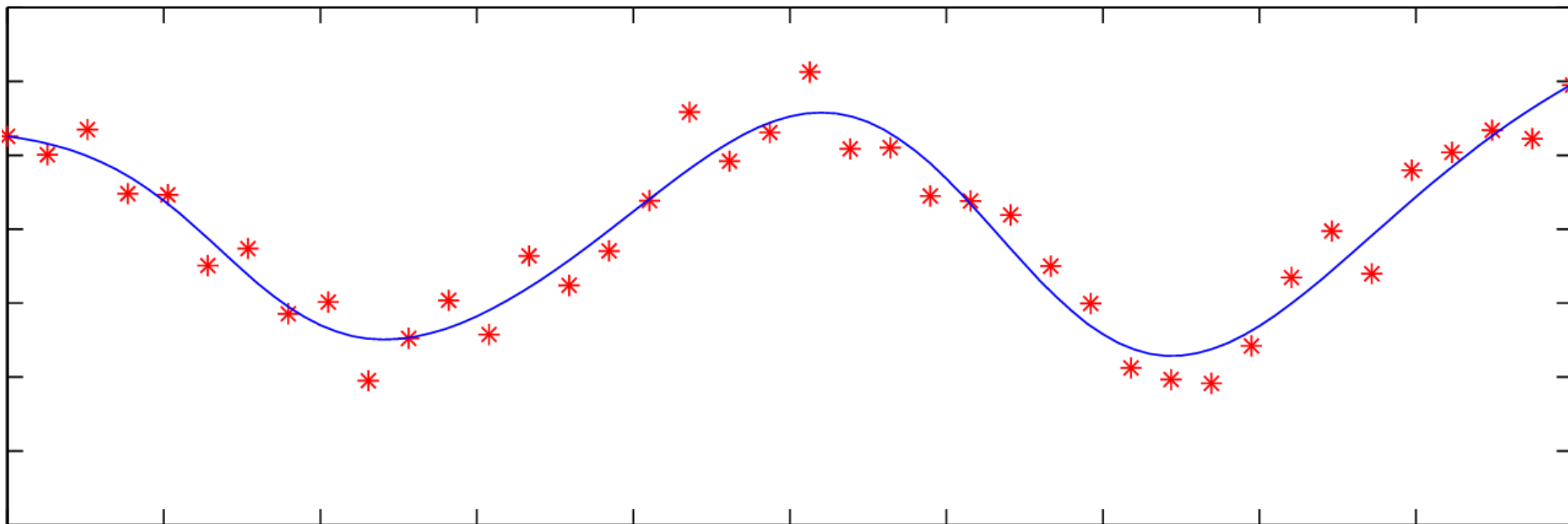
ML-model of degree "1"



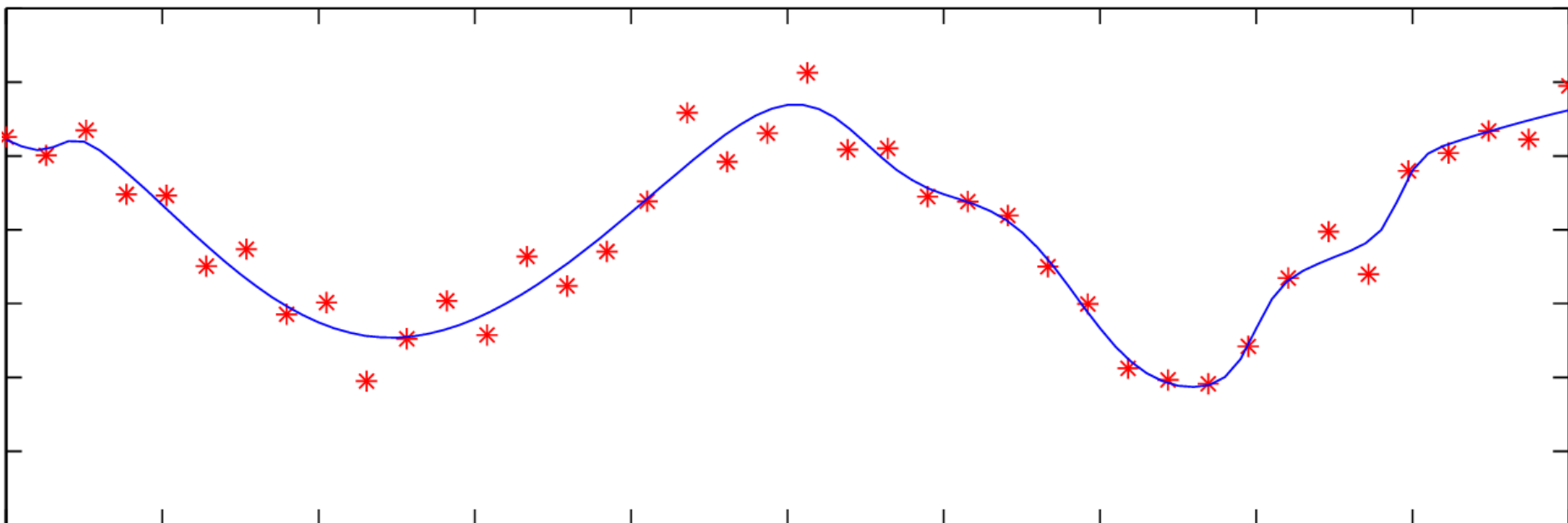
ML-model of degree "2"



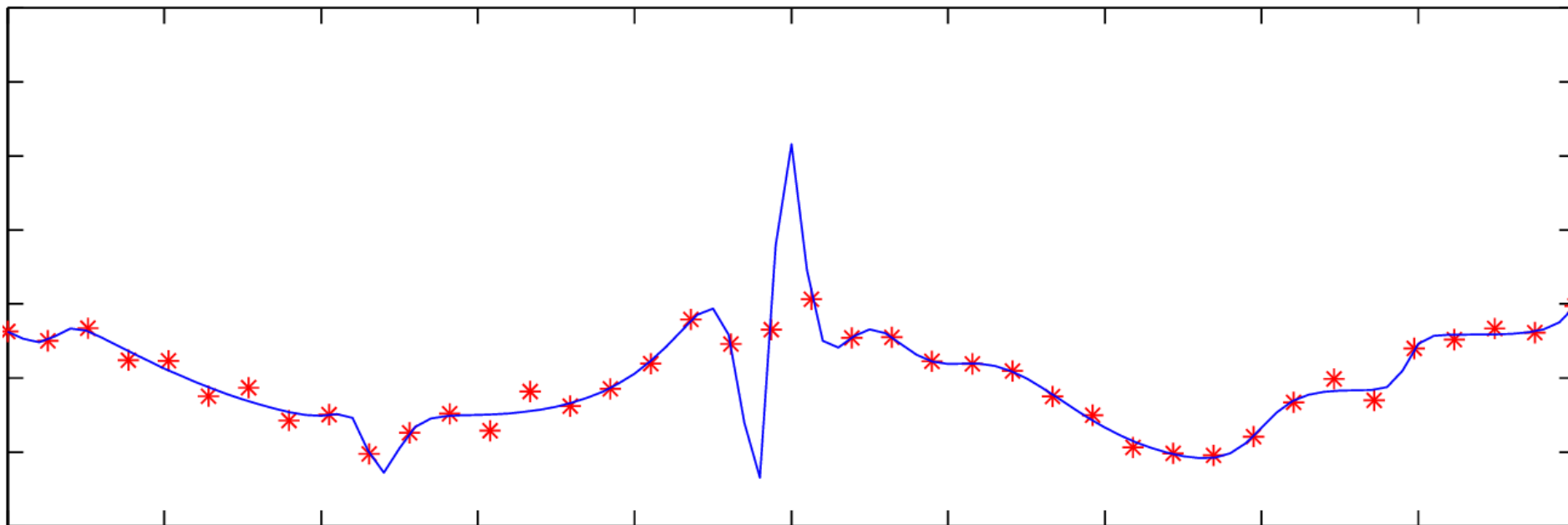
ML-model of degree "4"



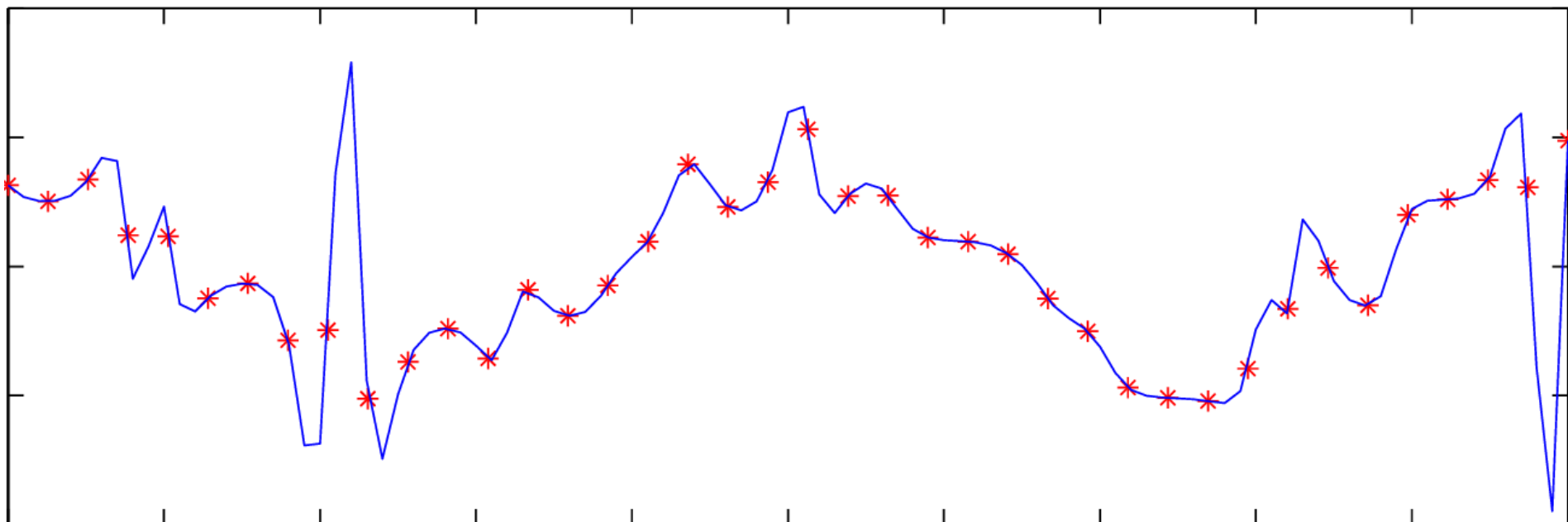
ML-model of degree "7"

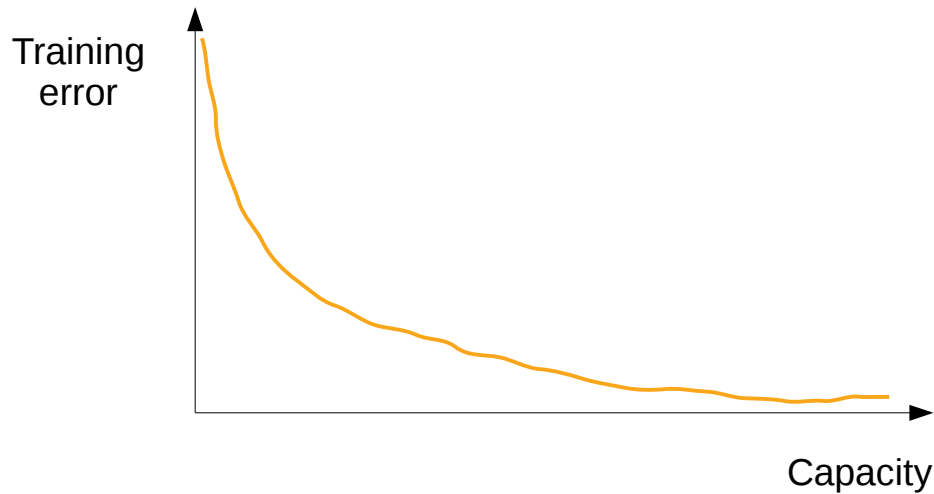


ML-model of degree "12"



ML-model of degree "25"





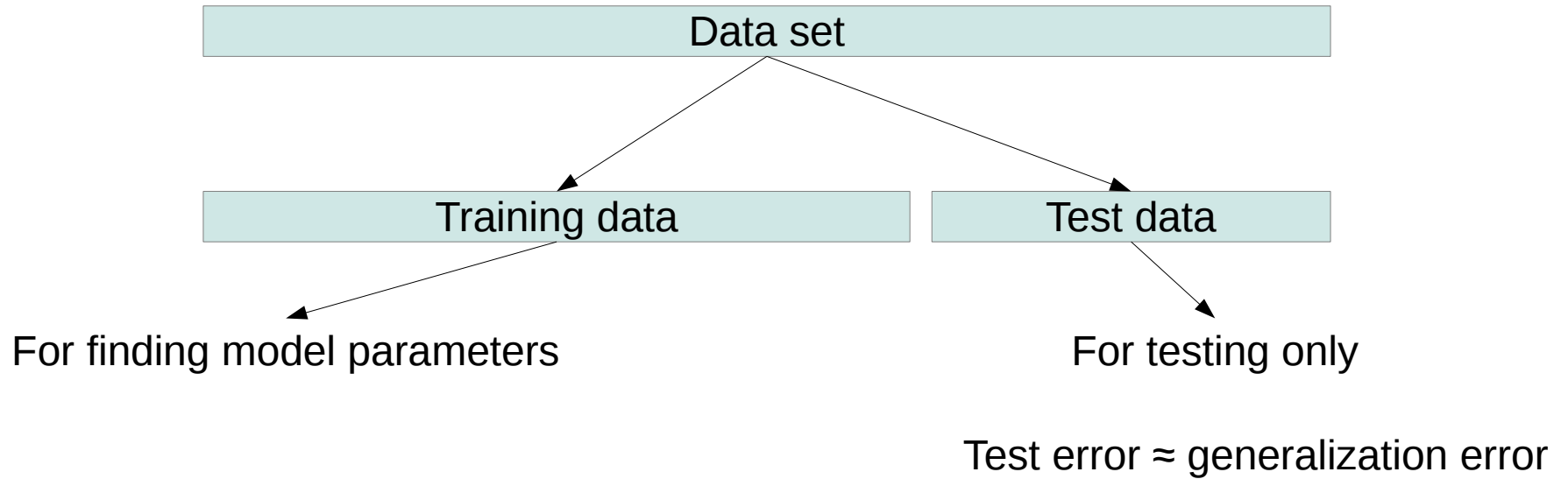
Problem: We cannot use training error or performance as an indicator how well my ML-model is doing!

The central challenge of machine learning is to perform well on **new previously unseen** data, **not** part of the training.

A model with such ability is said to **generalize**.

The **generalization performance/error** = expected performance/error on “new” data

Often,



We expect:

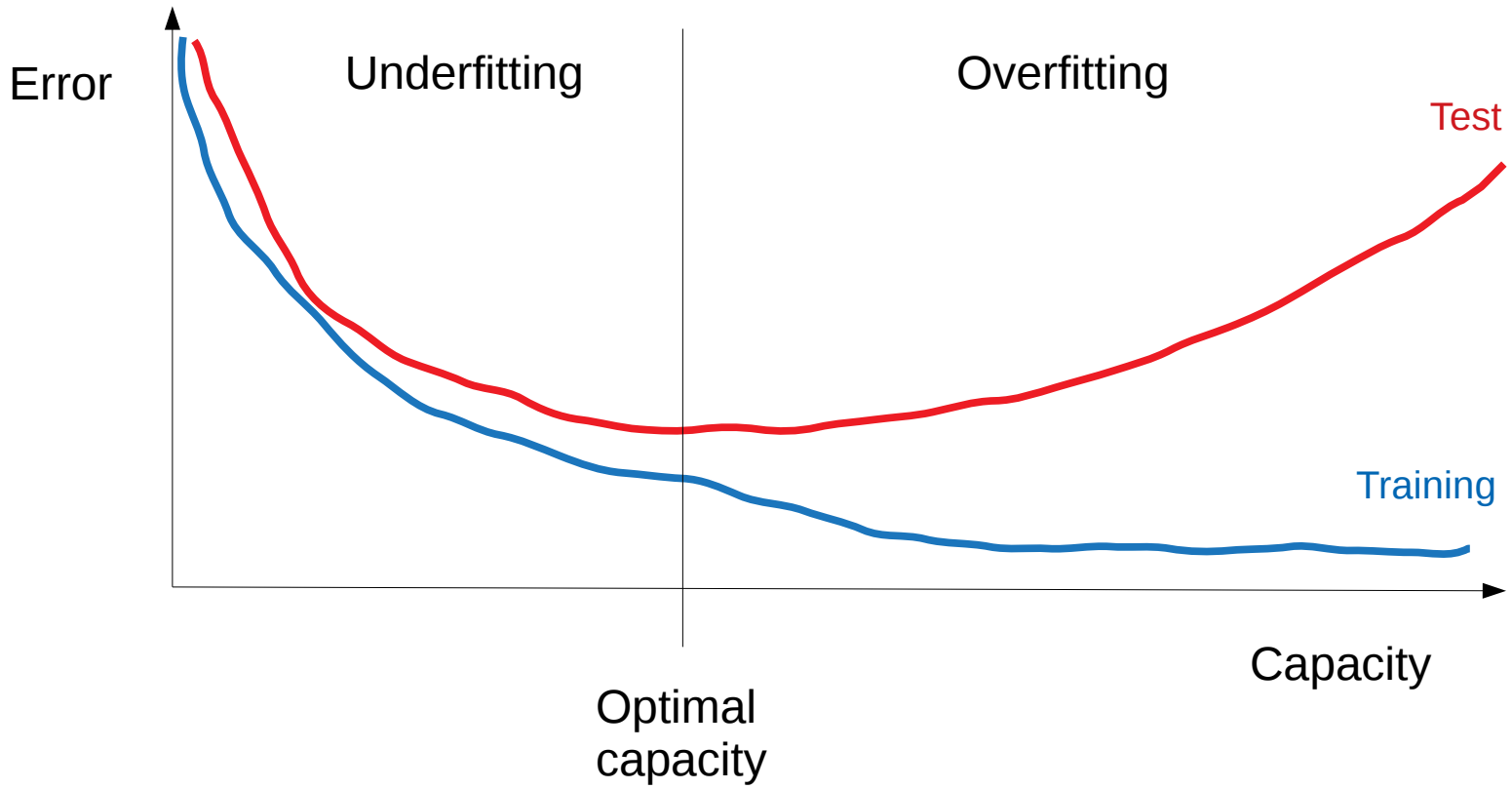
Test error \geq Training error

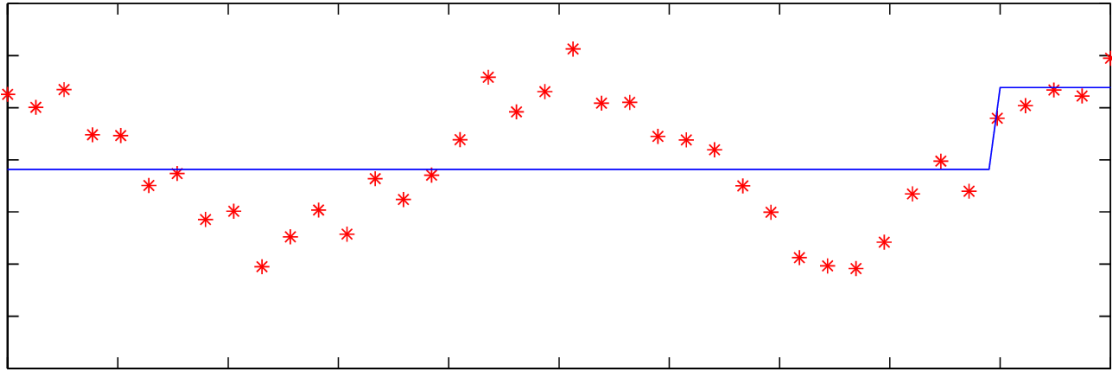
GOAL:

Minimize training error

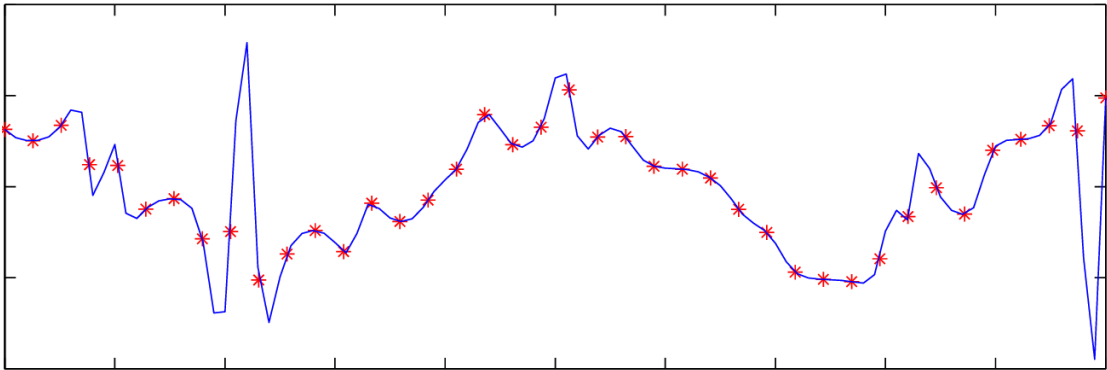
Make the gap between test error and training error as small as possible.

Overfitting and underfitting





Underfitting



Overfitting

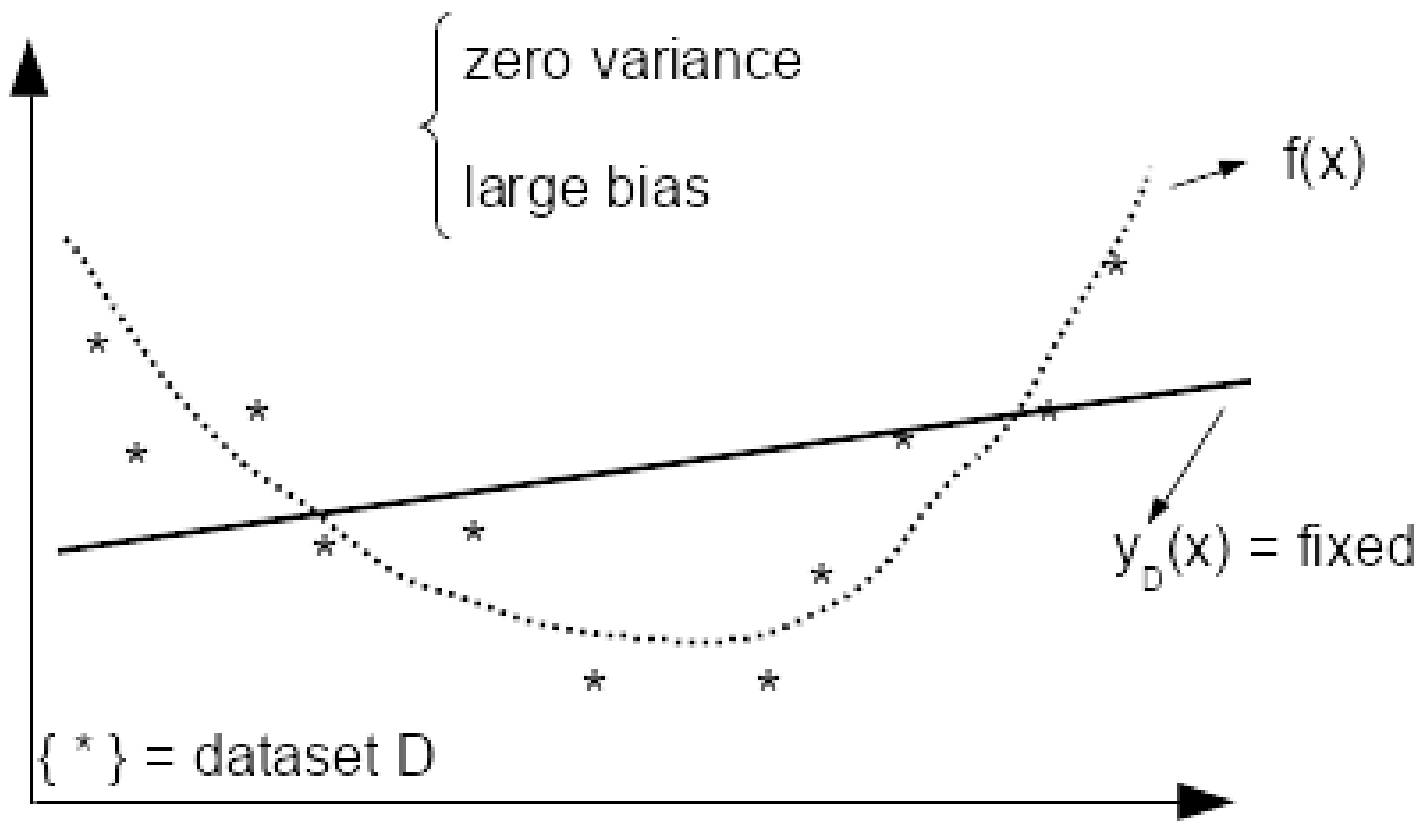
Bias-Variance tradeoff

D = Data set of size N

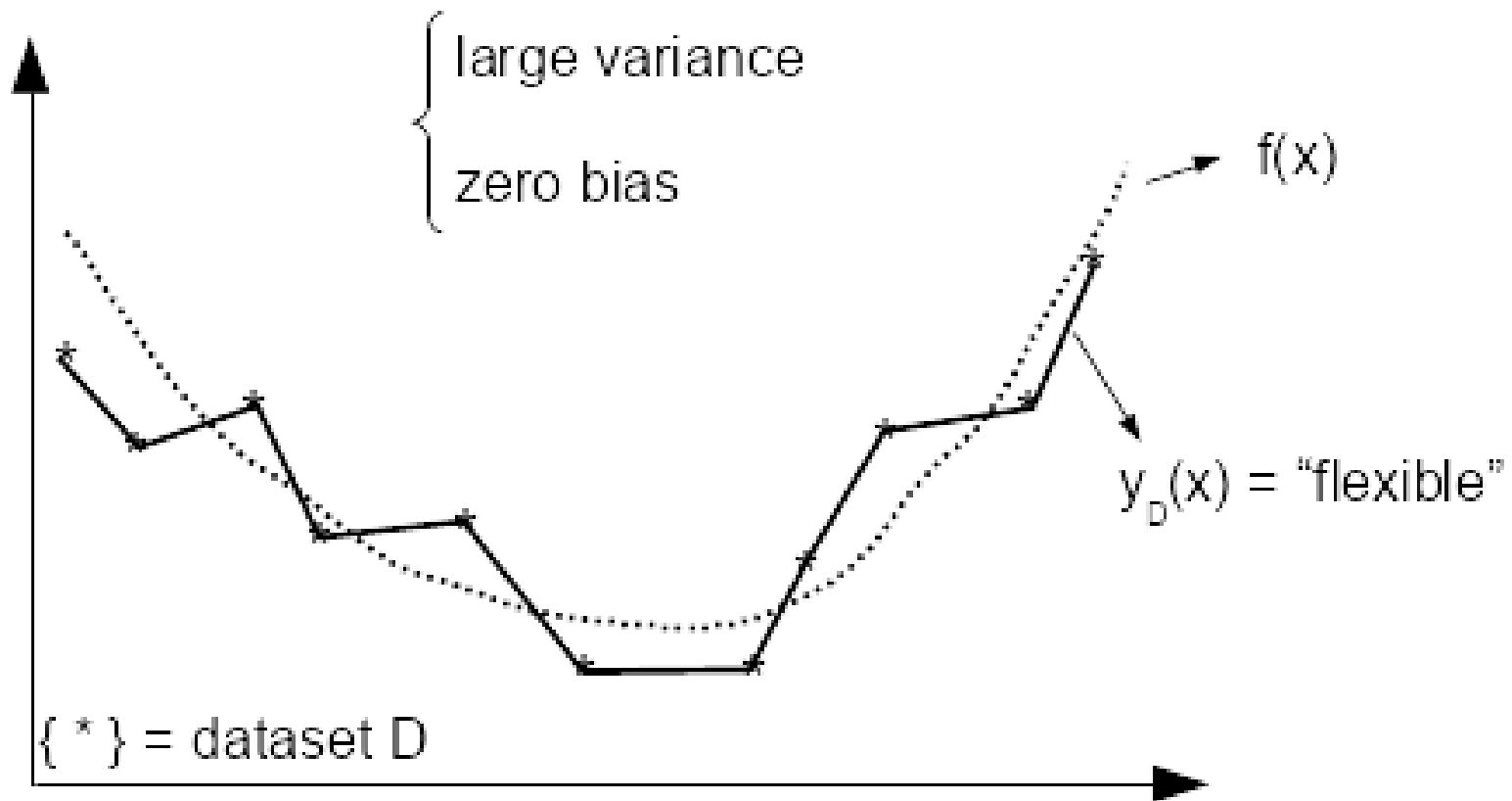
$y_D(\mathbf{x})$ = Model, trained on data set D

$\mathbf{E}[\cdot]$ = Expectation over many N -sized data sets D

$$\begin{aligned} \mathbf{E} \left[(y_D(\mathbf{x}) - f(\mathbf{x}))^2 \right] &= \\ &= \underbrace{\left(\mathbf{E} [y_D(\mathbf{x})] - f(\mathbf{x}) \right)^2}_{\text{bias}^2} + \underbrace{\mathbf{E} \left[(y_D(\mathbf{x}) - \mathbf{E} [y_D(\mathbf{x})])^2 \right]}_{\text{variance}} \end{aligned}$$



Underfitting situation



Overfitting situation

How do we find a balance between underfitting and overfitting?

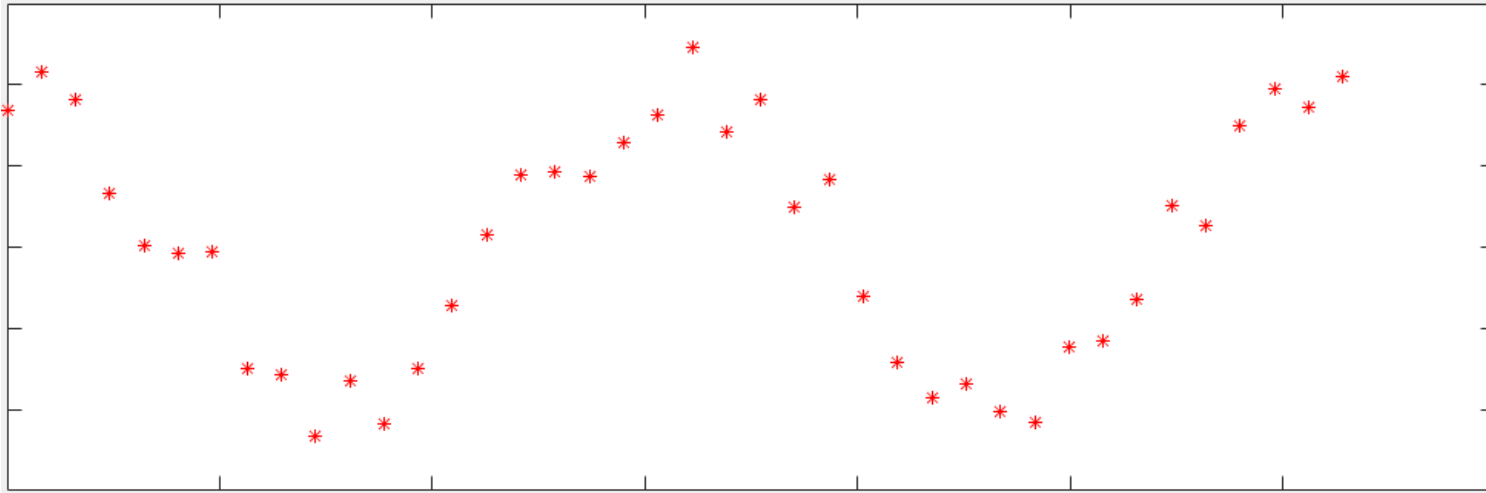
We want an efficient way of modifying the capacity of an ML model.

Regularization: Attempts to modify architectures, error functions or learning algorithms in order to reduce the capacity of a model.

Example: L2 norm regularization

$$E(\mathbf{w}) = \text{Loss function} + \alpha \|\mathbf{w}\|^2$$

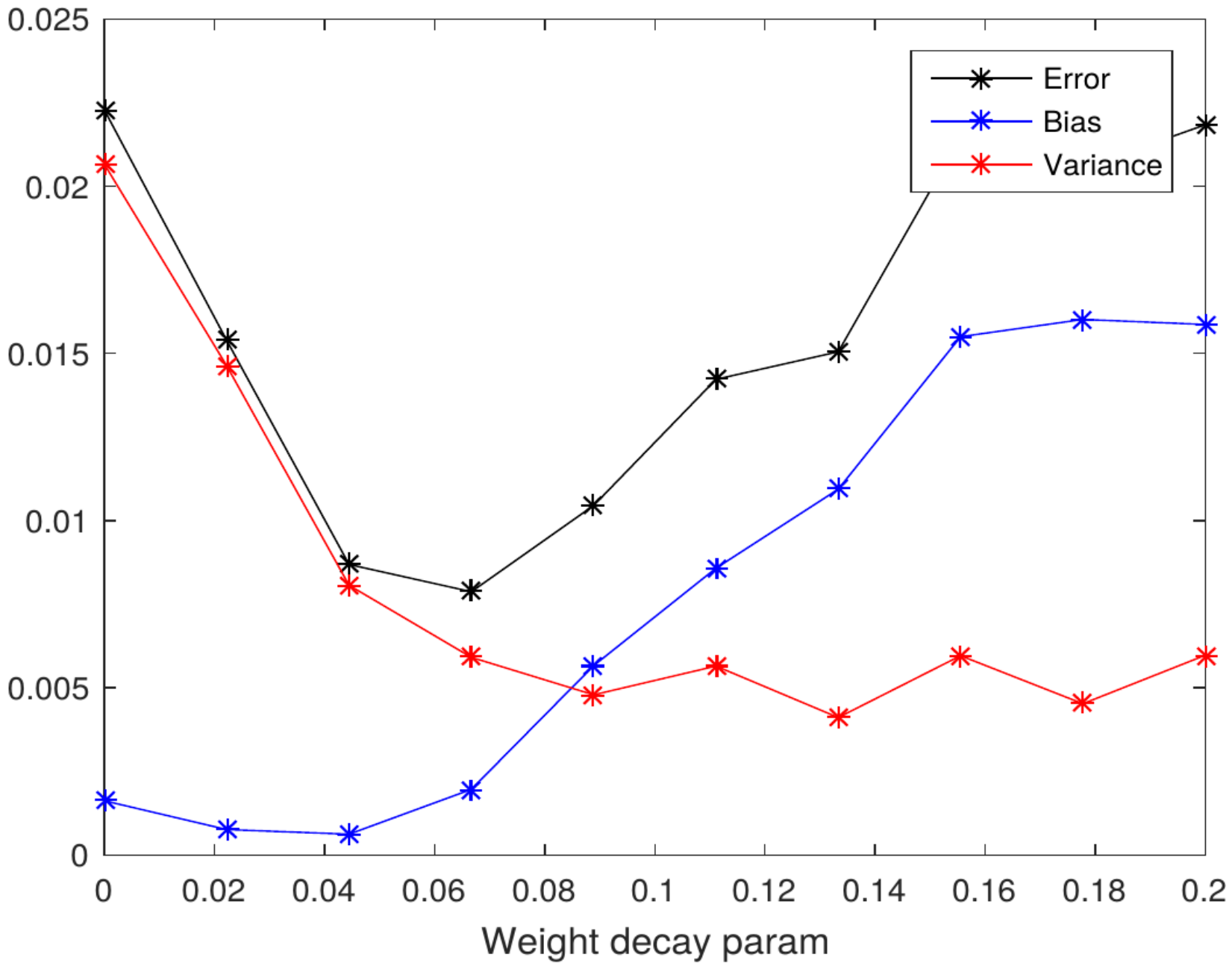
Simple regression problem



Could be $y(\mathbf{w}, \mathbf{x}) = w_0 + w_1x + w_2x^2 + w_3x^3 + \dots$

But in this case $y(\mathbf{w}, \mathbf{x}) = \sum_{j=1}^M w_j \tanh(x\tilde{w}_j + b_j) + w_o$

Minimize $\sum_n (y(\mathbf{w}, \mathbf{x}_n) - d_n)^2 + \alpha \underbrace{\|w_0, w_1, \dots, b_M\|^2}_{\text{Weight decay or L2}}$



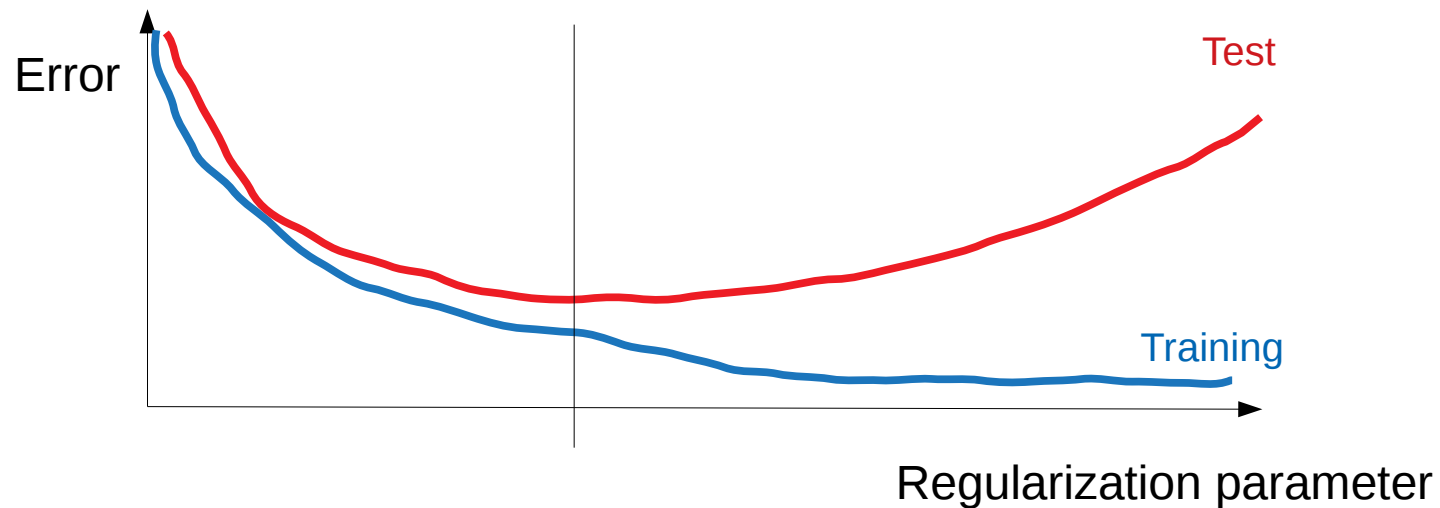
Hyper parameters and model selection

We have additional parameters that controls the ML model:

Size of the model: e.g. degree or number of hidden layers

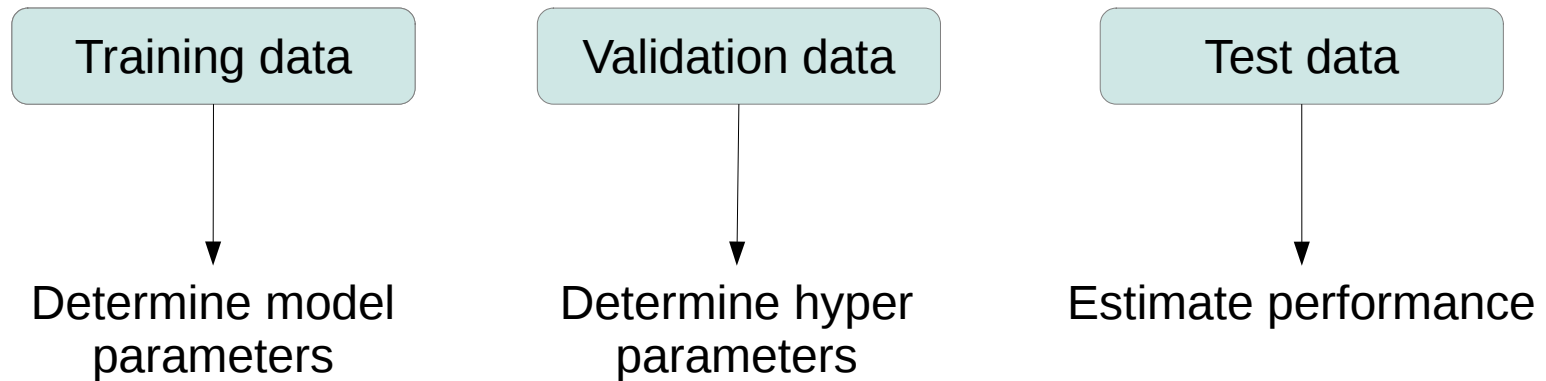
Regularization parameters: e.g. size of L2

Model selection = determination of hyper parameters



Model selection requires estimation
of generalization performance!!

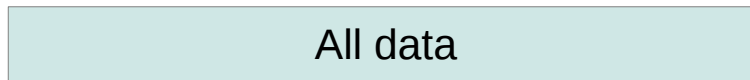
How can we do that?



Various approaches to estimate generalization performance

The holdout method

All data

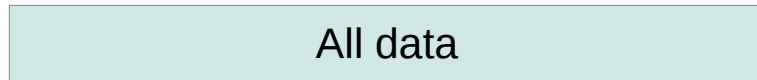


Training Validation

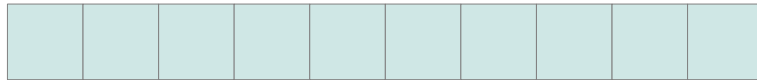


The K-fold cross validation method

All data



Split into K parts



...

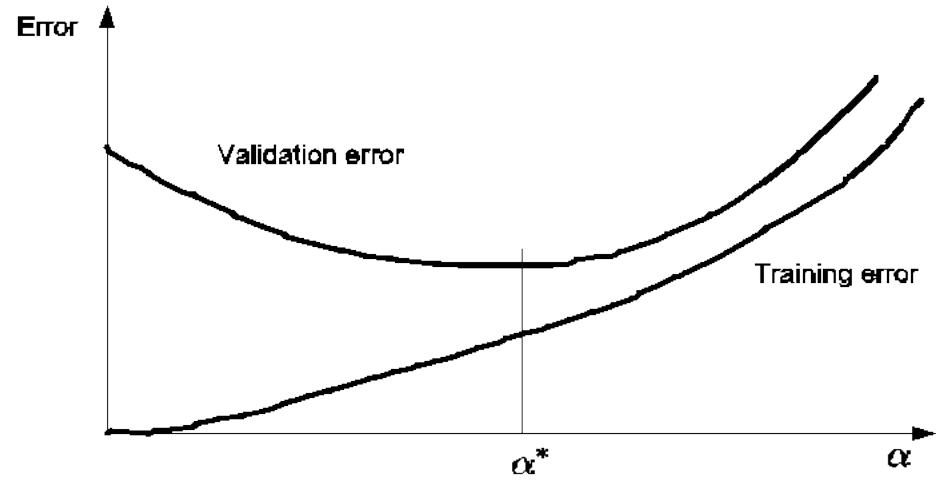


$$\hat{P} = \frac{1}{K} \sum_i P_i$$

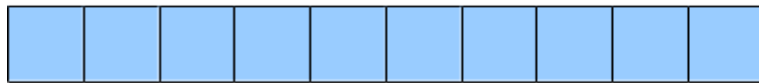
Sometimes repeat N times

$$\hat{P} = \frac{1}{NK} \sum_n \sum_i P_{in}$$

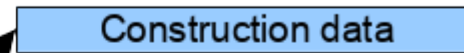
Nested K-fold cross validation!



Split into K parts



...



Split into M parts



...



Important!

