Lecture 15 – Exam Preparation

Analytics and Application

Our teaching portfolio consists of both analytics and domain-based courses at Bachelor, Master and PhD level

Domain	Course Name	Type	Semster	BSc	MSc	PhD
Domain & IS	Grundlagen Wirtschaftsinformatik	Course	SS+WS	✓		
Analytics	Data Science and Machine Learning (DSML)	Course	SS	✓		
Analytics	Analytics and Applications (AA)	Course	WS		✓	
Analytics	Advanced Analytics and Applications (AAA)	Course	SS		✓	
Domain & IS	Advanced Mathematical Optimization (AMO)	Course	WS		✓	
Domain & IS	Advanced Seminar Information Systems for Sustainable Society	Seminar	WS		✓	
Domain & IS	Information Systems Research – Analytics for a Sustainable Society	Seminar	SS			✓

Advanced Analytics and Applications

Sign up for our course **AAA** to learn about the use of information systems and advanced machine learning.

Specifically, we will cover topics around Soft Clustering, Spatial Analytics, Deep Learning, and Reinforcement Learning.

During **Lectures** we will discuss the inner workings of the algorithm, and during **Workshops** we will apply the discussed methods in various case studies.

When: Coming Semester, Wednesdays 14 PM – 17:30 PM

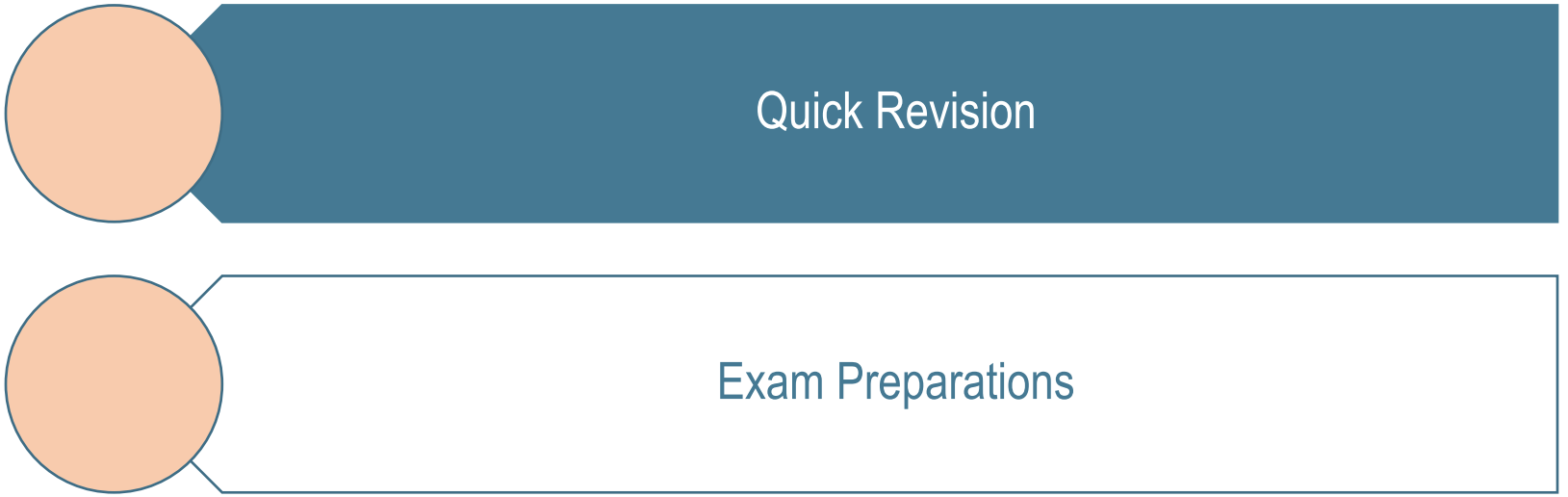
Where: tba

Lecturer: Prof. Chasin

Grading: Exam and Project (PO)

Ilias/Klips: 14277.0600

Agenda

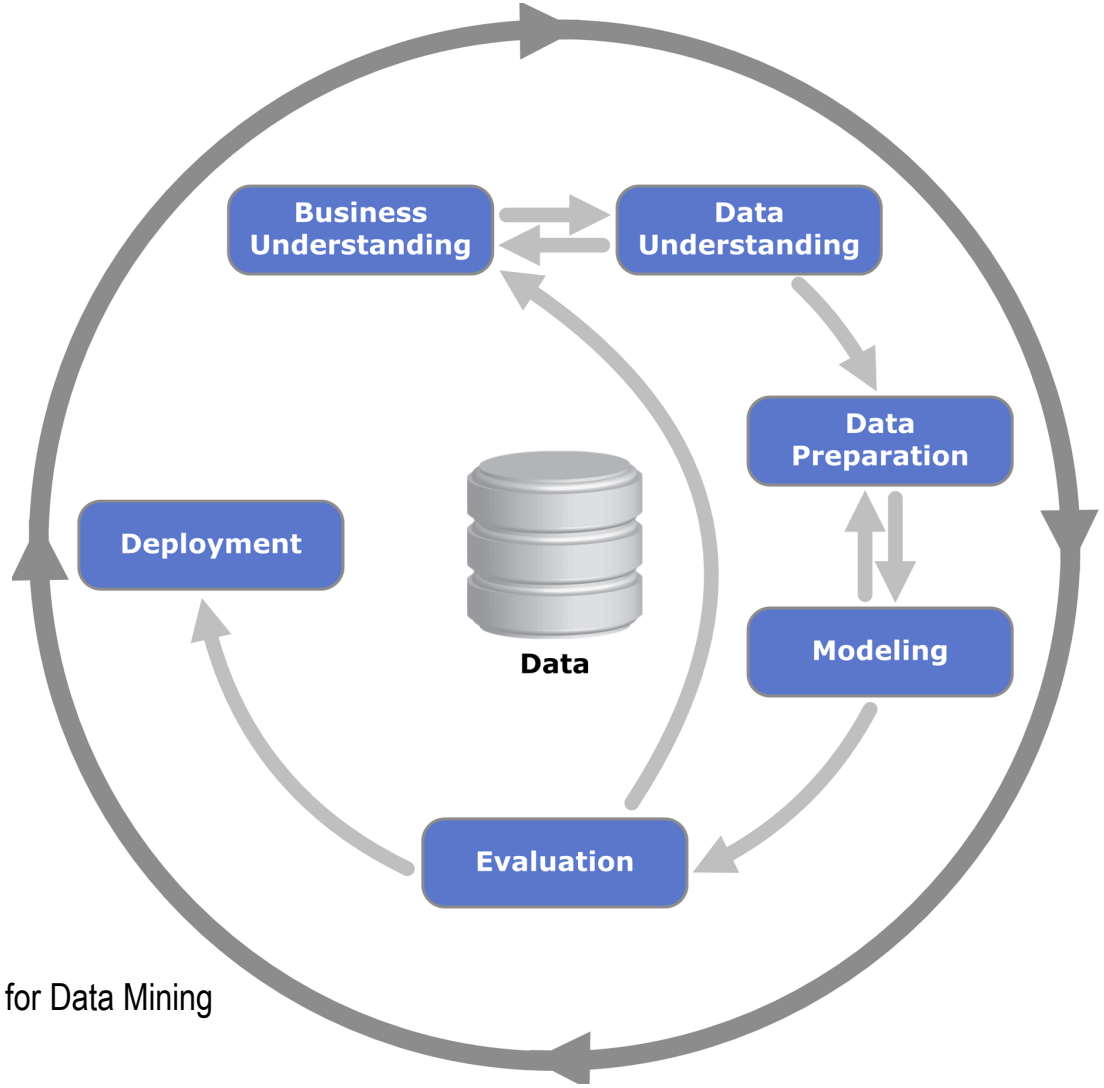


Course Schedule

#	Title	Topics	Recommended reading
#1 Oct 13	**Kick-off**	Introduction to course contents & objectives; admin & announcements	Shmueli et al., Ch. 1
#2 Oct 20	**Data Mining Process**	Core ideas in Data Mining & Explanation vs. Prediction	Shmueli et al. Ch. 2
#3 Oct 27	**Supervised Learning:** Regression (1/2)	Linear regression & Polynomial regression	Shmueli et al., Ch. 6
#4 Nov 03	**Supervised Learning:** Regression (2/2)	Performance evaluation & Regularization	Shmueli et al., Ch. 6
#5 Nov 10	**Supervised Learning:** Classification	Naive Bayes classifier & Maximum Likelihood & Logistic regression	Shmueli et al., Ch. 8 & 10
#6 Nov 17	**Supervised Learning:** Classification and Decision Trees	Classification tree & Regression tree & Performance evaluation	Shmueli et al., Ch. 9
#7 Nov 24	**Performance Evaluation**	Cross validation & Performance measures & Judging classifier performance	Shmueli et al., Ch. 5
#8 Dec 01	**Supervised Learning:** Combining methods	Ensembles: Bagging, Boosting & Uplift modeling	Shmueli et al., Ch. 13
#9 Dec 08	**Supervised Learning:** Artificial Neural Network	Intro to Artificial Neural Networks & applications	Shmueli et al., Ch. 11
#10 Dec 15	**Unsupervised Learning:** Cluster Analysis	Hard Clustering methods: K-Mean , K-means ++, Hierarchical clustering	Shmueli et al., Ch. 15
#11 Dec 22	**Unsupervised Learning:** Association Rules	Introduction to association rules & Collaborative filtering	Shmueli et al., Ch. 15
#12 Jan 12	**Times Series Analysis**	Handling time series & Forecasting methods & Smoothing methods	Shmueli et al. Ch. 16-19
#13 Jan 19	**Social Network Analysis**	Primer Graph Theory & Dijkstra for SP problems & Eigenvector centrality & Link Prediction	Shmueli et al. Ch. 19
#14 Jan 26	**Text Mining & NLP**	Introduction to Natural Language processing & Text Mining	Shmueli et al. Ch. 20
#15 Feb 02	**Wrap-up**	Course synthesis and announcements regarding exam preparations	none
Feb 02	**Deadline:** Group Projects	Submit your group projects via ILIAS by noon (12.00h)	none
Feb 15	**Exam 1**	**17.00 to 18.00**, sign up via KLIPS by **Jan. 27**, inspection date on March 11 (**upon request only**)	none
Mar 24	**Exam 2**	**17.00 to 18.00**, sign up via KLIPS by **March 11**, inspection date on April 22 (**upon request only**)	none

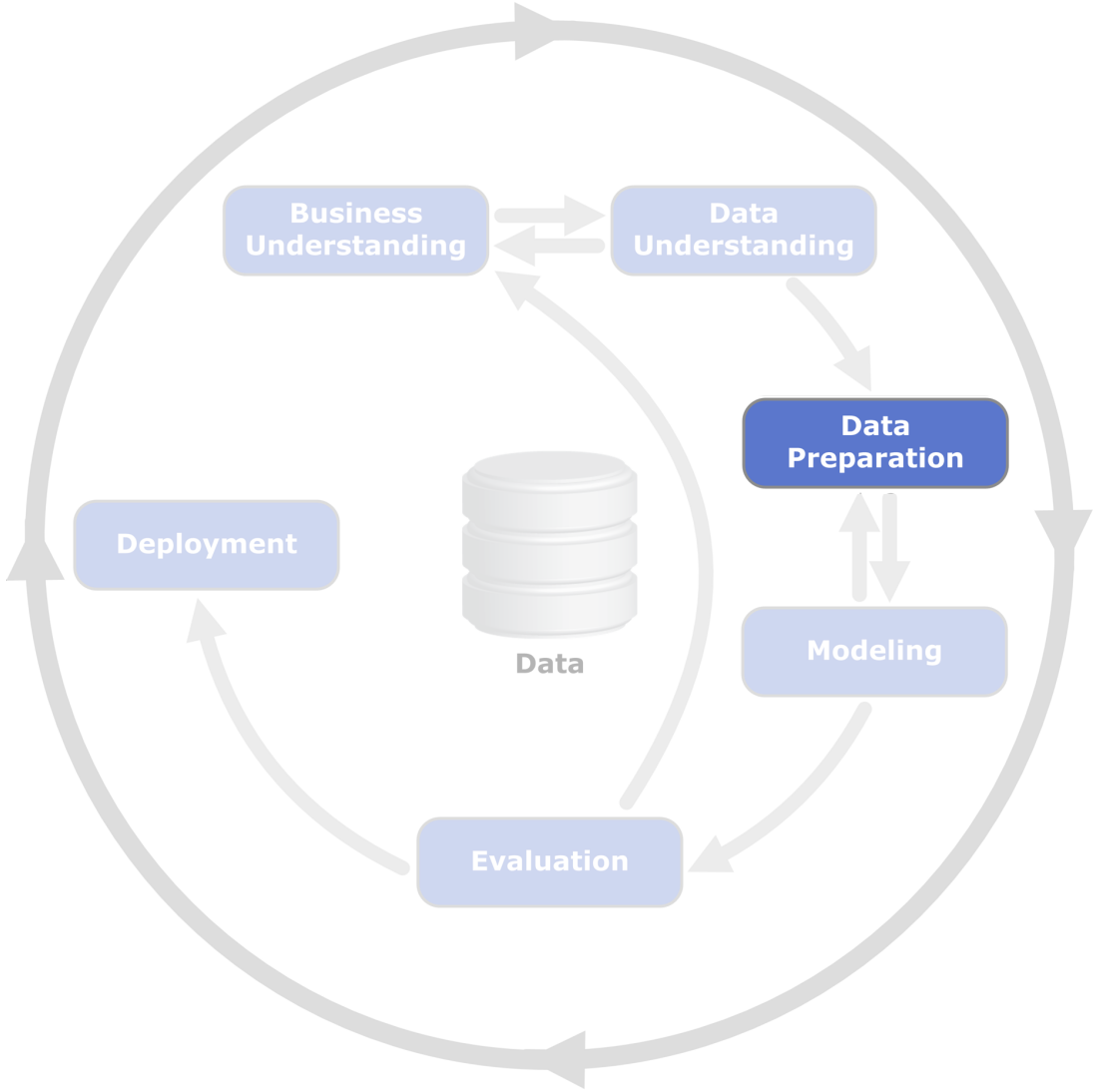
Table 1. Tentative Schedule

The CRISP* Data Science Process is a common way of describing the data science workflow



*CRISP – Cross Industry Standard Process for Data Mining

Data Preparation



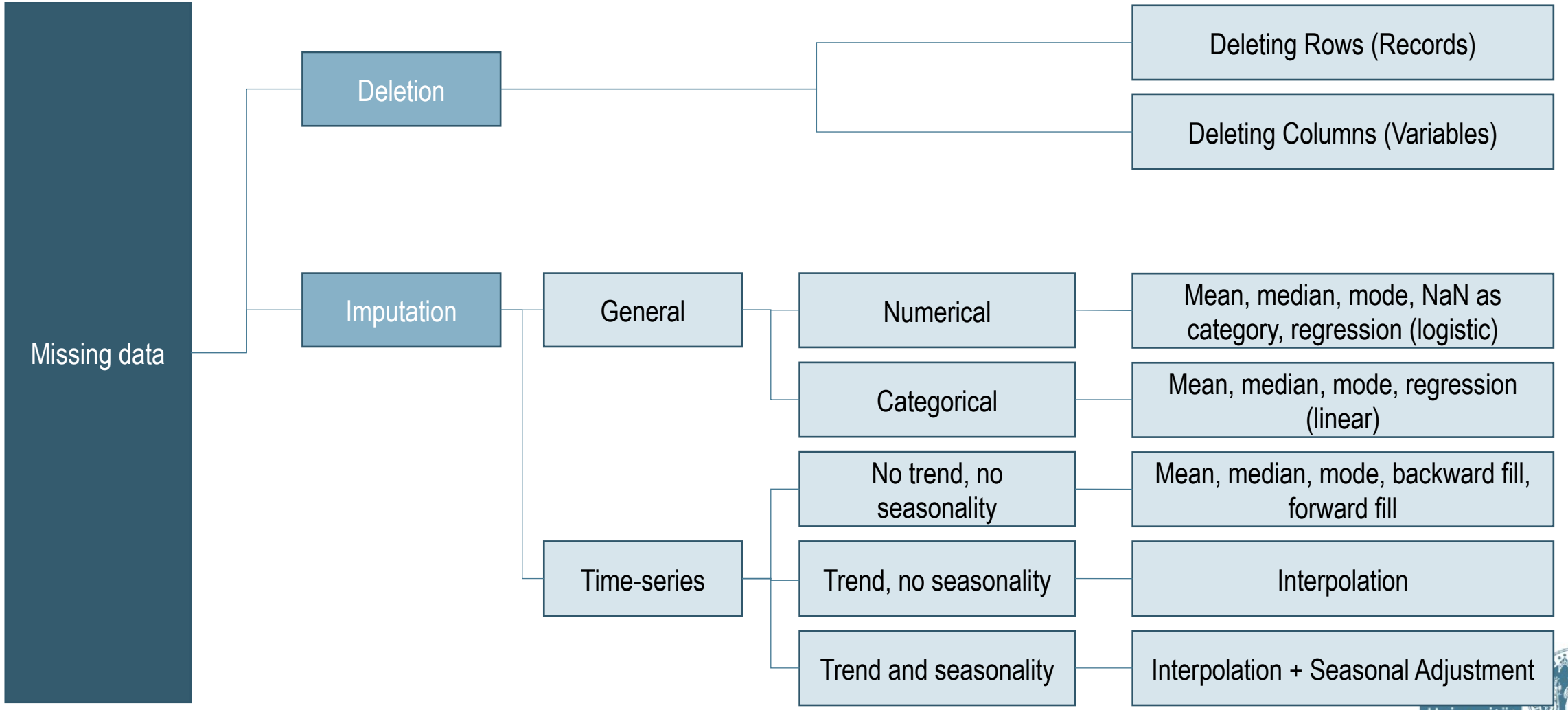
Data preparation encompasses all activities to construct and clean the data set.

- Data cleaning and preparation routines include, e.g.
  - Missing or invalid values elimination or imputation
  - Eliminating duplicate rows
  - Aligning formatting
  - Combining multiple data sources
  - Transforming and normalizing data (e.g. categorical to encoded features)
  - Engineering new features (e.g. via NLP, etc.)
- „Arguably the most time-consuming step of the entire DS process is data cleaning and preparation„
- Accelerate data preparation by automating common steps

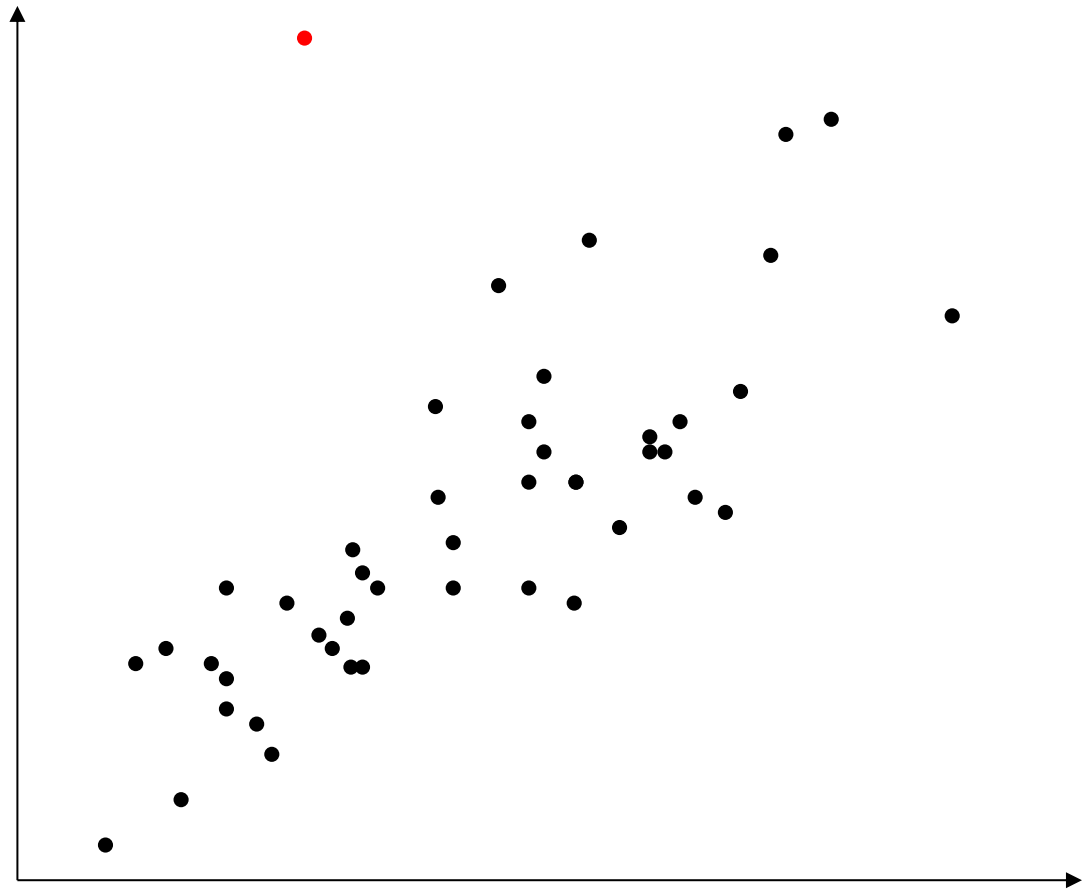
Data often comes in incomplete, erroneous or unsuitable formats which require cleaning and pre-processing

- Dealing with missing data
- Dealing with outliers
- Handling non-numerical variables
- Normalizing and re-scaling data
- Dimensionality reduction

Dealing with missing data – An overview



Outliers and how to deal with them



- Large datasets often include erroneous values resulting e.g from measurement errors
- This is often harmless, but in two cases it may not be
  - Outliers occur frequently in the data
  - Outliers exhibit significant distance from the rest of the data (e.g. misplaced decimal)
- Identify outliers via rule of thumb, e.g.
  - > 3 standard deviations
- Dealing with outliers is a **judgement call** – If the data is obviously wrong (e.g. negative value for age, etc.) we may remove it (i.e. treat it as a missing value), if the data is distant but plausible, we may decide to keep it

Normalizing and Re-scaling explained

Normalization (Standardization)

- Subtracting the mean and dividing by the standard deviation
- This is equivalent to computing a z-score

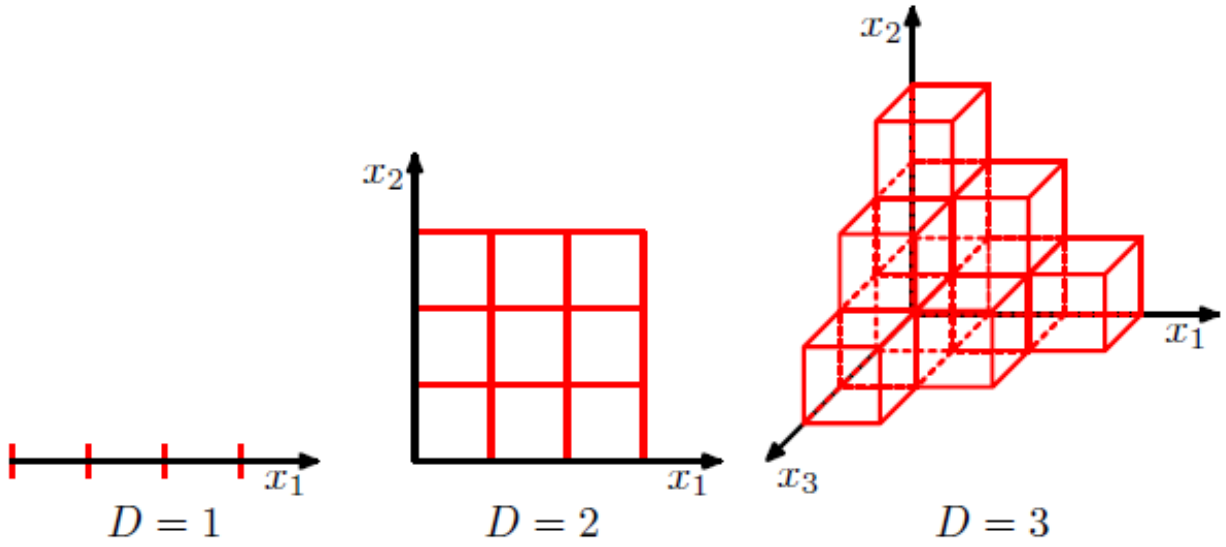
$$z = \frac{x - \bar{x}}{\sigma_x}$$

Re-Scaling

- Re-scaling each variable to [0,1] scale by subtracting the minimum and dividing by the range
- Subtracting by min(**x**) sets the origin to 0, while dividing by the range create [0,1] range

$$z = \frac{x - \min(x)}{\max(x) - \min(x)}$$

The curse of dimensionality – In high-dimensional spaces the number of cells increases exponentially with each added variable



- The number of cells increases exponentially with the dimensionality
- **Implication**: The number of observations needed increases significantly (due to density of observations in individual cells)
- The effect of a specific attribute exhibiting a wide range of values can outweigh another attribute with low value range but potentially high information value

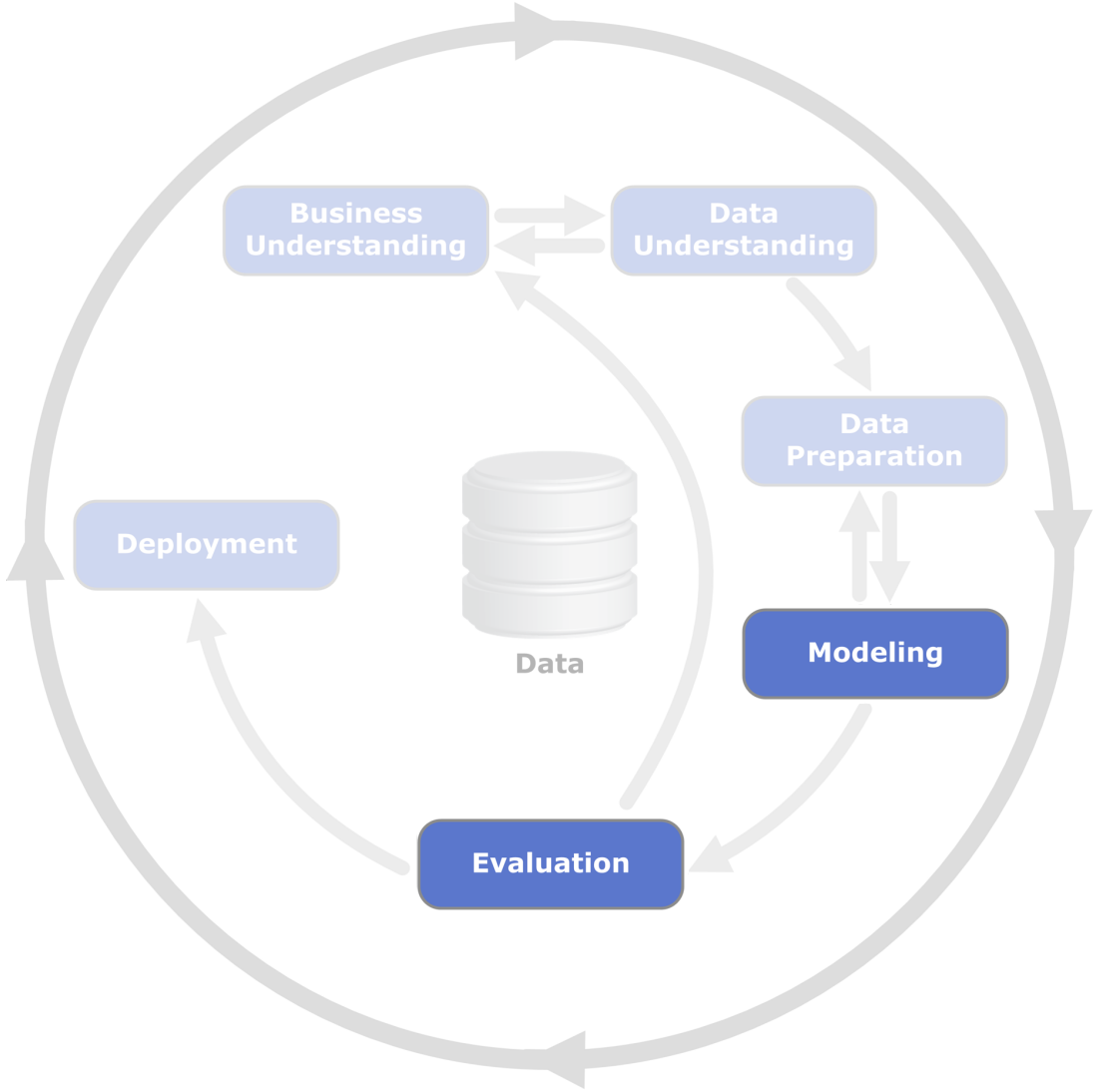
Source: Bishop (2006)

Two general approaches to reduce dimensionality of data

1. **Feature Selection:**
   - Decide which features to use based on the **variance per feature**
   - Decide which features to use based on **correlation coefficients between features**
   - **Filtering**: Pretend that only one variable exists and see how well it performs

2. **Feature Extraction:**
   - **Linear**
     - Principal Component Analysis
     - Linear Discriminant Analysis (Fisher)
     - Factor Analysis
   - **Nonlinear**
     - Kernel-based nonlinear transformations
     - Multidimensional scaling

Step 4 & 5: Modeling and Evaluation



Modeling builds on the prepared dataset

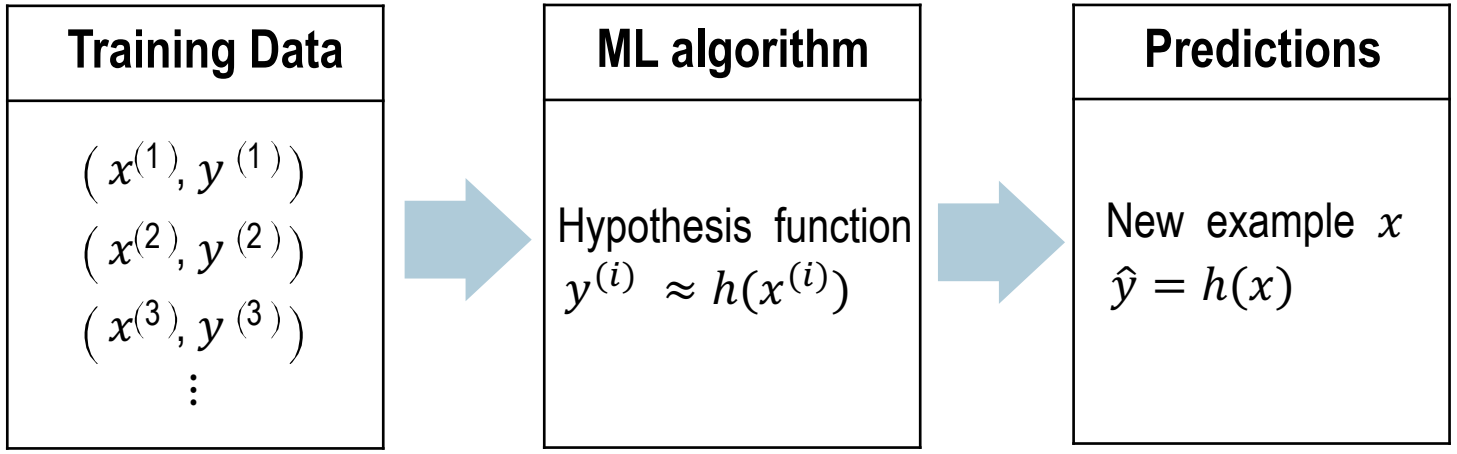
- Developing predictive or descriptive models
- Modeling is often a highly iterative process in which different features and models are tried

Model **evaluation** is performed during model development and before model deployment

- Assess the model's quality and it's performance in the real world – How reliable is it?
- Use statistical tests and common test metrics (R^2, RMSE, etc.) to compare model performance
- Ensure that the model properly addresses the business problem
- Refine model as needed

Machine learning

The basic process (supervised learning):



- This has been an example of a machine learning algorithm
  - Basic idea: in many domains, it is difficult to hand-build a predictive model, but easy to collect lots of data; machine learning provides a way to automatically infer the predictive model from data

Five key components that make up a ML model

	Term	Example
Input features	$x^{(i)} \in \mathbb{R}^n, i = 1, \dots, m$	$x^{(i)} = \begin{bmatrix} High\_Temperature^{(i)} \\ Is\_Weekday^{(i)} \\ 1 \end{bmatrix}$
Target features	$y^{(i)} \in \mathcal{Y}, i = 1, \dots, m$	$y^{(i)} \in \mathbb{R} = Peak\_Demand^{(i)}$
Model parameters	$\theta \in \mathbb{R}^n$	$\theta = (\theta_1, \theta_2)$
Hypothesis function	h_θ: $\mathbb{R}^n \rightarrow y$, predicts output given input	$h_\theta(x) = \sum_{j=1}^{n} \theta_j . x_j$
Objective function	ℓ: $\mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_+$, measures the difference between a prediction and an actual output	$\ell(\hat{y}, y) = (\hat{y} - y)^2$

At the core of each ML algorithm lies the canonical machine learning optimization problem

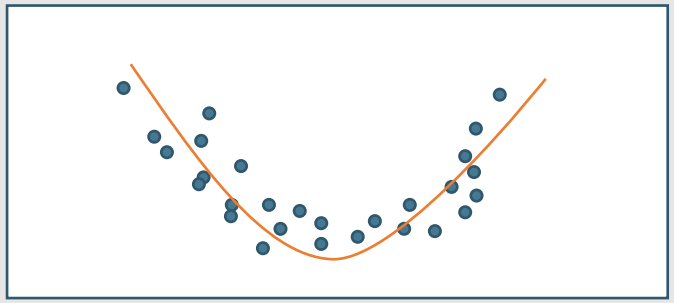
The canonical machine learning optimization problem:

$$\underset{\theta}{Minimize} \sum_{i=1}^{m} \ell(h_\theta(x^{(i)}), y^{(i)})$$

- Virtually **every machine learning algorithm** has **this form**, just specify
  1. What is the **hypothesis** function?
  2. What is the **loss (objective)** function?
  3. **How** do we **solve** the optimization problem?

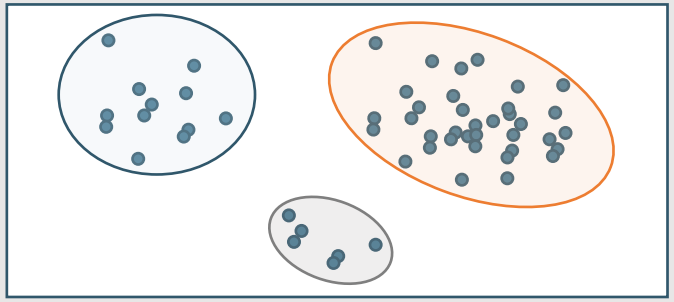
We have to differentiate between three fundamental Machine Learning concepts

Supervised Learning



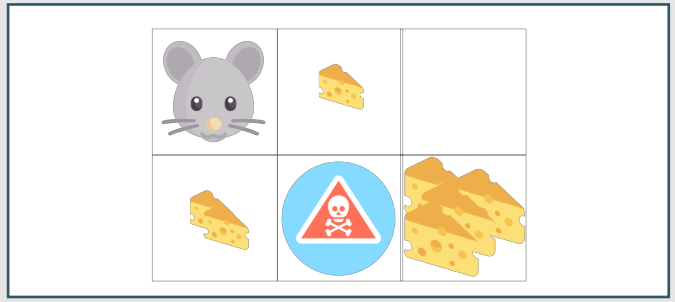
- Availability of **labeled** data
- Goal to learn a model that describes the relationship of **input features** and label
- Differentiation between **regression** (i.e. typically continuous targets) and **classification**
- Model performance **relatively easy** to evaluate

Unsupervised Learning



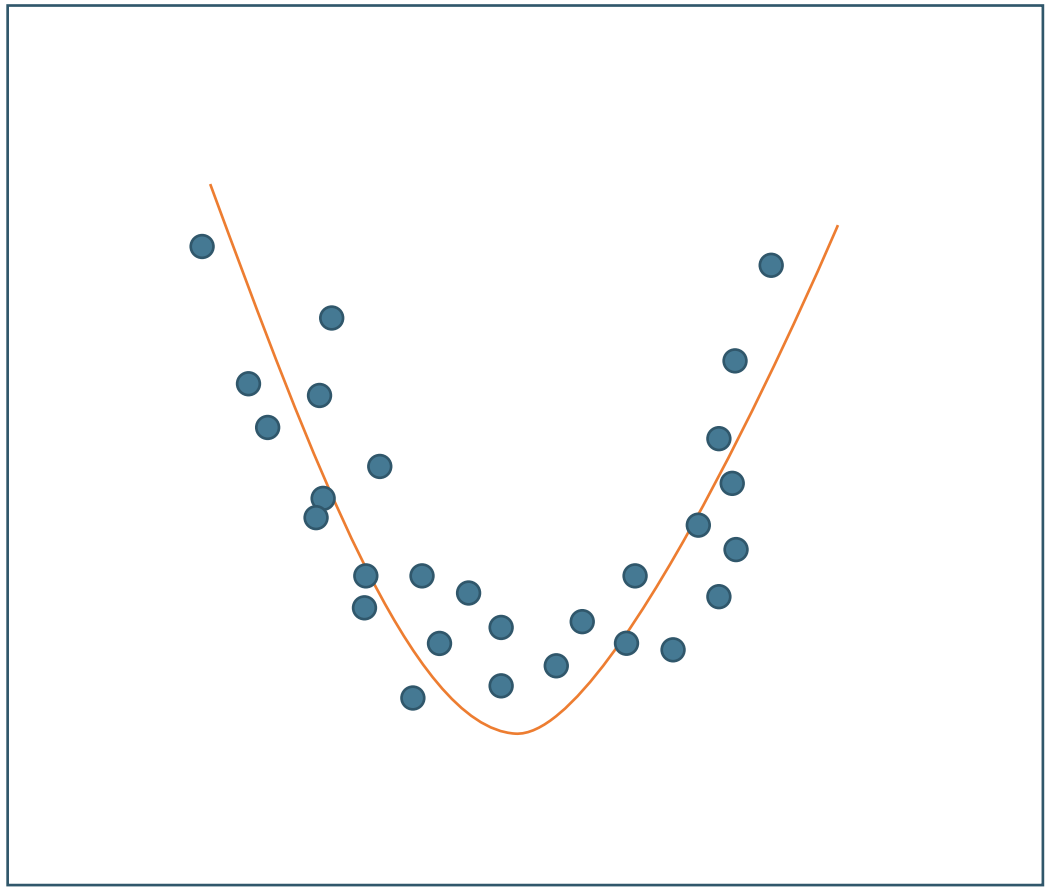
- Data **without** labels
- Goal to find certain structural **patterns** within the data
- Find **clusters** in data with similar characteristics
- Model performance **hard** to evaluate

Reinforcement Learning



- Fundamentally different from supervised and unsupervised learning
- Goal to improve **dynamic decision processes** by reacting and adapting to **reward signals**
- Closely connected to game theory
- Building block of strong **artificial intelligence**

Regression – Learning from labeled data to predict numerical outputs



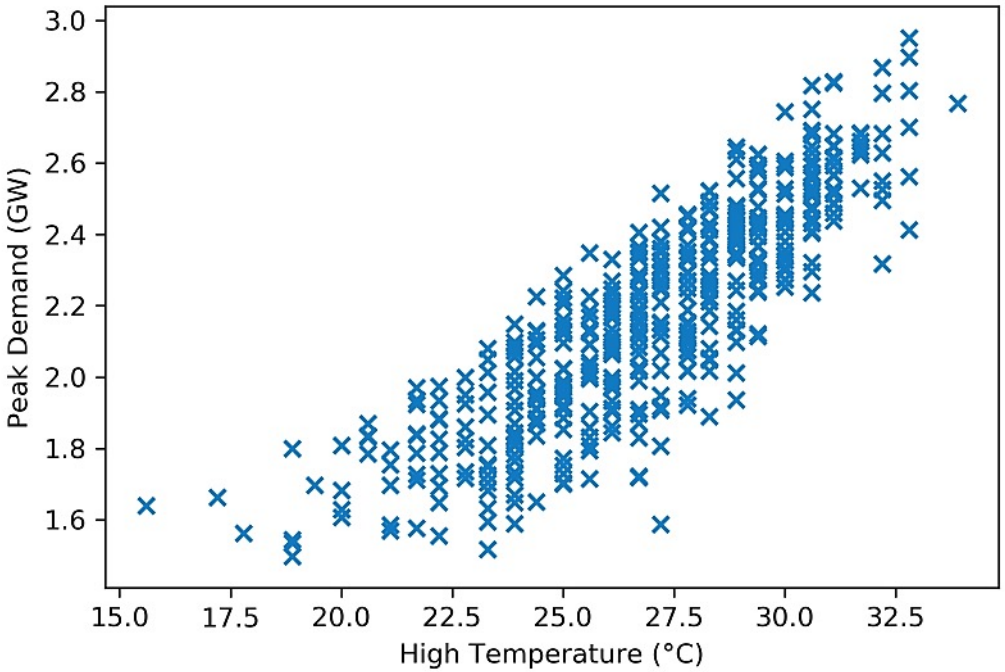
- Predict numeric predicting real-valued quantity $y \in \mathbb{R}$ (target value) based on input features
- **Linear** and **non-linear** relationship between input and target possible

We have covered Three common regression techniques in detail, but others are available – We provide a quick overview of these in the following

- **Linear Regression**
- **Polynomial Regression**
- **LASSO/Ridge Regression**
- KNN Regression
- Support Vector Regression
- Decision Tree Regression/Random forests
- Artificial Neural Networks
- …

* Having an understanding of equations for the highlighted models (in bold) is important. You do not need to memorize the equations.

Linear regression example



- Plot of high temperature vs. peak demand for summer months (June – August) for past six years

Linear regression – Hypothesis Function

$$\text{Peak\_Demand} \approx \theta_1 * \text{High\_Temperature} + \theta_2$$

- Let's suppose that the peak demand approximately fits a linear model
  - Peak_Demand ≈ θ_1*High_Temperature + θ_2
- Here θ_1 is the "slope" of the line, and θ_2 is the intercept
- How do we find a "good" fit to the data?
  - Many possibilities, but natural objective is to minimize some difference between this line and the observed data

How do we find parameters?

- How do we find the parameters θ_1 , θ_2 that minimize the objective function $E(\theta)$

$$E(\theta) = \sum_{i \in days} (\theta_1 High\_Temperature^{(i)} + \theta_2 - Peak\_Demand^{(i)})^2$$

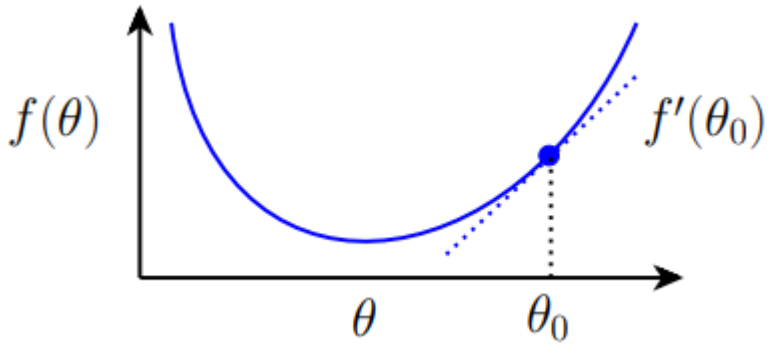
$$= \sum_{i \in days} (\theta_1 x^{(i)} + \theta_2 - y^{(i)})^2$$

How do we find parameters?

- General idea: suppose we want to minimize some function $E(\theta)$

$$E(\theta) = \sum_{i \in days} (\theta_1 x^{(i)} + \theta_2 - y^{(i)})^2$$

- Derivative is slope of the function, so **negative derivative** points “downhill”



Gradient in vector notation

- We can **simplify** the gradient computation (both notationally and computationally) substantially using matrix/vector notation

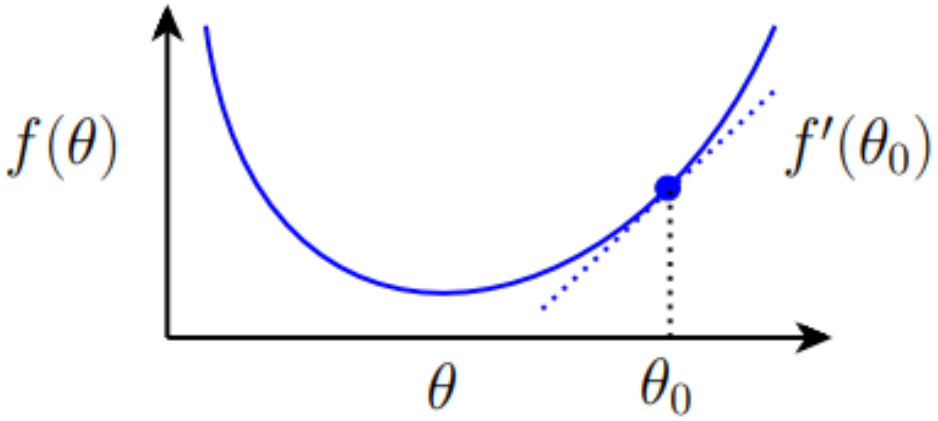
$$\frac{\partial E(\theta)}{\partial \theta_j} = \sum_{i=1}^{m} (\sum_{j=1}^{n} \theta_j x_j^{(i)} - y^{(i)})\ x_j^{(i)}$$

$$\Leftrightarrow\ \nabla_\theta E(\theta) = \sum_{i=1}^{m} x^{(i)} (x^{(i)^T} \theta\ -\ y^{(i)})$$

- Putting things in this form also makes it clearer how to analytically find the optimal solution for least squares

Solving least squares

- Gradient also gives a condition for optimality: gradient must equal zero (minimum of error function)



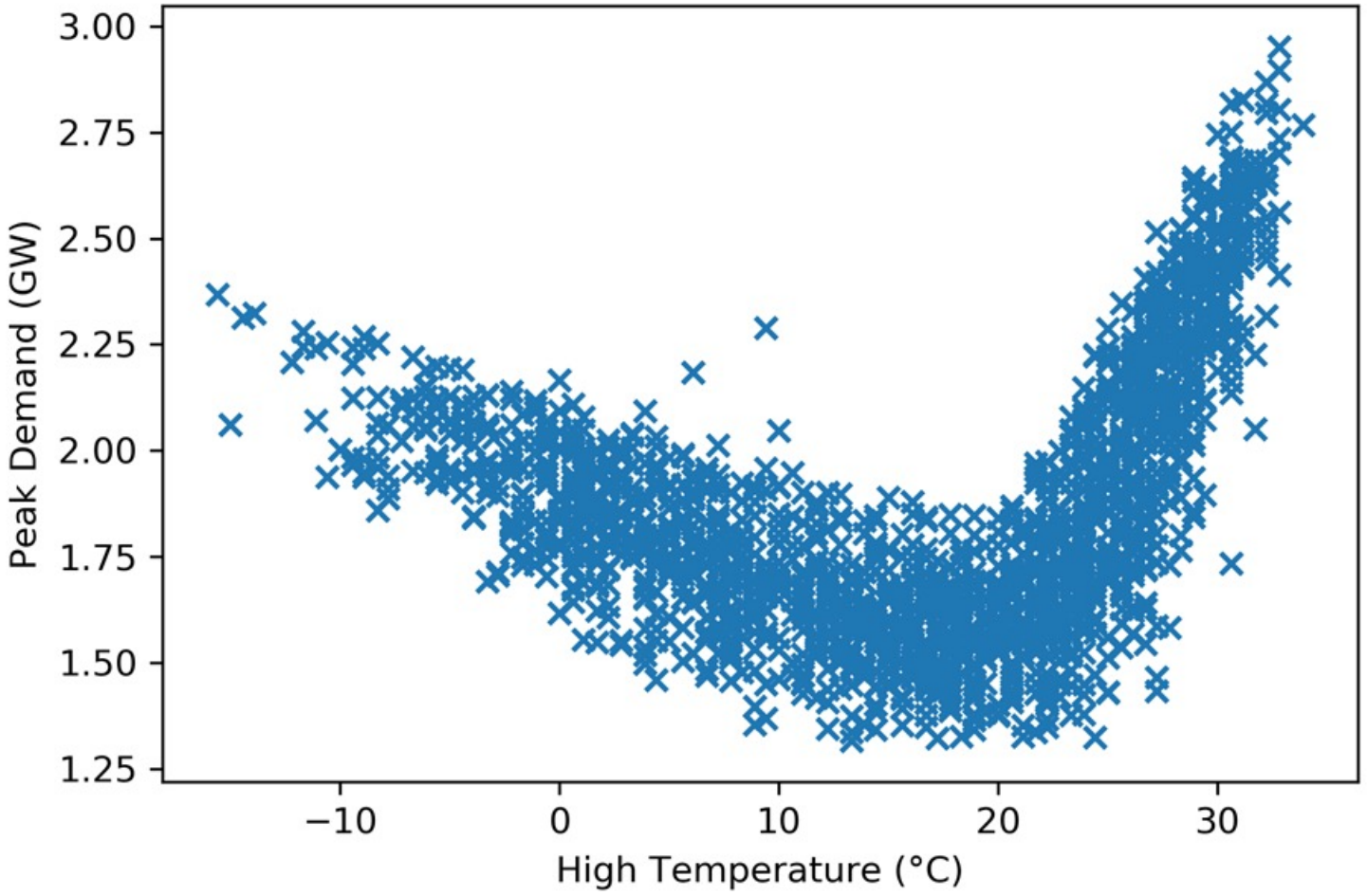
- Solving for $\nabla_\theta E(\theta) = 0$:

$$\sum_{i=1}^{m} x^{(i)} \left(x^{(i)^T} \theta - y^{(i)} \right) = 0$$

$$\Rightarrow \left(\sum_{i=1}^{m} x^{(i)} x^{(i)^T} \right) \theta - \sum_{i=1}^{m} x^{(i)} y^{(i)} = 0$$

$$\Rightarrow \theta^* = \left(\sum_{i=1}^{m} x^{(i)} x^{(i)^T} \right)^{-1} \left(\sum_{i=1}^{m} x^{(i)} y^{(i)} \right)$$

Non-Linear regression example



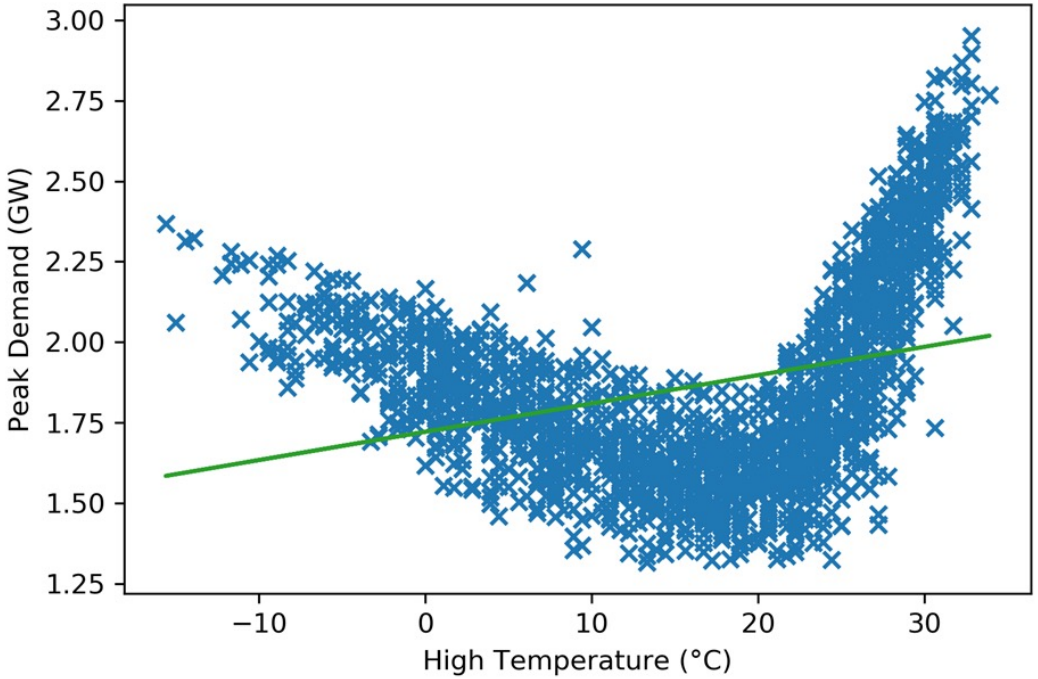
“Non-linear” regression by using non-linear features in linear models

- Thus far, we have illustrated linear regression as “drawing a line through through the data”, but this was really a function of our input features
- Though it may seem limited, linear regression algorithms are quite powerful when applied to **non-linear features** of the input data
- In our electricity consumption example , e.g.

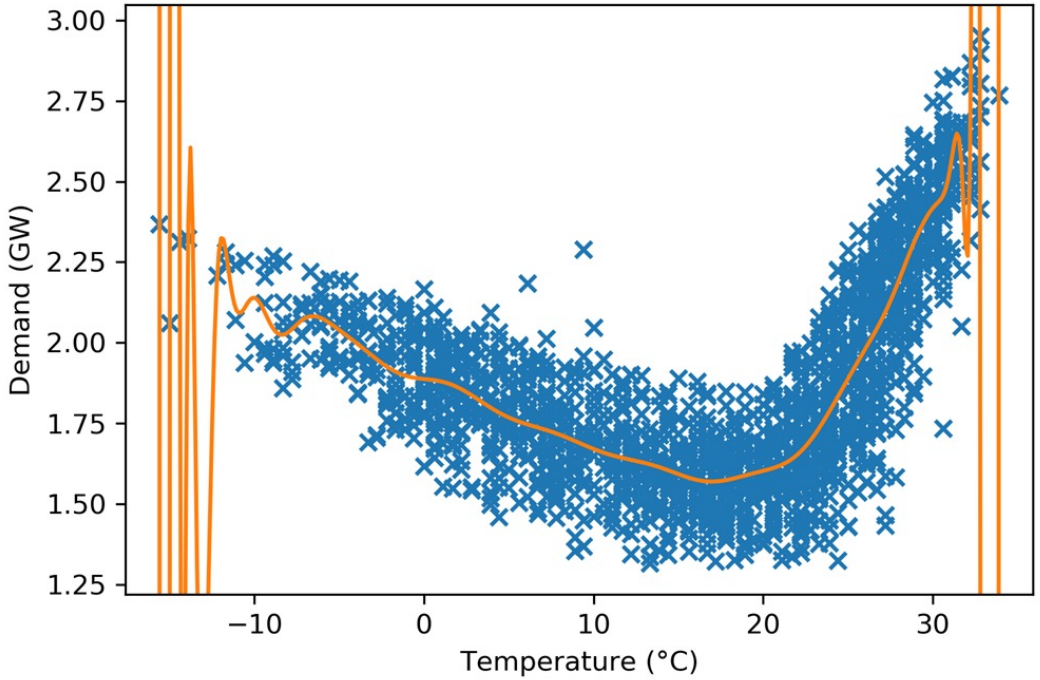
$$x^{(i)} = \begin{bmatrix} (High\_Temperature^{(i)})^2 \\ High\_Temperature^{(i)} \\ 1 \end{bmatrix}$$

There are two types of situations that we want to avoid in regression: Under- and Overfitting

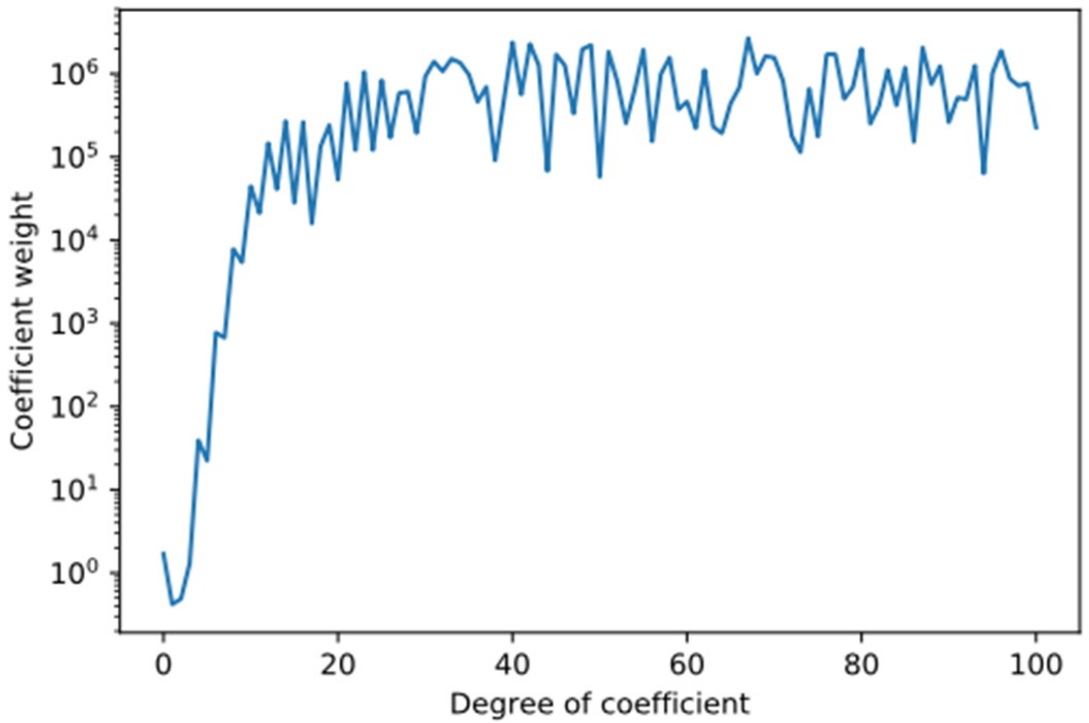
Underfitting



Overfitting



High model complexity comes with high coefficient weights – Controlling the size of parameters (regularization) as way to control model complexity



- But fitting these models also requires extremely large coefficients on these polynomials
- For 50-degree polynomial, the first few coefficients are:

  $\theta = 2.27\times10^4, 1.61\times10^5, 6.88\times10^4, -1.36\times10^5, \ldots$
- This suggests an **alternative way to control model complexity**: **keep the weights small (regularization)**, i.e. control the magnitude of the model parameters!

How do we control the size of the parameters? – We add a term in our objective (loss) function

L2 regularization (ridge regression)

$$\underset{\theta}{Minimize}\ \frac{1}{m}\sum_{i=1}^{m}\ell\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right) + \lambda\sum_{i=1}^{n}\theta^2$$

Note: The above regularization is referred to as L2 regularization (ridge regression), as we work with a squared regularization term

- This formulation trades off loss on the training set with a penalty on high values of the parameters (λ = **regularization parameter**).
- By varying λ from zero (**no regularization**) to infinity (**infinite regularization**, meaning parameters will all be zero), we can sweep out different sets of model complexity

Another option to control model complexity is L1 regularization (LASSO regression)

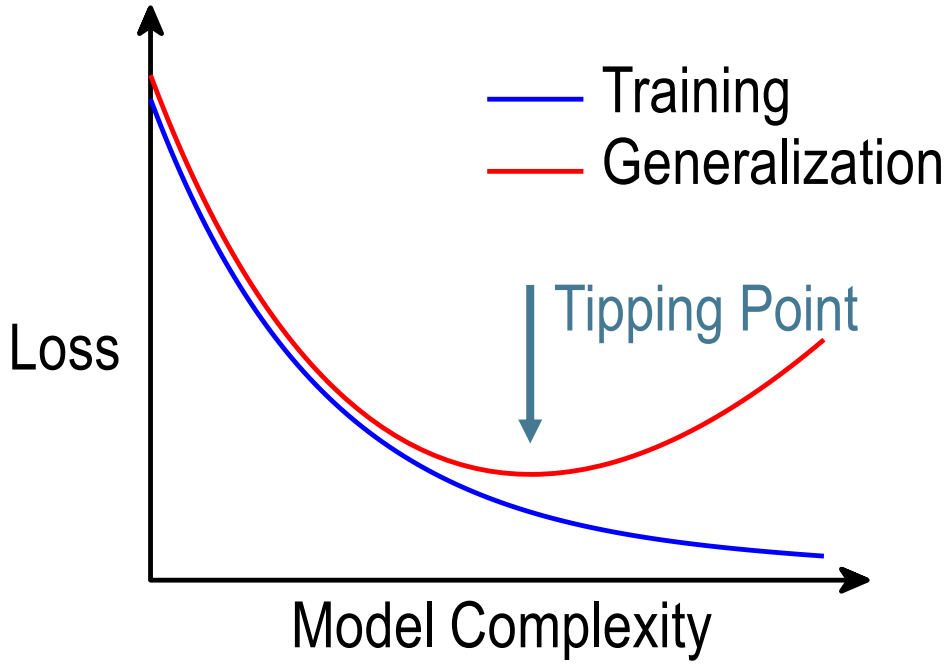
L1 regularization (LASSO regression)

$$\underset{\theta}{Minimize}\ \frac{1}{m}\sum_{i=1}^{m}\ell\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right) + \lambda\sum_{i=1}^{n}|\theta|$$

Note: The above regularization is referred to as L1 regularization (LASSO regression), as we work with an unsquared regularization term

- This formulation trades off loss on the training set with a penalty on absolute values of the parameters (λ = **regularization parameter**).
- This type of regularization (L1) **can lead to zero coefficients** of particular features so LASSO also **helps in feature selection**

Generalized version of under- and overfitting



- As a model becomes more complex (by adding more features, etc.), training loss always decreases; generalization loss decreases to a point, then starts to increase.

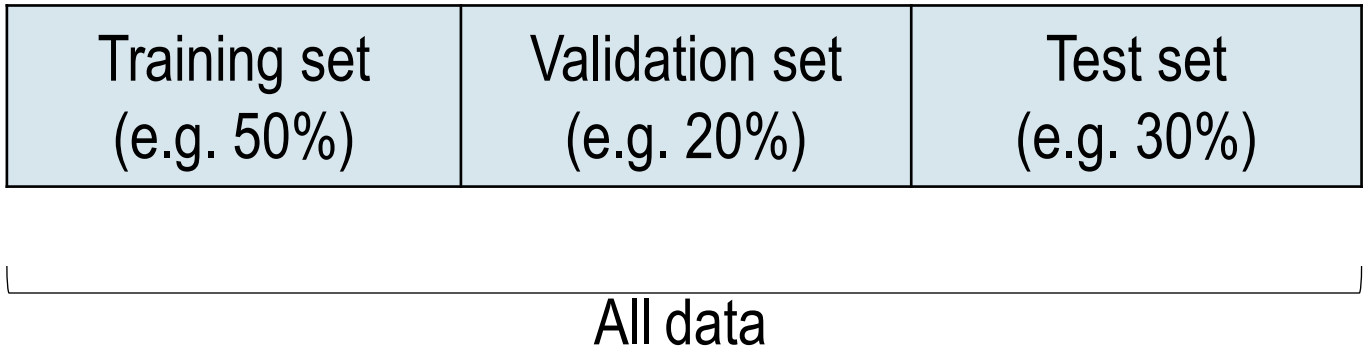
How to measure loss? – Absolute Regression Test Metrics

	Name	Term	Description
MSE/ MSD	Mean-Squared Error/Deviation	$\frac{1}{n}\sum_{i=1}^{n} e_i^2$	Average squared difference between the estimated values and what is estimated – Puts large emphasis on large deviations
RMSE	Root-Mean-Squared Error	$\sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2}$	Square root of average squared difference between the estimated values and what is estimated (i.e. square root of MSE) – Sets MSE to same units as dependent var.
MAE/ MAD	Mean Absolute Error/ Deviation	$\frac{1}{n}\sum_{i=1}^{n} \lvert e_i \rvert$	Average absolute error – Provides an indication of the absolute deviation (positive or negative) of the responses in the same units as the dependent var.
Average error	Average Error	$\frac{1}{n}\sum_{i=1}^{n} e_i$	Indicates whether the predictions are on average over- or underpredicting the target response

How to measure loss? – Relative Regression Test Metrics

	Name	Term	Description
R^2	R-Squared	$1 - \frac{Sum\ of\ Squares_{res}}{Sum\ of\ Squares_{total}}$	Proportion of variance in the dependent variable that is accounted for by the model
MAPE	Mean absolute percentage error	$100\% \times \frac{1}{n}\sum_{i=1}^{n} \lvert e_i / y_i \rvert$	Gives a percentage score of how predictions deviate on average from the actual values
nRMSE	Normalized RMSE	$\frac{RMSE}{\bar{y}}$	RMSE normalized by the dependent variable to give a ratio of the RMSE compared to the mean of the target value (not very commonly used)
nMAE	Normalized MAE	$\frac{MAE}{\bar{y}}$	MAE normalized by the dependent variable to give a ratio of the MAE compared to the mean of the target value (not very commonly used)

To obtain good test results partition your data into training, validation and test set prior to learning a ML model



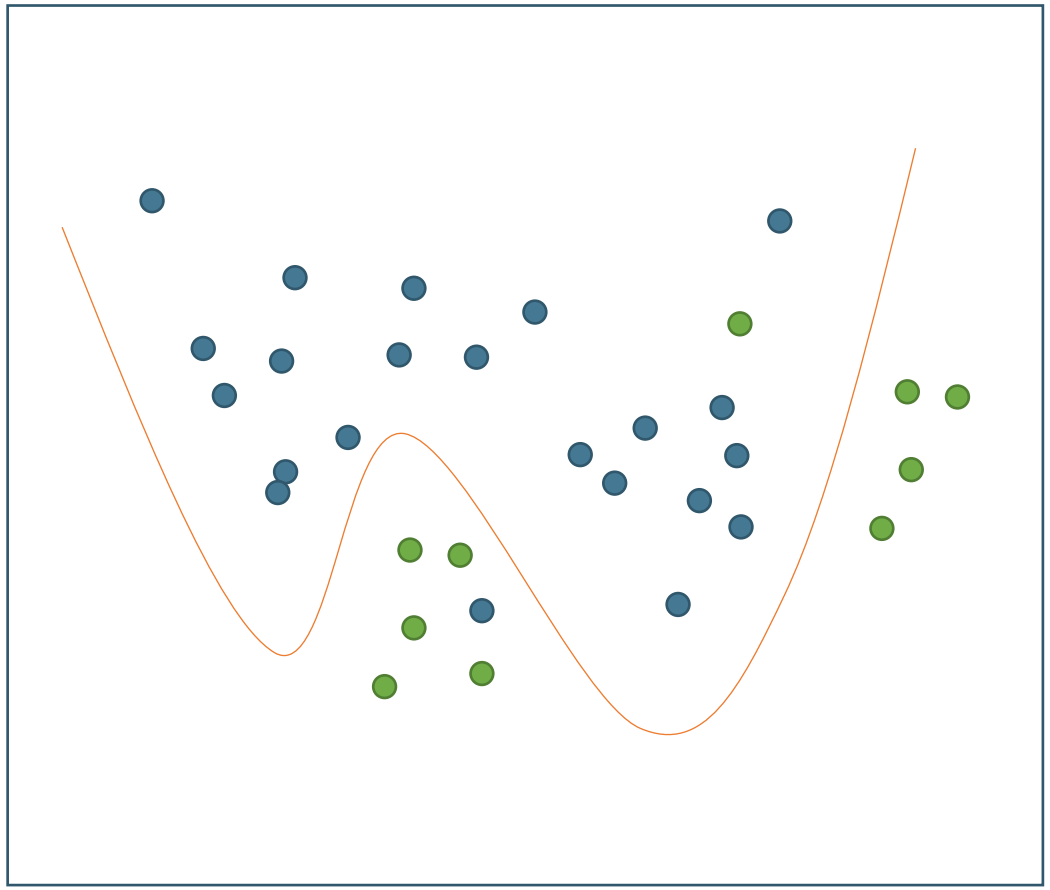
1. Divide data into training set, holdout set, and test set
2. Train algorithm on training set (i.e., to learn parameters), use holdout set to select hyperparameters
3. (Optional) retrain system on training + holdout
4. Evaluate performance on test set

Source: Bishop (2006)

In practice…

- "Leakage" of test set performance into algorithm design decisions in almost always a reality when dealing with any fixed data set (in theory, as soon as you look at test set performance once, you have corrupted that data as a valid set set)
- This is true in research as well as in data science practice
- **The best solutions:** evaluate your system "in the wild" (where it will see truly novel examples) as often a possible; recollect data if you suspect overfitting to test set; look at test set performance sparingly
- An interesting and very active area of research: adaptive data analysis (differential privacy to theoretically guarantee no overfitting)

Classification – Learning from labeled data to predict categorical outputs



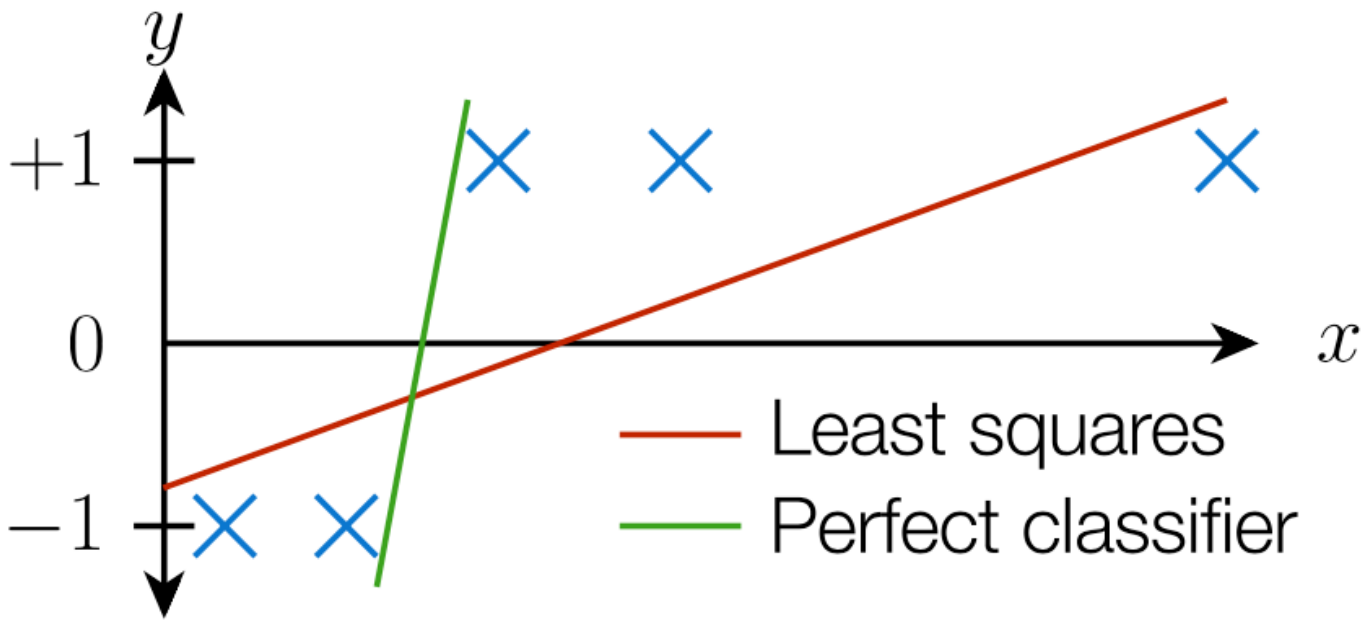
- Predict discrete-valued quantity y
  - **Binary classification:** $y \in \{-1, +1\}$
  - **Multi-class classification**: $y \in \{1,2, \dots, k\}$

We have covered selected classification techniques in detail but others are available – We provide a quick overview of these in the following

- **Linear Classification**
- **SVM Classification**
- **Logistic Regression**
- **Naïve Bayes' Classifier**
- **Classification Tree**
- Nonlinear Classification
- Ensemble Models
- ANN Classification

*Having an understanding of equations for the highlighted models (in bold) is important. You do not need to memorize the equations.

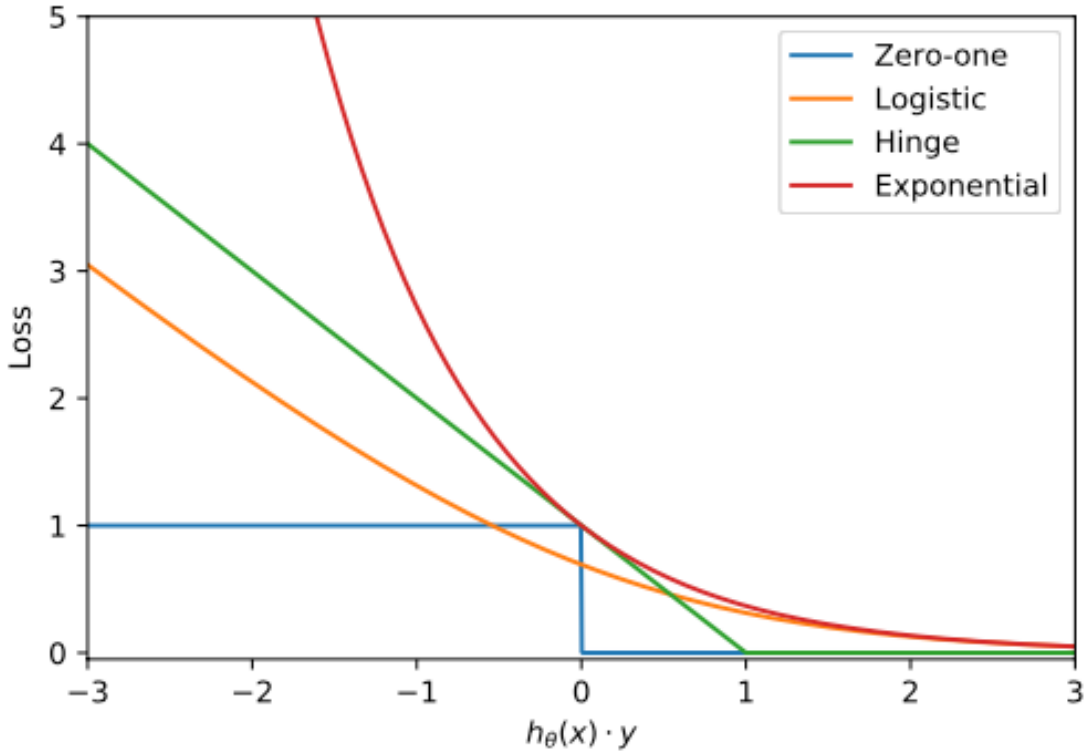
Loss functions for classification



- It is of course completely possible to classify the data perfectly with a linear classifier
- Least-squares loss function applied to classification aims at predicting exactly +1 or −1 on each data point
- Numbers much larger than one for a positive example will add to the loss

Alternative classification losses

Illustration of different loss functions



- Logistic loss: $l_{logistic} = \log(1 + \exp(-y\, h_\theta(x)))$
  - For large positive values of $h_\theta(x)\cdot y$ loss will be very close to zero; for large negative values the loss increases c. linearly
- Hinge loss: $l_{hinge} = \max\{1 - y\, h_\theta(x), 0\}$
  - As long as $h_\theta(x)\cdot y \geq 1$, this loss will be zero, whereas it will increase linearly for negative $h_\theta(x)\cdot y \geq 1x$.
- Exponential loss: $l_{exp} = \exp(-y\, h_\theta(x))$
  - This loss will go to zero for large $h_\theta(x)\cdot y$; but for negative $h_\theta(x)\cdot y$ the loss increases very quickly

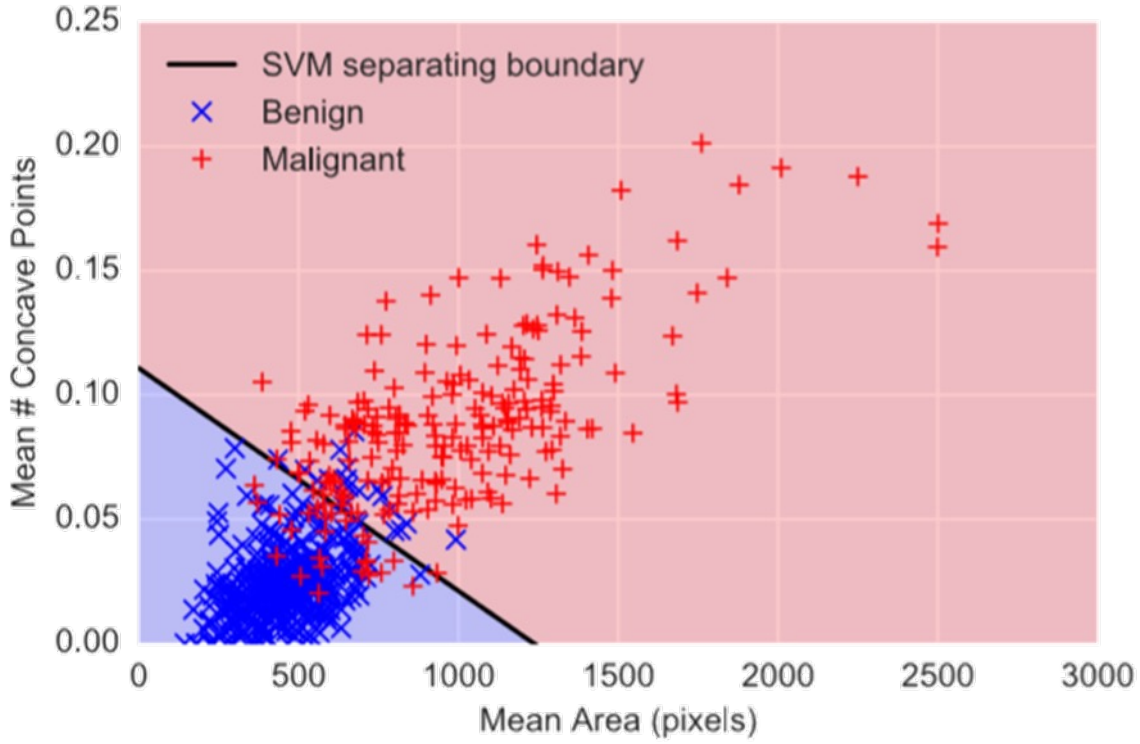
Solving classification tasks: Machine learning optimization

- With this notation, the “canonical” machine learning problem is written in the exact same way for classification
  - $\underset{\theta}{Minimize} \sum_{i=1}^{m} \ell\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right)$
- Unlike least squares, there is not an analytical solution to the zero gradient condition for most classification losses
- Instead, we solve these optimization problems using gradient descent (or a alternative optimization method, but we’ll only consider gradient descent here)
  - $Repeat\colon\ \theta := \theta - \alpha \sum_{i=1}^{m} \nabla_\theta \ell\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right)$

Support vector machine (SVM) – Linear hypothesis function with hinge loss

- A (linear) support vector machine (SVM) just solves the canonical machine learning optimization problem using **hinge loss** and **linear hypothesis**, **plus** an additional **regularization** term
  - $\underset{\theta}{Minimize} \sum_{i=1}^{m} max\left\{1 - y^{(i)}\theta^T x^{(i)}, 0\right\} + \frac{\lambda}{2} \|\theta\|_2^2$
- Even more precisely, the “standard” SVM does not actually regularize the "θ_i" (corresponding to the constant feature, but we’ll ignore this here)
- Updates using gradient descent:
  - $\theta := \theta - \alpha \sum_{i=1}^{m} -y^{(i)} x^{(i)} 1\left\{y^{(i)}\theta^T x^{(i)} \leq 1\right\} - \alpha\lambda\theta$

Support vector machine example

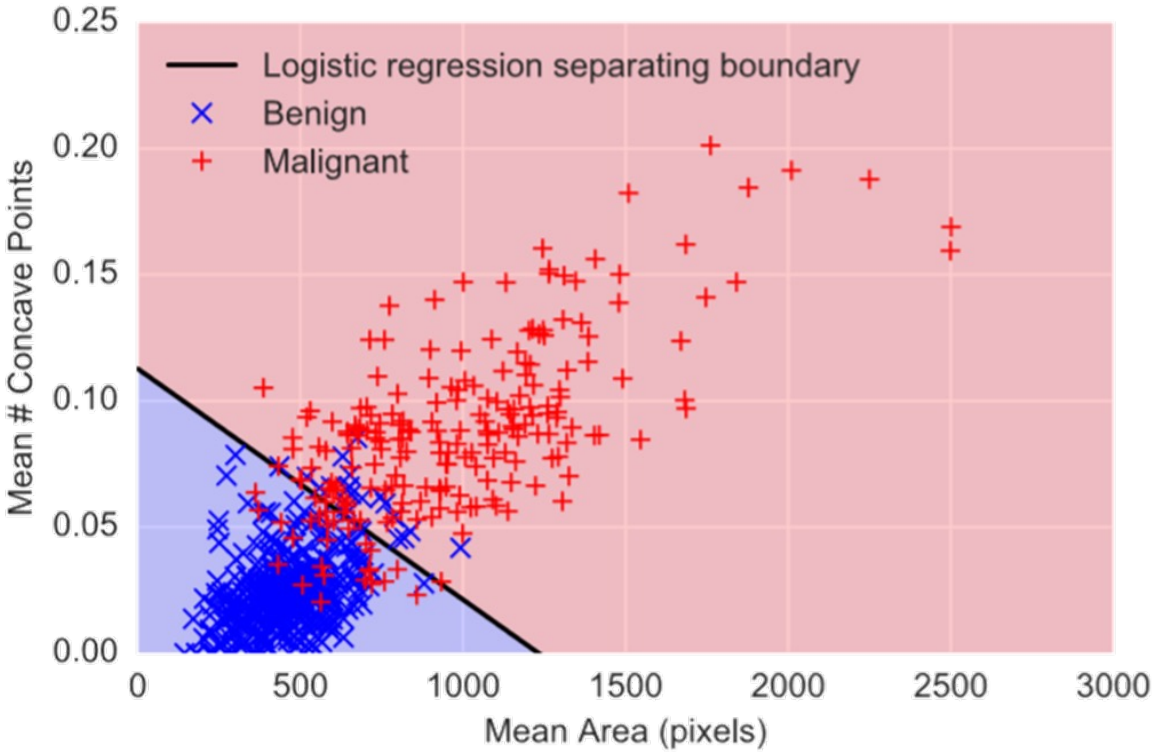


- Running support vector machine on cancer dataset, with small regularization parameter (effectively zero)
  - $\theta = \begin{bmatrix} 1.456 \\ 1.848 \\ -0.189 \end{bmatrix}$

Logistic regression – Linear hypothesis function and logistic loss

- Logistic regression just solves this problem using logistic loss and linear hypothesis function
  - $\underset{\theta}{Minimize} \sum_{i=1}^{m} \log(1 + \exp(-y^{(i)}\theta^T x^{(i)}))$
- Gradient descent updates (can you derive these?):
  - $\theta := \theta - \alpha \sum_{i=1}^{m} -y^{(i)}x^{(i)} \frac{1}{1+\exp(y^{(i)}\theta^T x^{(i)})}$
- Logistic regression also has a nice probabilistic interpretation: certain quantities give the *probability*, under a particular model, of an example being positive or negative
- We will consider this probabilistic setting more in the next lecture
- For now we are going to simply treat it as another loss minimization algorithm

Logistic regression example



- Running logistic regression on cancer data set, small regularization

Summary

- Logistic regression is similar to linear regression, except that it is used with a categorical response
- It can be used for explanatory tasks (=profiling) or predictive tasks (=classification)
- The predictors are related to the response Y via a nonlinear function called the **logit**
- As in linear regression, reducing predictors can be done via variable selection
- Logistic regression can be generalized to more than two classes

Bayes' rule

- A straightforward manipulation of probabilities:
  - $p(X_1|X_2) = \frac{p(X_1, X_2)}{\mathrm{p}(X_2)}$

$$= \frac{p(X_2|X_1)p(X_1)}{\mathrm{p}(X_2)}$$

$$= \frac{p(X_2|X_1)p(X_1)}{\sum_{x_1} p(X_2|x_1)p(x_1)}$$

Intuition

- Relate the actual probability to the measured test probability
- The quantities $p(X_2|X_1)$, $p(X_1)$, and $\mathrm{p}(X_2)$ may be easier to determine than the quantity of ultimate interest: $p(X_1|X_2)$
- Combined with the assumption of independence the Bayes' rule of conditional probability serves as basis for the Naïve Bayes Classifier

Naive Bayes assumptions

- We're going to find $p(Y|X)$ via Bayes' rule
  - $p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)} = \frac{p(X|Y)p(Y)}{\Sigma_y p(X|y)p(y)}$
- The devisor is just the sum over all values of Y of the distribution specified by the numeration, so we're just going to focus on the $p(X|Y)p(Y)$ term
- Modelling full distribution $p(X|Y)$for high-dimensional X is not practical, so we're going to make the **naive Bayes assumption**, that the elements x_i are conditionally independent given Y
  - $p(X|Y) = \prod_{i=1}^{n} p(x_i|Y)$

Evaluating classification models – Common classification error metrics derived from the confusion matrix

- Several common metrics are associated with entries of the confusion matrix (TP = true positive, FP = false positive, TN = true negative, FN = false negative)
  - TP Rate (also called Recall) = $\frac{TP}{TP+FN}$
  - FP Rate = $\frac{FP}{FP+TN}$
  - Precision = $\frac{TP}{TP+FP}$
  - Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$
- Different metrics can be standard for different domains

Confusion matrix

	Predicted Positive	Predicted Negative
Actual Positive	True Positive	False Negative
Actual Negative	False Positive	True Negative

Classification Trees

- Classification and Regression Trees are an easily understandable and transparent method for predicting or classifying new records
- A tree is a graphical representation of a set of rules
- Trees must be pruned to avoid over-fitting of the training data
- As trees do not make any assumptions about the data structure, they usually require large samples

Splits and Impurity

- Splits should be chosen such that impurity is reduced the most
- Impurity can be measured by the **Gini Index** as well as **Entropy**
- At each successive stage, compare two measures just mentioned across all possible splits in all variables

- Gini Index for rectangle *A* containing *m* records

$$I(A) = 1 - \sum_{k=1}^{m} p_k^2$$

p = proportion of cases in rectangle *A* that belong to class *k*

- Entropy can be measured using the following formula

$$entropy\,(A) = -\sum_{k=1}^{m} p_k \, log_2(p_k)$$

p = proportion of cases (out of *m*) in rectangle *A* that belong to class *k*

Advantages and Disadvantages of Trees

Advantages

- Easy to use, understand
- Produce rules that are easy to interpret & implement
- Variable selection & reduction is automatic
- Do not require the assumptions of statistical models
- Can work without extensive handling of missing data

Disadvantages

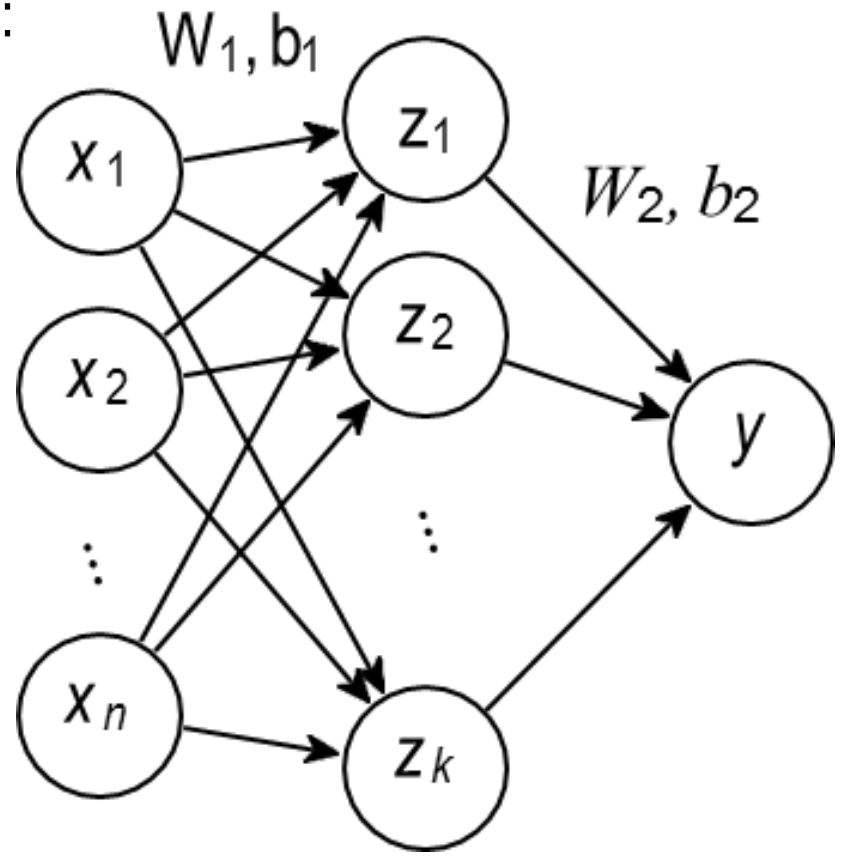
- May not perform well where there is structure in the data that is not well captured by horizontal or vertical splits
- Since the process deals with one variable at a time, no way to capture interactions between variables

Neural networks for machine learning

- The term “neural network” largely refers to the hypothesis class part of a machine learning algorithm:
  1. **Hypothesis:** non-linear hypothesis function, which involve compositions of multiple linear operators (e.g. matrix multiplications) and elementwise non-linear functions
  2. **Loss:** “Typical” loss functions for classification and regression: logistic, softmax (multiclass logistic), hinge, squared error, absolute error
  3. **Optimization:** Gradient descent, or more specifically, a variant called stochastic gradient descent
- ANN can be used both for regression and classification tasks!

Illustrating neural networks

- We draw neural networks using the same graphic as before (the non- linear function are always in implied in the neural network setting):



- Middle layer z is referred to as the hidden layer or activations
- These are the learned features, nothing in the data prescribed what values they should take, left up to algorithm to decide

Properties of neural networks

- A neural network with a single hidden layer (and enough hidden units) is a universal function approximator and can approximate any function over inputs
- In practice, not that relevant (similar to how polynomials can fit any function), and the more important aspect is that they appear to work very well in practice for many domains
- The hypothesis $h_\theta(x)$ is not a convex function of parameters $\theta = \{W_i, b_i\}$ so we have possibility of local optima
- Architectural choices (how many layers, how they are connected, etc.), become important algorithmic design choices (i.e. hyperparameters)

Ensemble Methods

Ensemble methods use multiple algorithms to obtain better predictive performance than could be obtained from any of the algorithms by itself

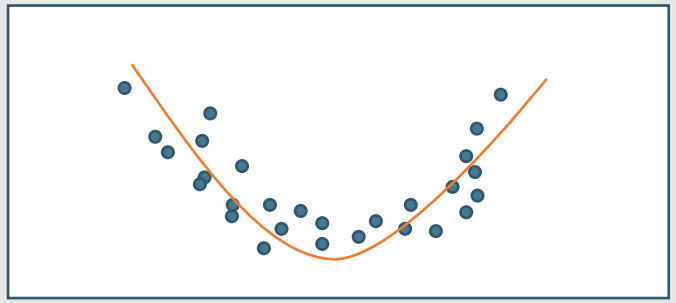
Using **multiple algorithms** usually increases model performance by:

- reducing variance: models are less dependent on the specific training data

- **Bagging** (or bootstrap aggregation) creates multiple data sets from the original training data by bootstrapping – **re-sample with repetition**. Runs several models and aggregates output with a voting system
- **Random Forest:** combines bagging with random selection of features (or predictors)
- **Boosting:** applies classifiers sequentially, assigning higher weights to observations that have been mis-classified by the previous methods

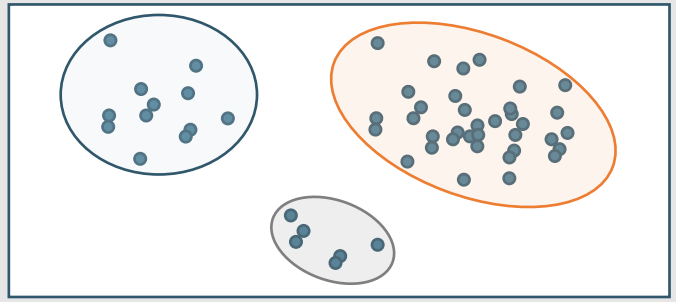
We have to differentiate between three fundamental Machine Learning concepts

Supervised Learning



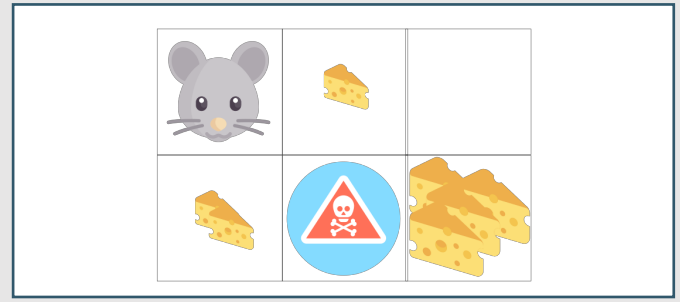
- Availability of **labeled** data
- Goal to learn a model that describes the relationship of **input features** and label
- Differentiation between **regression** (i.e. typically continuous targets) and **classification**
- Model performance **relatively easy** to evaluate

Unsupervised Learning



- Data **without** labels
- Goal to find certain structural **patterns** within the data
- Find **clusters** in data with similar characteristics
- Model performance **hard** to evaluate

Reinforcement Learning

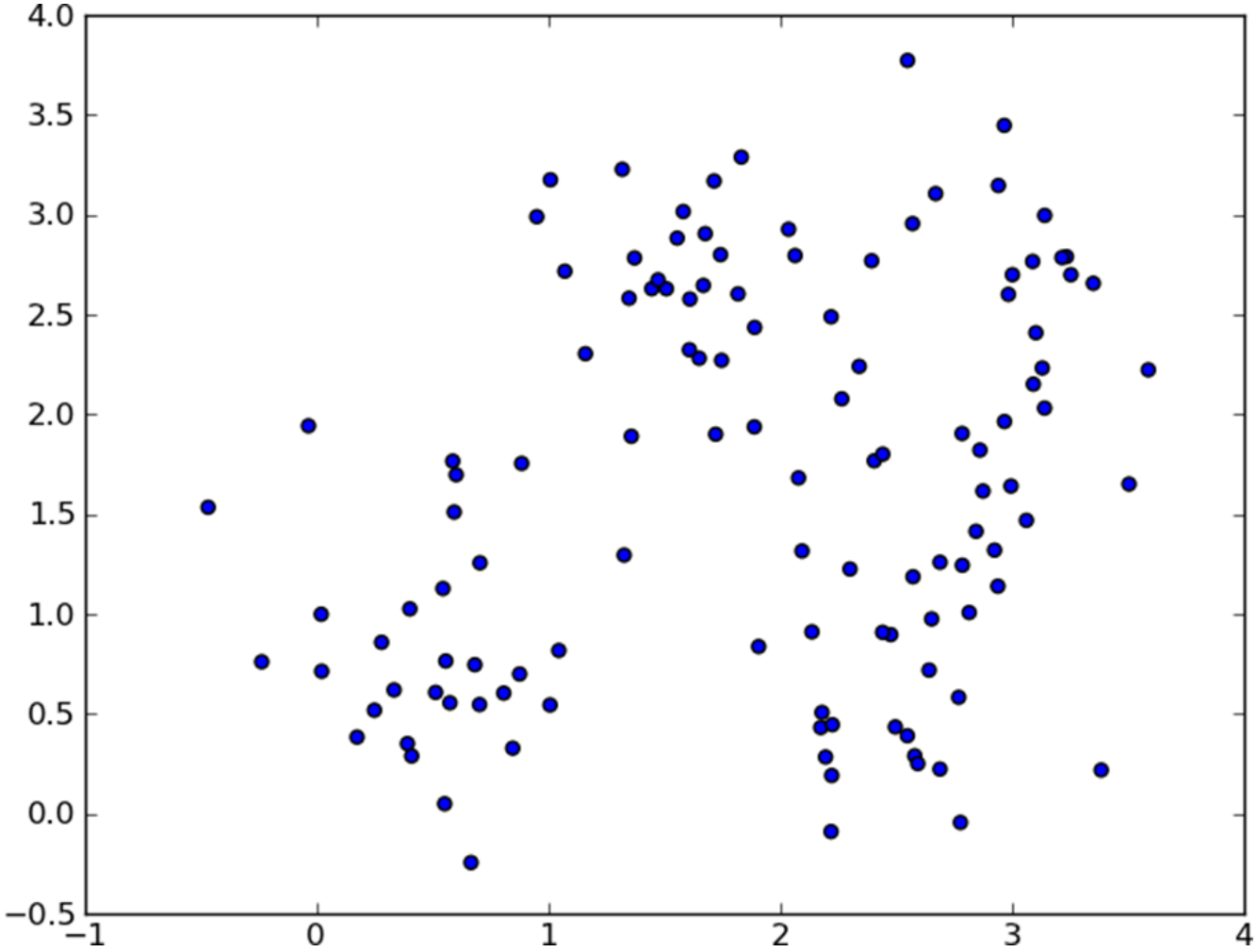


- Fundamentally different from supervised and unsupervised learning
- Goal to improve **dynamic decision processes** by reacting and adapting to **reward signals**
- Closely connected to game theory
- Building block of strong **artificial intelligence**

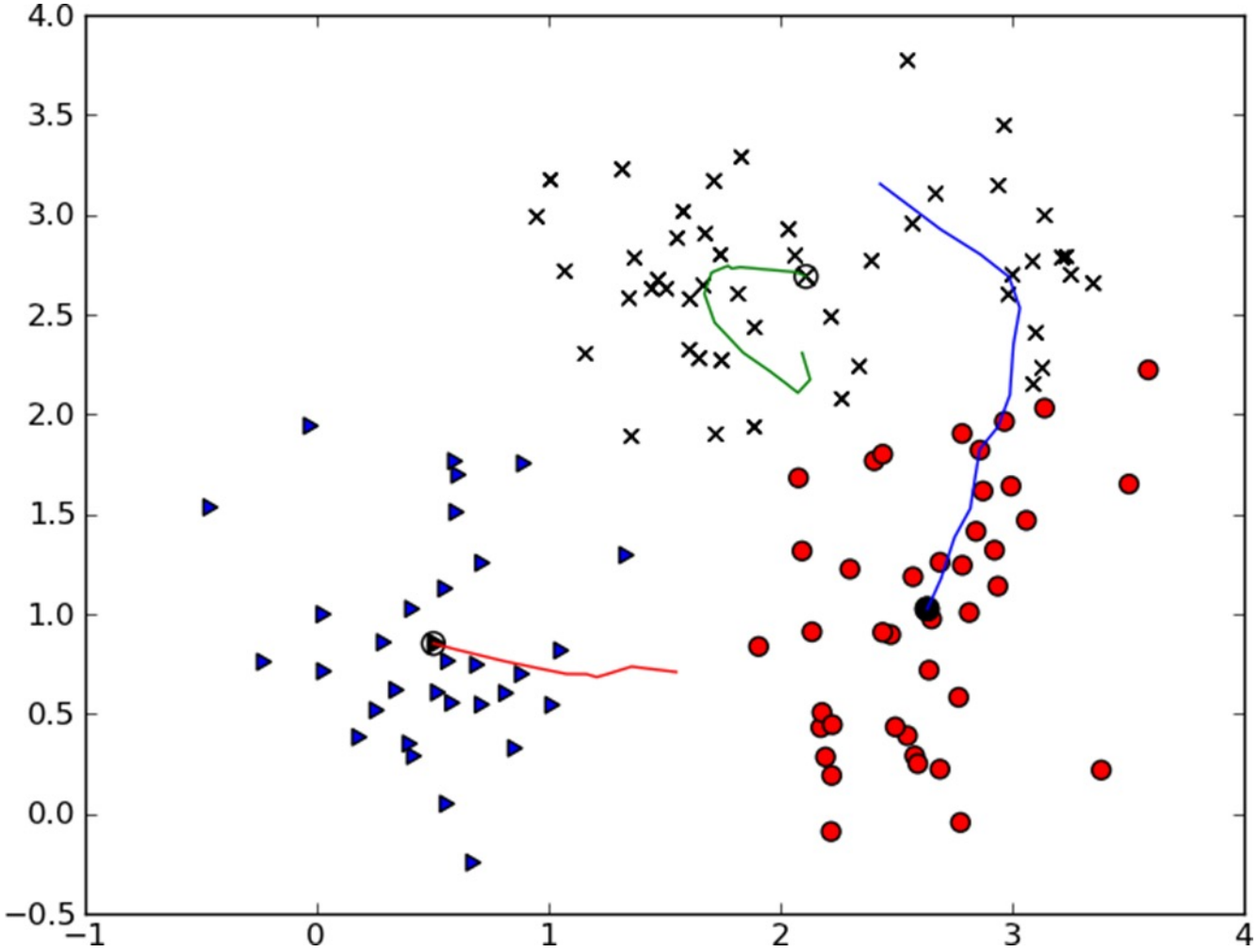
Two Types of Hard Clustering

- **Partitioning algorithms:** Construct various partitions and then evaluate them by some criterion.
  - k-means
  - k-means ++
- **Hierarchical algorithms:** Create a hierarchical decomposition of the set of objects using some criterion.
  - Hierarchical clustering

Example: k-means Clustering (1/2)



Example: k-means Clustering (2/2)

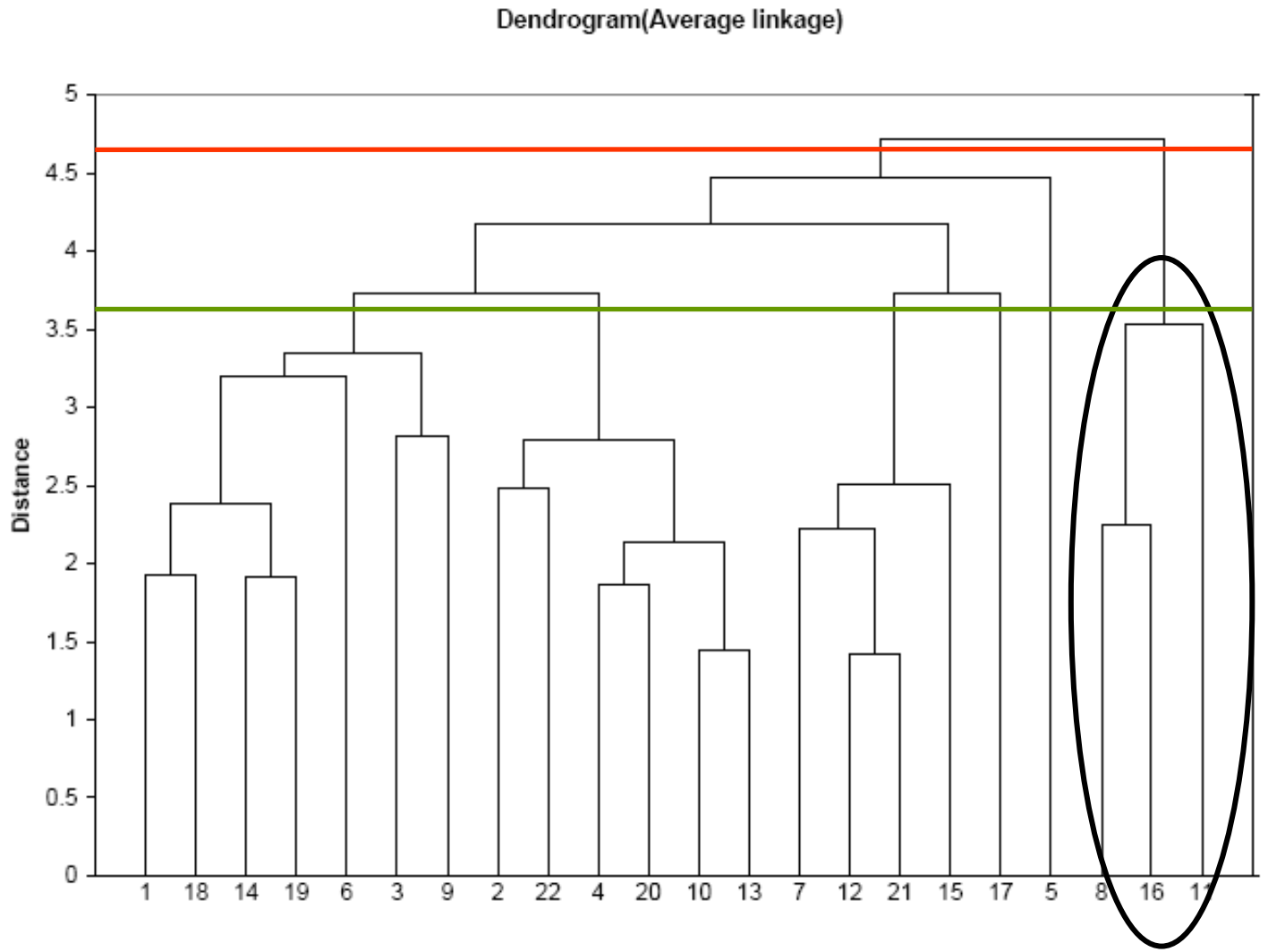


K-means++

Intuition

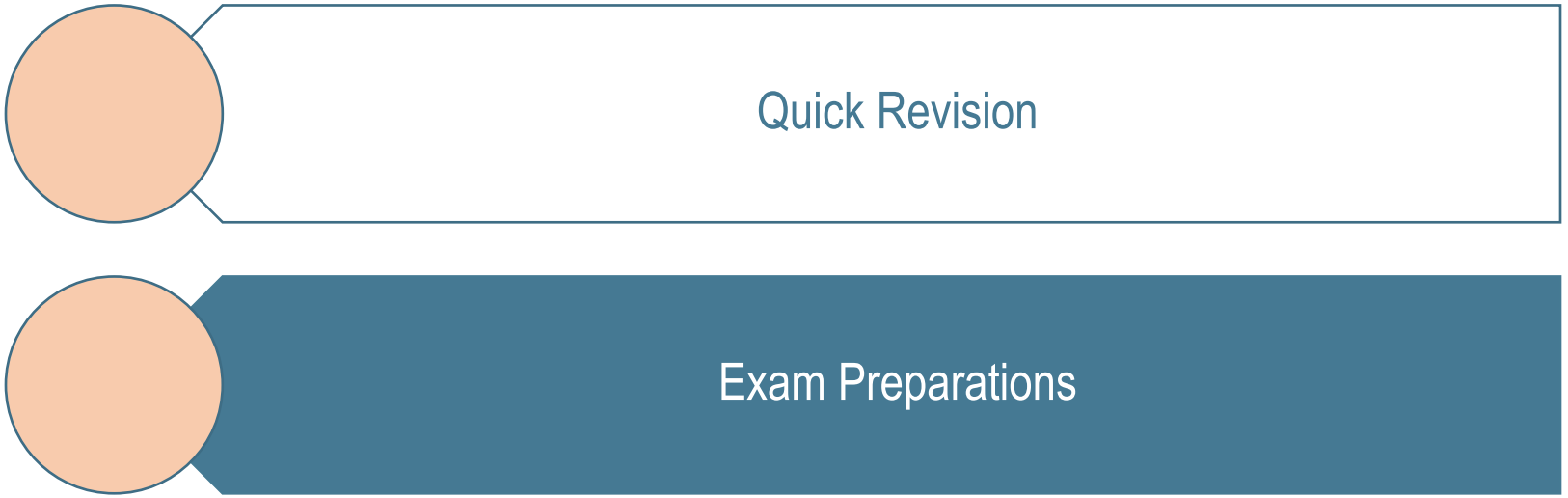
- Spreading out the *k* initial cluster centers is a good thing and helps avoid suboptimal clustering results
- Proceed as follows:
  1. Choose one center uniformly at random from among the data points
  2. For each data point *x* compute the distance between *x* and the nearest center that has already been chosen
  3. Choose one new data point at random as a new center, using a weighted probability distribution where a point *x* is chosen with probability proportional to distance squared
  4. Repeat Steps 2 and 3 until *k* centers have been chosen

Hierarchical Clustering



- For a given “distance between clusters”, a horizontal line intersects the clusters that are that far apart, to create clusters
- E.g., at distance of 4.6 (red line in next slide), data can be reduced to 2 clusters -- The smaller of the two is circled
- At distance of 3.6 (green line) data can be reduced to 6 clusters, including the circled cluster

Agenda



The exam will be a 1-hour closed-book examination – 50 points required to pass

Location and time	**Date:** 07/02/2022 & 10/03/2022 **Time:** 15:00-16:00 (please arrive well in advance) (both exams) **Location:** 100 Hörsaal II
Scope and format	60 minutes duration Written format Closed-book Language: English (provide all answers in English!) No calculators allowed and needed
Grading	Maximum of **100 points** available **Standard** university-wide **grading scheme**, i.e. 1,0; 1,3; 1,7; etc. **Passing grade: 4,0**

The exam will consist out of four building blocks – 100 points as maximum score

Block 1 – Multiple choice questions	▪ 5 multiple choice question (one question counting for two points) ▪ Single or multiple correct answers	10 points
Block 2 – Python questions	▪ One question (+ sub-questions) on Python, e.g. describing and/or correcting code	20 points
Block 3 – Written questions	▪ Three or four written questions testing deeper understanding of key topics covered throughout the course	70 points

Let's have a look at some examples – Block 1: Multiple Choice Questions

- **Tick your answers:**

EXAMPLES

- (2 point) You have been given data with descriptive features on students and info on whether they pass a class or not? You now wish to develop a predictive model. Which approaches could be most appropriate? Select **all** that apply.

		T	F
a)	K-means	☐	☐
b)	K-nearest-neighbors	☐	☐
c)	Linear classification	☐	☐
d)	Logistic classification	☐	☐

- (2 point) Which of the following is **false** about parameter optimization using gradient descent considering squared loss function?

		T	F
a)	The global optimum is always found	☐	☐
b)	Gradient descent may converge to a local optimum	☐	☐
c)	Parameter initialization is relevant	☐	☐
d)	The step size should be as small as numerically possible to guarantee convergence	☐	☐

- Block 1 is designed to give you a smooth start into the exam
- Multiple choice questions to test your general knowledge
- One or more correct answers

Let's have a look at some examples – Block 1: Multiple Choice Questions

- **Tick your answers:**

EXAMPLES

- (2 point) You have been given data with descriptive features on students and info on whether they pass a class or not? You now wish to develop a predictive model. Which approaches could be most appropriate? Select **all** that apply.

		T	F
a)	K-means	☐	☑
b)	K-nearest-neighbors	☑	☐
c)	Linear classification	☑	☐
d)	Logistic classification	☑	☐

- (2 point) Which of the following is **false** about parameter optimization using gradient descent considering squared loss function?

		T	F
a)	The global optimum is always found	☐	☑
b)	Gradient descent may converge to a local optimum	☑	☐
c)	Parameter initialization is relevant	☑	☐
d)	The step size should be as small as numerically possible to guarantee convergence	☑	☐

- Block 1 is designed to give you a smooth start into the exam
- Multiple choice questions to test your general knowledge
- One or more correct answers

Let's have a look at some examples – Block 2: Python Questions

- **Below, you see a snippet of a Python program** EXAMPLES **for a simple data reading and cleaning procedure.**
  - Explain what you see?
  - How would you modify the code so that the data cleansing is completed?

Snippet on next pages!

- Block 2 tests your basic knowledge of the Python Language in a data science context
- If you came to the workshops and participated in the team assignment you should be well prepared!

In [62]:
```
import numpy as np
import pandas as pd
import seaborn as sns
```

In [63]:
```
raw_data = pd.read_csv("iris.csv")
```

In [64]:
```
raw_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
number          150 non-null int64
Sepal.Length    148 non-null float64
Sepal.Width     149 non-null float64
Petal.Length    149 non-null float64
Petal.Width     150 non-null float64
Species         150 non-null object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.1+ KB
```

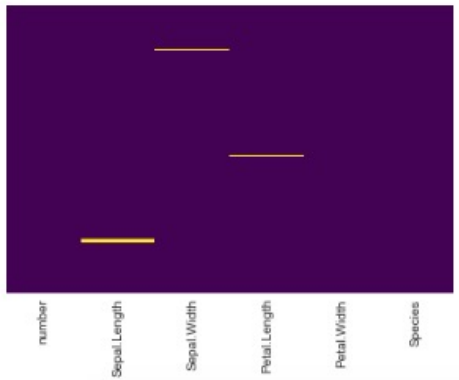
In [65]:
```
raw_data[raw_data["Sepal.Length"].isna()==True]  # There seem to be missing values (e.g. "Sepal.Length)
```

Out[65]:

	number	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
122	123	NaN	2.8	6.7	2.0	virginica
123	124	NaN	2.7	4.9	1.8	virginica

In [66]:
```
raw_data.dropna()     # drop all NaNs

sns.set_style("whitegrid")
sns.heatmap(raw_data.isnull(),yticklabels=False,cbar=False, cmap="viridis")   # check graphically for NaNs
plt.show()
```



```
In [73]: import numpy as np
         import pandas as pd
         import seaborn as sns

In [63]: raw_data = pd.read_csv("iris.csv")

In [69]: raw_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
number          150 non-null int64
Sepal.Length    148 non-null float64
Sepal.Width     149 non-null float64
Petal.Length    149 non-null float64
Petal.Width     150 non-null float64
Species         150 non-null object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.1+ KB
```

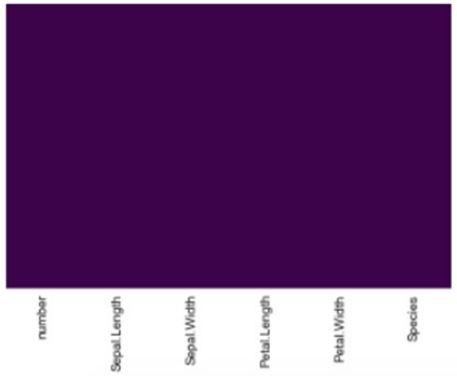
```
In [70]: raw_data[raw_data["Sepal.Length"].isna()==True]  # There seem to be missing values (e.g. "Sepal.Length)
```

Out[70]:

	number	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
122	123	NaN	2.8	6.7	2.0	virginica
123	124	NaN	2.7	4.9	1.8	virginica

```
In [72]: raw_data.dropna(inplace=True)     # drop all NaNs

         sns.set_style("whitegrid")
         sns.heatmap(raw_data.isnull(),yticklabels=False,cbar=False, cmap="viridis")   # check graphically for NaNs
         plt.show()
```




```
raw_data.dropna(inplace=True)     # drop all NaNs

sns.set_style("whitegrid")
sns.heatmap(raw_data.isnull(),yticklabels=False,cbar=False, cmap="viridis")
plt.show()
```

- A simple „inplace=True“ confirms that you wish to permanently drop NaN values from the DataFrame

```
In [68]: import numpy as np
         import pandas as pd
         import seaborn as sns

In [63]: raw_data = pd.read_csv("iris.csv")

In [64]: raw_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
number          150 non-null int64
Sepal.Length    148 non-null float64
Sepal.Width     149 non-null float64
Petal.Length    149 non-null float64
Petal.Width     150 non-null float64
Species         150 non-null object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.1+ KB
```

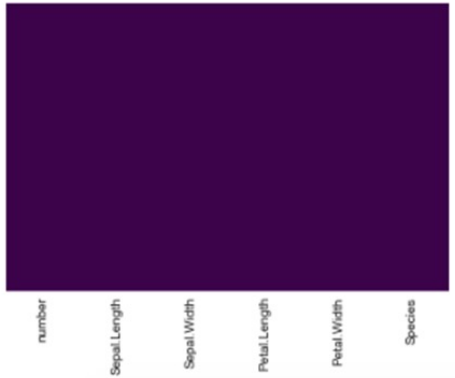
```
In [65]: raw_data[raw_data["Sepal.Length"].isna()==True]  # There seem to be missing values (e.g. "Sepal.Length)
```

Out[65]:

	number	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
122	123	NaN	2.8	6.7	2.0	virginica
123	124	NaN	2.7	4.9	1.8	virginica

```
In [67]: raw_data_clean = raw_data.dropna()     # drop all NaNs

         sns.set_style("whitegrid")
         sns.heatmap(raw_data_clean.isnull(),yticklabels=False,cbar=False, cmap="viridis")   # check graphically for NaNs
         plt.show()
```




```
raw_data_clean = raw_data.dropna()     # drop all NaNs

sns.set_style("whitegrid")
sns.heatmap(raw_data_clean.isnull(),yticklabels=False,cbar=False, cmap="viridis")
plt.show()
```

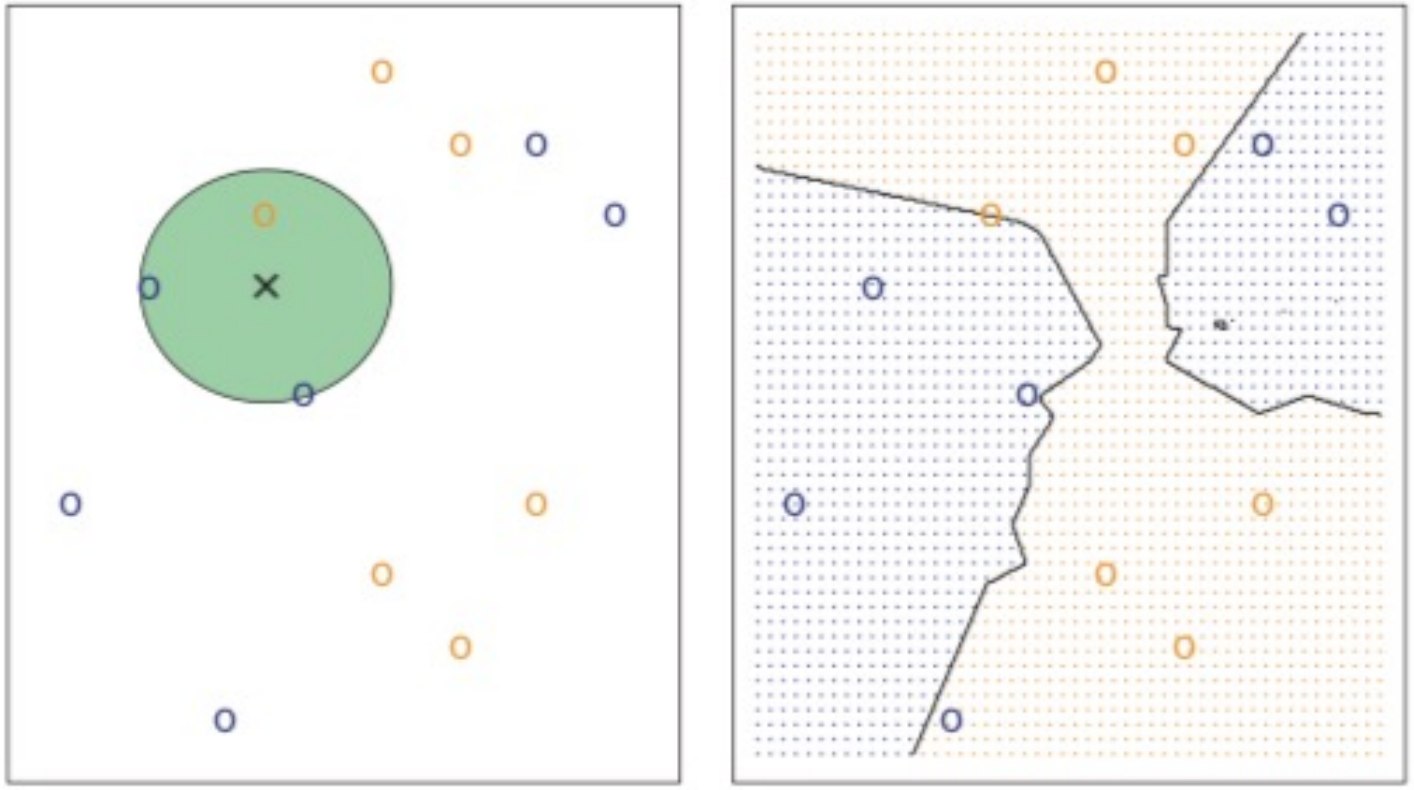
- Alternatively you can also define a new DataFrame (preferred if you do not want to permanently change the original DataFrame)

Let's have a look at some examples – Block 3: Written Questions

- **K-Nearest-Neighbors** EXAMPLES
- (20 points) You have a large high-dimensional set of binary labeled data which you want to build a classification model for. You opt for the K-nearest-neighbors model to classify new data.
  1. Explain the basic working principles of the k-nearest neighbor approach.
  2. How would you determine the optimal value for k? What happens if your k is…
     - … too high?
     - … too low?
  3. What are potential other approaches you might want to consider for this classification problem?

- In Block 3 we test you understanding of the covered concepts
- Give written and to-the-point answers to the questions – No page-long essays!

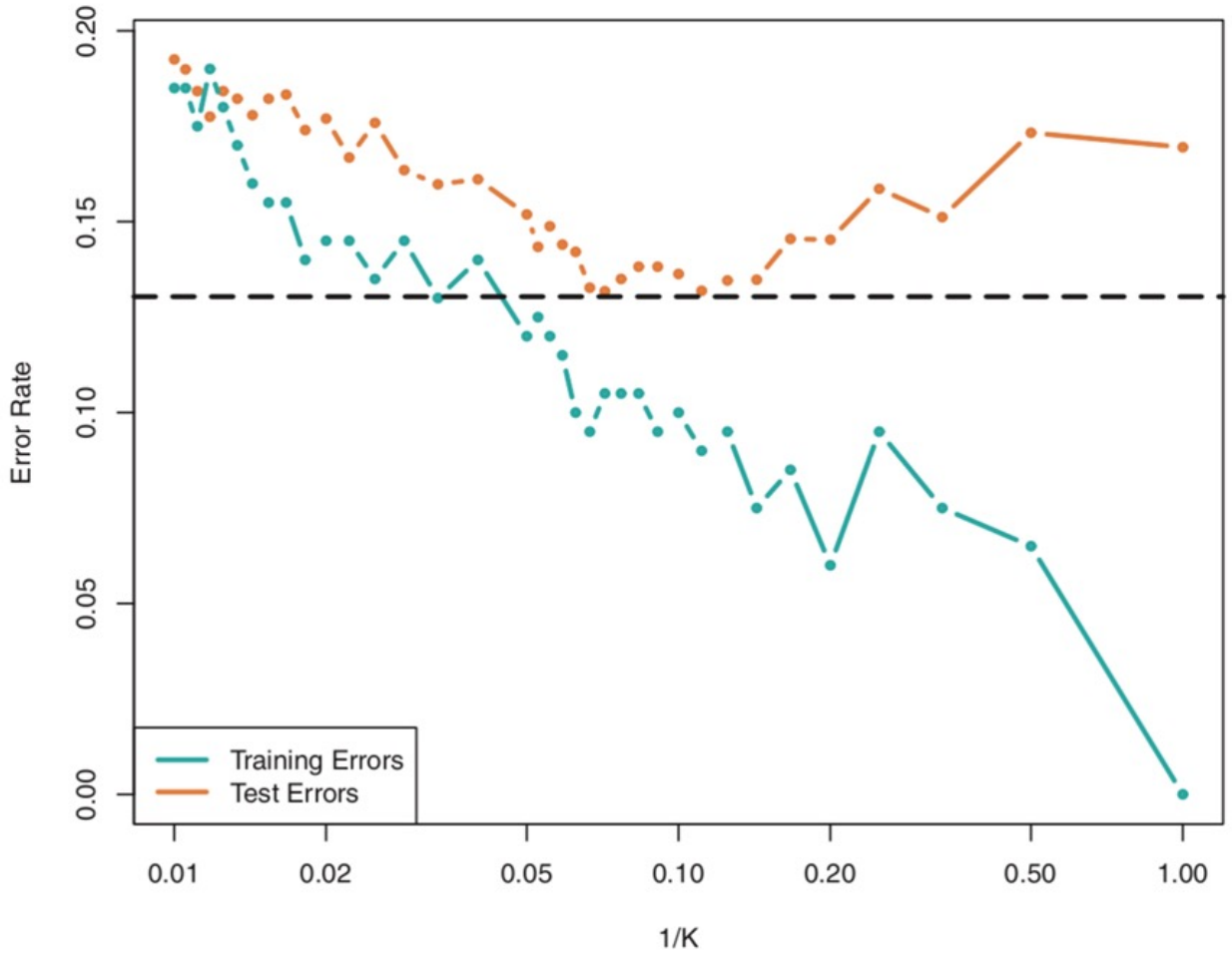
KNN – Basic intuition



- KNN classification is intuitively very simple
- A data point is classified by a majority vote of its neighbors, i.e. the object is assigned to the class most common among its k nearest neighbors
- k is a positive integer hyperparameter

Source: Elements of Statistical Learning

KNN – Defining correct k



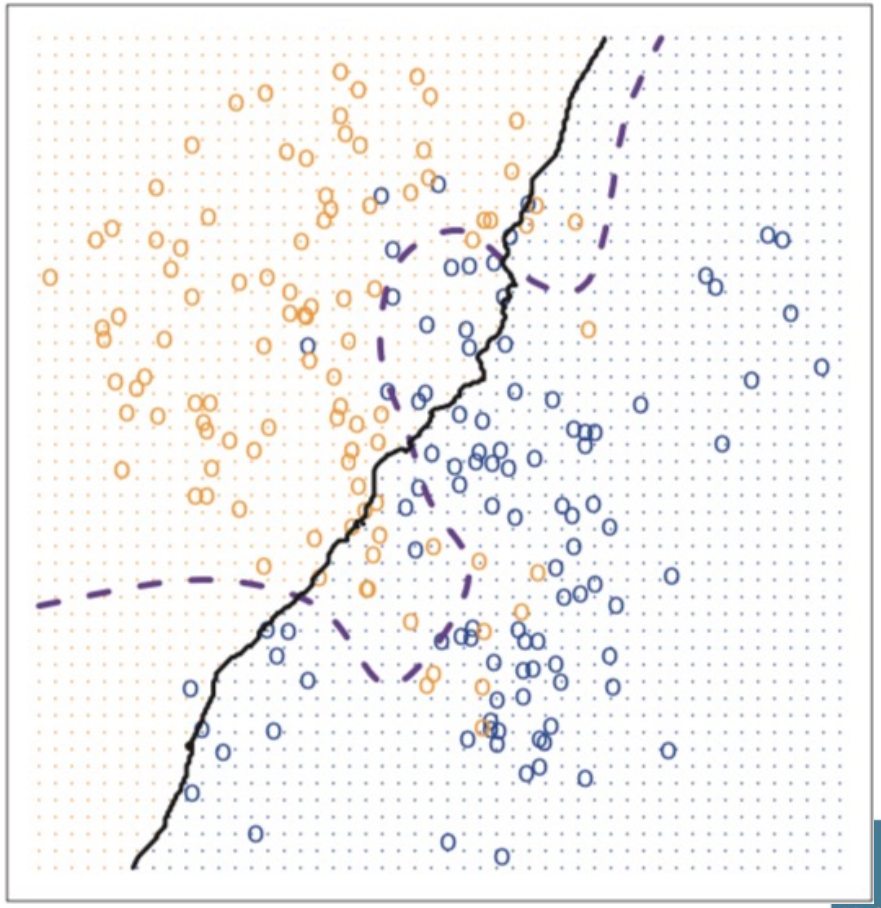
- The choice of K has a drastic effect on the KNN classifier obtained
- Plotting error rates versus 1/k illustrates how optimal k can be found
- With k = 1 or v. small, the KNN training error rate is 0, but the test error rate may be quite high – **We experience overfitting due to too high flexibility of the classifier**
- If k is large, both training and test error rate are high – **We experience underfitting due to too low flexibility of the classifier**

Source: Elements of Statistical Learning

KNN – Under- and Overfitting illustrated

KNN: K=1

KNN: K=100



Source: Elements of Statistical Learning

KNN – Potential alternatives

- Alternative classification approaches:
  - Linear Classifiers (with diff. loss functions)
  - Logistic regression (also essentially a linear classifier)
  - Support Vector Machines (also essentially a linear classifier with hinge loss)
  - Neural networks (MLP)
  - …

Here are some general straight-forward tips of how to make the most of the exam

- **Come prepared** – Although straight-forward it does pay to have read and understood the lecture material. There will not be a lot of time to actually start thinking about things during the exam
- **Focus your efforts** – Be smart as to where to focus your efforts. Collect “easy points” but make sure to focus on the ”valuable” questions
- **Manage time** – Given the relatively short amount of time available the scope of the exam is relatively large. This is intentional and will require you to manage time effectively. If you do not have the answer to a question move on to the next one.

Any further questions?

Contact

For general questions and enquiries on **research**, **teaching**, **job openings** and new **projects** refer to our website at www.is3.uni-koeln.de

For specific enquiries regarding this course contact us by sending an email to the **IS3 teaching** address at is3-teaching@wiso.uni-koeln.de

Follow us on **Twitter** at **@IS3_UniCologne** to stay up to date with recent publication and presentations of our group