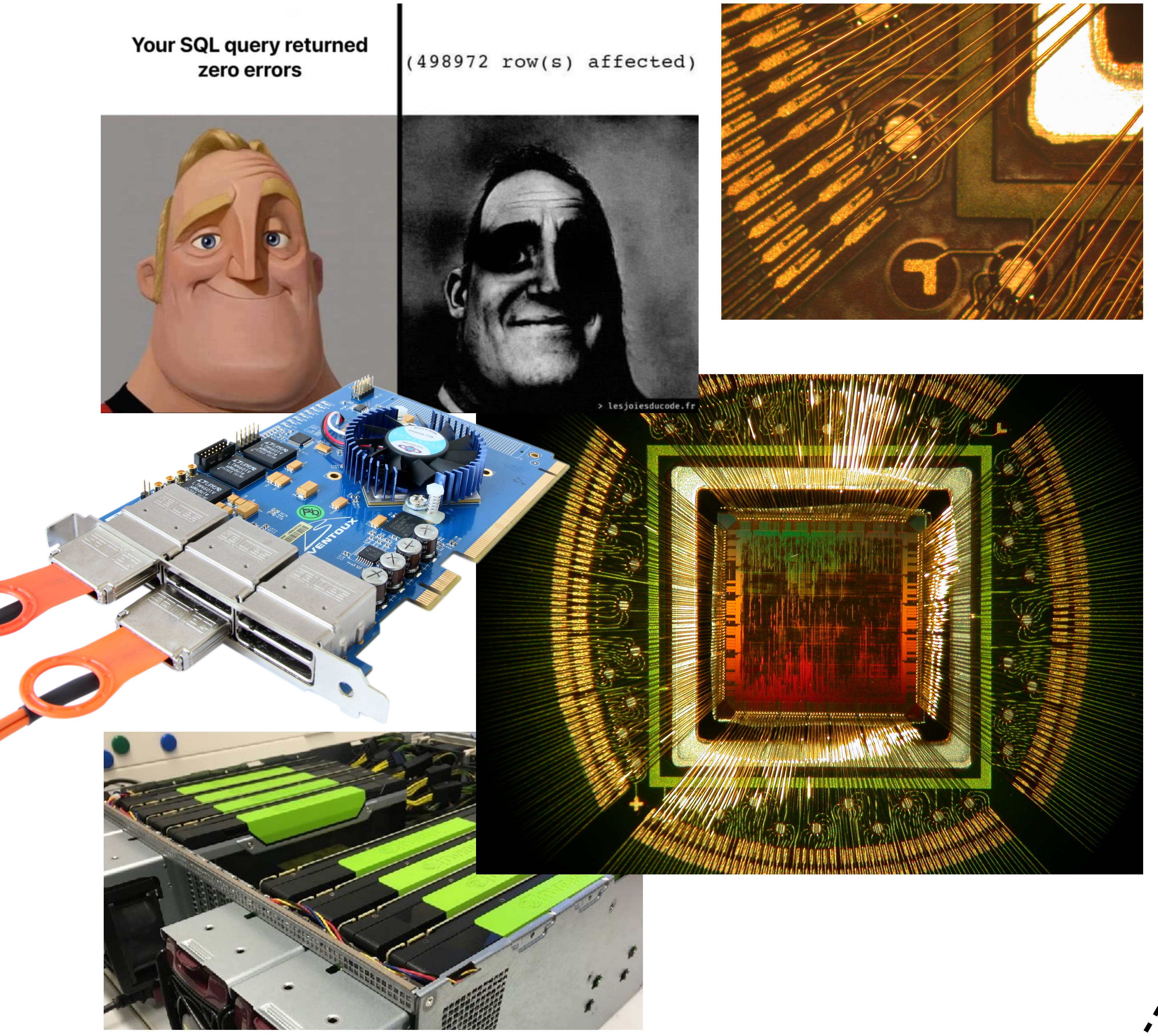
NEURAL NETWORKS FROM SCRATCH
LECTURE 01 - INTRODUCTION

Hendrik Borras, Robin Janssen
{hendrik.borras, robin.janssen}@ziti.uni-heidelberg.de,
HAWAII Lab, Institute of Computer Engineering
Heidelberg University

ABOUT US

From: database engineer, HW designer (ASICS, FPGA), HPC



$$\mathbf{x}^l = \Phi(\mathbf{W} \oplus \mathbf{x}^{l-1} + \mathbf{b}^l)$$

Neural Architectures

Compiler

Plethora of HW

$$perf[\frac{ops}{s}] = p[Watt] \cdot e[\frac{ops}{J}]$$

$$P = afCV^2 + VI_{leakage}$$

To: vertically integrated approach to efficient ML => HW systems for AI

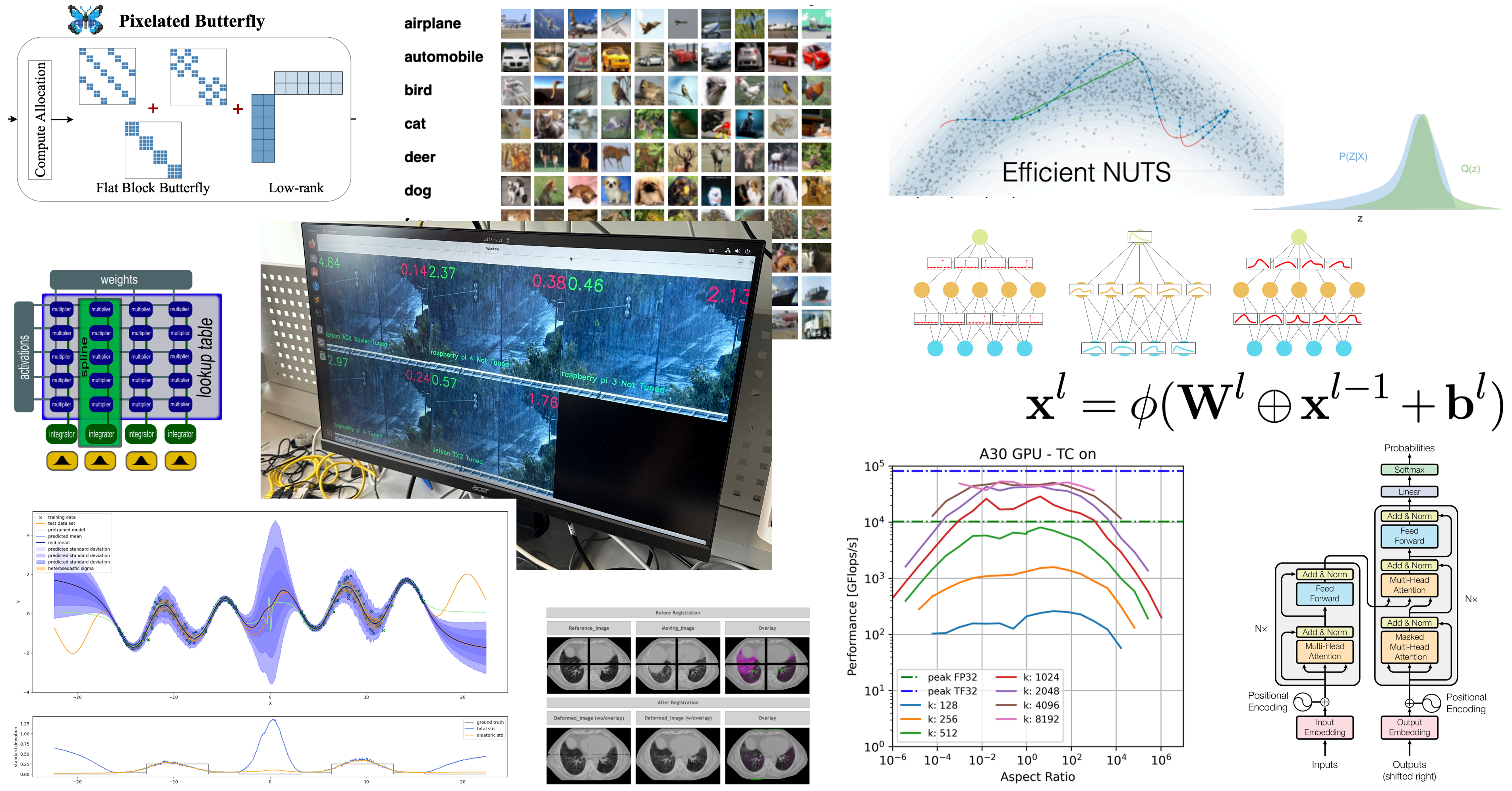
MAIN RESEARCH DIRECTIONS

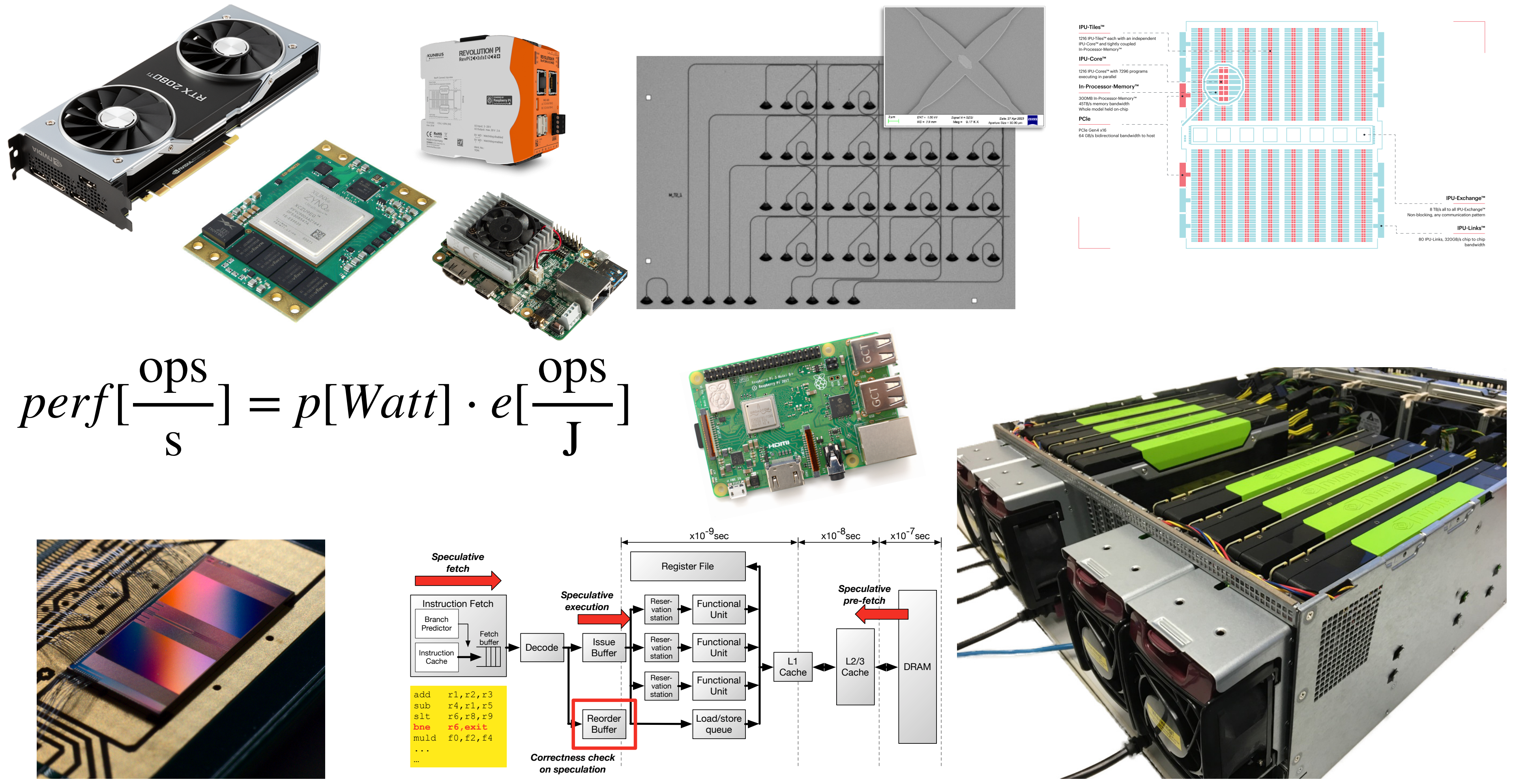
Robust ML

- Approximations: Bayesian, Probabilistic, Deep Ensembles, Repulsive Ensembles (LLMs)
- Understanding and using methods
- Assessment of costs
- Translating among methods
- Mapping to noisy HW

Scalable ML

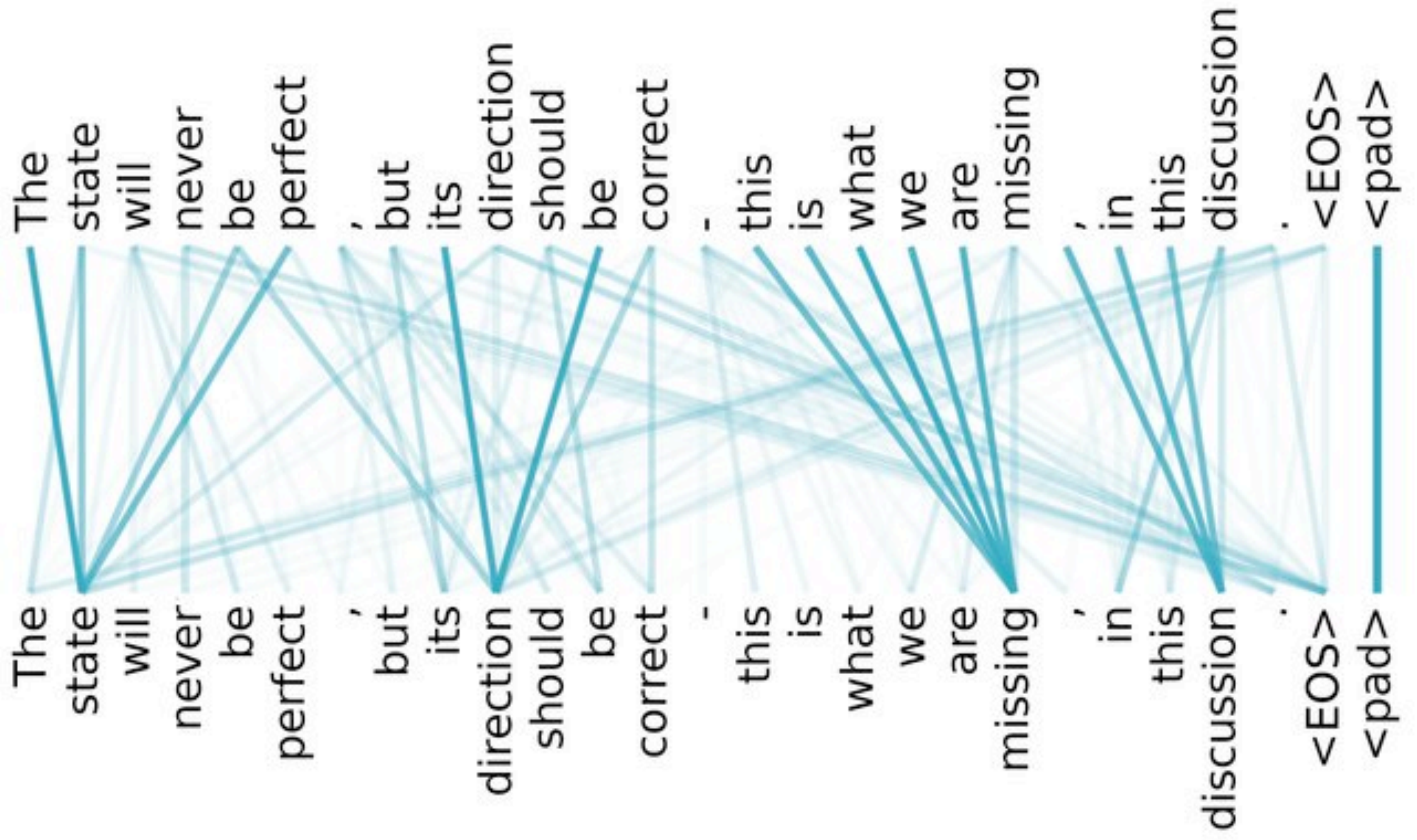
- GreenML - compression for training of large-scale models
- Khunjerab - connecting RRAM to LLMs
- AMD’s Fused GPU/CPU
- Model compression (NAS)
- Mapping to noisy memory





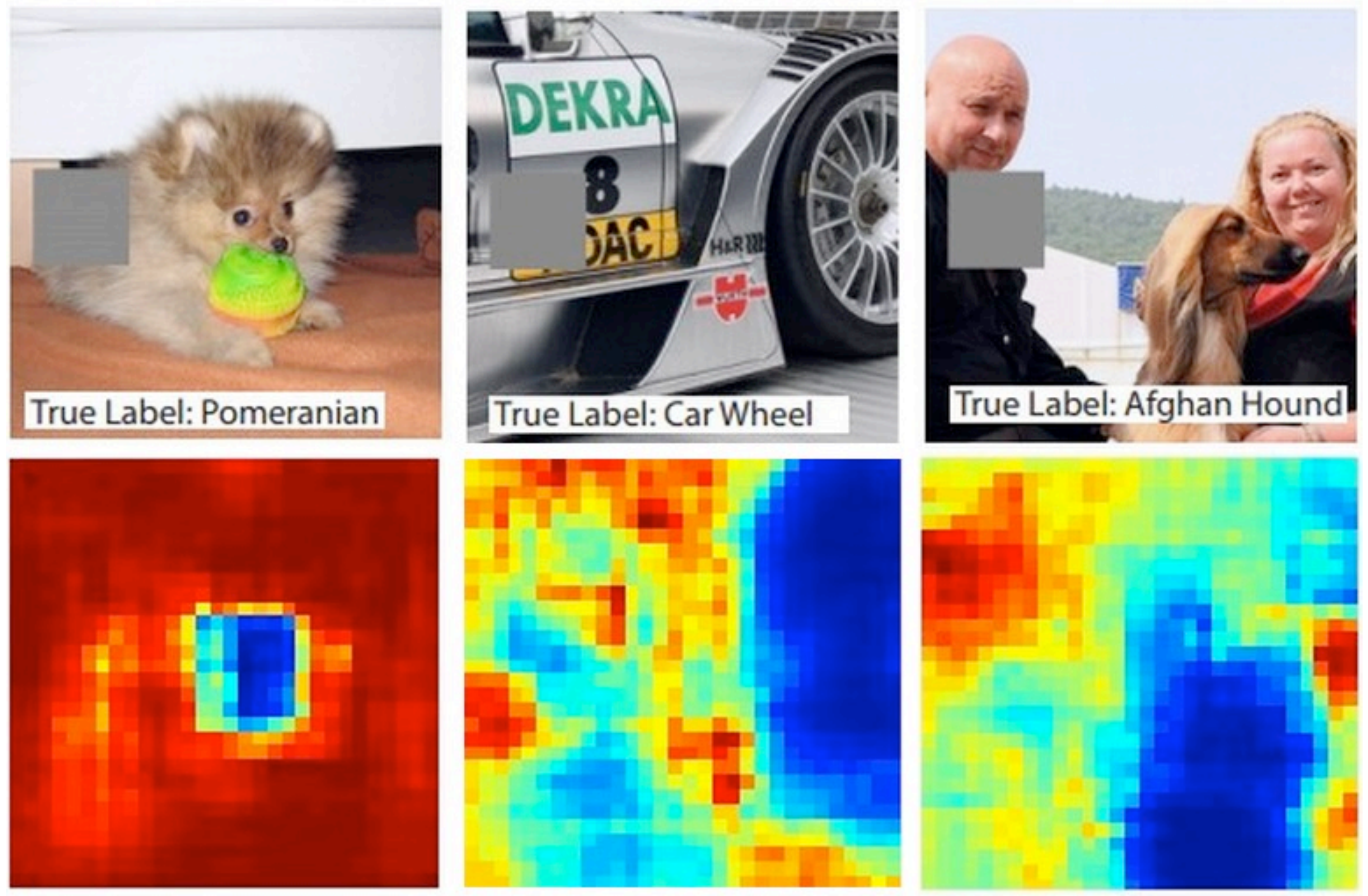
ML APPLICATIONS

Language Processing



Robotics

Image Processing



Speech Recognition

MODERN ML

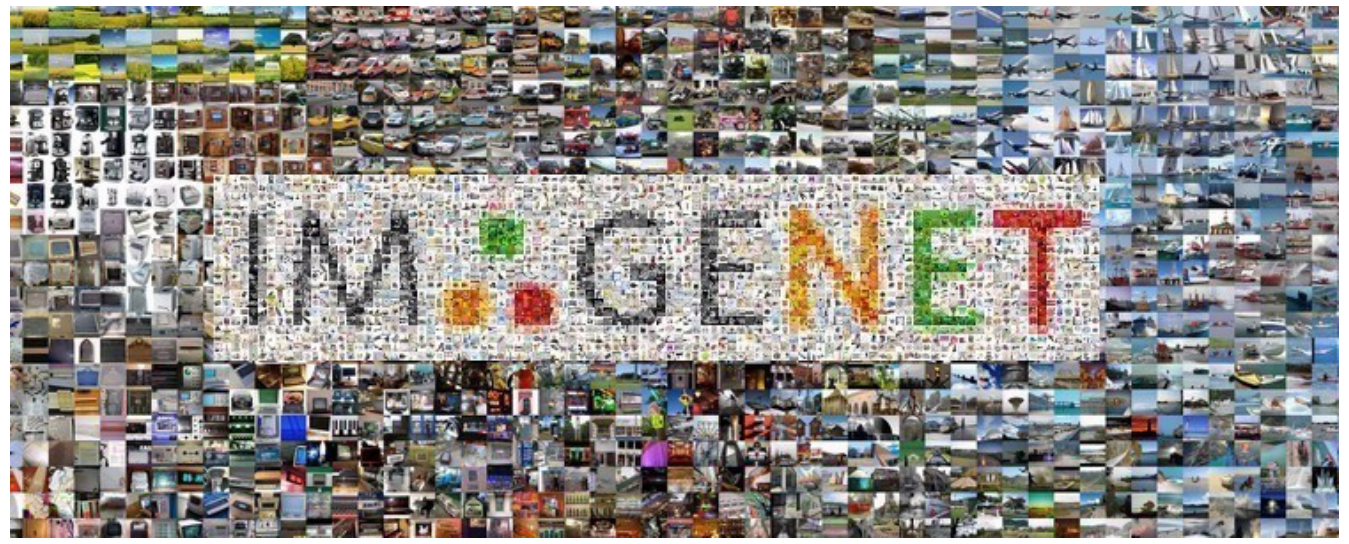
Training: ~ $O(10^{18})$ OPs
Inference: ~ $O(10^{9})$ OPs

Image & video: classification, object localization & detection

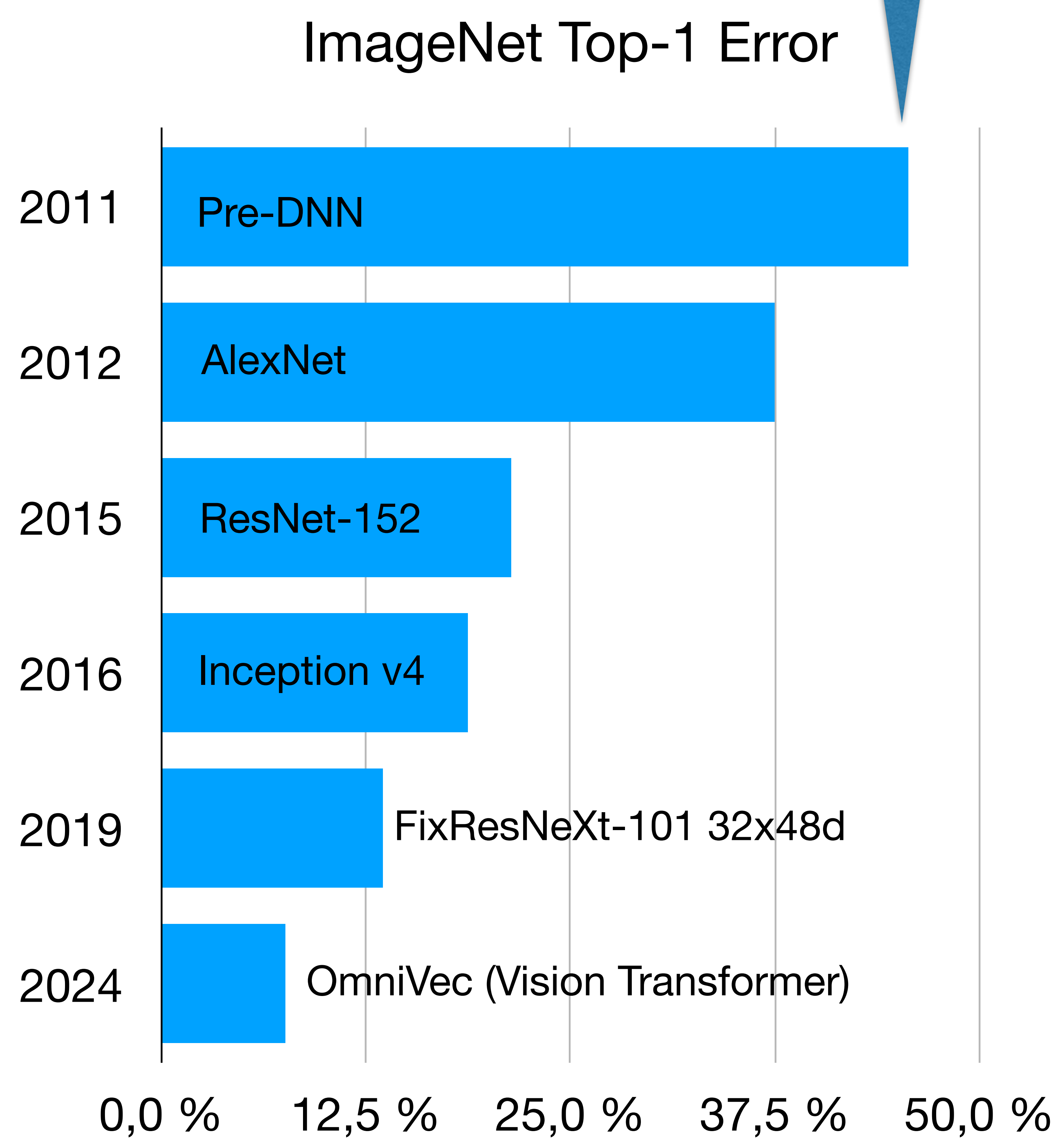
Speech and language: speech recognition, natural language processing

Medical: imaging, genetics of diseases

Various: game play, robotics



IMAGENET: 1000 classes



Artificial Neural Networks (ANNs) deliver state-of-the-art accuracy for many AI tasks
… at the cost of extremely high computational complexity

DATASET COMPARISON

	Type	Dataset Samples	Dataset Size
MNIST	Image	60k train + 10k test	~ 45 MB
CIFAR-10	Image	50k train + 10k test	~ 176 MB
ILSVRC2015	Image	1.38M	~ 150 GB
FineVideo	Video	43k	~ 600 GB (3.4k hours)
The Pile	Text	211M (documents)	~ 825 GB
LLAMA Pretraining Sets	Text		~ 4.7 TB

Trains on a reasonable laptop in ~ 10min

Trains in ~ 3 weeks on 2k A100 GPUs consuming ~ 449 MWh [1]

[1] Touvron, H., "LLaMA: Open and Efficient Foundation Language Models", 2023, https://arxiv.org/abs/2302.13971

HARDWARE LOTTERY HYPOTHESIS

"Tooling […] has played a disproportionately large role in deciding which ideas succeed and which fail"

HW determines which ideas succeed

- ANNs == matrix-matrix ops == excellent performance of GPUs
- Most ML researchers ignore hardware

Recent trends

- Convolutions and transformers (attention heads, based on softmax)
- GPT-3: 175B parameters (800GB of state); Alphafold-2: 23TB of training data

What if another processor was existing, e.g. excelling in processing large graphs?

- Probabilistic graphical models, sum-product networks, graph neural networks, etc.?

PROCESSOR SPECIALIZATION IS CONSIDERED HARMFUL FOR INNOVATION

ORGANIZATION

OBJECTIVES

Objectives: The students …

- … learn about the mathematical foundations of machine learning
- … start applying their skills by implementing a least squares fit
- … continue on to multi-layered models by implementing a multi-layer perceptron (MLP) from scratch
- … experience first-hand the requirement of using parallel architectures, in our case GPUs, when scaling up neural networks and learn how to bring their models to the GPU
- … apply their acquired knowledge on more complex architectures, datasets, problems or optimizers during the project
- … implement a more complex models/techniques based on their acquired knowledge as their final project

METHODS & PREREQUISITES

Methodology

- Strong focus on learning from hands-on experience
- Learning to implement neural networks starting with pure Python without *any* auto-grad packages, with usage of the common numerical packages (numpy, CuPy, Scikit-learn) following after -> Allows for a look under the hood not easily possible using modern ML libraries
- Students can choose from a large selection of final project topics based on their specific personal interests

Prerequisites

General knowledge of machine learning (either from lectures or from self-study)

Passed exams in:

- Einführung in die Praktische Informatik (IPI) OR Programmierkurs (IPK)
- Lineare Algebra 1 (MA4) OR Mathematik für Informatik 1 (IMI1)

Practical experience:

- Intermediate proficiency in Python

ORGANIZATION

Lectures – 2 hours/week

- Lecturers:
  - Hendrik Borras (hendrik.borras@ziti.uni-heidelberg.de)
  - Robin Janssen (robin.janssen@ziti.uni-heidelberg.de)
- Time: Wednesday, 13:00 st.

Exercises – 2 hours/week

- Groups of 2 or 3 students
- Time: Wednesday, after the lecture
- Mixture of programming and experiment exercises

Project-Based Grading

- Work to be done in groups – individual work must be visible
- Students implement, document, and present an ML program
- Grades are determined by the quality of the project, report, and presentation at poster session

ORGANIZATION

Both “Anfängerpraktikum” and “Fortgeschrittenenpraktikum” are possible

- They will differ in the amount of work expected for the projects
- All other things, like lectures and final presentation are the same

Who would like to do an AP? Who would like to do an FP?

- A minimum of two people per are required to make sure groups can be formed

ASSIGNMENTS

Practical exercises: Coding and experiments

Goal:

- General understanding of basic DNN building blocks
- Building a common code base for the projects

Comments on LLM use:

- Generally no restrictions, however:
- Code is trivially solvable by Copilot or [insert random LLM]
- Learning is greatly diminished with LLM use
- Recommendation: Turning off Copilot or similar assistances during the exercises

AGENDA

Datum	Vorlesung	Übung
15.10	Einführung & Machine Learning 1	Polynomial curve fitting
22.10	Machine Learning 2	MLP from scratch
29.10	GPUs & CuPy	Cluster access, GPU acceleration and experiment visualization
05.11	Project formalities and suggestions	Project proposal development
12.11	No lecture	
19.11	Project proposal discussion and kick-off	
…	N-times Project updates and questions	
KW 10	Poster session	
KW 12	Reports due	

ADDITIONAL MATERIAL

Papers

Badillo et al.: An Introduction to Machine Learning (https://ascpt.onlinelibrary.wiley.com/doi/pdfdirect/10.1002/cpt.1796)

Horowitz: Computing's energy problem (and what we can do about it) (https://ieeexplore.ieee.org/document/6757323)

Textbooks

Goodfellow et al.: Deep Learning (https://www.deeplearningbook.org, https://github.com/janishar/mit-deep-learning-book-pdf)

Hwu et al.: Programming Massively Parallel Processors (https://www.sciencedirect.com/book/9780323912310/programming-massively-parallel-processors)

Other

Deep Learning Cheat Sheet (https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-deep-learning)

ADDITIONAL MATERIAL

https://hawaii.ziti.uni-heidelberg.de/teaching/ap_nn_from_scratch_materials_wise2025/

LINEAR AND POLYNOMIAL REGRESSION

Learning, generalization, model selection, regularization, overfitting

With material from Andrew Ng (CS229 lecture notes) and Christopher Bishop (Pattern Recognition and Machine Learning)

SUPERVISED LEARNING

Based on the given housing data, is it possible to learn to predict the costs of other houses?

➡ Prediction of "Unseen data"

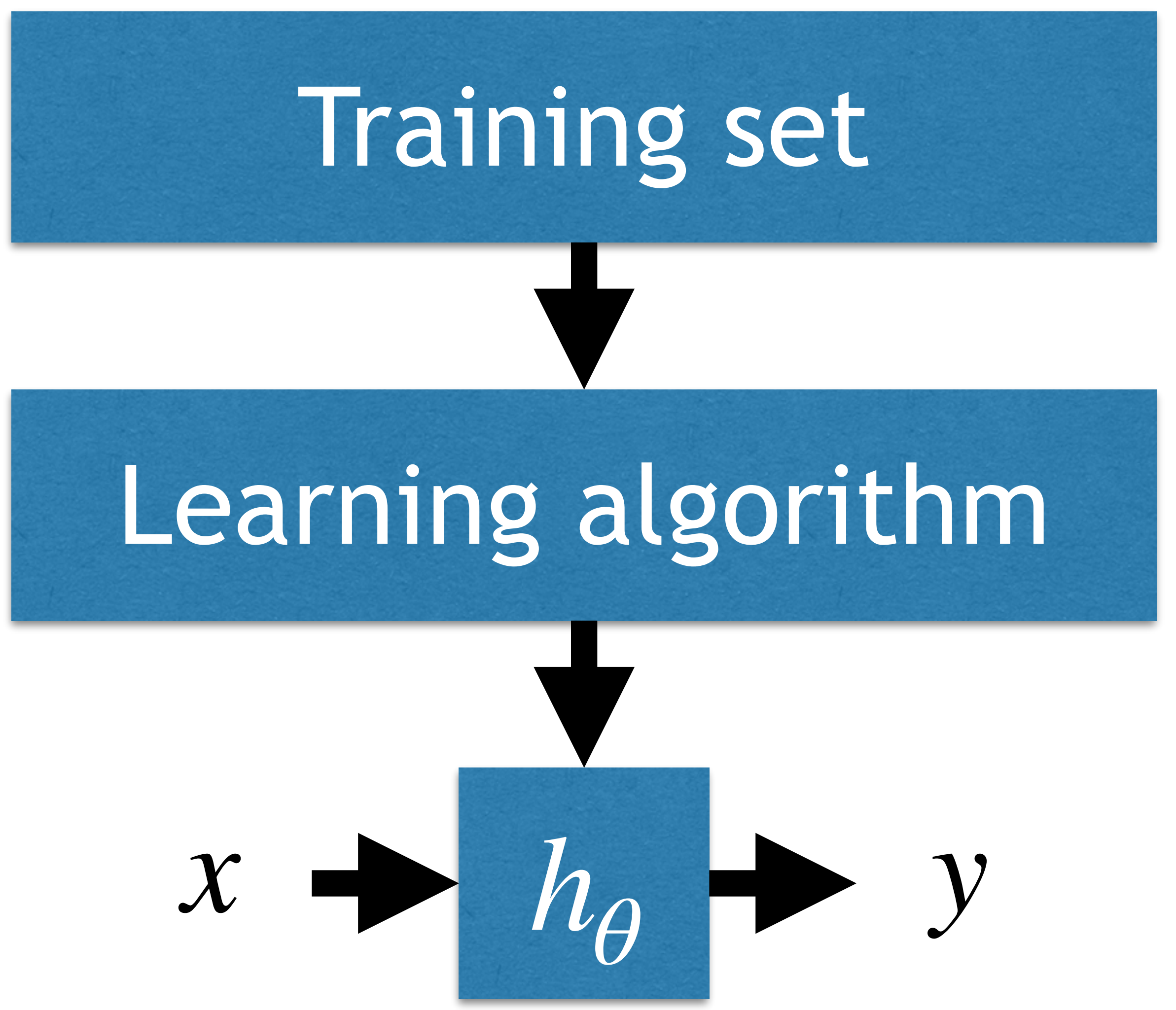
Notation

- $x^{(i)}$: Input features of sample i
- $t^{(i)}$: Target variable (or output variable or label) of sample i
- $(x^{(i)}, t^{(i)})$: Training sample (or observation) i
- Training set: set of all training samples (size N)

Supervised learning problem: find good prediction function $y = h_\theta(x)$

- θ (theta) are the parameters (weights) of the model
- Classification (discrete) vs. regression (continuous) problem

Living area (feet^2)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮



LINEAR REGRESSION

$$\mathbf{x} = (x_1, x_2)^T = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^2$$

Living area (feet^2)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

Supervised learning: choose function h

$$y = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Simplification given D model parameters:

$$h_\theta(\mathbf{x}) = h(\mathbf{x}) = \sum_{d=1}^{D} \theta_d x_d = \theta^T \mathbf{x}$$ (model intercept θ_0 by $x_0 = 1$)

Learning: make $h(x)$ close to t for the N training samples we have

Cost (or error or loss) function “how close is that”: $J(\theta) = \frac{1}{2} \sum_{n=1}^{N} \left(h_\theta(x^{(n)}) - t^{(n)}\right)^2$

Least-squares method to find the optimal parameters by minimizing this sum of squared residuals

GRADIENT DESCENT

Choose θ such that $J(\theta)$ is minimal

Start with initial guess of θ, repeatedly perform gradient descent:

$\theta_d := \theta_d - \alpha \frac{\partial}{\partial \theta_d} J(\theta)$, simultaneously for all $d = 1, \ldots, D$ and learning rate α

$$\frac{\partial}{\partial \theta_d} J(\theta) = \frac{\partial}{\partial \theta_d} \frac{1}{2} \sum_{n=1}^{N} (h_\theta(x) - t)^2 = \frac{2}{2} \sum_{n=1}^{N} (h_\theta(x) - t) \cdot \frac{\partial}{\partial \theta_d} \Big((\sum_{i=1}^{D} \theta_i x_i) - t \Big) = \sum_{n=1}^{N} (h_\theta(x) - t) x_d$$

Hint: remember chain rule of calculus - for $f(x) = u\big(v(x)\big), f'(x) = u'\big(v(x)\big) v'(x)$

=> Update rule: $\theta_d := \theta_d + \alpha \sum_n \big(t^{(n)} - h_\theta(x^{(n)})\big) x_d^{(n)}$

Magnitude of update is proportional to error term

Which set of the training samples (elements n) to consider for one update?

BATCH GRADIENT DESCENT

Only one global optima as J is a convex quadratic function

Batch gradient descent: $\forall d \in D$

$$\theta_d := \theta_d + \alpha \sum_{n=1}^{N} \left(t^{(n)} - h_\theta(x^{(n)})\right) x_d^{(n)}$$

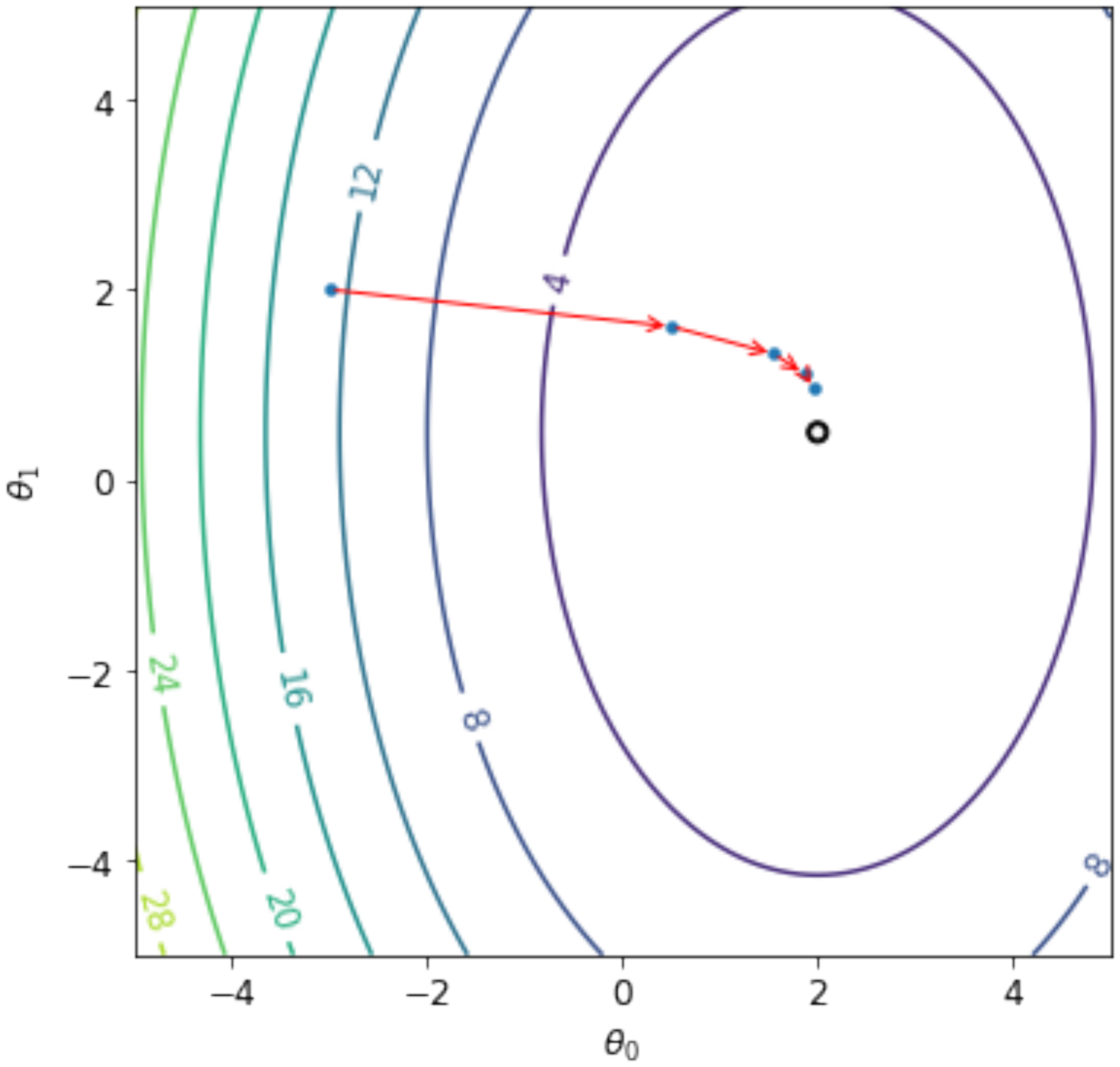
Repeat until convergence

Looks at every training sample ($\forall n \in N$) on every step

Number of steps depend on convergence

Guaranteed to be optimal, but expensive

Cost function



STOCHASTIC (INCREMENTAL) GRADIENT DESCENT

Scanning the complete data set for every step can be costly

Stochastic gradient descent is based on randomly selecting training samples to perform gradient descent

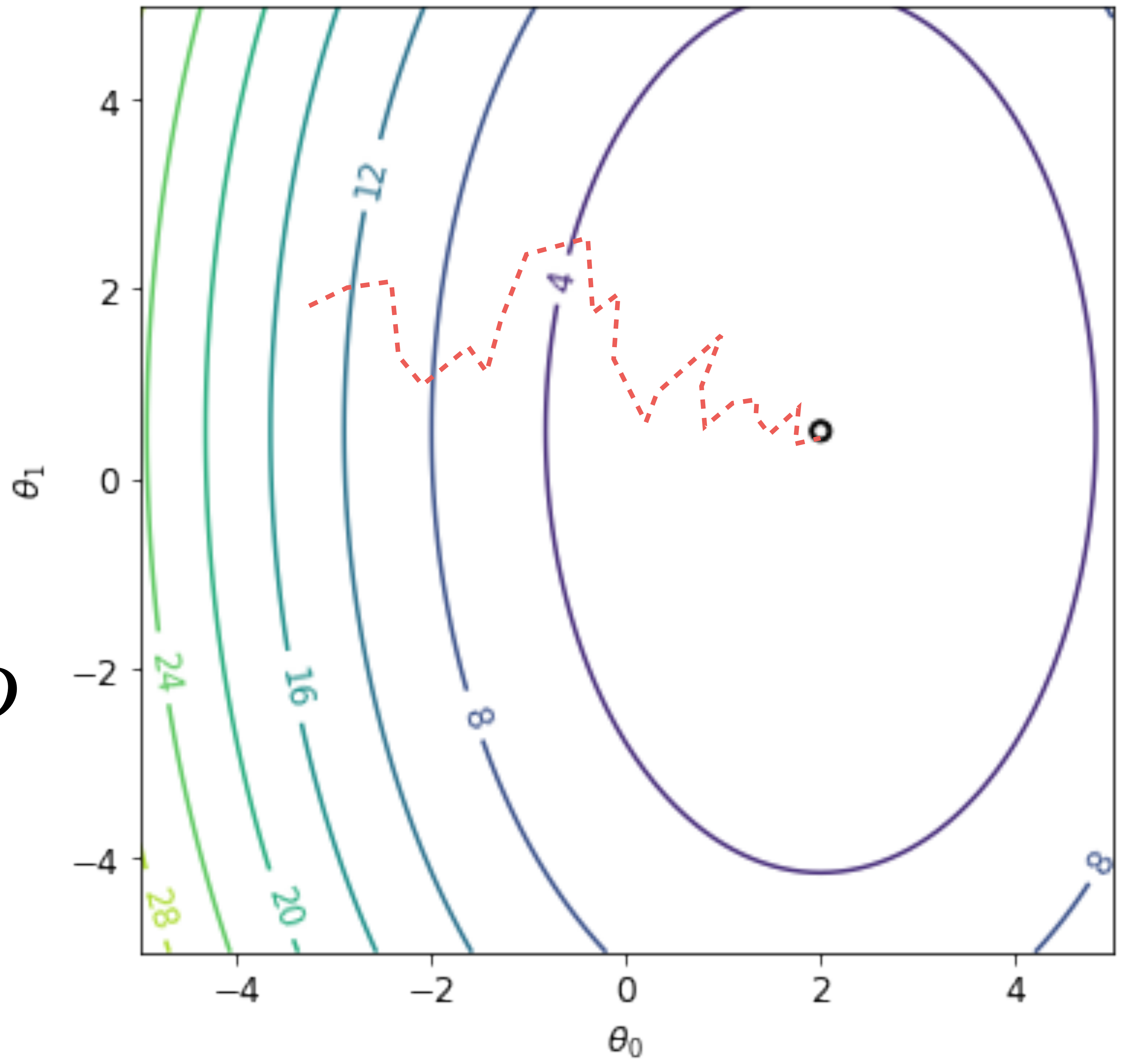
for all n in N:

$$\theta_d := \theta_d + \alpha\left(t^{(n)} - h_\theta(x^{(n)})\right)x_d^{(n)}; \forall d \in D$$

Repeat until convergence

Makes progress for each training sample

Cost function



POLYNOMIAL CURVE FITTING

Training set: N observations of $\mathbf{x} = (x_1, \ldots, x_N)^T$ and $\mathbf{t} = (t_1, \ldots, t_N)^T$

Ground truth: $t = sin(2\pi x)$, but (Gaussian) noise present

Many data sets have an underlying regularity, but observations are corrupted by random noise

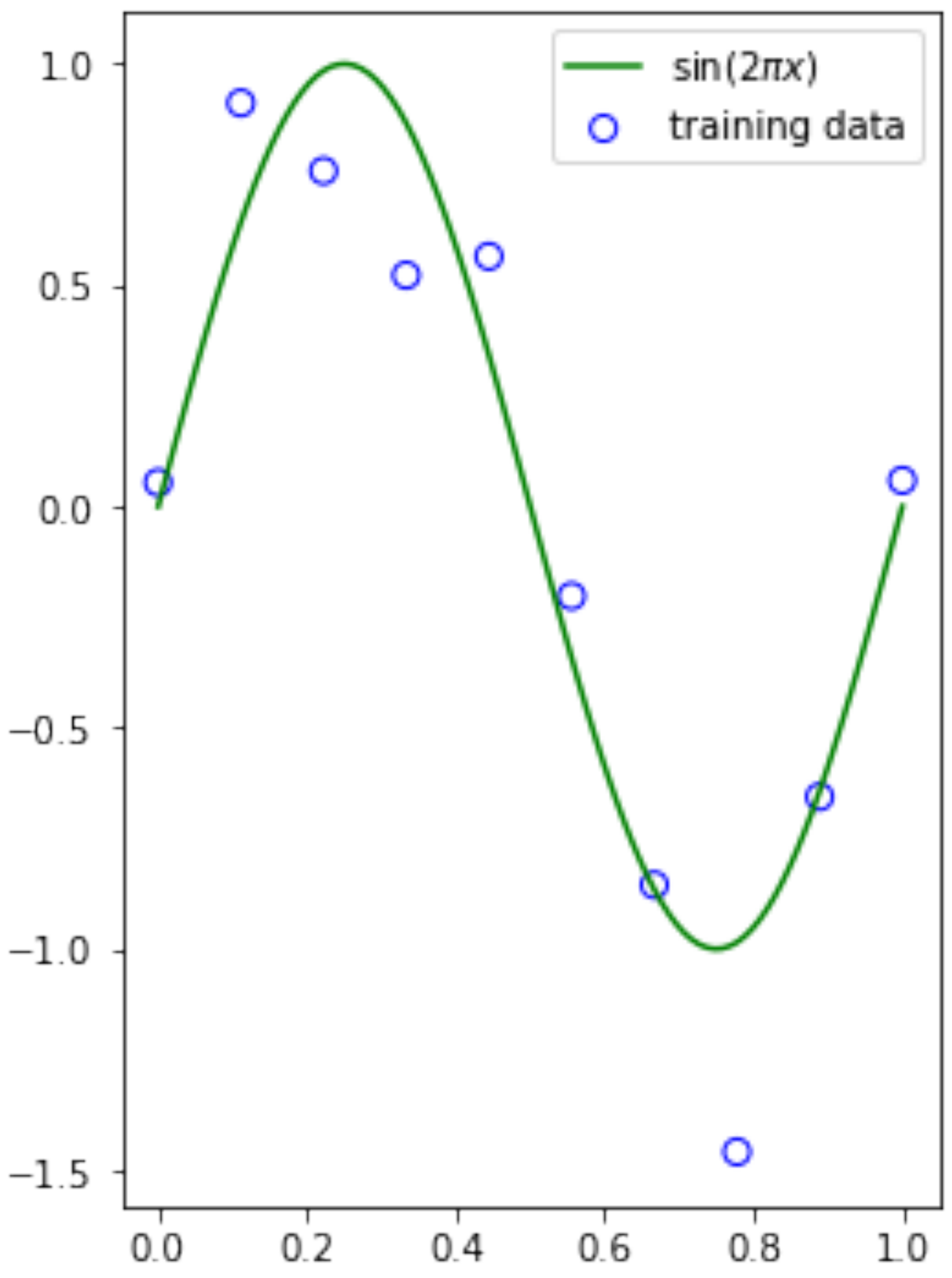
Objective: make good predictions $\hat{y}$ of new values $\hat{x}$

Generalize from a finite data set

Model: polynomial function of order of M

$$h(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{m=0}^{M} w_m x^m$$

Although $h(x, \mathbf{w})$ is a nonlinear function of x, it is a linear function of the coefficients $\mathbf{w}$ => linear model



FITTING

Determine the coefficients $\mathbf{w}$ by fitting to N training samples

Minimize error function $E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(h(x_n, \mathbf{w}) - t_n\right)^2$

Again: quadratic function of coefficients $\mathbf{w}$

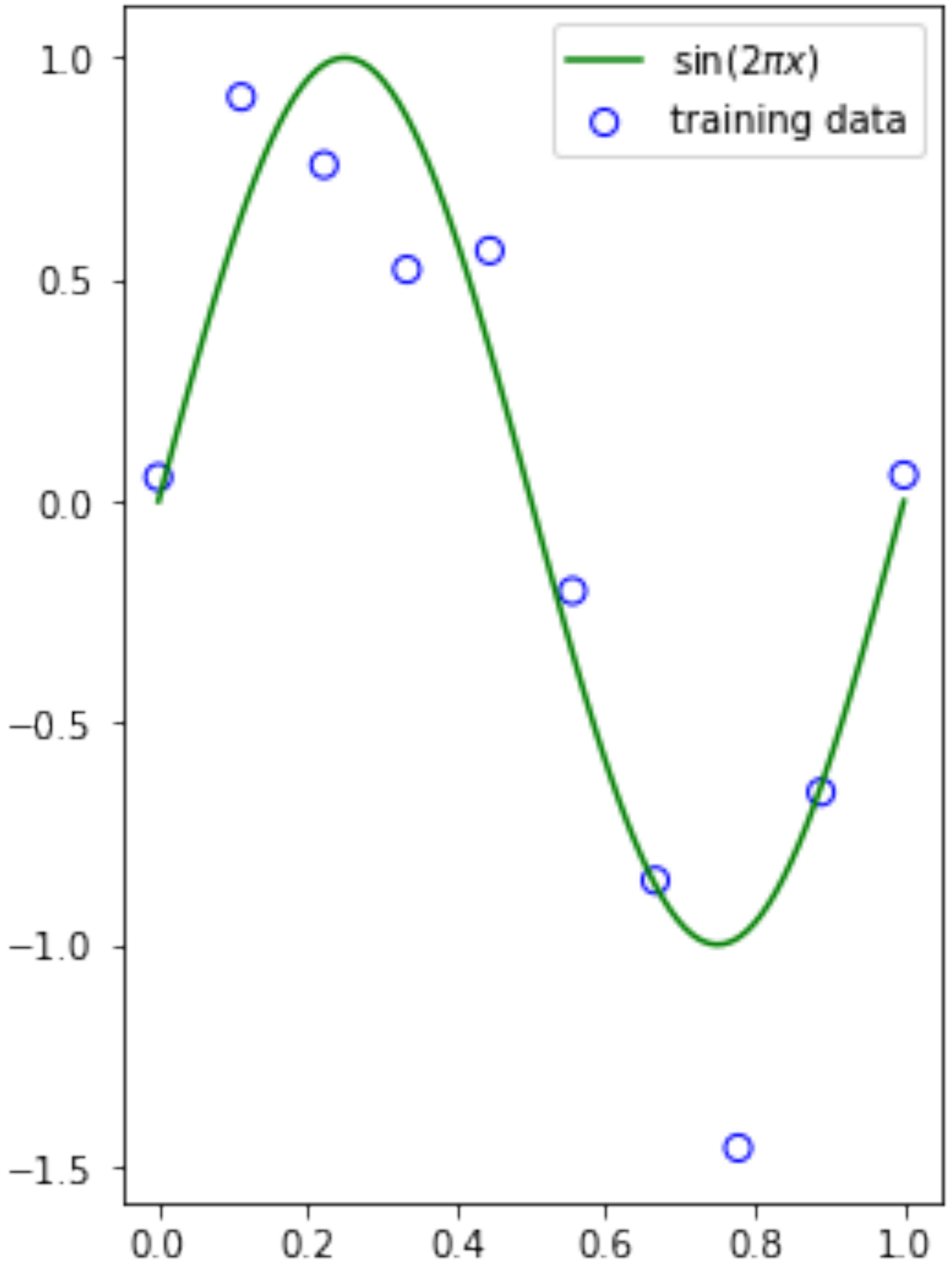
=> partial derivates (with respect to the coefficients) are linear in the elements of $\mathbf{w}$

=> unique solution $\mathbf{w}^*$

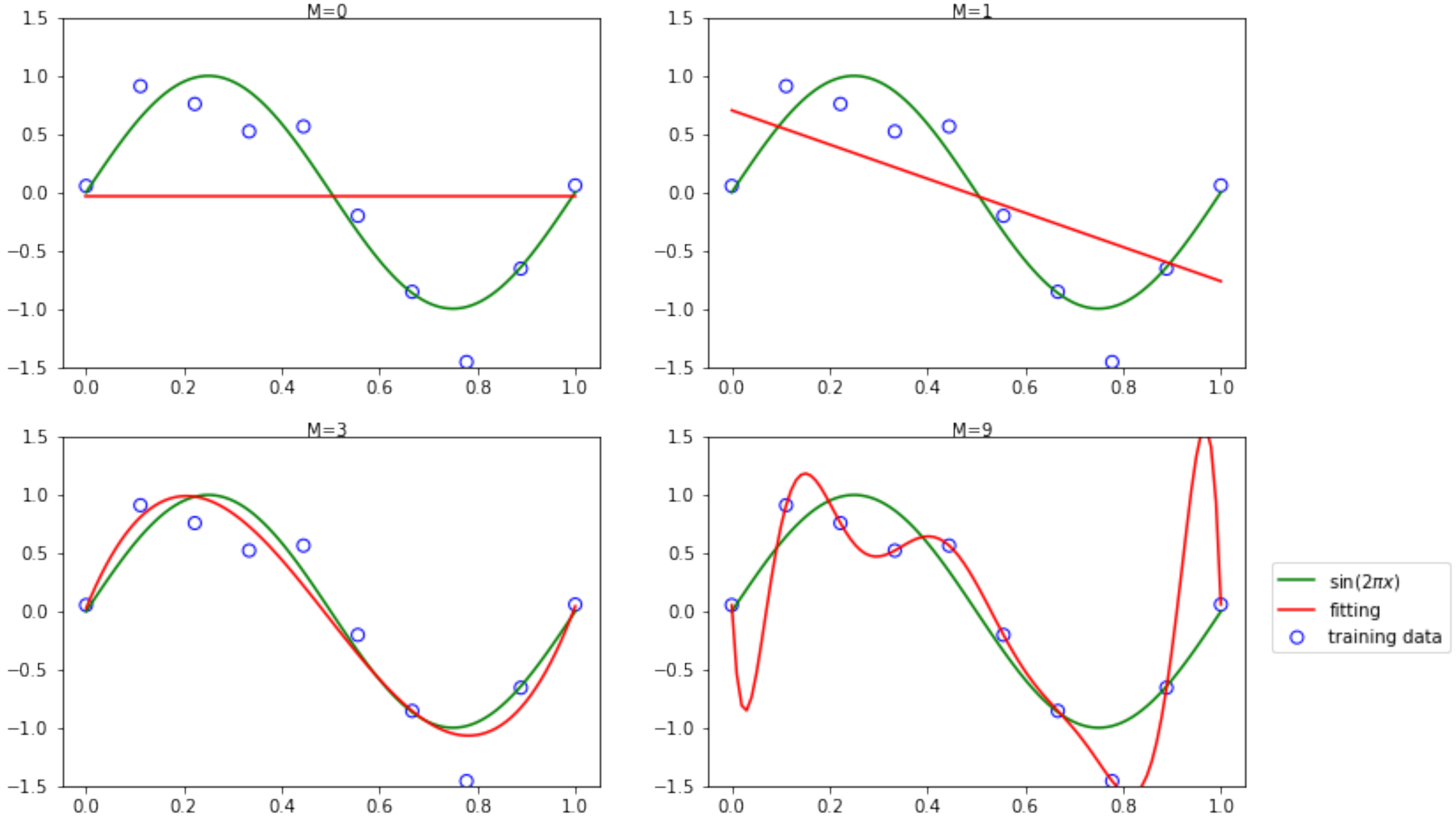
But what about order M?

$$h(x, \mathbf{w}) = \sum_{m=0}^{M} w_m x^m$$

=> model selection



MODEL SELECTION



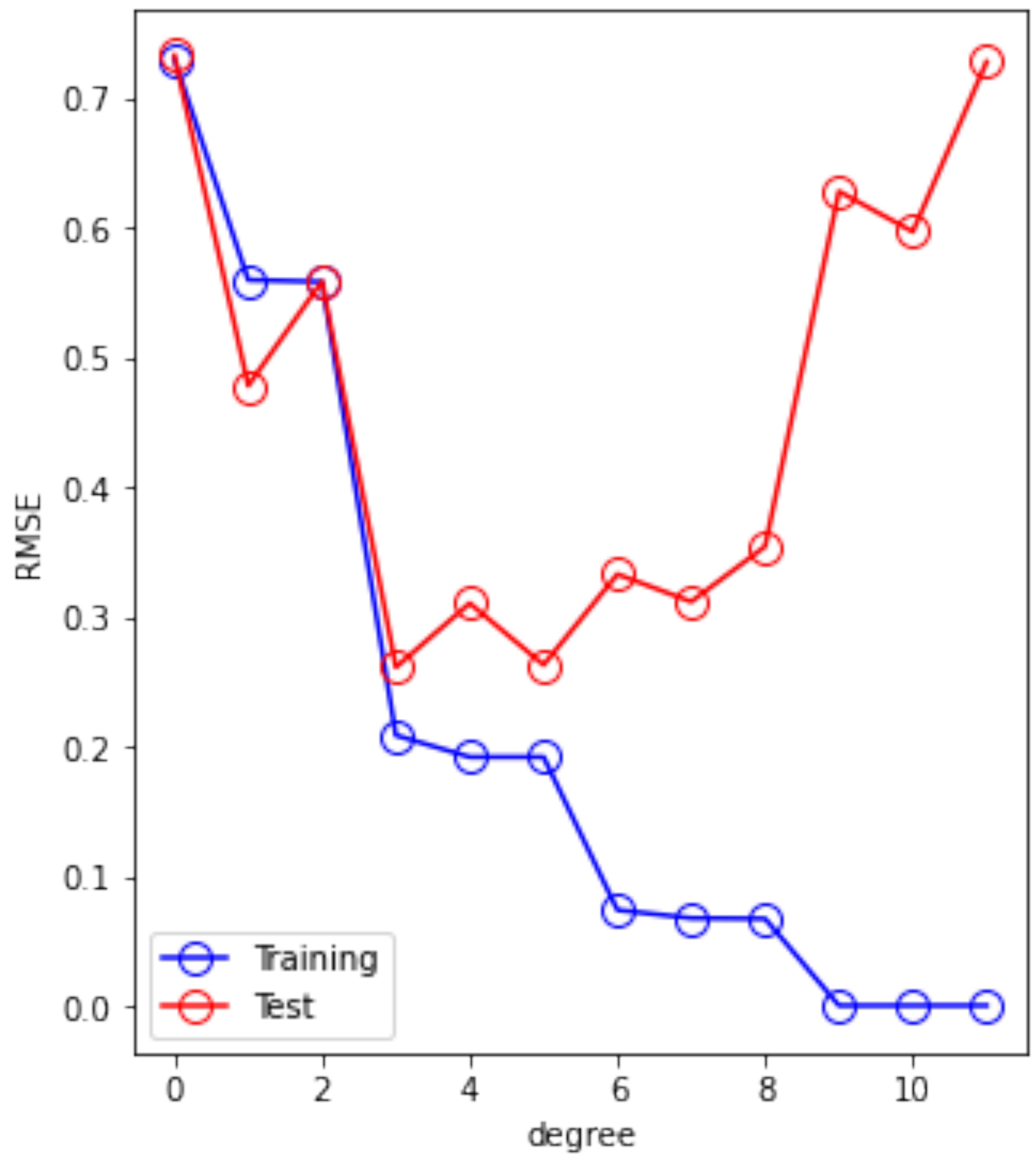
GENERALIZATION AND OVERFITTING

Good generalization: making accurate predictions for new (unseen) data

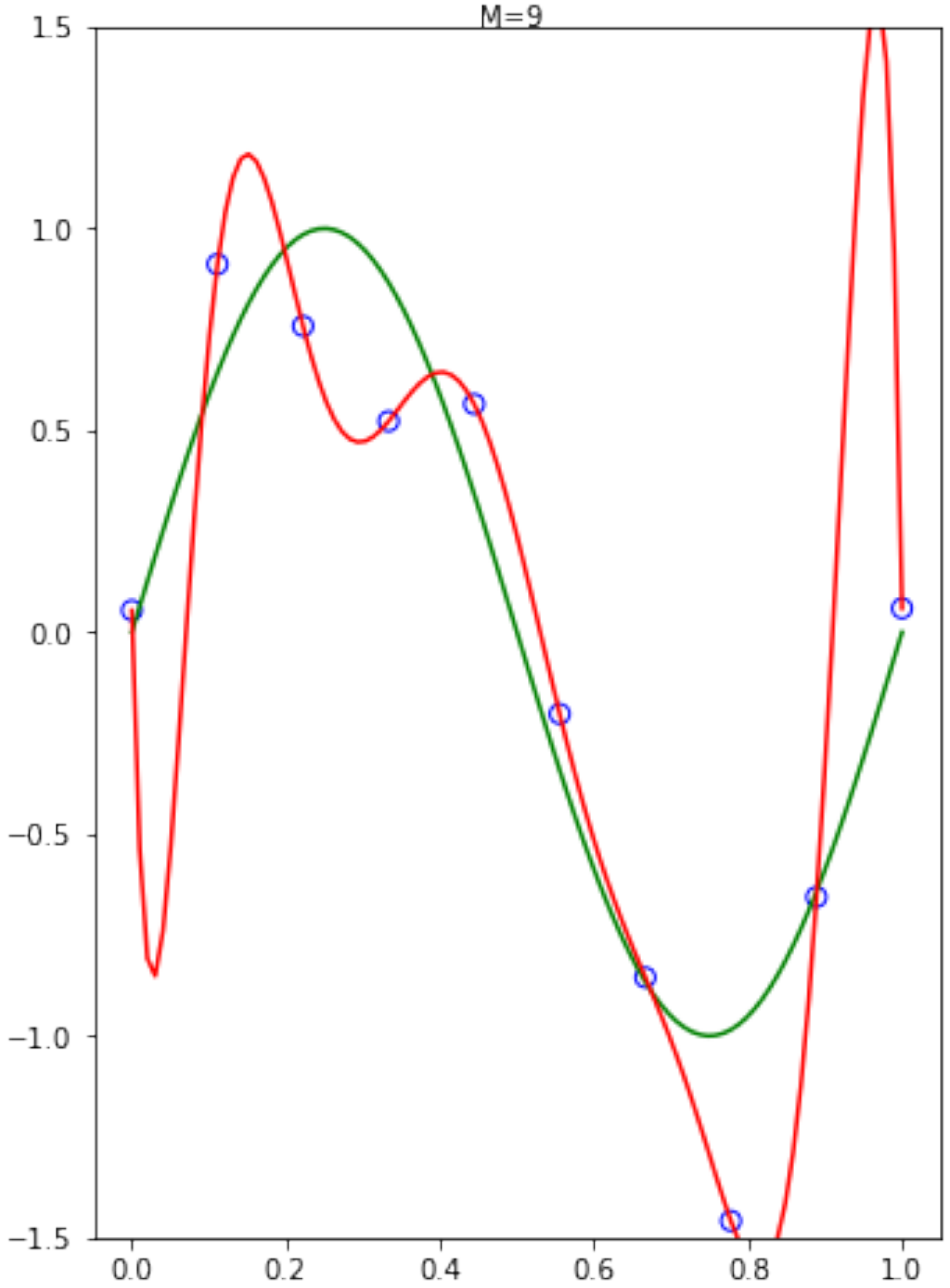
- Test set: here generated the same way as training set
- Usual procedure: Split dataset into training, test (and sometimes validation) sets, with the test set remaining unknown to the model during training (Very important!)

Identify overfitting

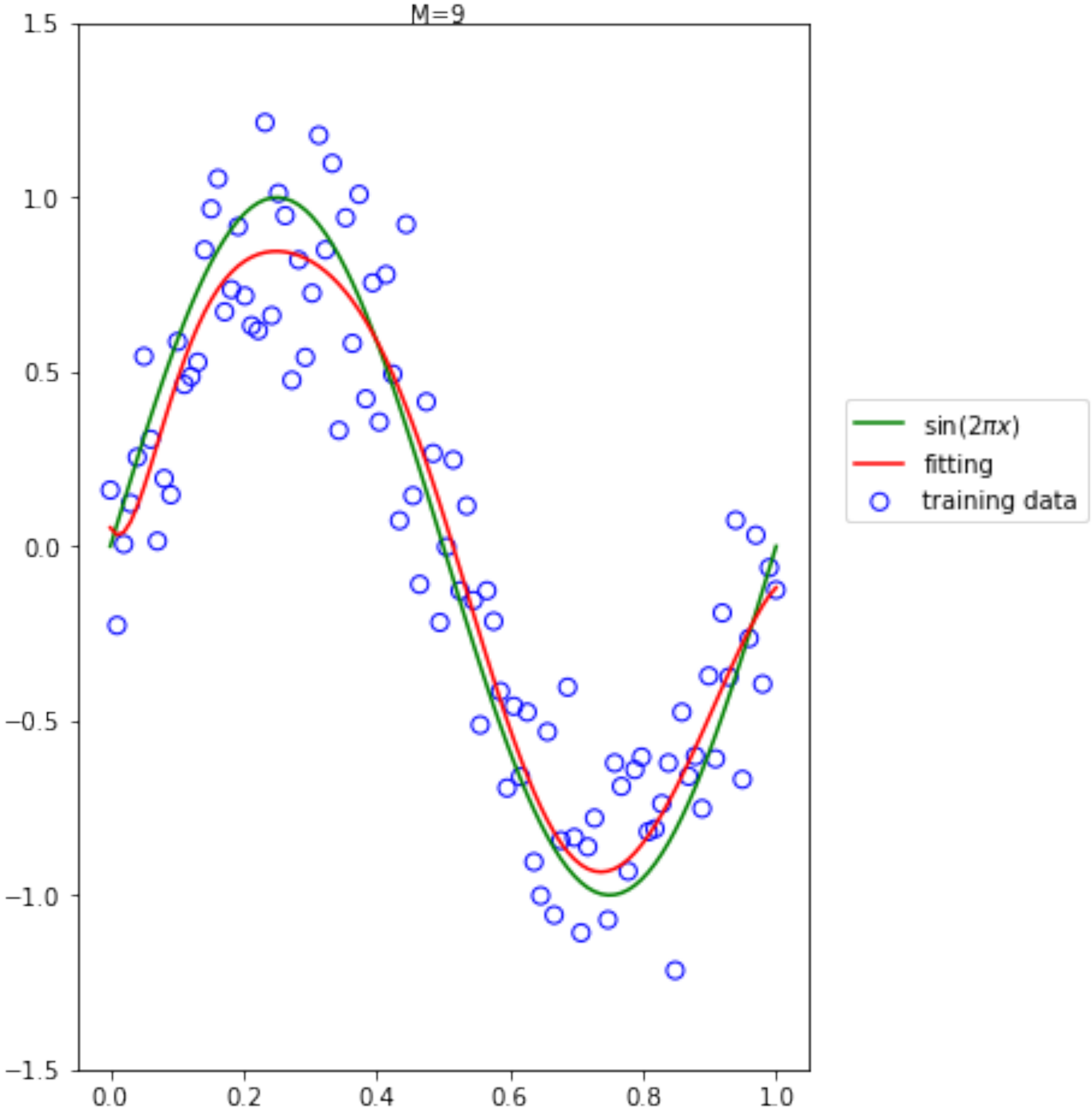
- Training error: $E(\mathbf{w}^*)$ for the training set
- Test error: $E(\mathbf{w}^*)$ for the test set
- If datasets are of different size: $E_{RMS} = \sqrt{2E(\mathbf{w})/N}$



MODEL SELECTION DEPENDS ON DATA SET SIZE



N=10



N=100

REGULARIZATION

Regularization can control overfitting by adding a penalty term to the error function

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(h(x_n, \mathbf{w}) - t_n\right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|$$

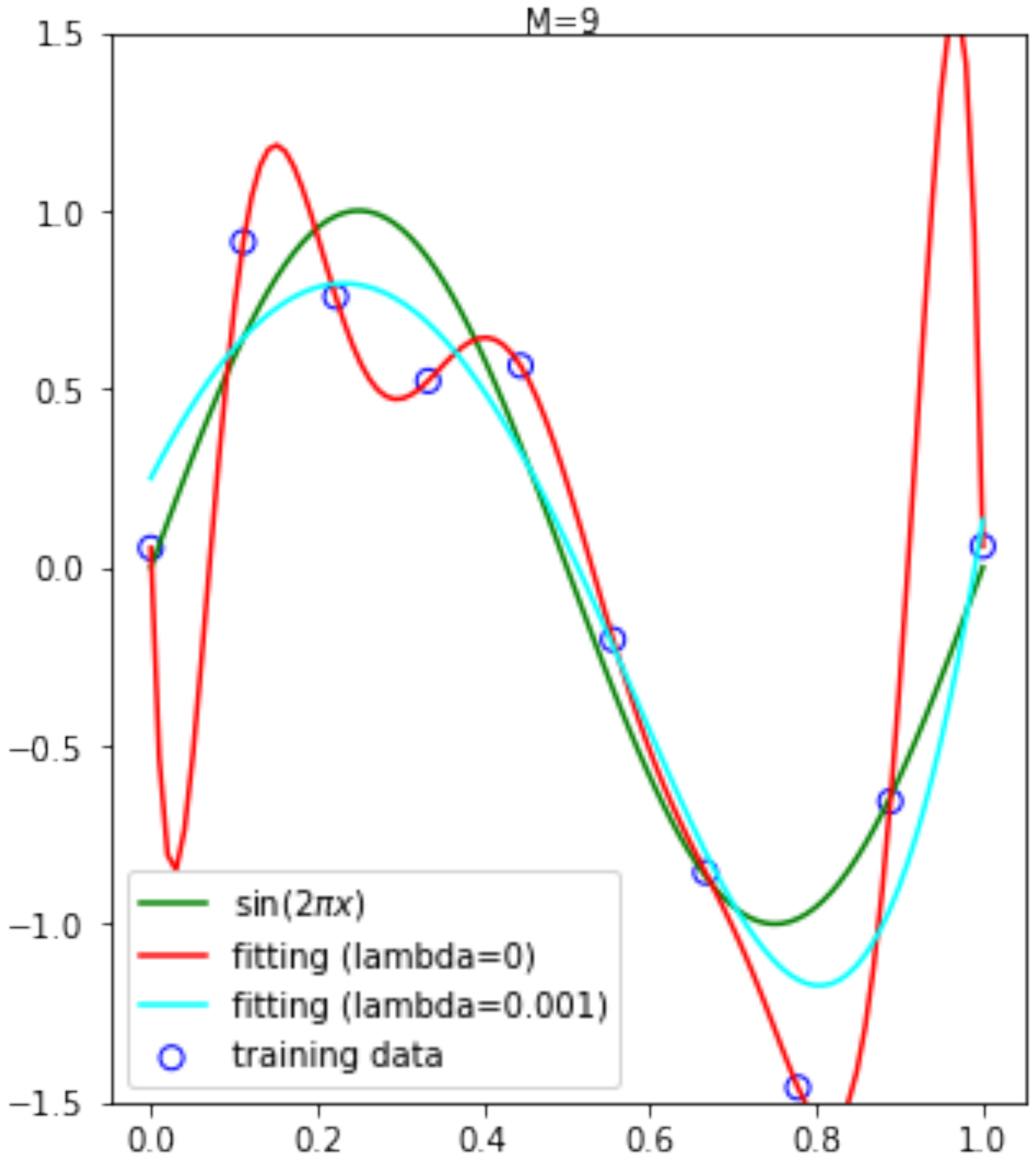
where $\|\mathbf{w}\| = \mathbf{w}^T\mathbf{w}$

λ governs the relative importance of the regularization term

Such shrinkage methods reduce the value of the coefficients

Quadratic regularizer: *ridge regression* or *weight decay* or *L2 regularization*

Validation set to optimize either M or λ



WRAPPING UP

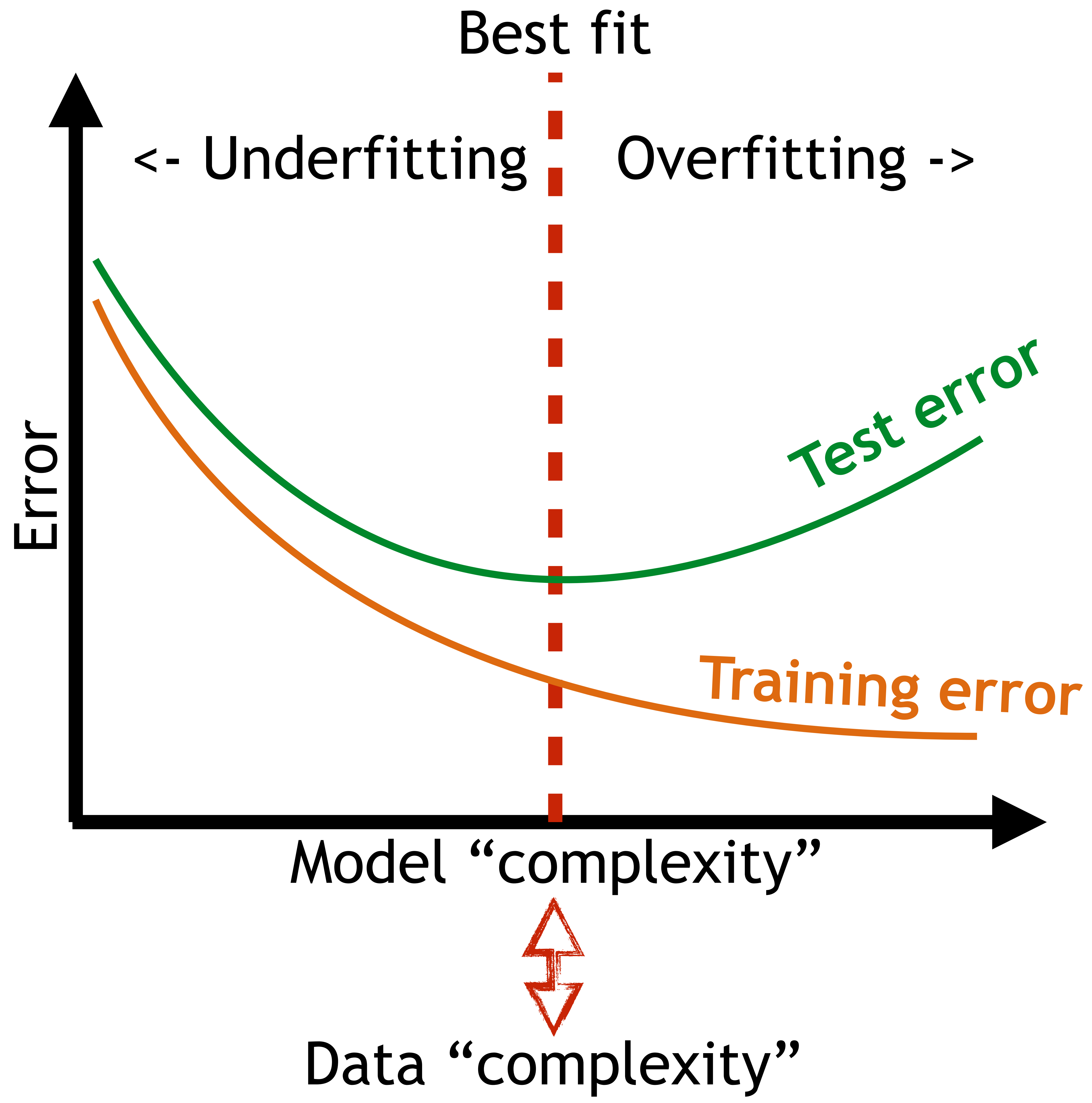
SUMMARY

This course: Teaches how Neural Networks actually work under the hood

Linear regression example

- Parameters and how they are learned
- Generalization and model selection
- Overfitting and regularization
- Linear models are *not* universal approximators

Artificial Neural Networks (ANNs) *are* universal approximators, but their gains are paid in higher computational complexity and lower interpretability



EXERCISE

Online at the materials page

- Curve fitting on a simple dataset
- Code templates provided
- First with numpy as a reference
- Second with manually implemented SGD
- Investigation of: Over/under-fitting, dataset size

To be submitted via e-mail by:

- Wednesday 9:00

Discussion after the next lecture

https://hawaii.ziti.uni-heidelberg.de/teaching/ap_nn_from_scratch_materials_wise2025/