Introduction to Machine Learning?

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,

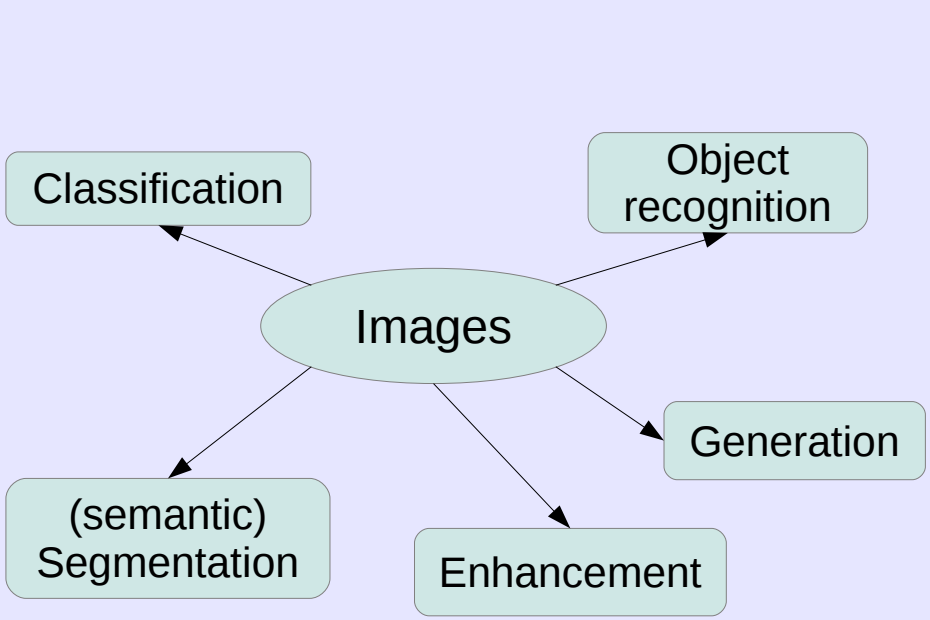
if its performance at tasks in T, as
measured by P, improves with
experience E.

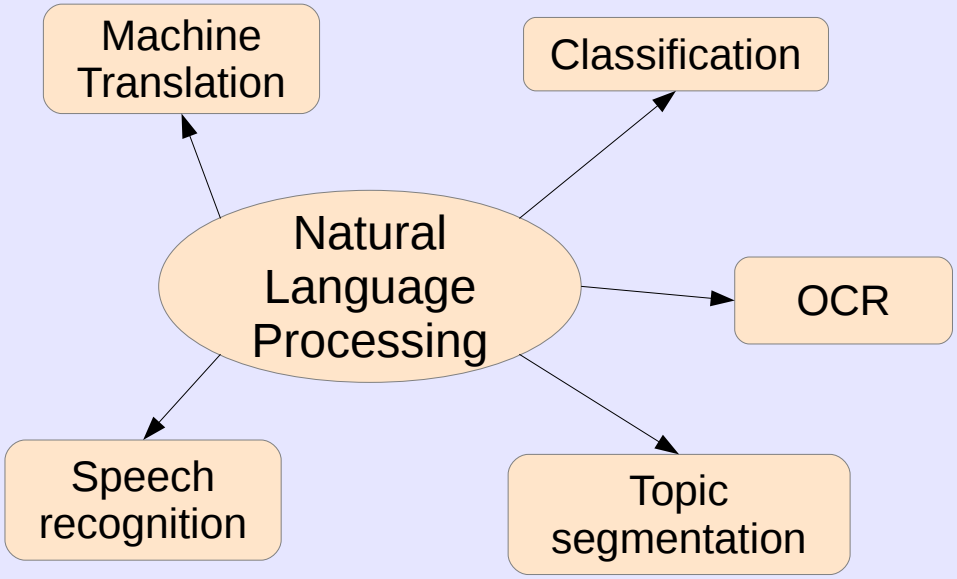
- Tom Mitchell

Contents of this lecture

- Introduction
- (E, T, P)
- Capacity, overfitting and underfitting
- Hyper parameters and model selection
- Maximum Likelihood
- Stochastic gradient descent

Some application areas





Healthcare

Bioinformatics

Data security

Smart cars

Financial trading

Recommendations

Online search

Fraud detection

Playing games

A famous example!

Each digit is a 28x28 “grayscale” image.

60 000 images in the training set.

10 000 image in the test set.

Best result so far are based on (deep) convolutional neural networks
or deep neural networks with an error rate of ***0.21%*** (21 misclassified images)

Some ML models

Neural Networks

Random Forest

Support Vector Machine

Decision Tree

PCA

Simulated Annealing

Naive Bayes

Bayesian Network

Genetic Algorithms

K-means

K-Nearest Neighbors

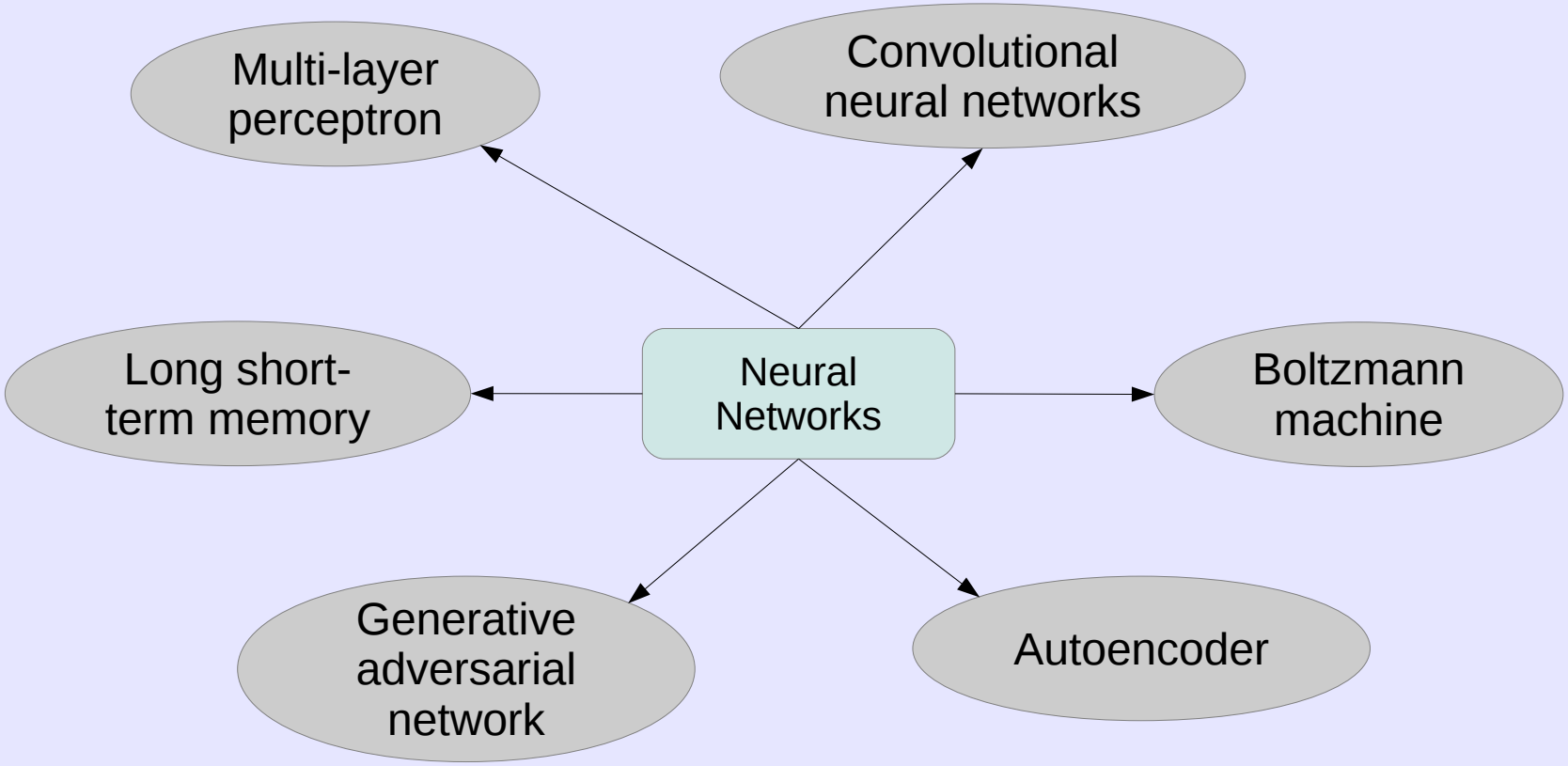
Logistic Regression

Boosting

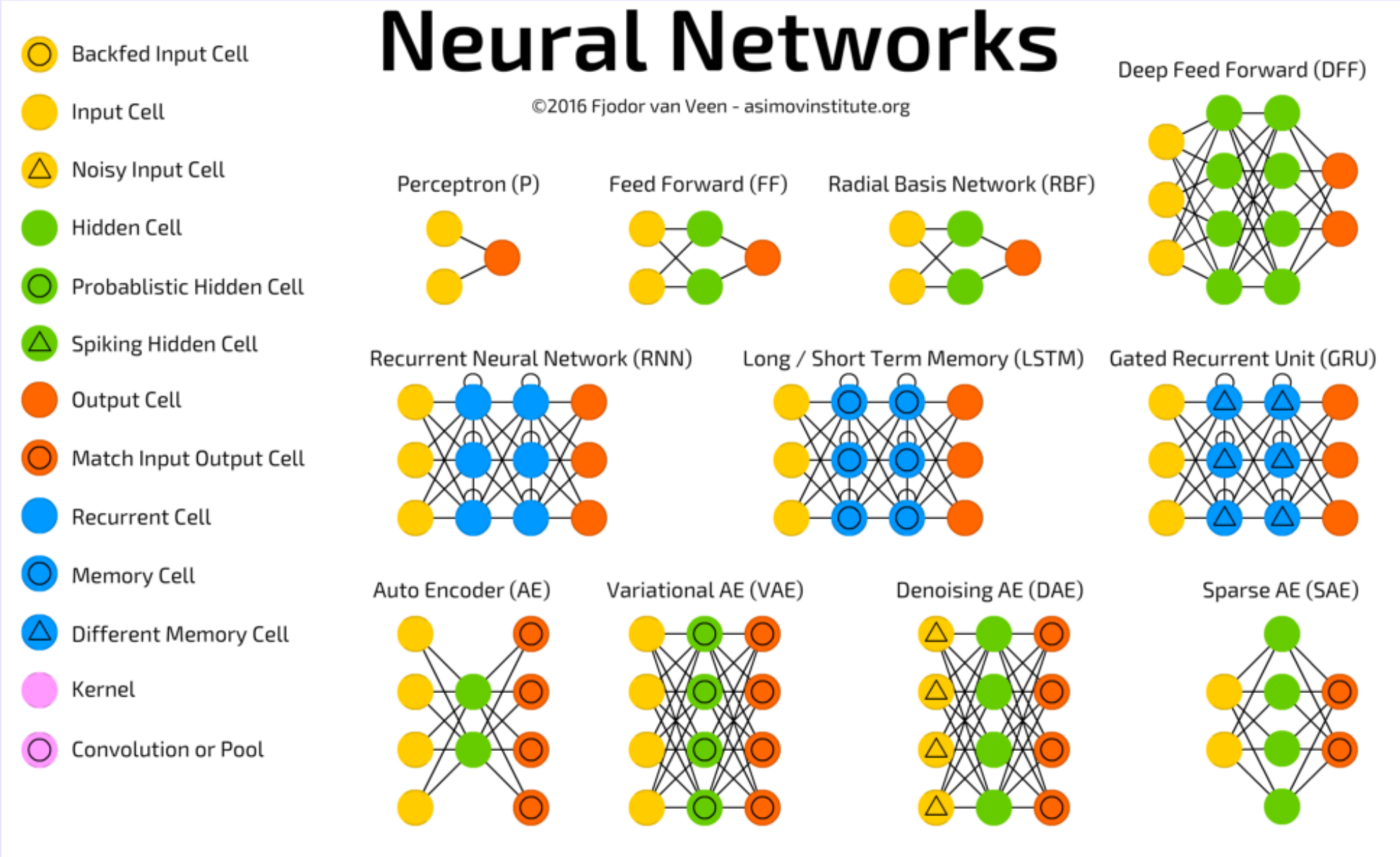
Factor Analysis

ICA

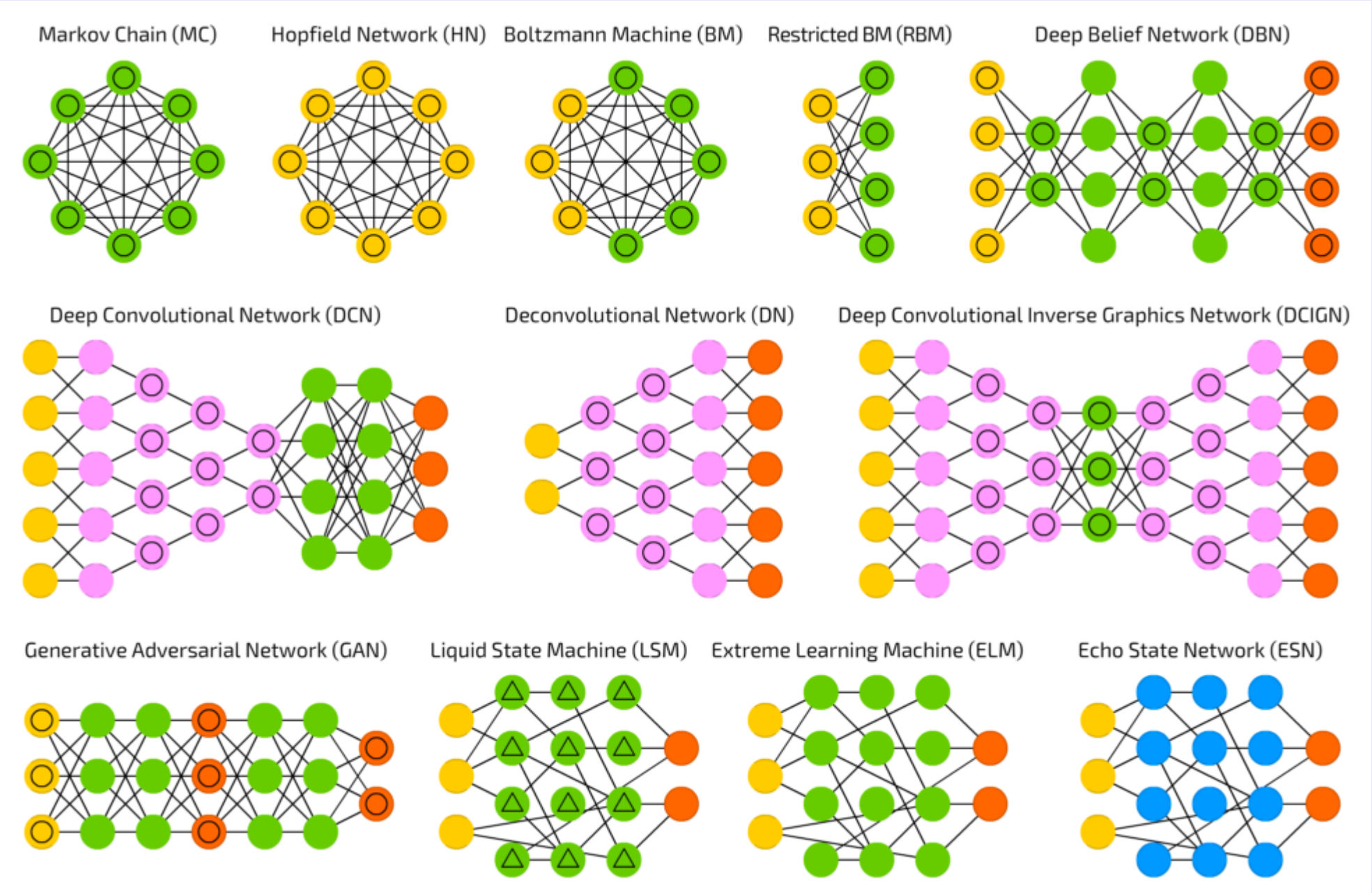
Hidden Markov Model


Multi-layer perceptron
Convolutional neural networks
Long short-term memory
Neural Networks
Boltzmann machine
Generative adversarial network
Autoencoder

+ many more...



http://www.asimovinstitute.org/neural-network-zoo/



http://www.asimovinstitute.org/neural-network-zoo/

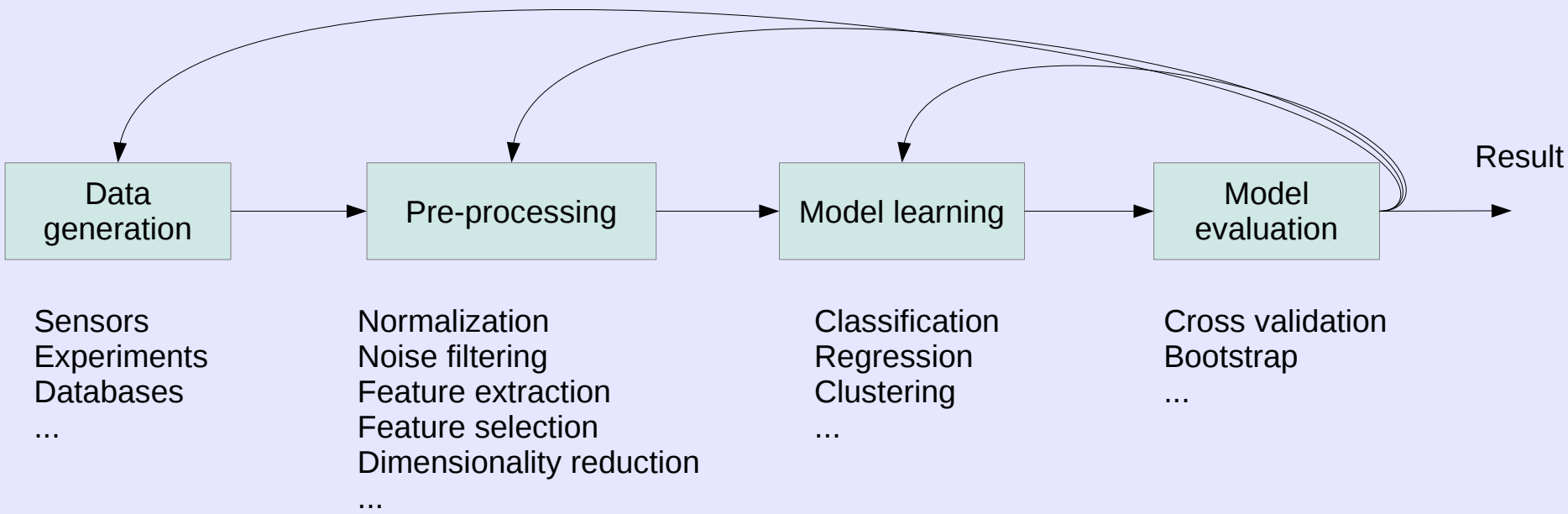
According to Tom Mitchell

Learning =

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

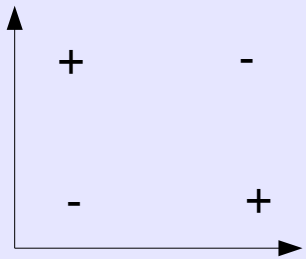
Learning!

Typical learning process

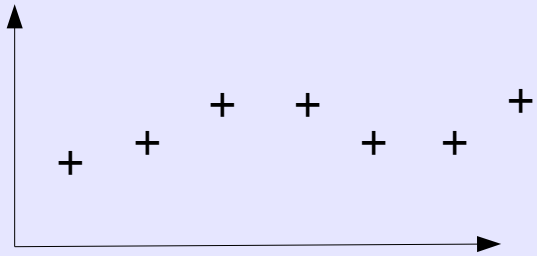


Tasks (*T*)

Some common tasks:



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

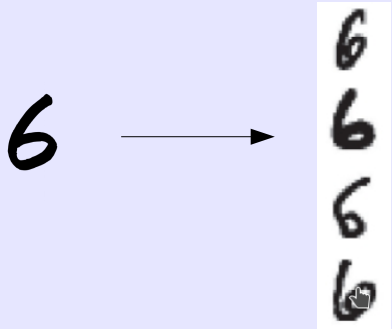


Regression: Predict a numerical output given an input

Translation: sequence in → sequence out

It's raining cats and dogs →
Det regnar katter och hundar

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



Performance measures (P)

Accuracy: 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:

Supervised learning and Unsupervised learning

For all machine learning problems we a data set

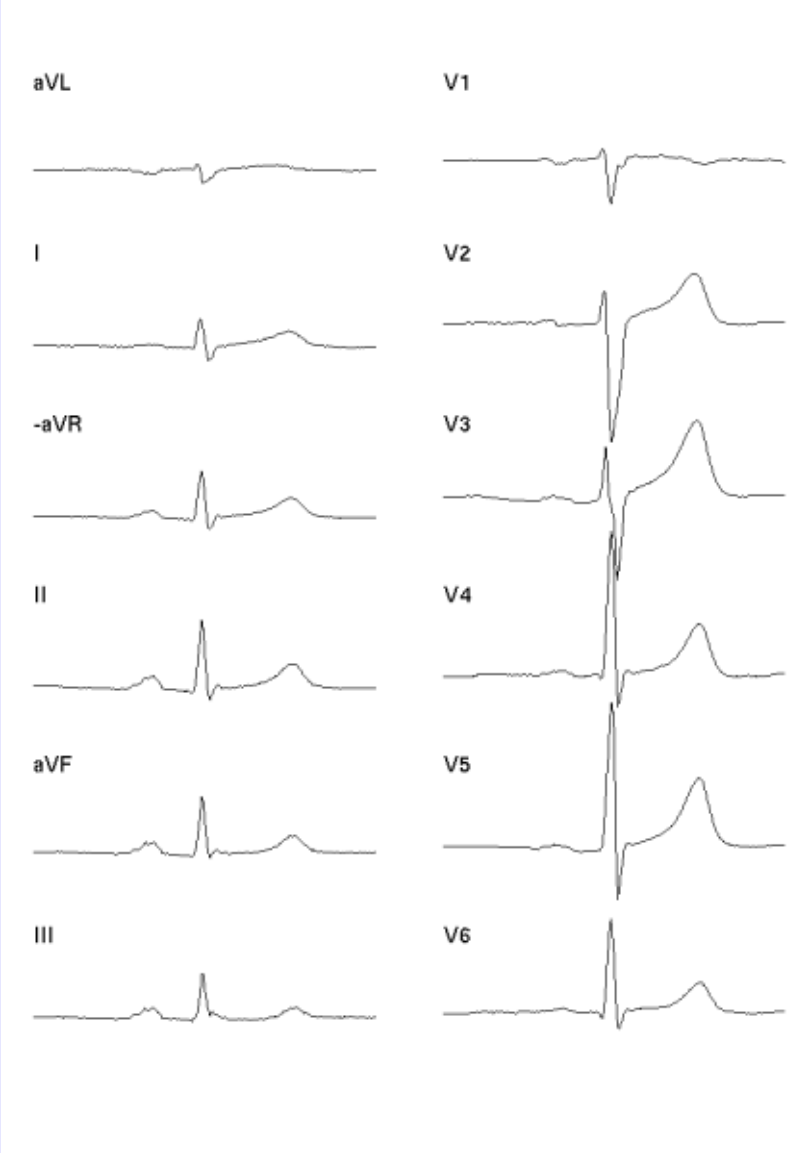
$$\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\} \longrightarrow \text{Unsupervised learning}$$

If we add labels

$$\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\} + \{\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_N\} \longrightarrow \text{Supervised learning}$$

An example

Classification of ECGs



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

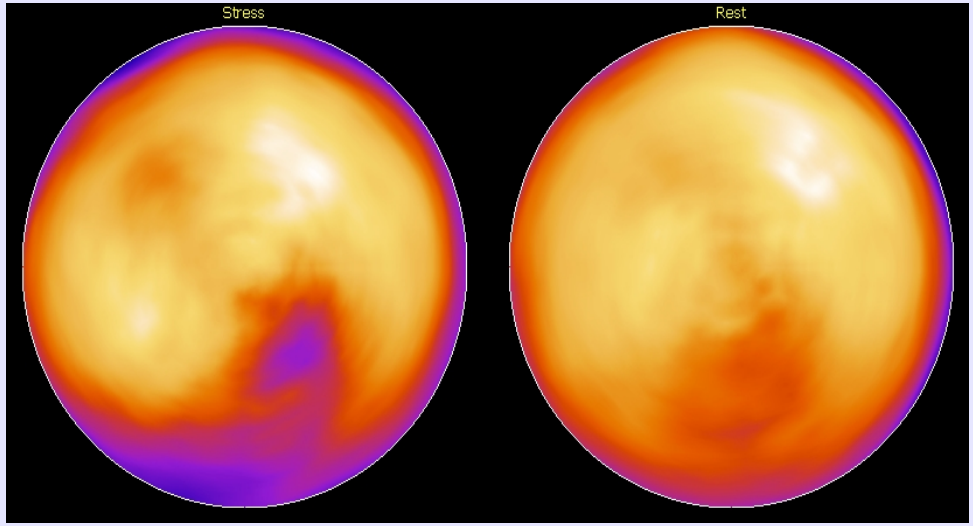
Extracted features
Raw signal

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

AMI or not AMI
Ischemia or not ischemia

Another example

Classification of “heart” images



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

Extracted features
Image (RGB pixel values)

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

AMI or not AMI
Ischemia or not ischemia

Experience (E)

Another very important learning algorithm is:

Reinforcement learning

It can be described as an interaction between the learning algorithm and its environment.

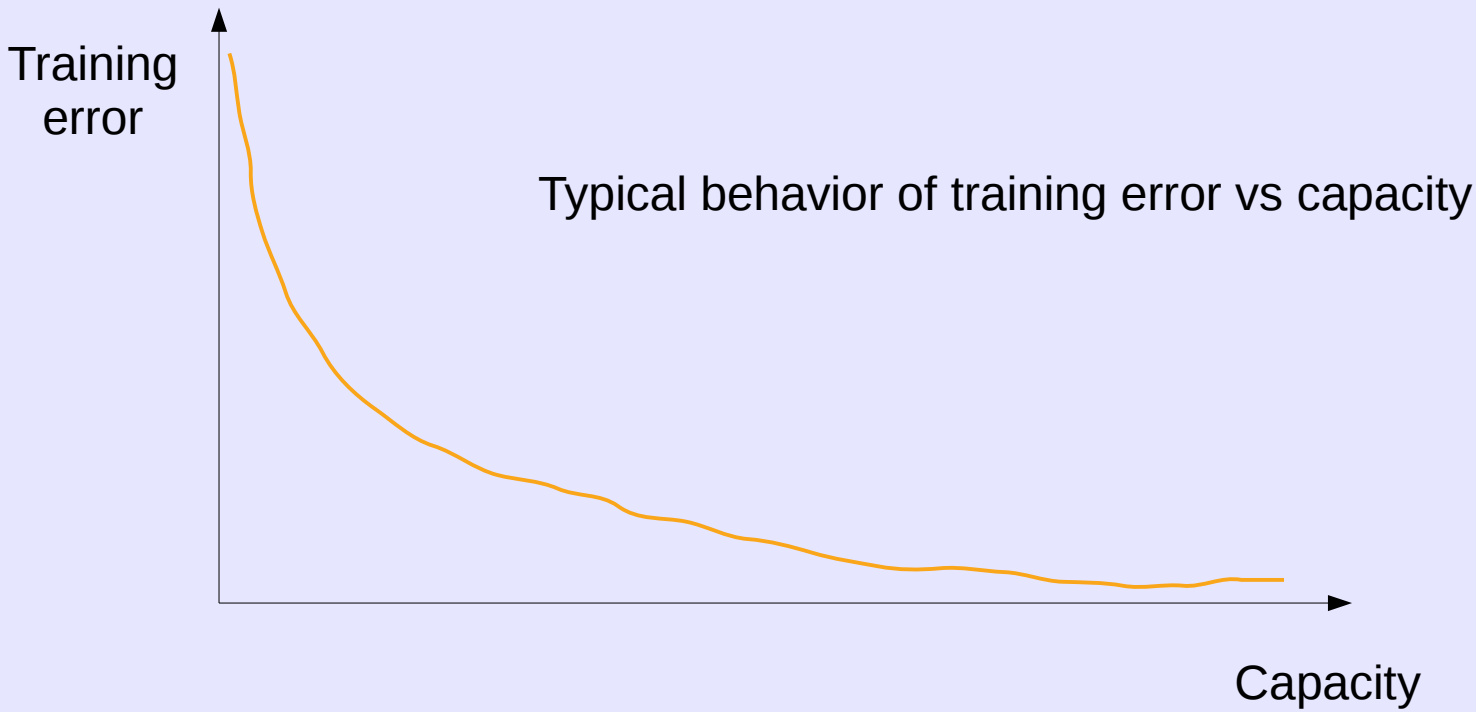
The data set is not fixed!

$$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$$

Capacity, overfitting and underfitting

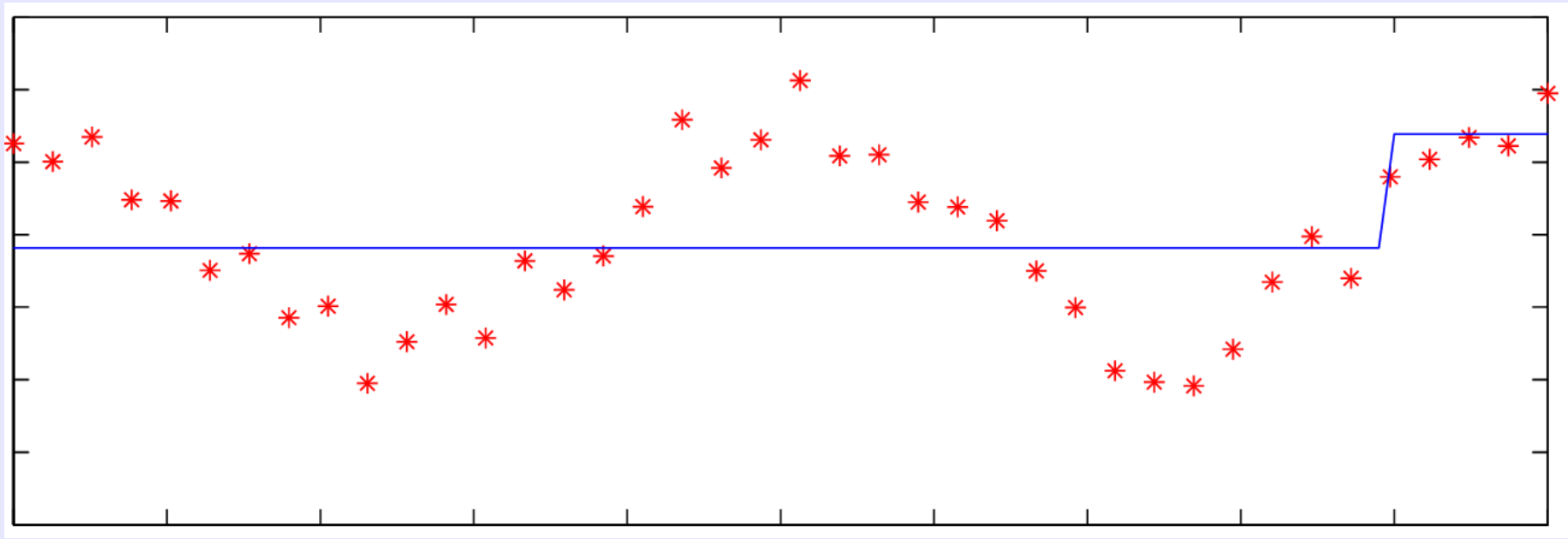
We have a data set *D* of cases (data points) used to train a model.

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).

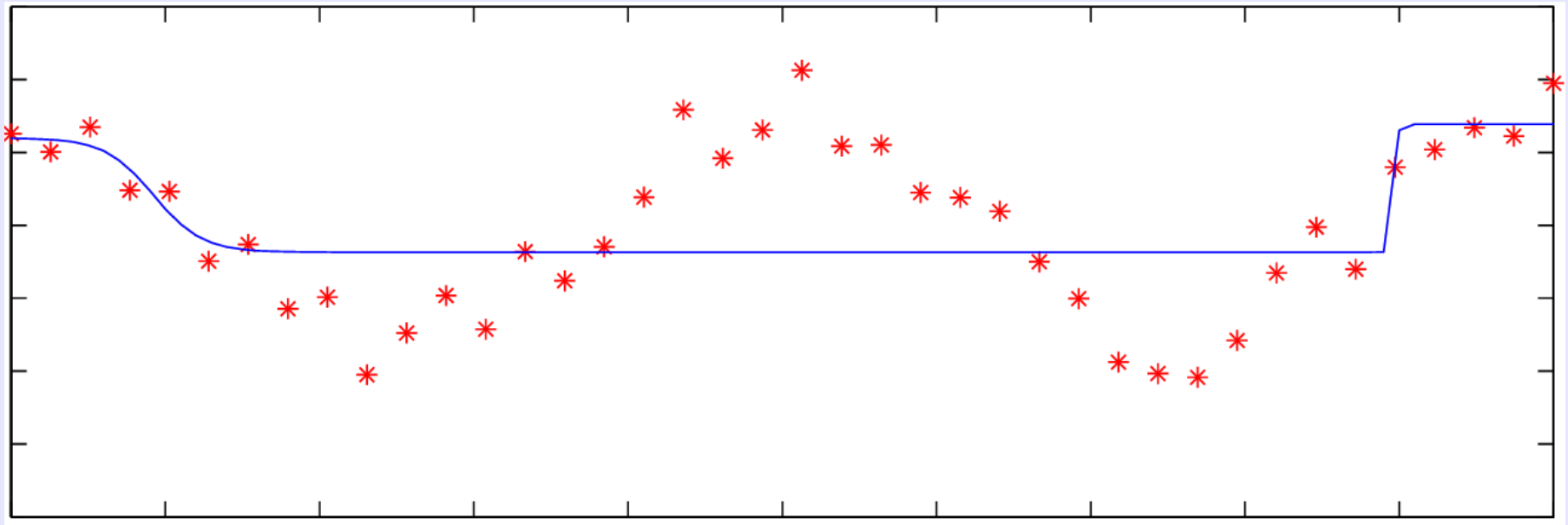


Typical behavior of training error vs capacity

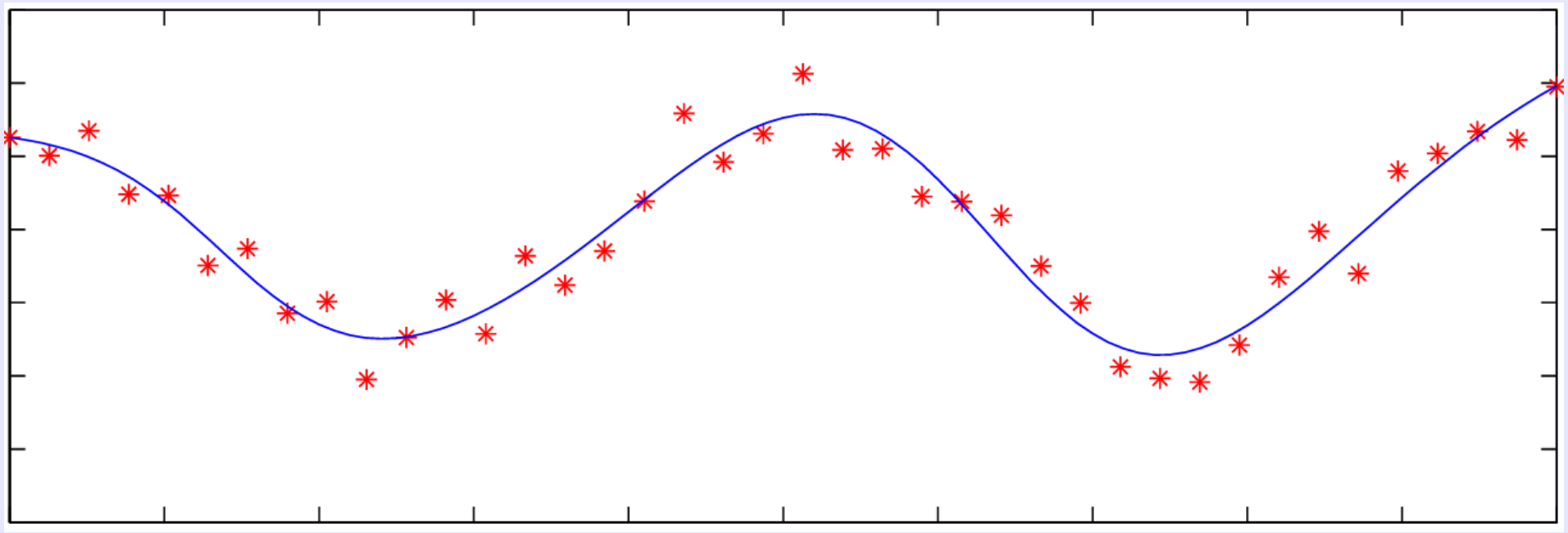
ML-model of degree “1”



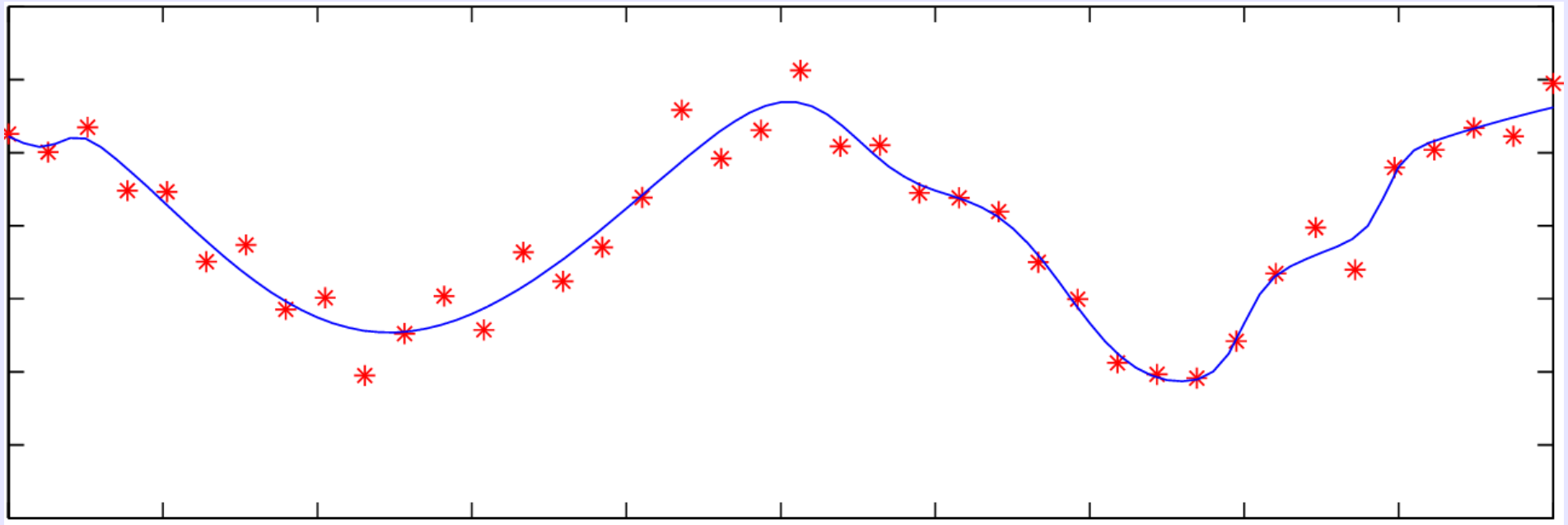
ML-model of degree “2”



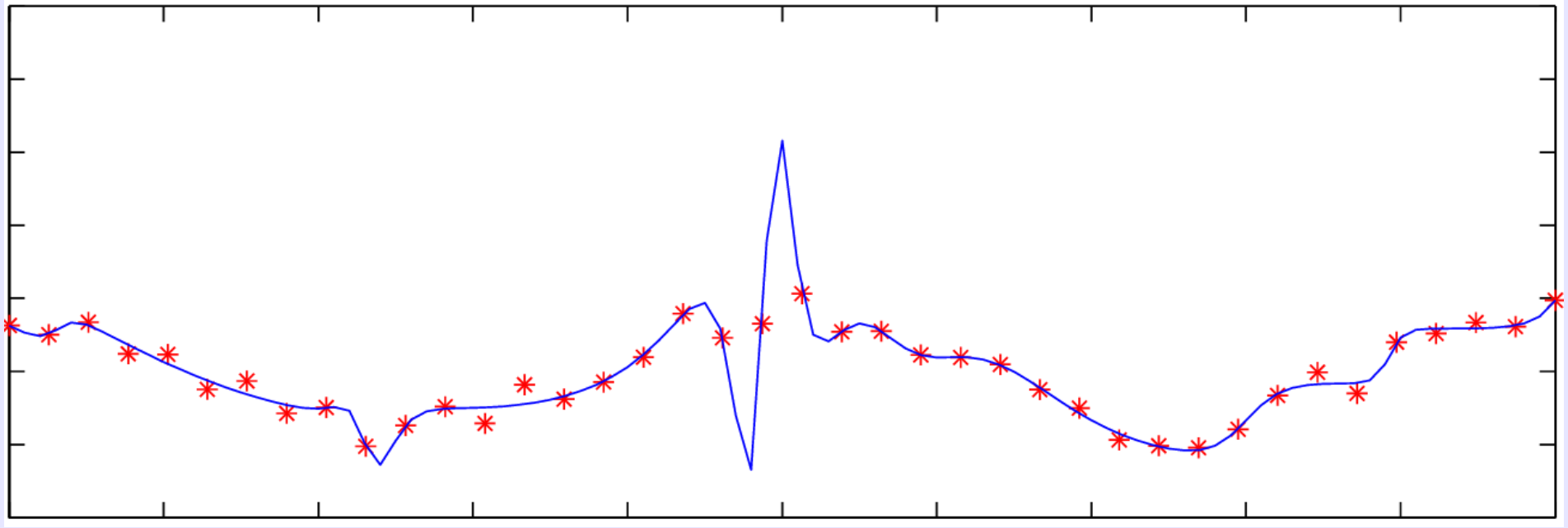
ML-model of degree “4”



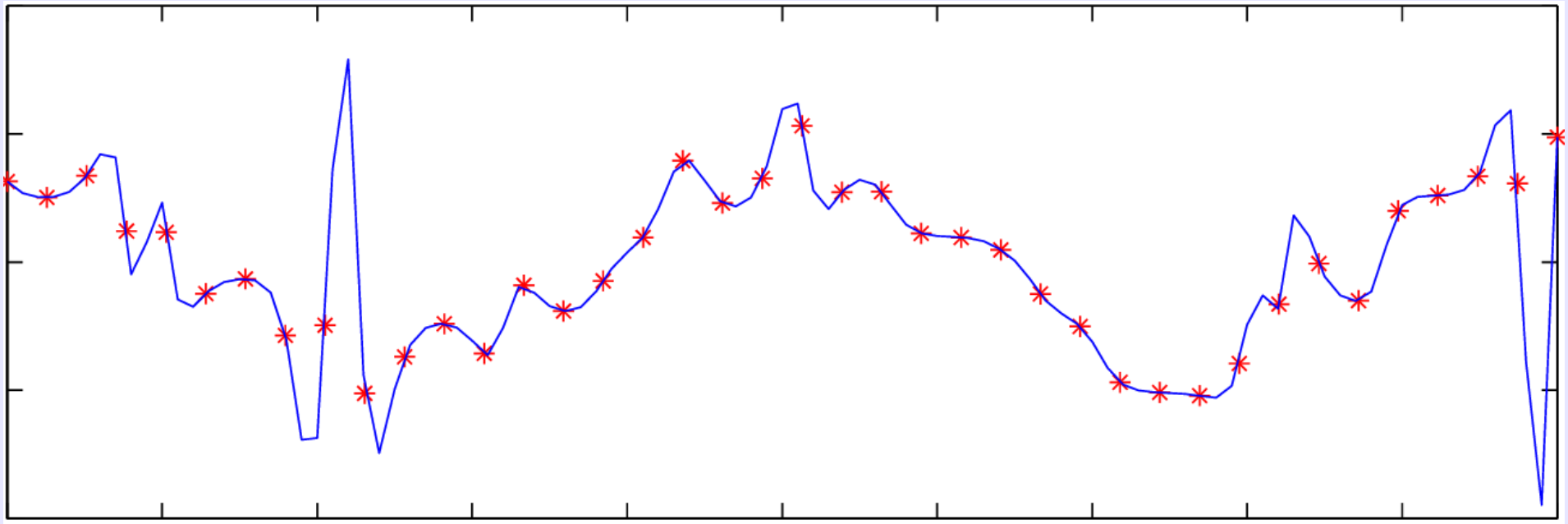
ML-model of degree “7”

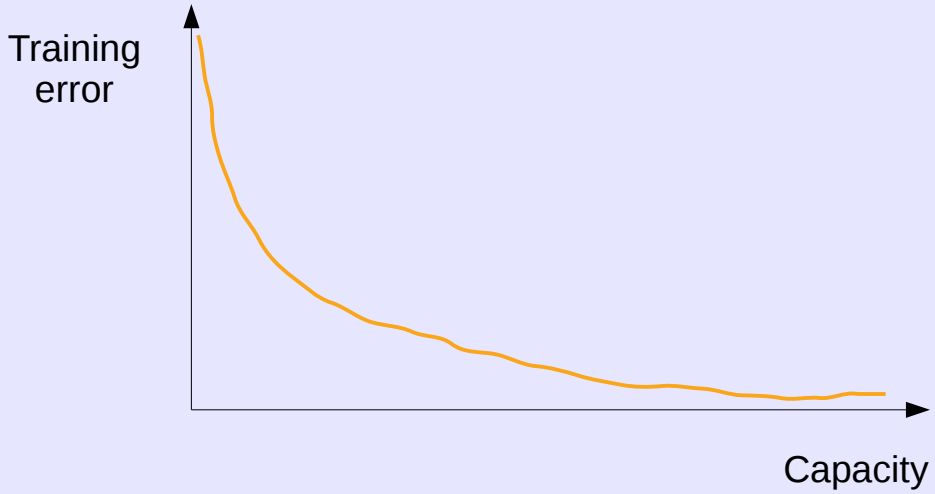


ML-model of degree “12”



ML-model of degree “25”





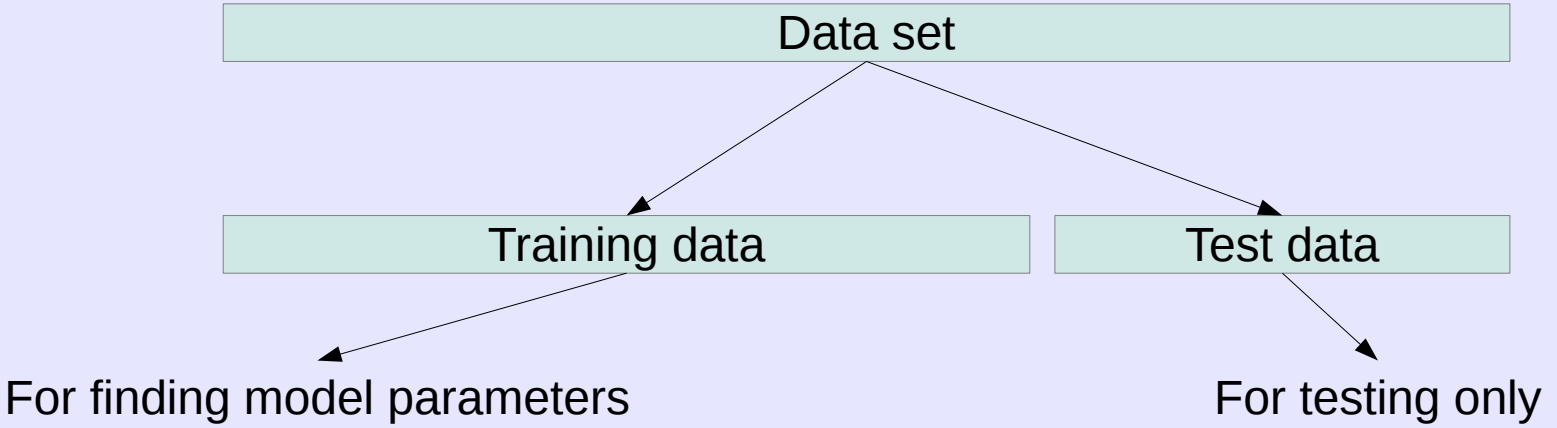
Problem: We cannot use training error or performance as an indicator how well my ML-model is doing!

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” inputs

Often,



Test error ≈ generalization error

We expect:

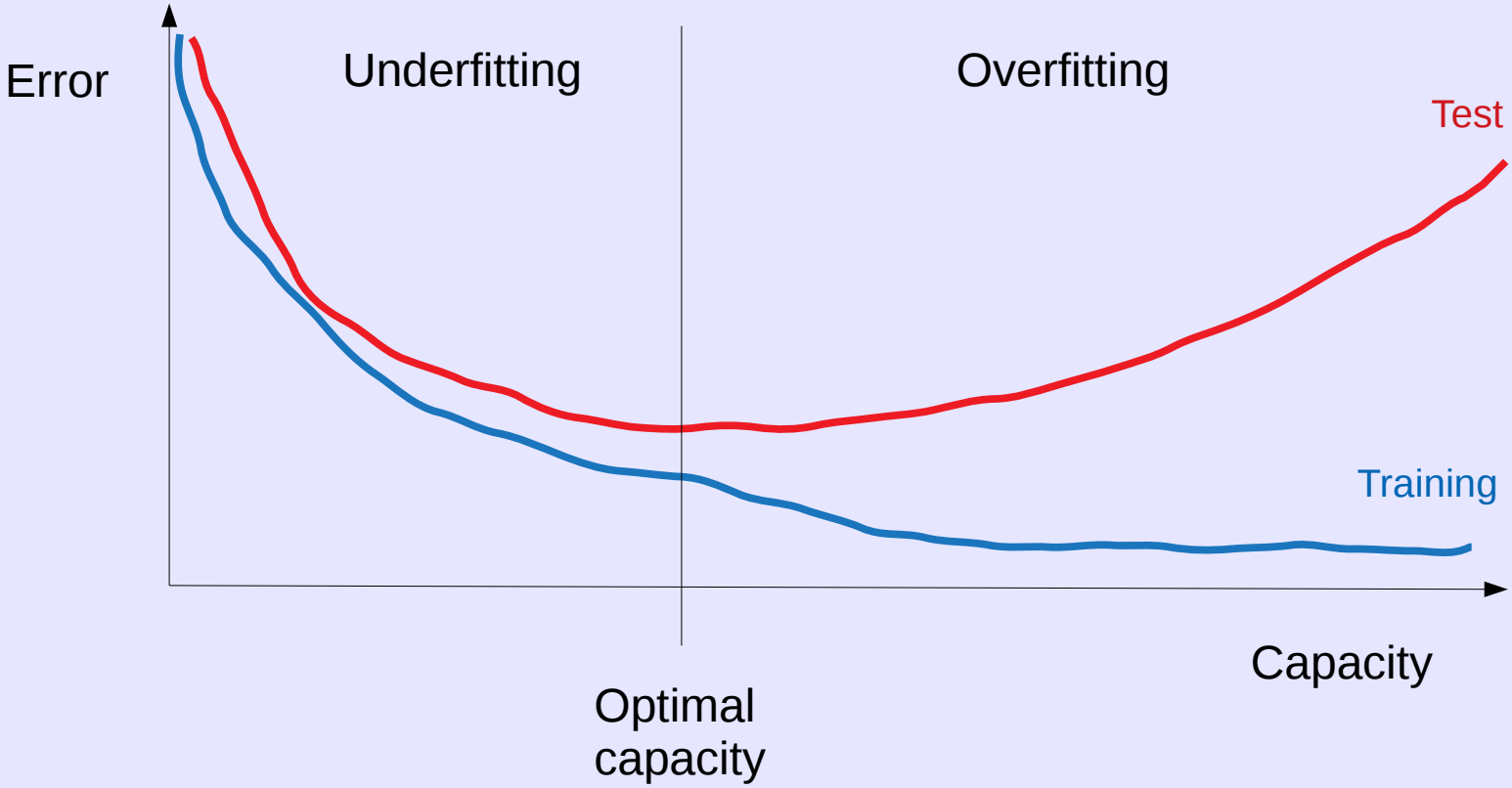
Test error ≥ Training error

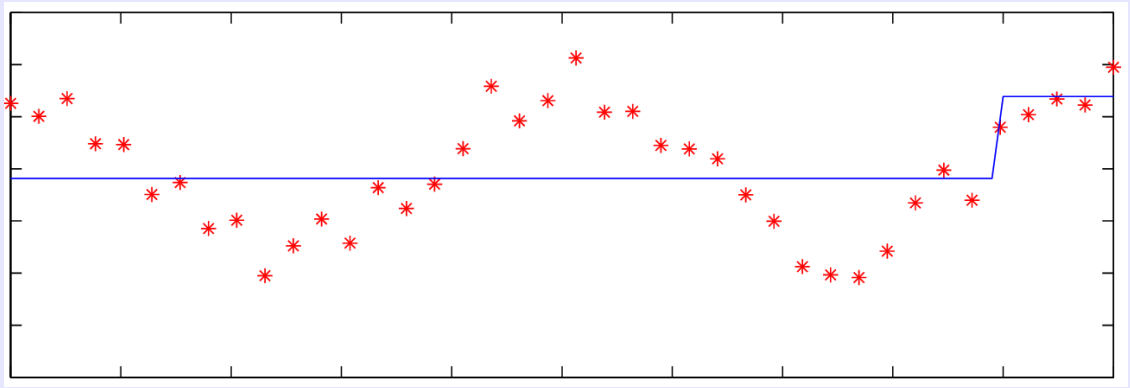
GOAL:

Minimize training error

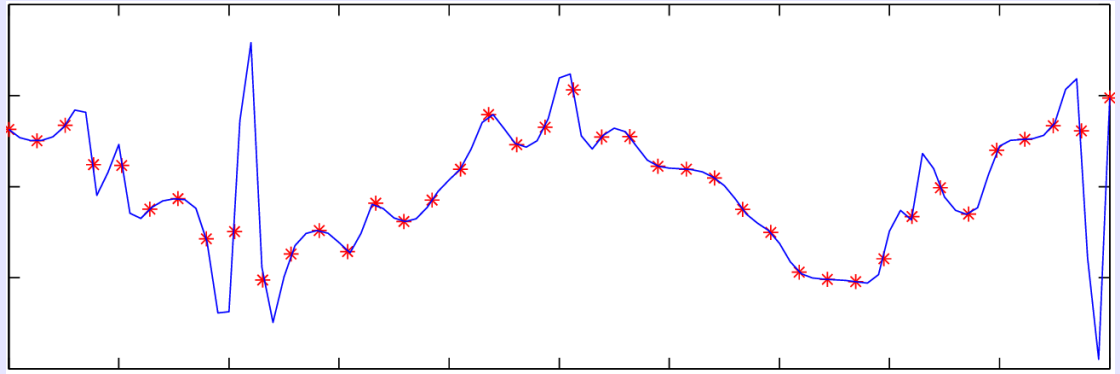
Make the gap between test error
and training error as small as possible.

Overfitting and underfitting





Underfitting



Overfitting

Bias-Variance tradeoff

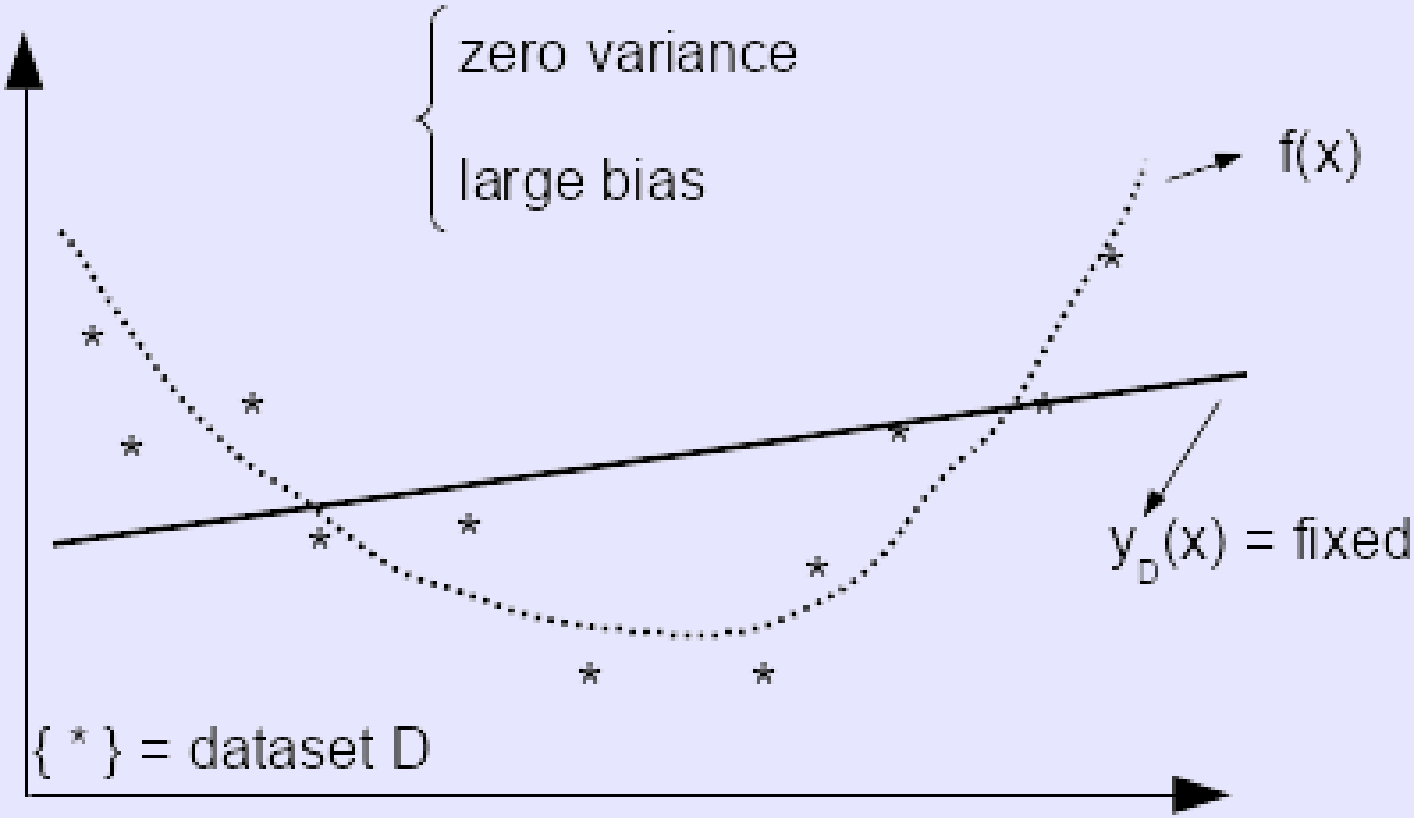
$D =$ Data set of size N

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

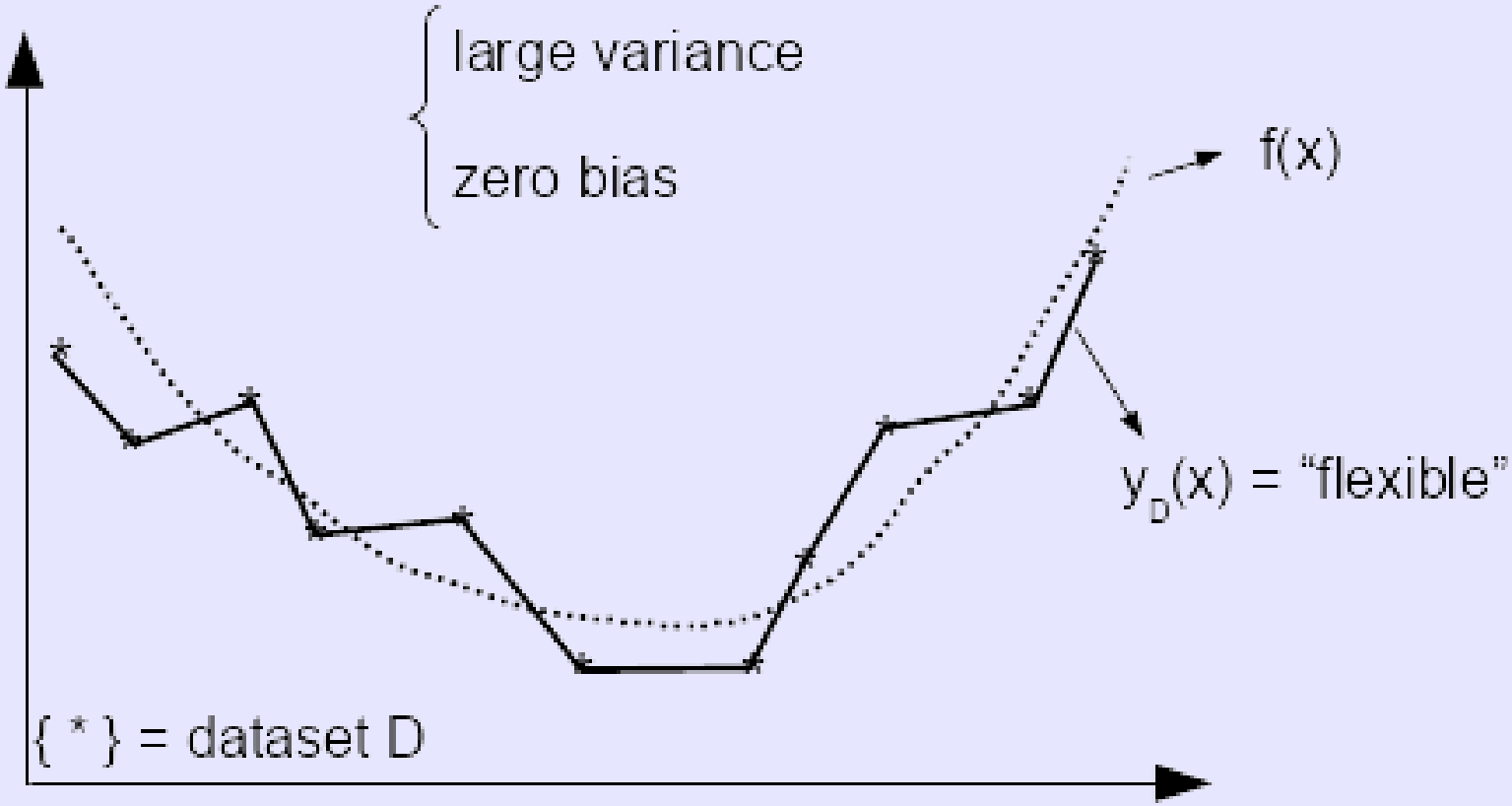
$\mathrm{E}\left[\cdot\right] =$ Expectation over many N-sized data sets D

$$\mathrm{E}\left[\left(y_{\mathrm{D}}(\mathbf{x}) - f(\mathbf{x})\right)^2\right] =$$

$$= \underbrace{\left(\mathrm{E}\left[y_{\mathrm{D}}(\mathbf{x})\right] - f(\mathbf{x})\right)^2}_{\mathrm{bias}^2} + \underbrace{\mathrm{E}\left[\left(y_{\mathrm{D}}(\mathbf{x}) - \mathrm{E}\left[y_{\mathrm{D}}(\mathbf{x})\right]\right)^2\right]}_{\mathrm{variance}}$$



Underfitting situation



Overfitting situation

How do we find a balance between underfitting and overfitting?

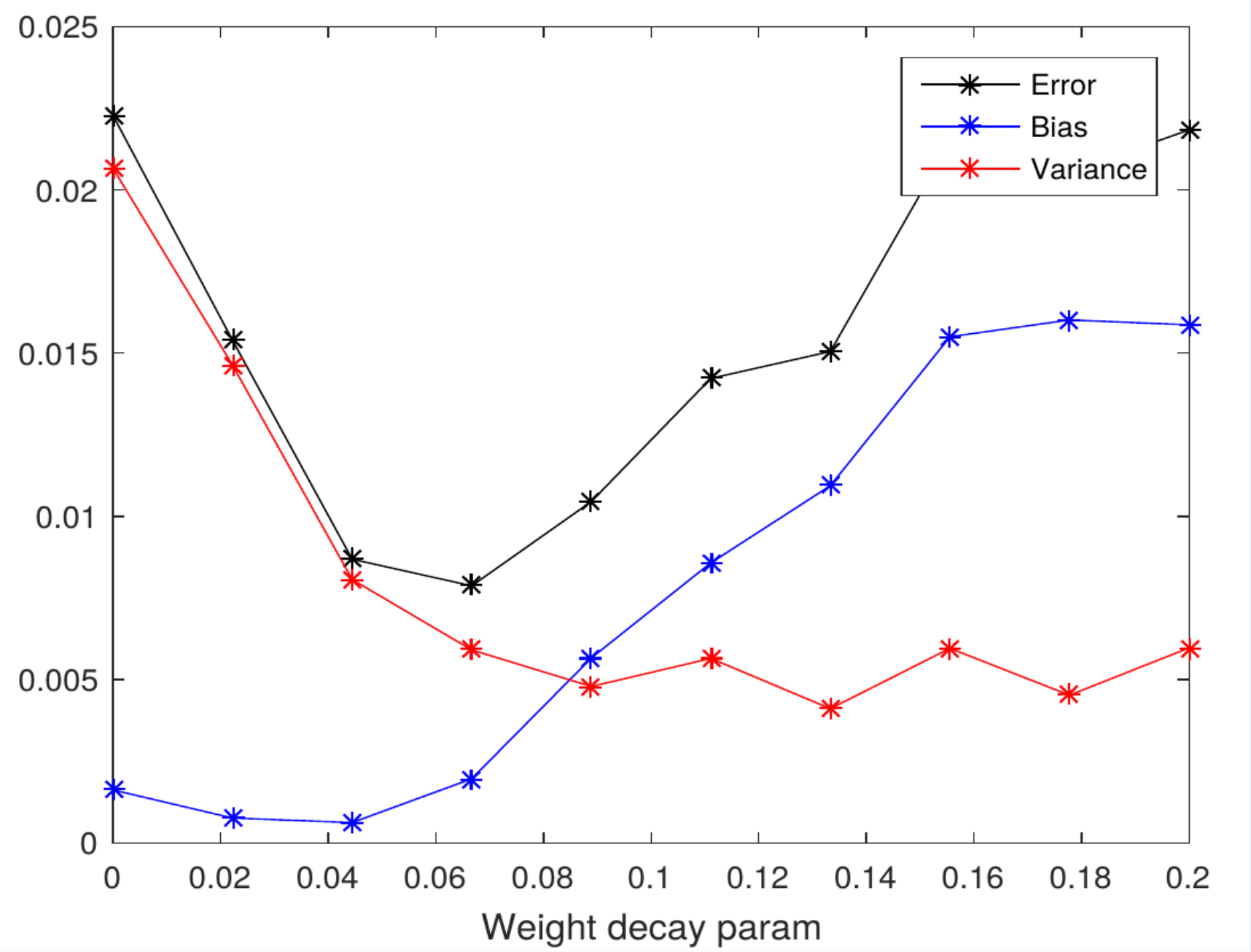
We want an efficient way of modifying the capacity of an ML model.

Regularization: Attempts to modify architectures, error functions or learning algorithms in order to reduce the generalization error

Example: L2 norm regularization

$$E(\mathbf{w}) = \mathrm{MSE} + \alpha ||\mathbf{w}||^2$$

Simple regression problem



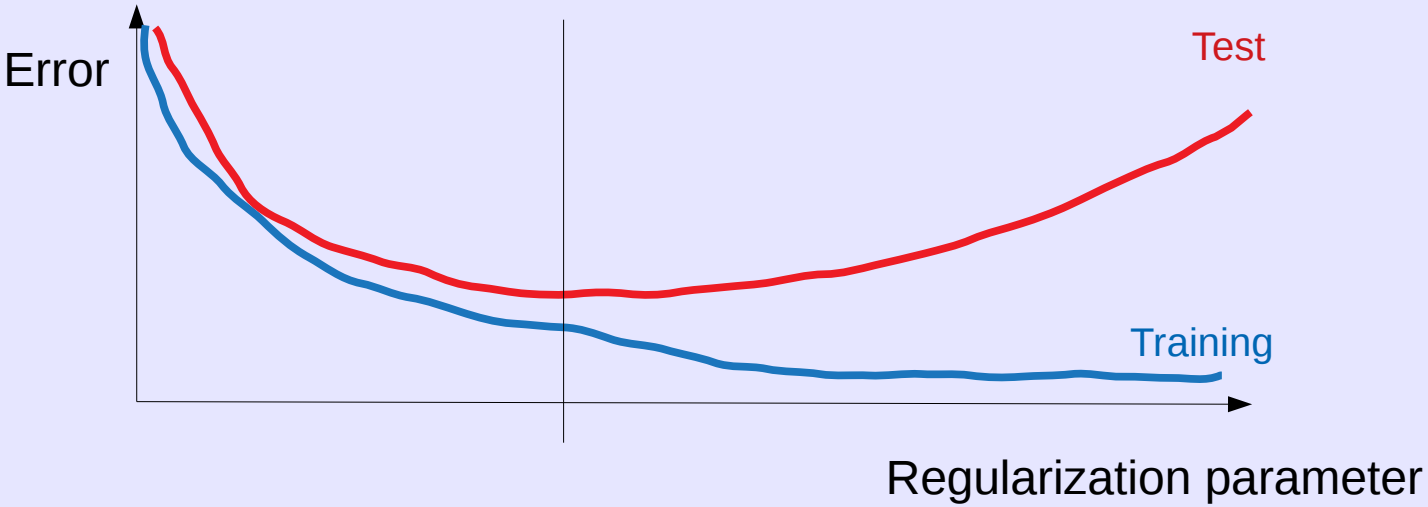
Hyper parameters and model selection

We have additional parameters that controls the ML model:

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

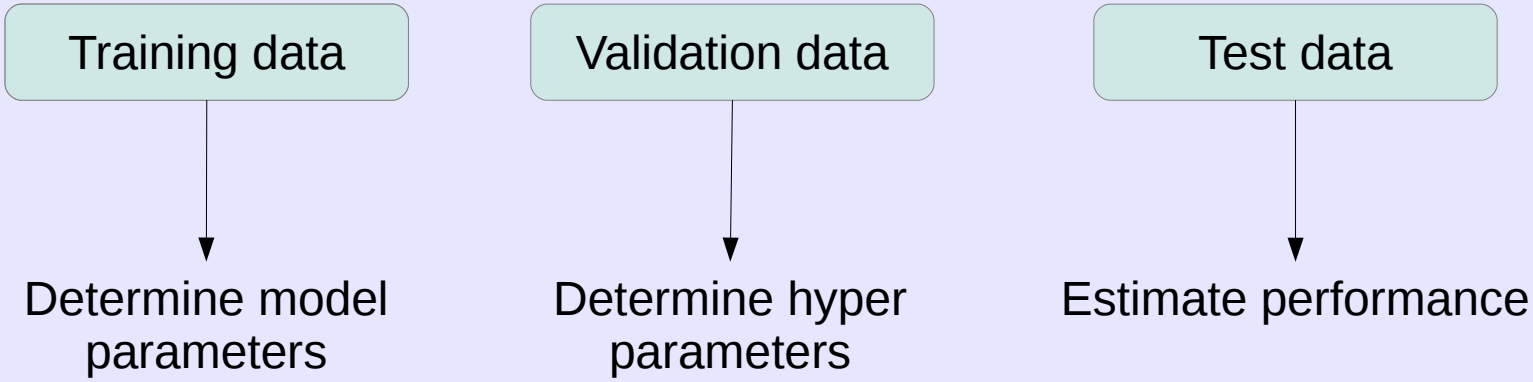
Regularization parameters: size of L2 norm or “dropout” parameter

Model selection = determination of hyper parameters



Model selection requires estimation
of generalization performance!!

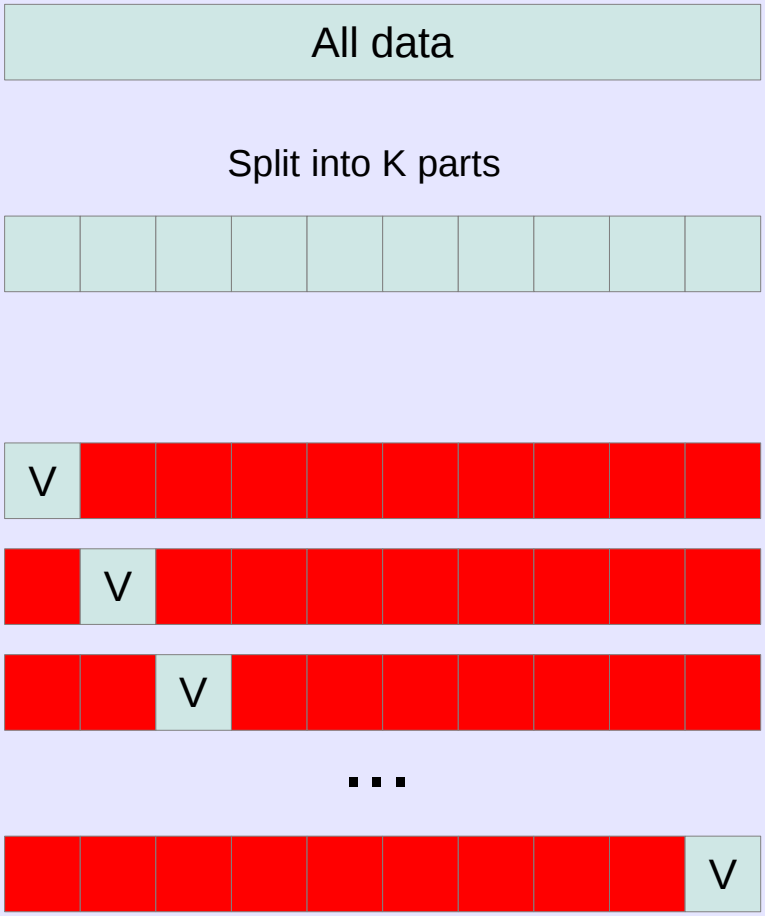
How can we do that?



Various approaches to estimate generalization performance

The holdout method

The K-fold cross validation method

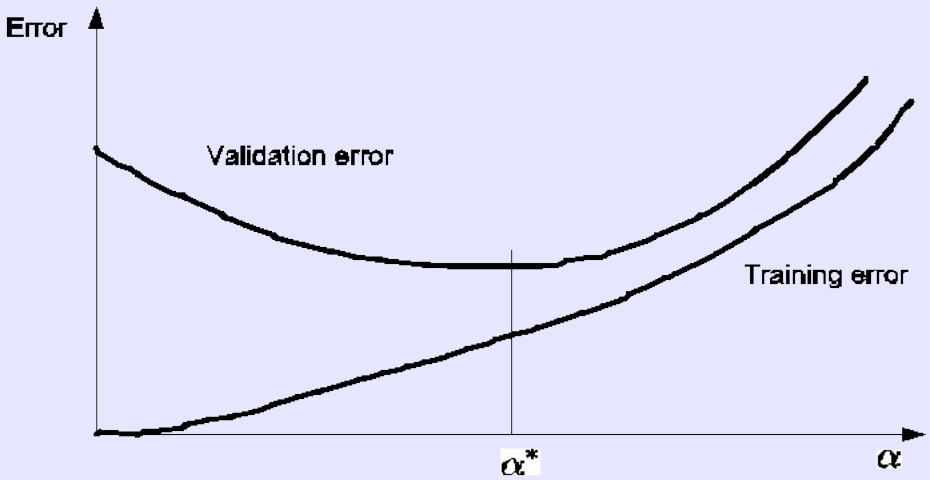


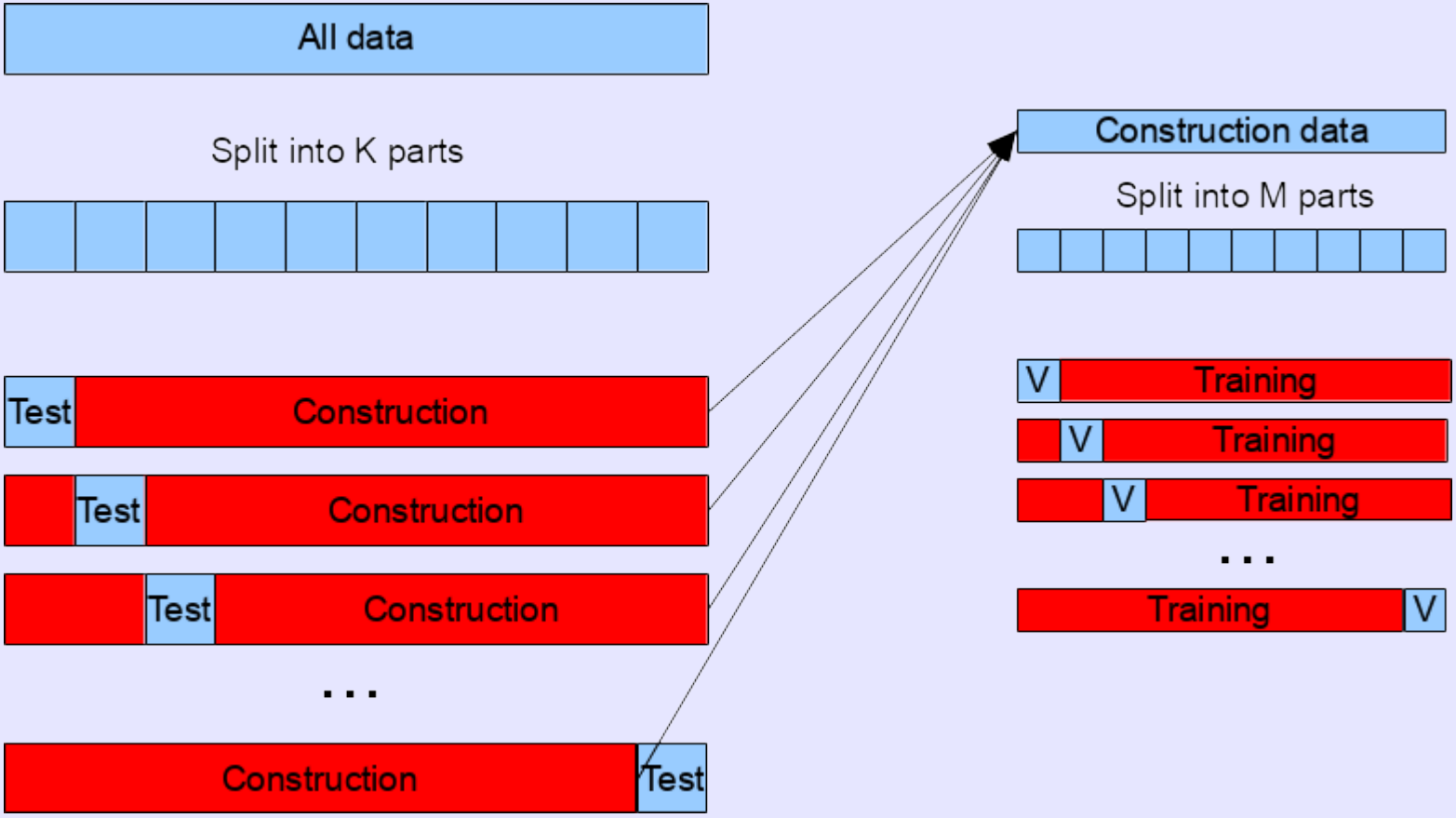
$$\hat{P} = \frac{1}{K} \sum_i P_i$$

Sometimes repeat *N* times

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

Sometime we need two loops!





Error function (sometimes also called the loss function) =
The function to minimize during training

How do we decide the error function?

A very common approach is to use the ***Maximum Likelihood principle***

$X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ Training data drawn from $p_{\mathrm{data}}(\mathbf{x})$

$p_{\mathrm{model}}(\mathbf{x}; \theta)$ Family of distributions modeled by θ

Maximum Likelihood

$$\theta = \underset{\theta}{\operatorname{argmax}}\ p_{\mathrm{model}}(X; \theta)$$

$$= \underset{\theta}{\operatorname{argmax}} \prod_{n}^{N} p_{\mathrm{model}}(\mathbf{x}_n; \theta)$$

Use the log instead

$$\theta = \underset{\theta}{\operatorname{argmax}} \sum_{n}^{N} \log p_{\text{model}}(\mathbf{x}_n; \theta)$$

We can define the loss function to be

$$E(\theta) = -\sum_{n}^{N} \log p_{\text{model}}(\mathbf{x}_n; \theta)$$

Conditional log likelihood

$$E(\theta) = -\sum_{n}^{N} \log P_{\text{model}}(\mathbf{d}_n | \mathbf{x}_n; \theta)$$

Stochastic gradient descent

Training an ML model, especially neural network models,
involves minimizing the loss function with respect to model parameters

Very often one have to rely on numerical minimization procedures.
The most common approach is ***gradient descent*** based methods

$$\Delta\theta_i = -\eta \frac{\partial E(\theta)}{\partial \theta_i}$$

Now very often

$$E(\theta) = \frac{1}{N} \sum_n^N E_n(\theta)$$

We can write

$$\Delta\theta_i = \frac{1}{N} \sum_n \Delta\theta_{ni}$$

per pattern update

Stochastic gradient descent (SGD)

We can approximate the gradient with a smaller set of patterns

$$\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m\} \qquad \Delta\theta_i \approx \frac{1}{m}\sum_k^m \Delta\theta_{ki}$$

$$m \ll N$$

Each set $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m\}$ is called a minibatch of patterns.

There are many methods centered around the SGD idea!!

Optimizers available in “Keras”

RMSprop

SGD

Adadelta

Adamax

Adagrad

Adam

Nadam

Some common loss functions

Regression problem

$$\begin{array}{ll} \text{targets} & \{d_1, d_2, ..., d_N\} \\ \text{inputs} & \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\} \end{array}$$

$$E(\theta) = \frac{1}{N} \sum_{n}^{N} \left(d_n - y(\mathbf{x}_n; \theta)\right)^2$$

Mean squared error

Binary classification problem

$$\begin{array}{ll} \text{targets} & \{d_1, d_2, ..., d_N\},\ d_n \in \{0, 1\} \\ \text{inputs} & \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\} \end{array}$$

$$E(\theta) = -\frac{1}{N} \sum_{n}^{N} \left[d_n \log y(\mathbf{x}_n; \theta) + (1 - d_n) \log \left(1 - y(\mathbf{x}_n; \theta)\right) \right]$$

Cross entropy error