Photogrammetry & Robotics Lab

Machine Learning for Robotics and Computer Vision

Introduction

Jens Behley

What is Machine Learning?

- Classical definition by Tom Mitchell:

> A computer program is said to **learn** from *experience* E with respect to some class of *task* T, and *performance measure* P, if its performance at tasks in T, as measured by P, improves with experience E.

From: Tom M. Mitchell, Machine Learning, McGraw-Hill, 1997

But what is Machine Learning?

- Rather broad definition that captures the many faces of Machine Learning (ML)

- **Goal:** We want to design algorithms that **automatically** extract valuable information from **data** via adapting a **model**

- Adaptation of the model based on data is what is called **learning**

Examples from Everyday Life

- Many things are powered by ML:
  - Spam filter for your emails
  - Auto correct on smart phones
  - Recommendations on streaming services
  - Speech recognition in personal assistants
  - …

- Often tasks where it's hard to write a program by fixed rules
- ML solves this by learning from a large set of examples (= experiences)

Examples in Robotics & Computer Vision

- Perception in self-driving cars
- Semantic interpretation of images

Contents of the Course

- **Part I**: Traditional ML methods
  - Basics, ML terminology, general ML models for classification, regression & clustering

- **Part II**: Deep Learning for Vision Tasks
  - Convolutional Neural Networks (CNN), learning CNNs, current research topics

People

Jens Behley

Lucas Nunes

Lecture, Exercises, and Exam

- Lectures as **video** recordings
- Tutorials & questions via **Zoom**
- **eCampus** website for further information
- Homework assignments
- **Deadlines:** see eCampus deadlines

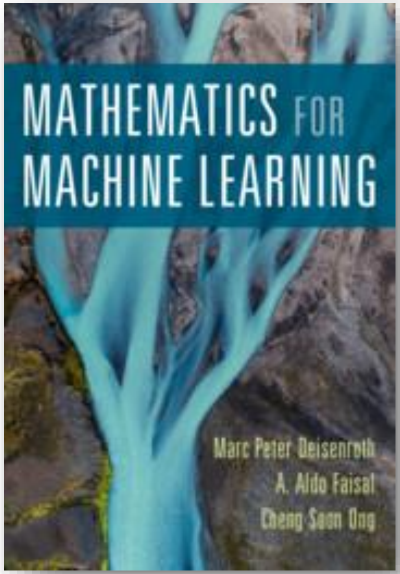
- **Oral** exam (most likely via Zoom)

Homework Assignments

- **Exam Admission: at least 50%** of the points overall
- Coding (Python) is an essential part of the homework assignments
- Assignment/submission via **eCampus**
- No plagiarism! Copying of solutions is not accepted **(zero tolerance policy)**

Books

- Most of the contents are based on two freely available books



https://mml-book.com

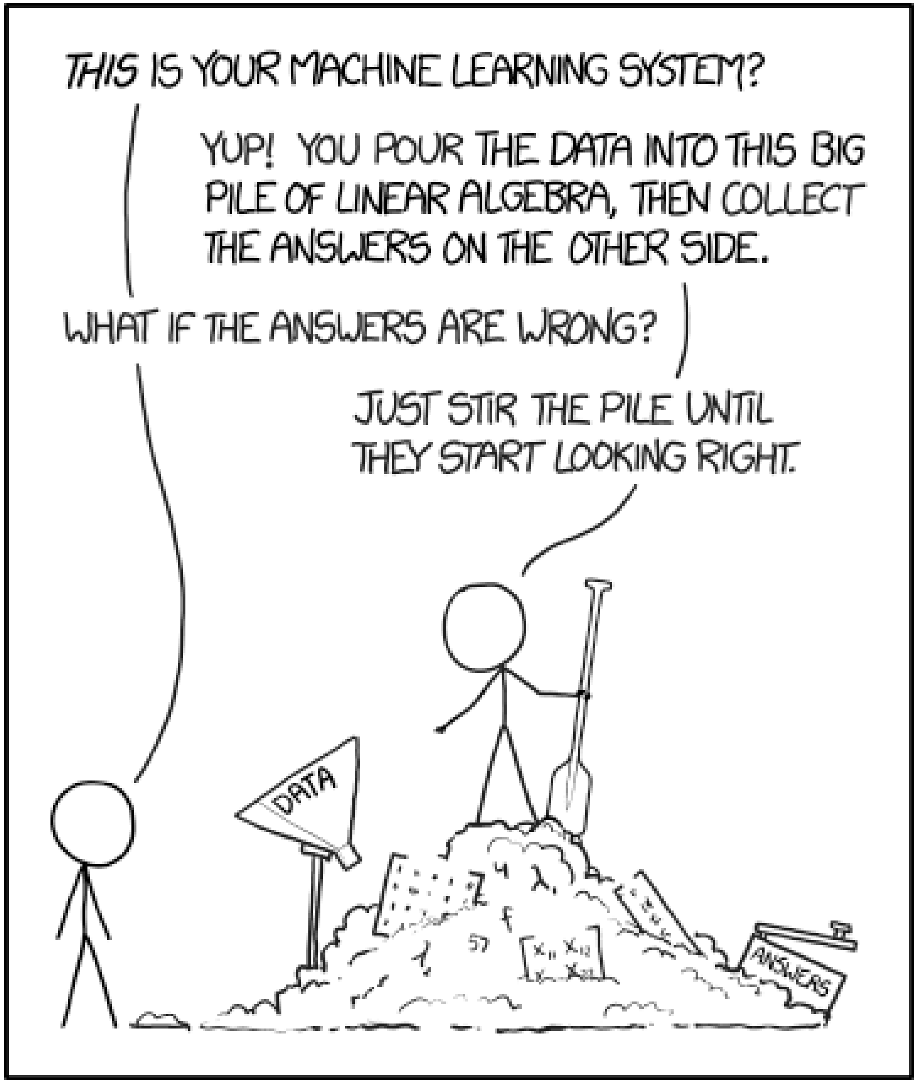
[MML]

http://www.computervisionmodels.com/

[CVMLI]

[MML Chap. 9.3]

Doing a better job...



Source: https://xkcd.com/1838/

High-level overview

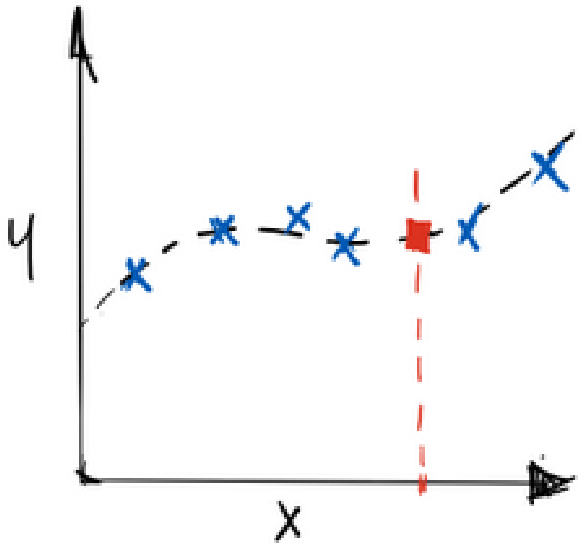
- **Main components** of ML algorithm
  1. **Data**
  2. **Model**
  3. **Learning**

- Here, high-level overview of the concepts
- In next lectures, we will dive into specific models for specific tasks and cover learning

Types of Learning

- **Supervised** Learning
  - Examples with target values (= labels) available
  - Targets can be continuous or discrete
  - **Goal:** Learn predictors for inferring the target value for unseen data
- **Unsupervised** Learning
  - Examples without explicit targets (unlabeled)
  - **Goal:** Analysis of the data or learn generative models or representations of the data

Regression vs. Classification



$\mathbf{x}$	y
(0.8, 0.2, 0.3, 0.9)	1
(0.3, 0.2, 0.3, 0.4)	2
(0.1, 0.9, 0.4, 0.1)	1
(0.5, 0.1, 0.2, 0.2)	3
(0.1, 0.2, 0.8, 0.7)	?

- **Regression**
  - Input: example $\mathbf{x} \in \mathbb{R}^D$
  - Output: target values $y \in \mathbb{R}$ (continuous)
- **Classification**
  - Input: example $\mathbf{x} \in \mathbb{R}^D$
  - Output: class $y \in \{1, \ldots, k\} = \mathcal{Y}$ (discrete)

Data

Where do we get the data?

- Research: (Benchmark) datasets available that provide data in the right format
  - Advantage: authors of the dataset ensured that data is ready to go
  - Disadvantage: Specific domain & dataset biases, may not fit your task

- Getting data for new tasks & domains
  - Retrieve from databases or crawl internet
  - Collect/record new data (robot, camera, …)

Unstructured Data

- Data from multiple sources is usually unstructured
- Data can have different types of fields
  - Categorical values, e.g., “red”, “green”, …
  - Strings, e.g., names, …
  - Missing values, e.g., “ ”, “NULL”, …
  - Misspelled values
  - etc.

Data Wrangling

- Process of transforming raw data into a usable format

- Main task: cleaning and converting values

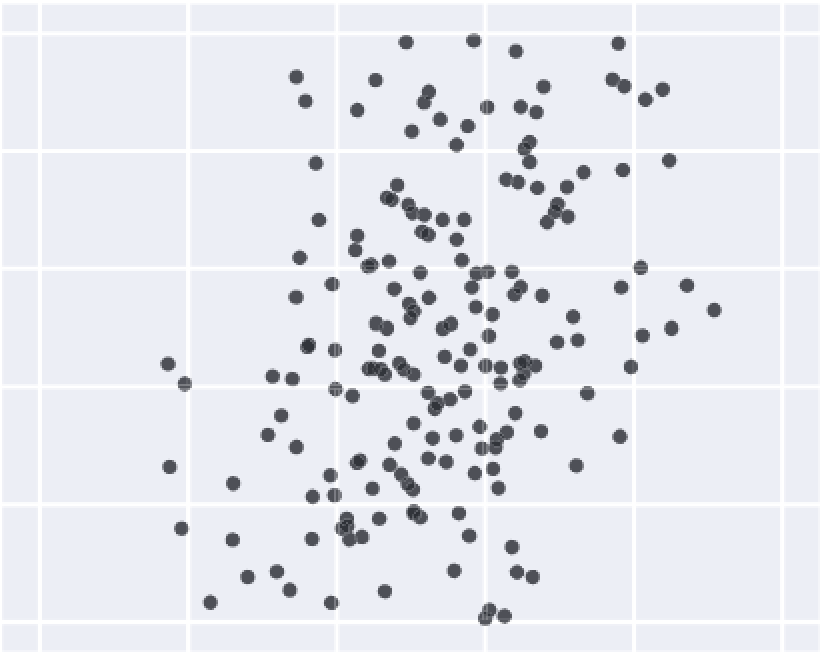
- Handling missing values
  - Replace with 0 (= “ignore feature”)
  - Impute values, e.g., mean of other values, regression, …
  - Drop example (worst option)

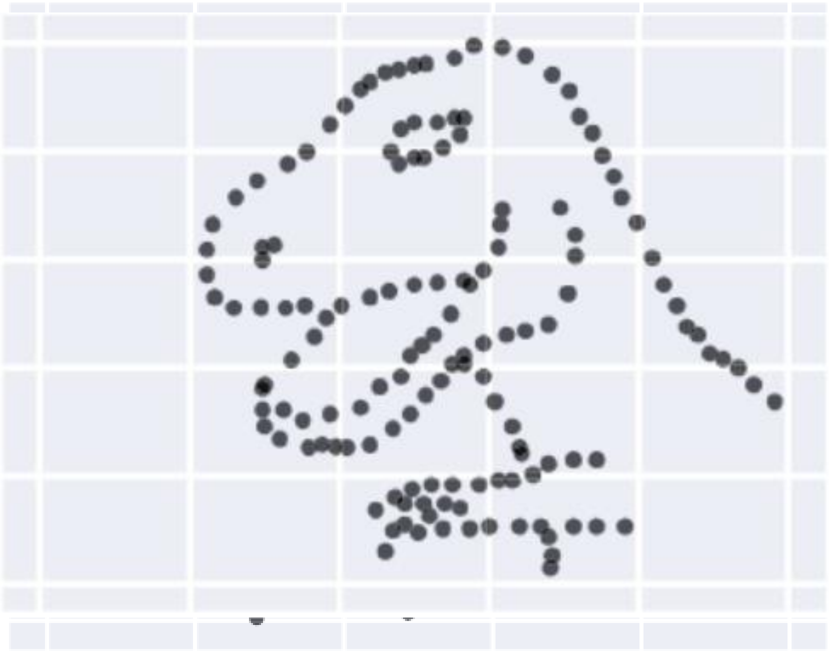
[MML Chap. 8.1.1]

Visualize your data!

- Summary statistics are not enough!
- Let’s say you have a dataset $\mathcal{X}, \mathbf{x}_n \in \mathbb{R}^2$, with following statistics (mean and standard deviation):

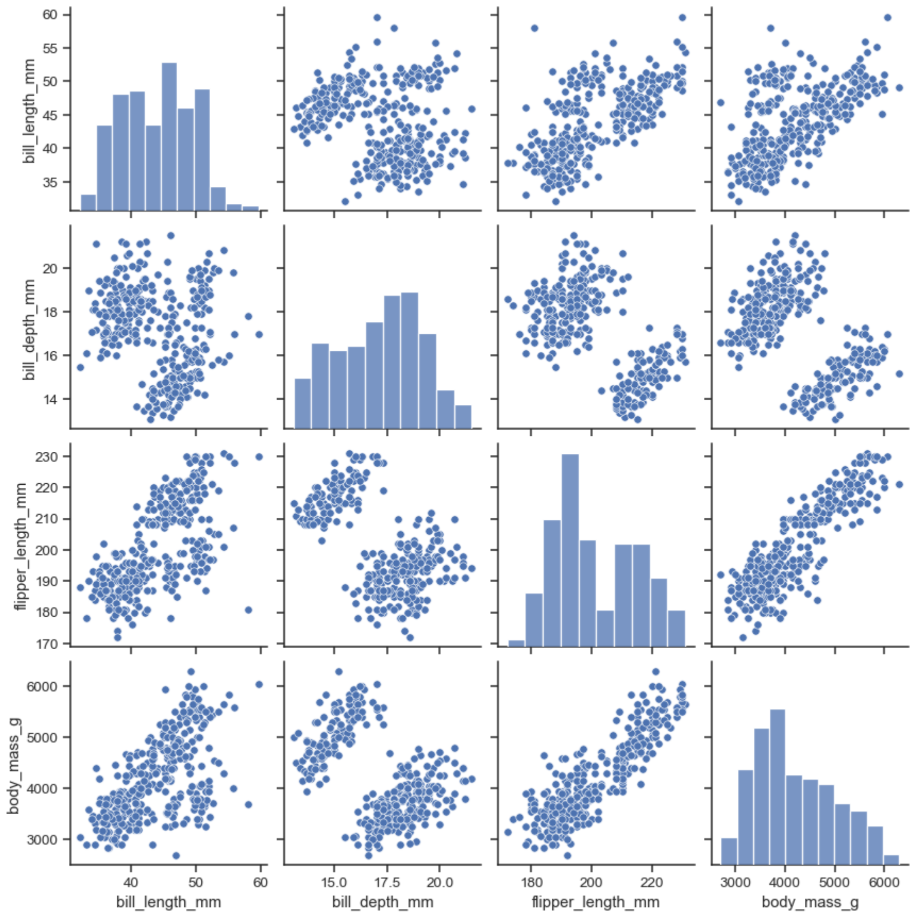
$$\mu = (54.28, 47.83), \ \sigma = (16.76, 26.93)$$





From: https://www.autodesk.com/research/publications/same-stats-different-graphs

Plotting multi-dimensional Data



Source: seaborn.pairplot

- Plot pairs that might show dependencies and relations

Data in Supervised Learning

- Examples in supervised learning given as set of tuples

$$\mathcal{X} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n), \ldots, (\mathbf{x}_N, y_N)\}$$

  - Data vectors $\mathbf{x}_n \in \mathbb{R}^D$ called **feature vector**
  - Entry $x_d \in \mathbb{R}$ of data vector
    $$\mathbf{x}_n = (x_1, \ldots, x_d, \ldots, x_D)$$
    is called **feature**, attribute, or covariate
  - y is called **label**, target, response, or annotation
  - Data used for learning is called **training set** $\mathcal{X}_{\text{train}}$

[MML Chap. 8.1.1]

Example: Apples vs. Pears

- Let's say we want to classify fruit images
- **Features:** "roundness" and "redness"
- Image from which we determine the following features
  - $x_1 \in [0, 1]$ = ratio of radii of fitted ellipse radii
  - $x_2 \in [0, 1]$ = avg. of red values of fruit pixels
- Output:
  - $y = \{0, 1\}$ → {Apple, Pear}

  - → **binary classification** problem

Example: Apple vs. Pears

- Gathered images, curated them, extracted features

© Abhijit Tembhekar

→ (0.95, 0.96)

© I, NibbiP

→ (0.15, 0.5)

© PiccoloNamek

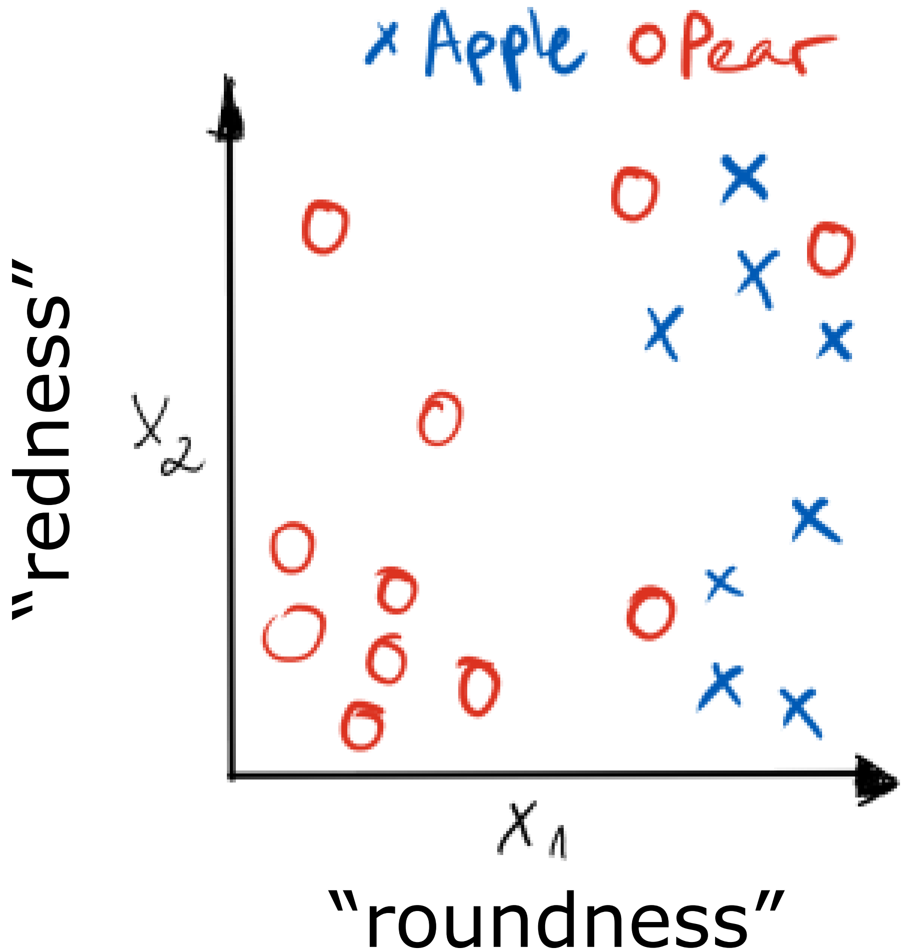
→ (0.96, 0.1)

© CDC

→ (0.95, 0.9)

Example: Apple vs. Pears

- We get the following **feature space** where each point corresponds to an example:



Model

Function vs. Probabilistic Model

- Different views on the role of the model as a predictor

- Models as **functions** $f : \mathbb{R}^D \mapsto \mathbb{R}$ producing an output y for given input $\mathbf{x} \in \mathbb{R}^D$:

$$y = f(x)$$

- Models as **probability distribution** $P(y|\mathbf{x})$:

$$\hat{y} = \arg\max_{c \in \mathcal{Y}} P(y = c|\mathbf{x})$$

- Advantage: Get confidence in prediction

[MML Chap. 8.1.2, 8.1.3]

Parameters & Hyperparameters

- The model is specified by adjustable **parameters** θ that are learned from data
- Amount of model parameters determine the **capacity** of the model

- **Hyperparameters** = Parameters of the model that are not learned from data
- Hyperparameters are selected in advance

[MML Chap. 8.1.4]

Example: Apple vs. Pears (cont) k-Nearest Neighbor Classifier

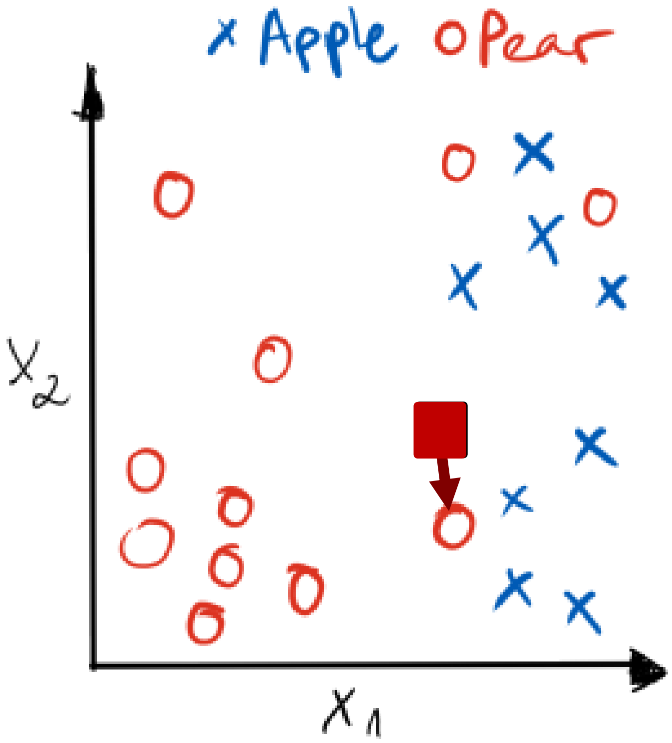
- Example: **k-Nearest Neighbor Classifier**

- Let $\mathcal{N}_k(\mathbf{x})$ the k-nearest neighbors of $\mathbf{x}$ from $(\mathbf{x}_n, y) \in \mathcal{X}_{\text{train}}$
- Then $P(y = c|\mathbf{x})$ is given by

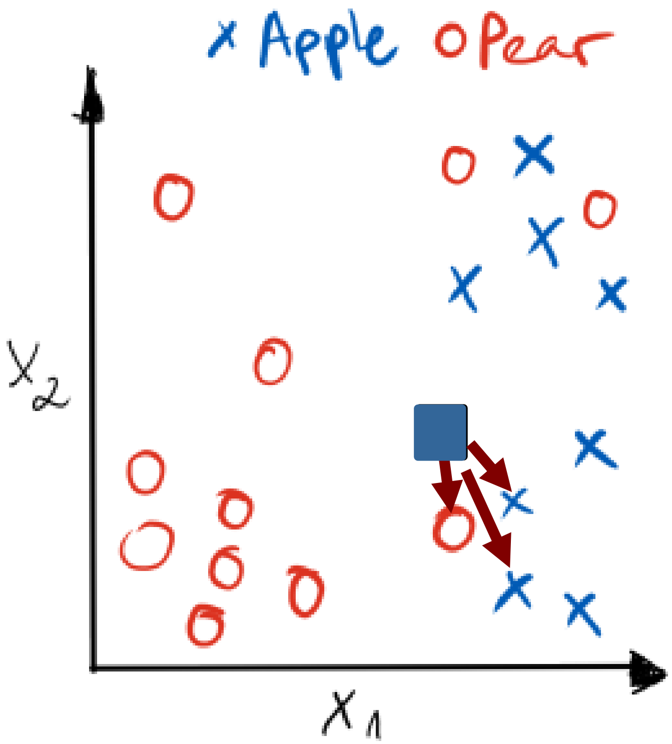
$$P(y = c|\mathbf{x}) = \frac{1}{k} \left|\{(x_k, y_k) \in \mathcal{N}_k(\mathbf{x}) \mid y_k = c\}\right|$$

- **Hyperparameter:** number of neighbors k

Example: Apple vs. Pears (cont) k-Nearest Neighbor Classifier



k=1



k=3

- Depending on k different outputs

Learning

Learning the model

- Learning is adapting the **parameters** θ of the model given the training set $\mathcal{X}_{\text{train}}$
- In most cases its maximizing the likelihood

$$P(y_1, \ldots, y_N | \mathbf{x}_1, \ldots, \mathbf{x}_n, \theta)$$

- Assumption: set of examples are **<u>i</u>ndependent and <u>i</u>dentically <u>d</u>istributed**
- Thus, learning is an optimization problem:

$$\theta^\star = \arg\max_\theta \prod_{n=1}^N P(y_n | \mathbf{x}_n)$$

[MML Chap. 8.1.4, 8.3]

Example: Apple vs. Pears (cont)
k-Nearest Neighbor Classifier

- Learning the k-Nearest Neighbors classifier is simple ➔ store the training set $\mathcal{X}_{\text{train}}$

- How well does the model perform?
- How to chose hyperparameter k?

Evaluation

Measuring Performance

- **Goal:** Learn models that performs well on unseen data → generalization
- How do we measure performance? Metrics!
- **Training error** = how well do we determine target values y_n for each $\mathbf{x}_n$
- Other metrics of interest possible

- **Problem:** training error just tells us performance on seen data; should be ideally close to zero

Training and Test set

- **Solution:** Evaluate model performance on part of the data not used for training!

- Data used for learning, called **training**, is called **training set** $\mathcal{X}_{\text{train}}$
- Data used for evaluating trained model is **test set** $\mathcal{X}_{\text{test}}$

- It holds $\mathcal{X}_{\text{train}} \cap \mathcal{X}_{\text{test}} = \emptyset$!
- Test set **not** used for determining the parameters!
- Test set should be only used seldomly!

Human in the Loop Phenomenon

- Test set is our *estimate* how algorithm works on **unseen** data
- But you might be tempted to just "look" at the test data and select hyperparameters or parameters that "look" good on the test data
- Overfitting to test data and biased result possible → no insight about generalization

- "Good looking" model might fail miserably in real world application!

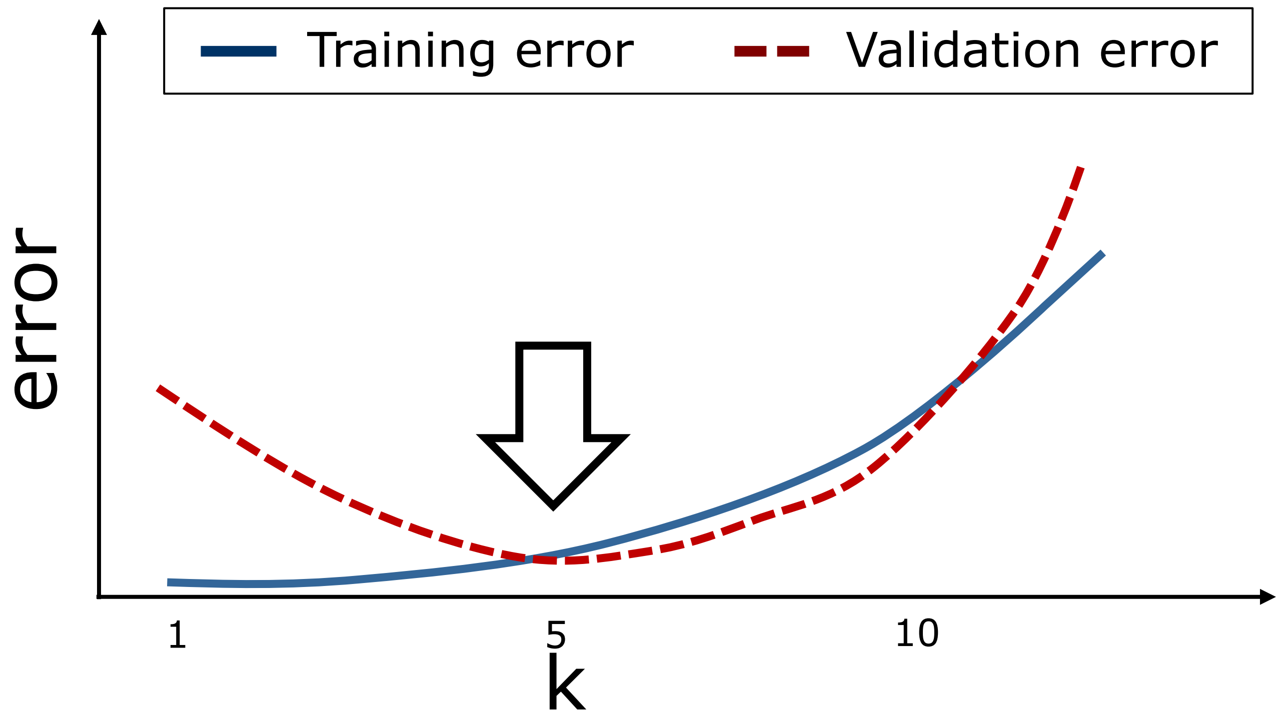
Validation set

- **Selecting hyperparameters** should happen on a so-called **validation set**
- Validation set is usually part from the training set
- But: validation set is also not used for training
  - With fixed hyperparameters sometimes still used for re-training the model)

- **Validation set ≠ test set!**

Example: Apple and Pears

- Selecting k such that validation error is minimized.
- Error = $\frac{|\{\mathbf{x}_n \in \mathcal{X} | \hat{y} \neq y_n\}|}{|\mathcal{X}|}$ with $\hat{y} = \arg\max_{c \in \mathcal{Y}} P(y = c|\mathbf{x})$



Again, don't fool yourself!

- Tuning on test set will lead to wrong conclusions!
- Good performance on your test set will not translate to good performance in real world if you select parameters on the test set!
- All hyperparameter choices should only happen on the validation set

Summary

- Covered high-level overview of ML algorithms
- Introduced some ML terminology
- High-level overview of the ML pipeline

- Next lectures:
  - More on regression and classification models
  - Mostly, details on more complex models and learning them

See you next week!