Introduction to Data Mining

Prof. Dr. Stephan Trahasch
Offenburg University of Applied Sciences

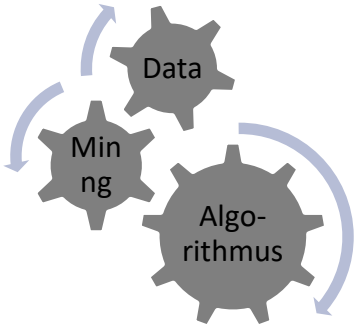
Definition of Data Mining

Data Mining is a non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. (Fayyad et al. 1996)

Extraction of

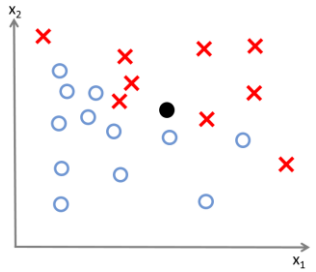
Implicit, new and (hopefully) useful

patterns.

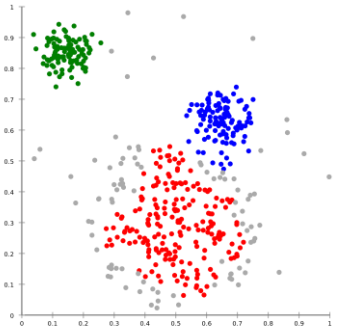


Data Mining Tasks

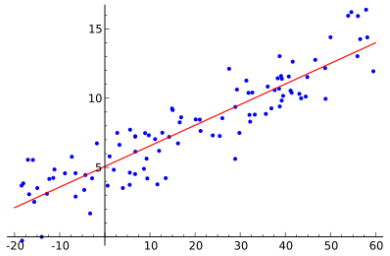
Classification



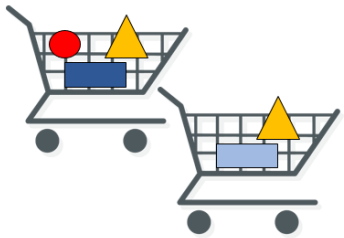
Clustering



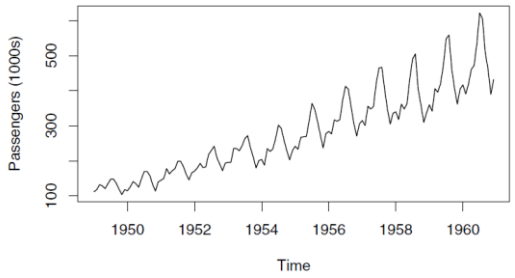
Regression



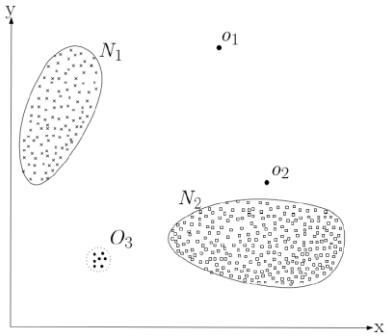
Association Pattern



Time Series



Outlier Analysis



Data Mining

Requirements: We have data

Task: Find patterns and underlying rules in data.

How can we solve the problem automatically?
→ Data Mining Algorithms

Challenges:

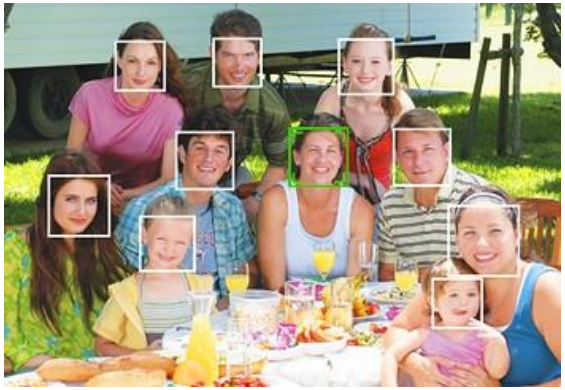
- How to automatically measure the quality of the result?
- How to decide, if a pattern is useful or not?

Learned patterns and rules can be used for prediction.

Keep in mind that there are not always underlying patterns and rules that can be learned, even if we have a lot of data!

Types of prediction problems

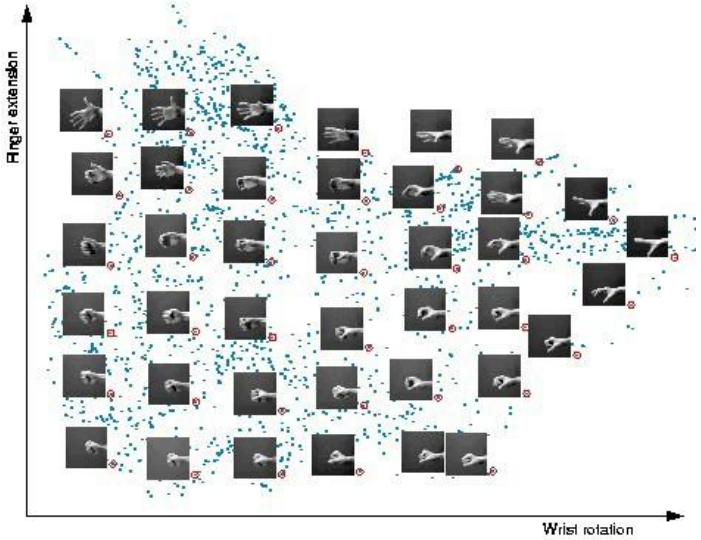
- **Supervised learning**
  - "Labeled" training data
  - Every example has a desired target value, a "best answer"
  - Reward prediction being close to target



57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0	1
78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0	0
69,F,180,0,115,85,40,22,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0	1
18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	0
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0	0
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0	0
84,F,210,1,135,105,39,24,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0	1
89,F,135,0,120,95,36,28,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,1,0,0	1
49,M,195,0,115,85,39,32,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0	0
40,M,205,0,115,90,37,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	0
74,M,250,1,130,100,38,26,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	0
77,F,140,0,125,100,40,30,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,1	1

Types of prediction problems

- Supervised learning
- **Unsupervised learning**
  - No known target values
  - No targets = nothing to predict?
  - Reward “patterns” or “explaining features”
  - Often, data mining



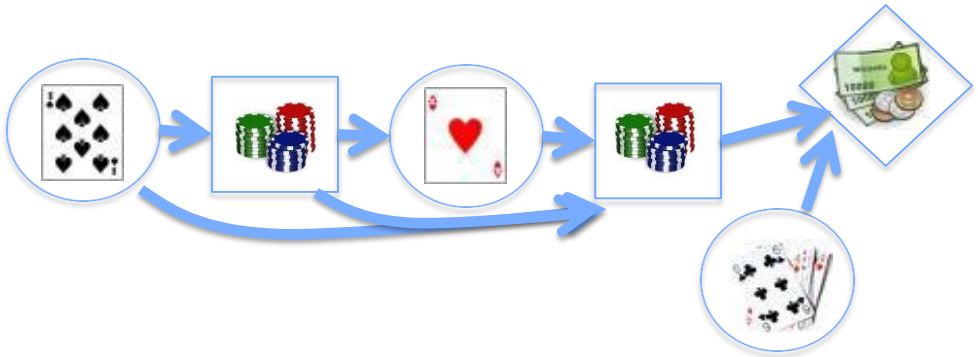

```
57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0
78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0
69,F,180,0,115,85,40,22,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0
18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0
84,F,210,1,135,105,39,24,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0
89,F,135,0,120,95,36,28,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0
49,M,195,0,115,85,39,32,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0
40,M,205,0,115,90,37,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
74,M,250,1,130,100,38,26,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
77,F,140,0,125,100,40,30,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,1
```

Types of prediction problems

- Supervised learning
- Unsupervised learning
- **Semi-supervised learning**
  - Similar to supervised
  - some data have unknown target values
  - Example: medical data
    Lots of patient data, few known outcomes
  - Example: image tagging
    Lots of images on Flickr, but only some of them tagged

Types of prediction problems

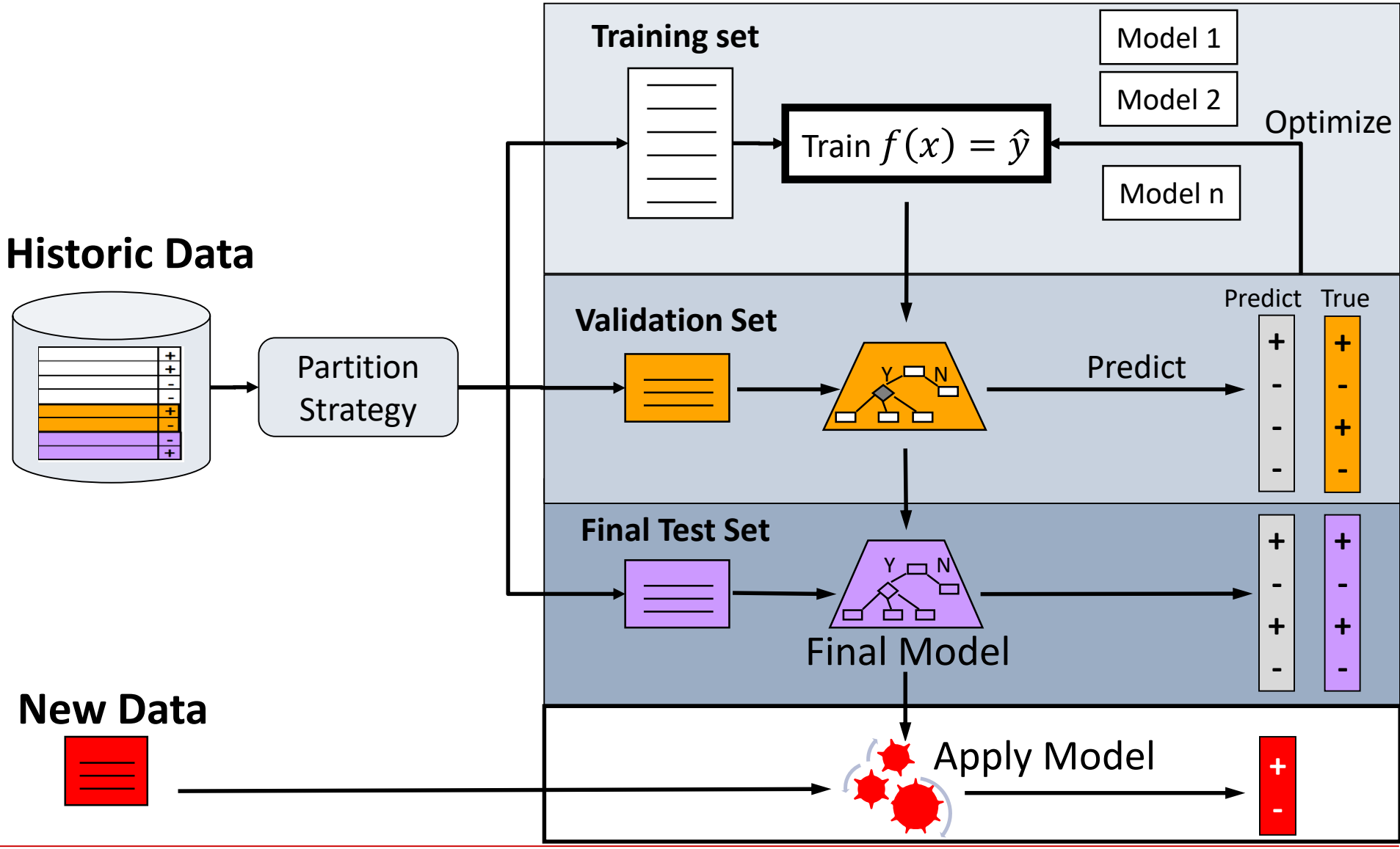
- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- **Reinforcement learning**
  - "Indirect" feedback on quality
  - No answers, just "better" or "worse"
  - Feedback may be delayed



Terminology

Object/concept	Objects of interest like customer, product, machine
Features or attributes	A set of characteristic features or attributes that describe the object (quantitativ and/or qualitative)
Label or target	A specific attribute to be predicted
Sample data	Data of observations. One observation is called an instance or example.
Model	A algorithm or function that assigns an output value of the target attribute to objects. This function minimizes an error function or optimizes a quality function.

Training, validation and test set



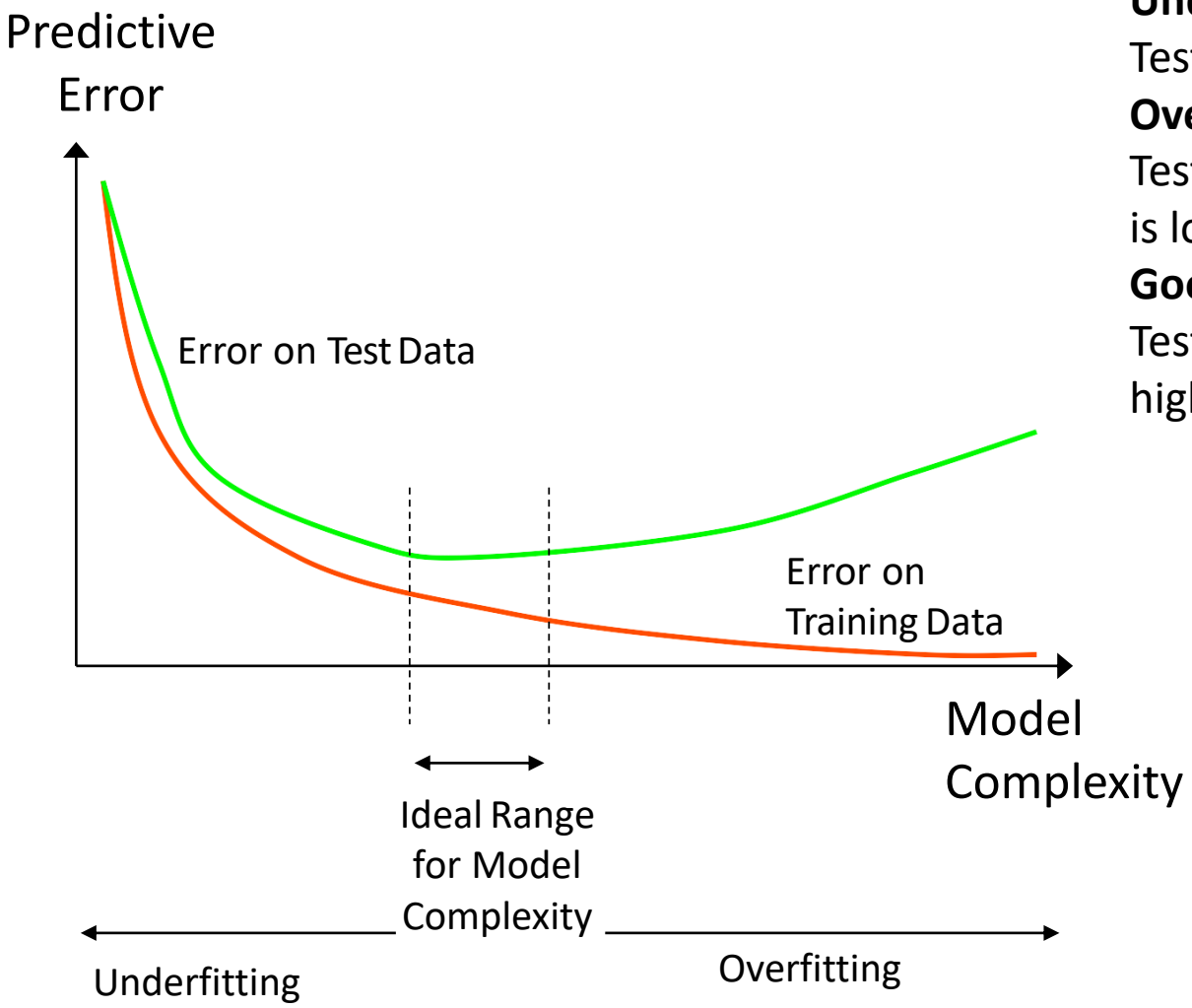
Hyperparameter

Hyperparameters are, for example, weightings on penalty straps, number of decision trees in Random Forets, number of hidden nodes of a neural network,. ....

Division of data into training, validation and test set.

- The **model is trained on the training data** with different values of the hyperparameter.
- We optimize and select the model with the appropriate hyperparameters, which showed the **best performance on the validation data**. → Training Error
- The error of this optimized model is calculated with the **test data**. → Test Error

How Overfitting affects Prediction



Underfitting
Test and training error are both high

Overfitting
Test error is high while training error is low

Good fit
Test error is low, and only slightly higher than the training error

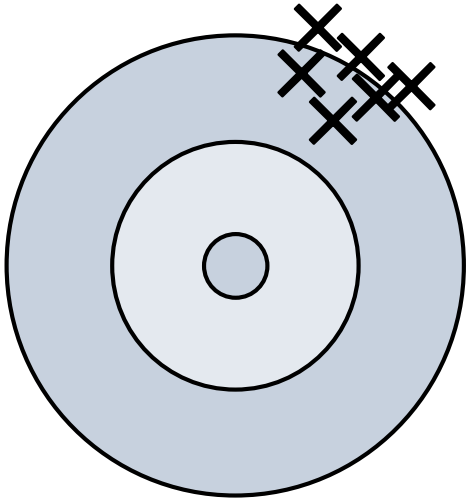
Bias-Variance Tradeoff

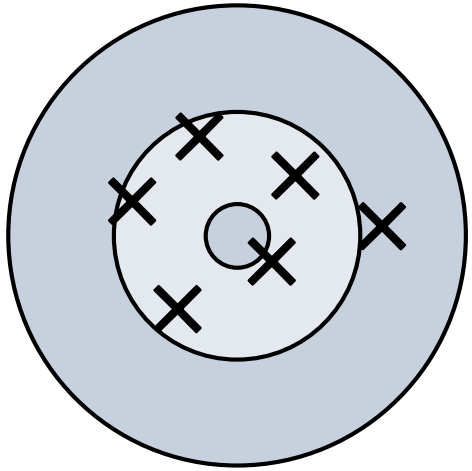
- **Bias:** difference between what you expect to learn and truth
  - Measures how well you expect to represent true solution
  - Decreases with more complex model

- **Variance:** difference between what you expect to learn and what you learn from a from a particular dataset
  - Measures how sensitive learner is to specific dataset
  - Increases with more complex model

Bias-Variance Tradeoff

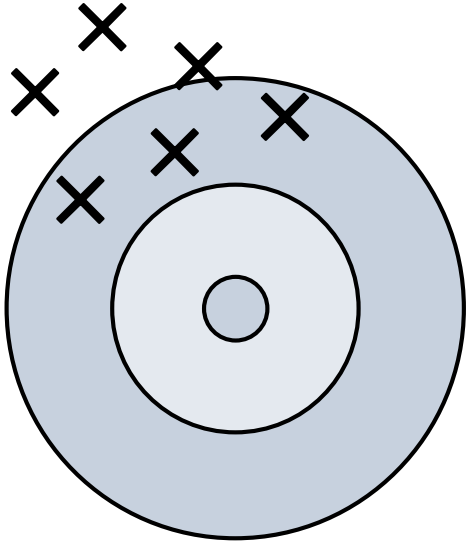
High bias
Low variance

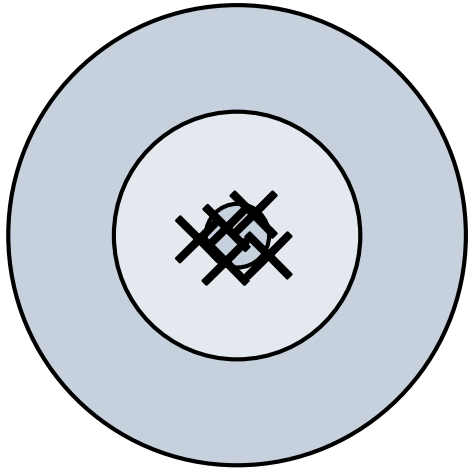




Low bias
High variance

High bias
High variance





Low bias
Low variance

Ensemble Methods, Random Forests and AdaBoost

Prof. Dr. Stephan Trahasch

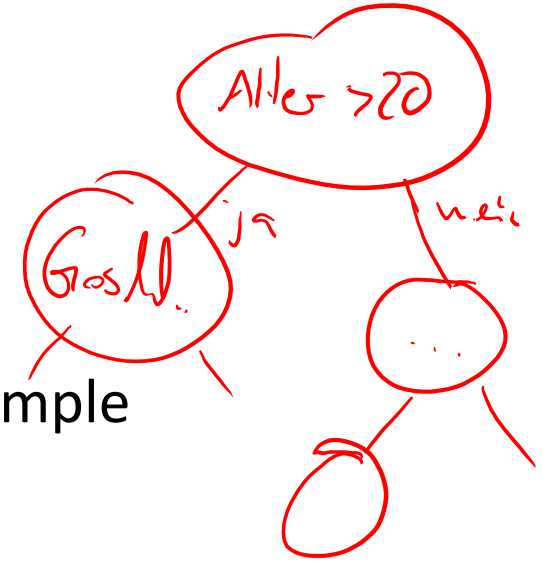
Offenburg University of Applied Sciences

V 0

Bias and Variance Decomposition

Bias

the part of the error that is caused by the model



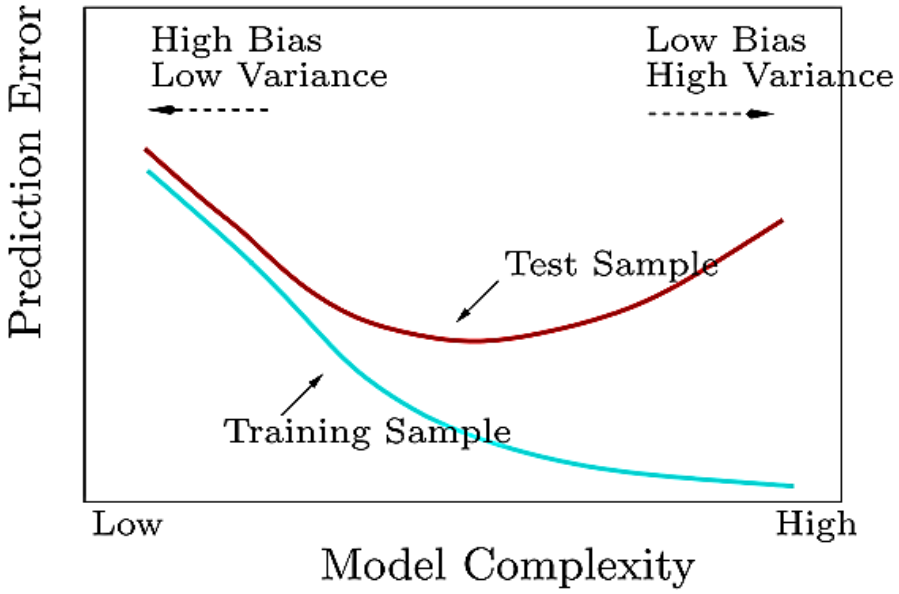
Variance

the part of the error that is caused by the data sample

Bias-Variance Trade-off

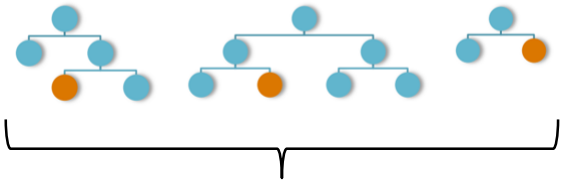
- Algorithms that can easily adapt to any given decision boundary are very sensitive to small variations in the data and vice versa
- Models with a low bias often have a high variance
  - e.g., nearest neighbor, unpruned decision trees
- Models with a low variance often have a high bias
  - e.g., decision stump, linear model

Bias-Variance Trade-off



Random Forest

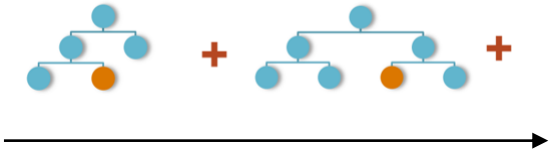
Variance ↓



Voting

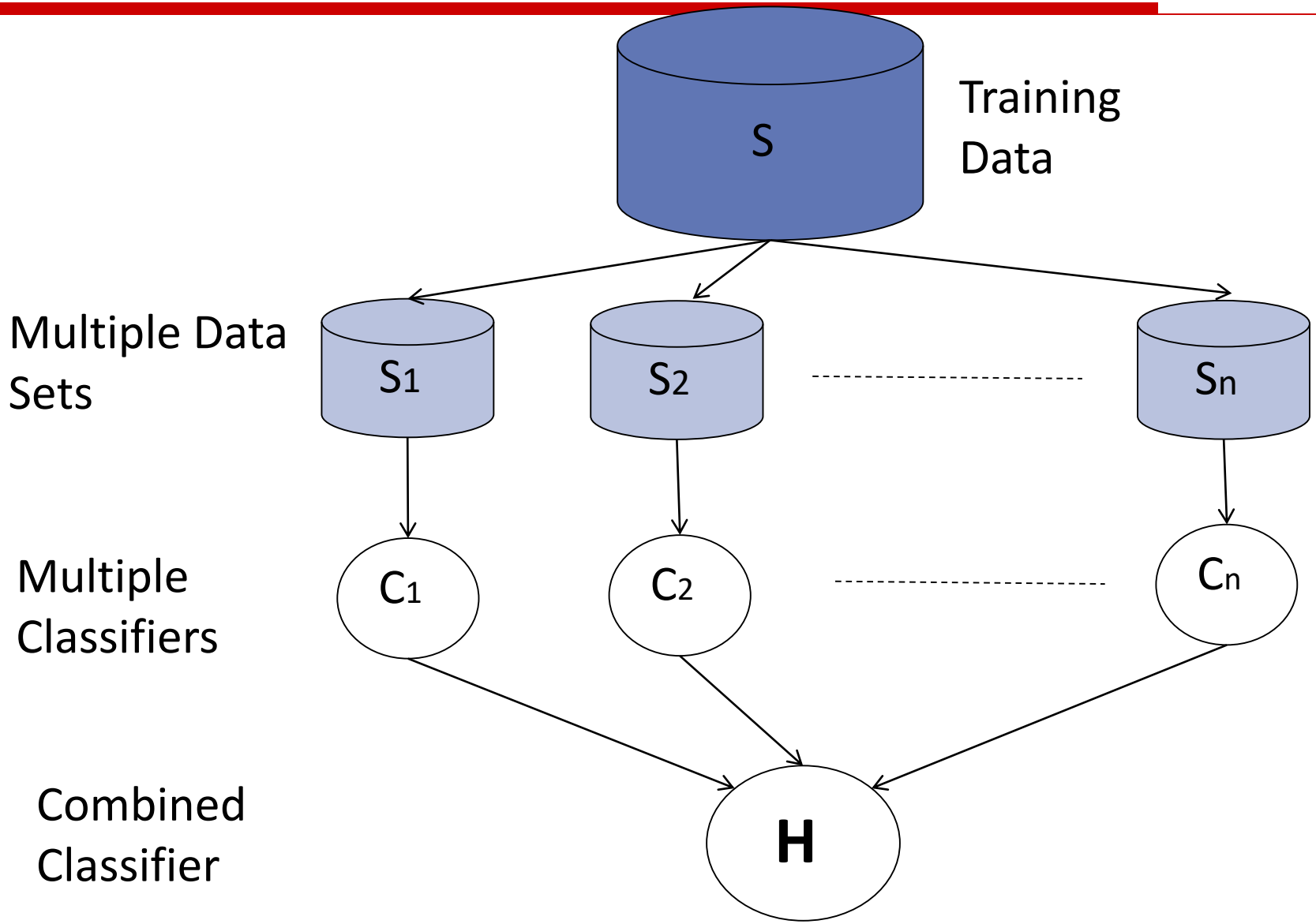
Booosting

Bias ↓



Weighted Sum

Basic Idea of Meta Learning



Ensemble Methods

- Bootstrapping → Aufteilung der Daten
- Bagging (Breiman 1994)
- Boosting (Schapire 1989)
- Adaboost (Schapire 1995)
- Random Forest (Breiman 2001)

Prediction of classes by combining several models that have been trained.

Bootstrap – also called 0.632 Bootstrap

Given a record with n instances (observations).
Bootstrap sample: draw a sample of the same length from the original dataset with replacement.

Original Data	1	2	3	4	5	6	7	8	9	10
Bootstrap 1	7	8	10	8	2	5	10	10	5	9
Bootstrap 2	1	4	9	1	2	3	2	7	3	2
Bootstrap 3	1	8	5	10	5	5	9	6	3	7
…										

For each set of size n, the probability that a given example appears in it is $\Pr(x \in B_i) = 1 - \left(1 - \frac{1}{n}\right)^n \xrightarrow[n\to\infty]{} 0.6322$

On average, less than 2/3 of the examples appear in any single bootstrap sample.

Bagging - Aggregate Bootstrapping
Models may differ when learned on different data samples

- Idea of bagging:
  - create samples by picking examples with replacement
  - learn a model on each sample
  - combine models
- Samples
  - differ in the subset of examples
  - replacement randomly re-weights instances
- Uses usually the same base learner of B bootstrap versions of the original dataset and calculates mean value from the predictions of the models
- Reduces the variability of the prediction
- Works better with predictions that show small bias but have a higher variance. E. g. decision trees

Bagging: Basic approach

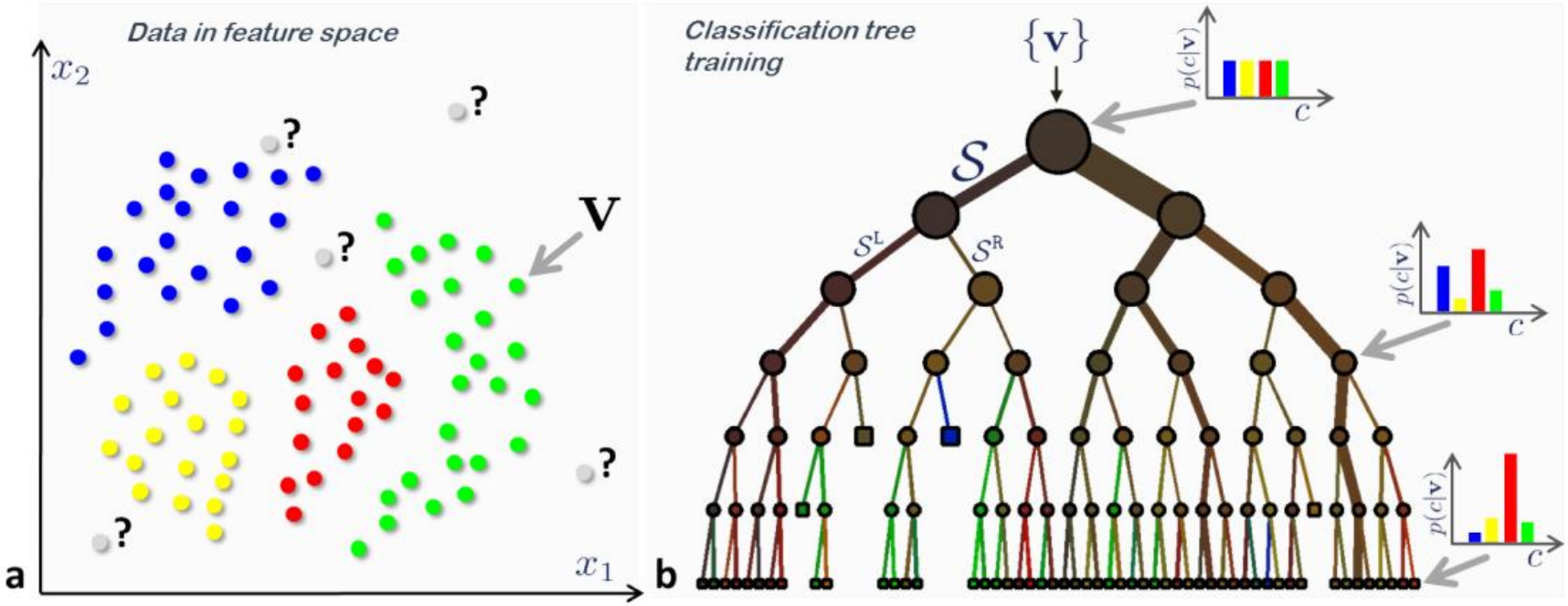
1. for m = 1 to t // t ... number of iterations
   a) draw with replacement a bootstrap sample D_m of data
   b) learn a classifier C_m from D_m
2. for each test example
   a) try all classifiers C_m
   b) predict the class that receives the highest number of votes

- variations are possible
  e.g., size of subset, sampling w/o replacement, etc.
- many related variants
  - sampling of features, not instances
  - learn a set of classifiers with different algorithms

Boosting

- Idea of boosting
  - train a set of classifiers, one after another
  - later classifiers focus on examples that were misclassified by earlier classifiers
  - weight the predictions of the classifiers with their error
- Realization
  - perform multiple iterations
  - Each time using different example weights
  - weight update between iterations
    - increase the weight of incorrectly classified examples
    - so they become more important in the next iterations (misclassification errors for these examples count more heavily)
  - combine results of all iterations
    - weighted by their respective error measures

Decision Tree splits input data into cases



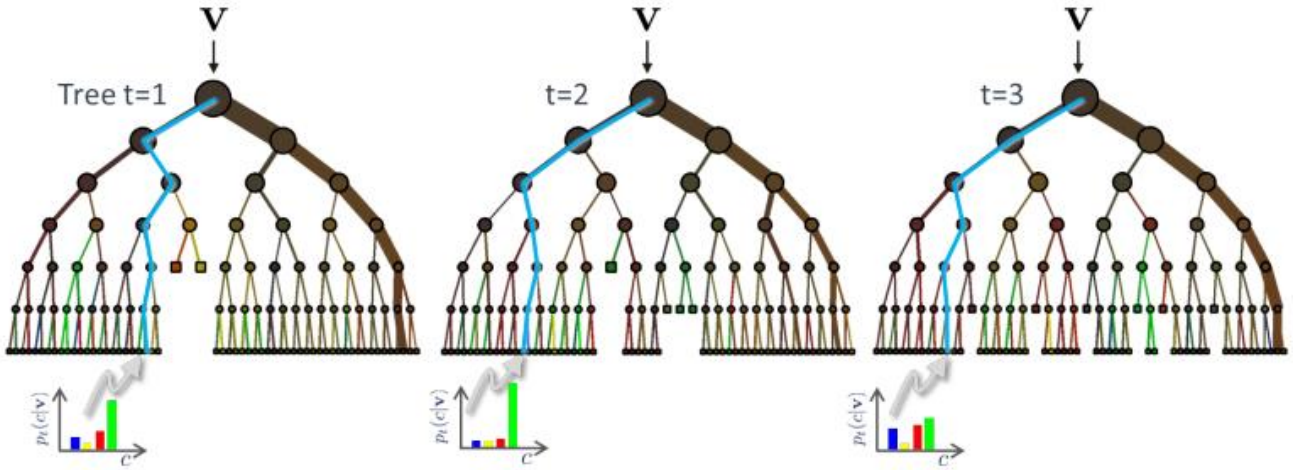
Criminisi, A., Shotton, J., and Konukoglu, E., "Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning," MSR-TR-2011-114, 2011.

Random Forest
Combines bagging and random attribute subset selection

- Build the tree from a bootstrap sample
- Instead of choosing the best split among all attributes, select the best split among a random subset of k attributes
- is equal to bagging when k equals the number of attributes

- There is a bias/variance tradeoff with k:
  - The smaller k, the greater the reduction of variance but also
  - the higher the increase of bias

Random Forest



- Random Forest consists of many "Trees", hence "Forest".
- Random “because the trees are randomly generated”
- It is not the whole dataset, but bootstrap learning samples are used
- At each node only random selection of attributes
- Assigning a value to each record based on the decision of each tree → majority

randomForest, ranger

Random Forest for Classification or Regression

1. For b = 1 to B ~500

   a) Draw a bootstrap sample **Z*** of size *N* from the training data.

   b) Grow a random forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

      i. Select *m* variables at random from the *p* variables.

      ii. Pick the best variable/split-point among the *m*.

      iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$

Classification:

Let $\hat{C}_b(x)$ be the class prediction of the *b*th random-forest tree.

Then $\hat{C}_{rf}^B(x) = \text{majority vote}\{\hat{C}_b(x)\}_1^B$

Regression: $\hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$

Bootstrap: Out-Of-Bag (OOB)

Observations that are not included in the bootstrap sample are called Out-Of-Bag (OOB).

Out-of-Bag error rate can be calculated for any observation over whole forest and corresponds to leave-one-out error rate.

Unlike most models, cross-validation is performed while Random Forest is being adapted.

→ no cross-validation necessary

Advantages of Random Forest

- Suitable for high-dimensional data where the number of attributes exceeds the number of objects observed
- yet robust against overfitting
- Quantifies the importance of individual variables
- Good performance compared to classics such as support vector machines and neural networks
- Straightforward estimation of the error rate
- Consistent estimation of target values
- Problem with a single tree:
  has high variance → wrong decision in "high" nodes runs through subsequent nodes → many trees reduce this effect

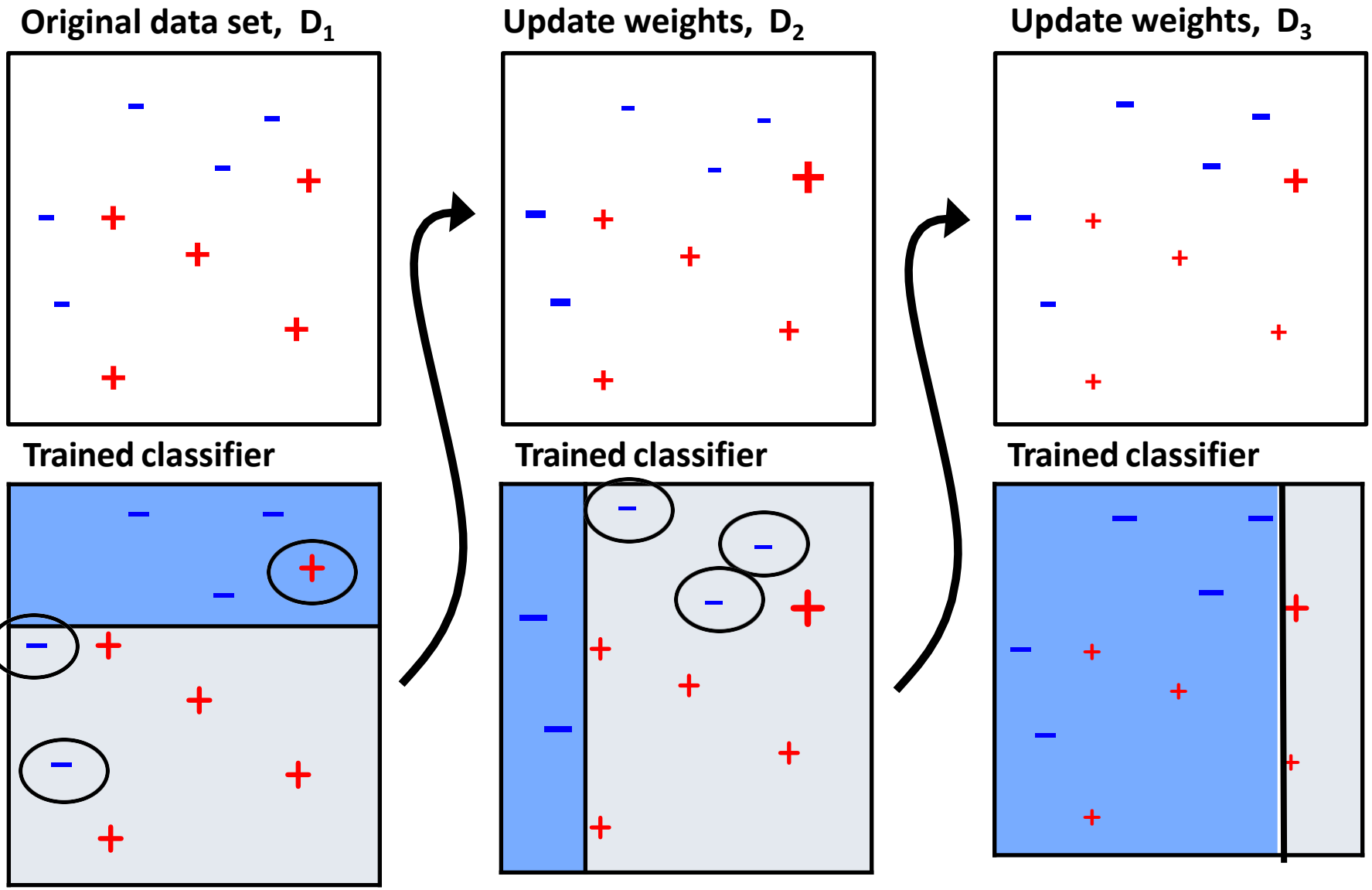
Differences to an standard Decision Tree

- Each tree is trained with a “bootstrap resample of data”. “Bootstrap resample of data set with N samples: Make new data set by drawing with replacement N samples; i.e., some samples will probably occur multiple times in new data set”
- For each split, randomly select m attributes from the total p attributes ($m = \sqrt{p}$)
- No Pruning

Boosting

- Ensembles
  - Weighted combinations of classifiers
  - “Committee” decisions
  - Equal weights (majority vote) in a simple case
  - Might want to weight unevenly – up-weight good experts
- **Boosting**
  - Focus new experts on examples that others get wrong
  - Train experts sequentially
  - Errors of early experts indicate the “hard” examples
  - Focus later classifiers on getting these examples right
  - Combine the whole set in the end
  - Convert many “weak” learners into a complex classifier

Example: Classes +1 , -1



Algorithm AdaBoost.M1 - Adaptive Boosting

1. Initialize example weights $w_i = \frac{1}{N} \quad (i = 1, \ldots N)$
2. for m = 1 to B // B ... number of iterations
   a) learn a classifier C_m using the current example weights
   b) compute a weighted error estimate

$$err_m = \frac{\sum w_i \text{ of all incorrectly classifed } e_i}{\sum_{i=1}^{N} w_i}$$

   c) compute a classifier weight $\alpha_m = \frac{1}{2}\ln\left(\frac{1-err_m}{err_m}\right)$
   d) for all **correctly** classified examples e_i: $w_i \leftarrow w_i e^{-\alpha_m}$
   e) for all **incorrectly** classified examples e_i: $w_i \leftarrow w_i e^{+\alpha_m}$
   f) normalize the weights w_i so that they sum to 1

Boosting – Algorithm AdaBoost.M1

…

3. for each test example
   a) try all classifiers C_m
   b) predict the class that receives the highest sum of weights α_m

Boosting

- Each of the trees can be rather small, with just a few terminal nodes.
- They learn slowly.
- In general, statistical learning approaches that learn slowly tend to perform well.
- Unlike bagging and random forests, boosting can overfit if the number of trees is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select the number of trees.
- In boosting, unlike in bagging, the construction of each tree depends strongly on the trees that have already been grown.

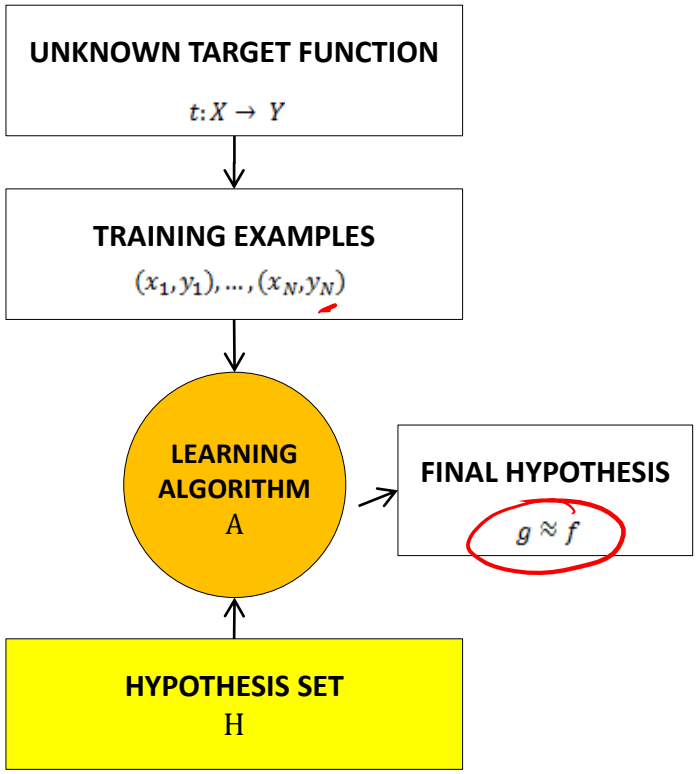
Support Vector Machines
SVM

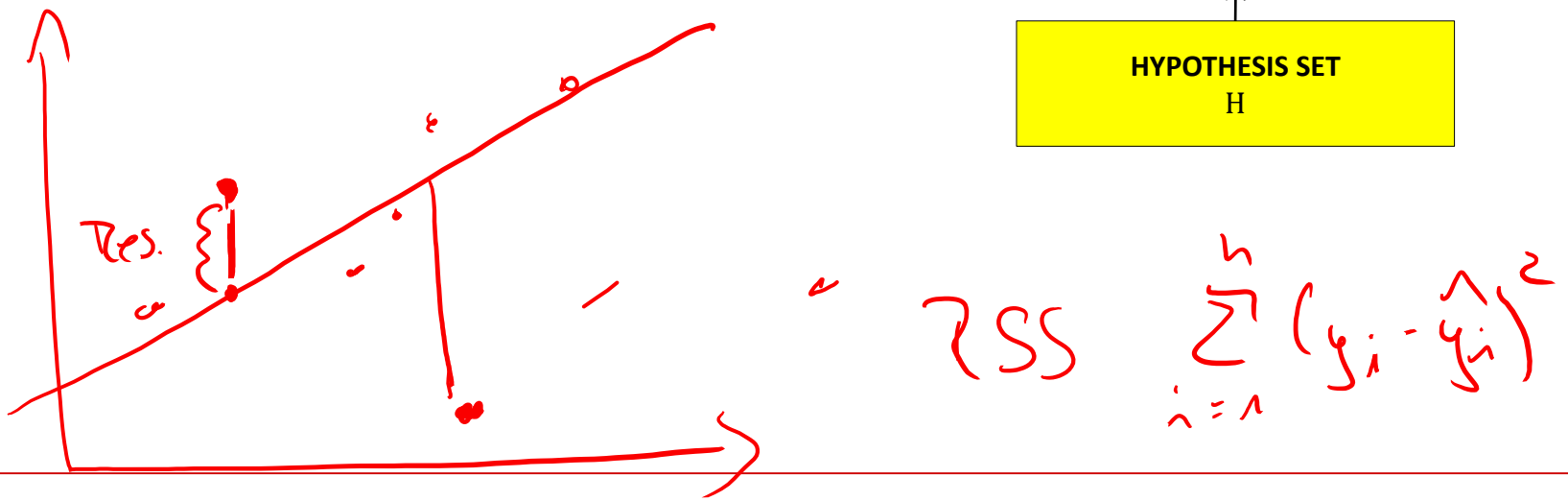
Prof. Dr. Stephan Trahasch

Offenburg University of Applied Sciences

Support Vector Machine as „linear classifier“

- Learn the optimal linear separation of data elements
- Allows later classification of unknown test data based on the learned linear hyperplane
- No machine in the true sense of the word



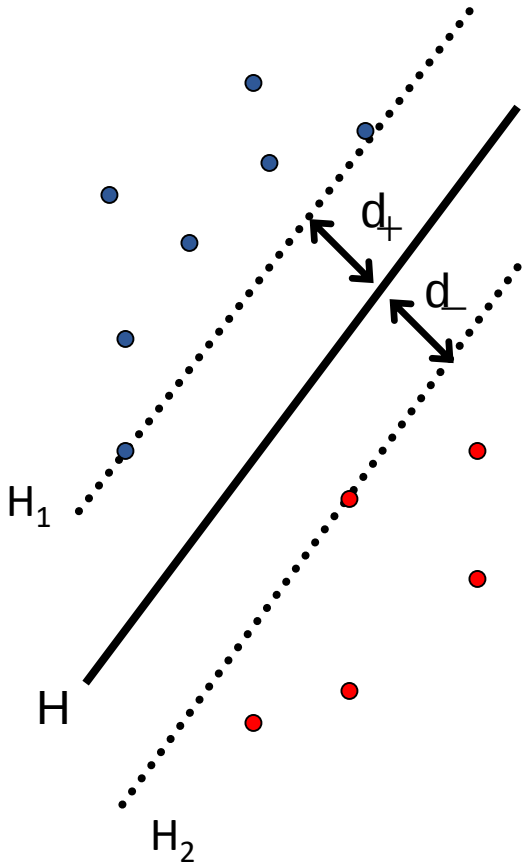


Optimal hyperplane

Observations are called linearly separable if there is a hyperplane H that separates the positive and negative examples.

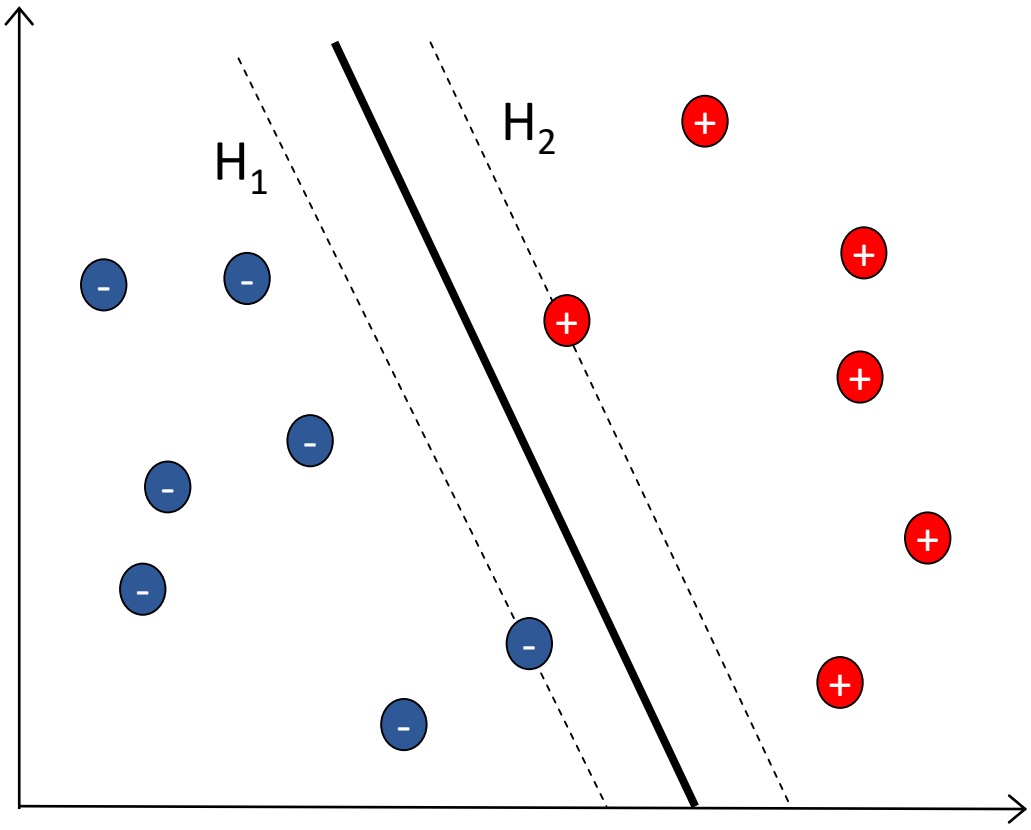
H is a **optimal hyperplane**, if its distance d to the next positive and to the next negative example is maximum.

There is a clearly defined optimal hyperplane.



Margin

$$H = \{x | x^T \beta + \beta_0 = 0\}$$



$$x_+^T \beta + \beta_0 \geq +1$$

$$x_-^T \beta + \beta_0 \leq -1$$

$$y_i = \begin{cases} +1 & \text{for } + \text{ samples} \\ -1 & \text{for } - \text{ samples} \end{cases}$$

$$y_i(x_+^T \beta + \beta_0) \geq +1$$

$$y_i(x_-^T \beta + \beta_0) \geq +1$$

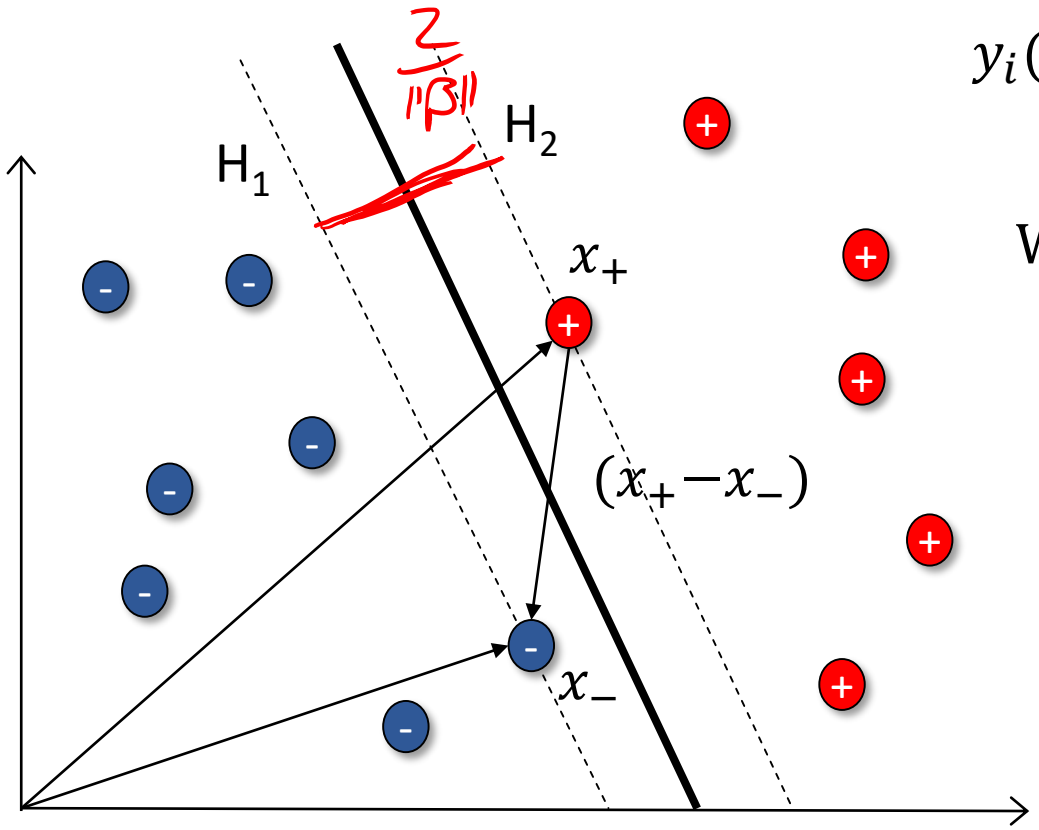
$$y_i(x_{+/-}^T \beta + \beta_0) - 1 \geq 0$$

$$y_i(x_i^T \beta + \beta_0) - 1 = 0$$

For x_i in gutter.

Margin

$$H = \{x | x^T \beta + \beta_0 = 0\}$$



$$y_i(x_{+/-}^T \beta + \beta_0) - 1 \geq 0$$

$$y_i(x_i^T \beta + \beta_0) - 1 = 0 \quad \text{For } x_i \text{ in gutter.}$$

$$\text{Width} = (x_+ - x_-) * \frac{\beta}{\|\beta\|} = \frac{2}{\|\beta\|}$$

The optimal hyperplane is determined by the nearest points of C+ and C-.

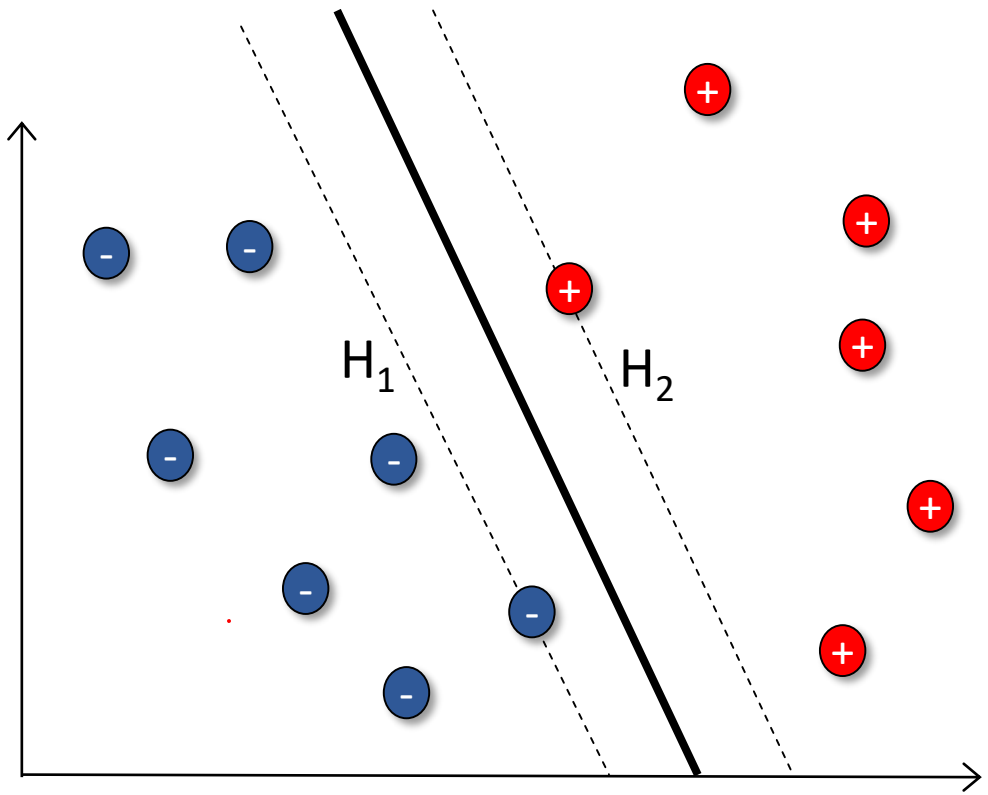
Margin

After construction there is no example between H_1 and H_2, i.e.

$$x^T\beta + \beta_0 \geq +1 \quad \forall \vec{x} \in C_+$$
$$x^T\beta + \beta_0 \leq -1 \quad \forall \vec{x} \in C_-$$

Distance $d(H_1, H_2) = \frac{2}{\left\|\vec{\beta}\right\|}$

is called Margin and should be maximized.



Maximum Margin

By maximizing the margin we find an optimal hyperplane within the set of possible separating hyperplanes.

To maximize $\frac{2}{\left\|\vec{\beta}\right\|}$ we can also minimize $\left\|\vec{\beta}\right\|^2$.

Minimization of $\frac{1}{2}\left\|\vec{\beta}\right\|^2$ is a convex quadratic optimization problem.

There is a clearly defined, optimal hyperplane

$$H = \{x \mid x\beta + \beta_0 = 0\}$$

The quadratic optimization problem can be solved in $O(n^3)$.

What do we know now?

- Maximizing the margin of a hyperplane results in a clear definition of the optimal separating hyperplane.
- We have to minimize the length of the normal vector w.
  - Formulation as Lagrange function
  - Formulation as a dual optimization problem
- The learning outcome is a linear combination of support vectors.
- We only have to calculate the scalar product with the examples.

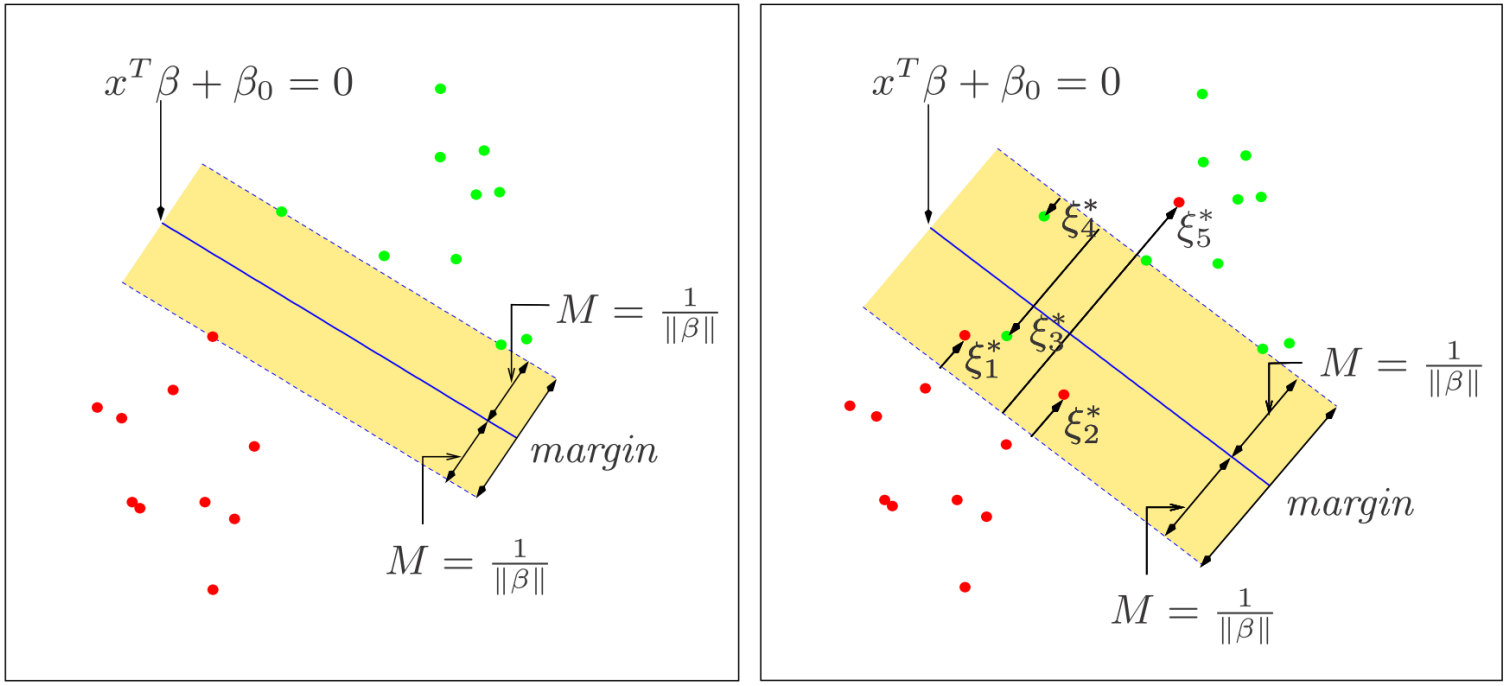


FIGURE 12.1. *Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width* $2M = 2/\|\beta\|$. *The right panel shows the nonseparable (overlap) case. The points labeled* ξ_j^* *are on the wrong side of their margin by an amount* $\xi_j^* = M\xi_j$; *points on the correct side have* $\xi_j^* = 0$. *The margin is maximized subject to a total budget* $\sum \xi_i \leq$ *constant. Hence* $\sum \xi_j^*$ *is the total distance of points on the wrong side of their margin.*

Hastie, T., Tibshirani, R., & Friedman, J. H. (2017). *The elements of statistical learning: Data mining, inference, and prediction* (12th ed). *Springer series in statistics*. New York: Springer.

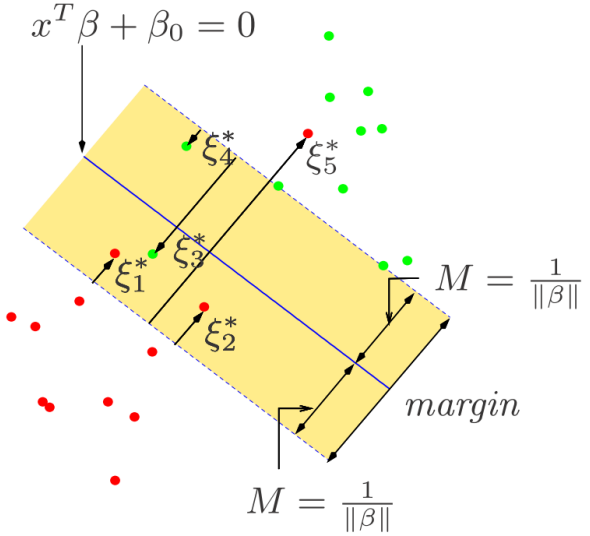
Relaxation

We allow for some points to be on the wrong side of the margin.

Define the slack variables $\xi = (\xi_1, \xi_2, \ldots, \xi_n)$

$$y_i(x_i^T \beta + \beta_0) \geq 1 - \xi_i$$

with $\xi_i \geq 0 \;\; and \; \sum_i \xi_i \leq Constant \; C$



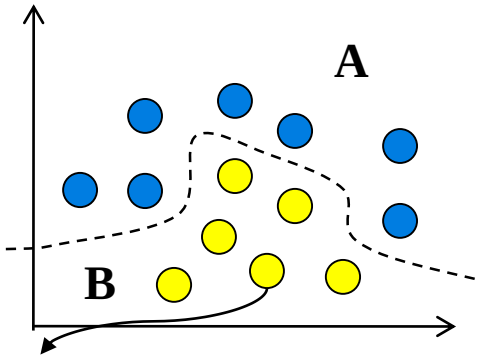
Optimization problem: $\frac{1}{2}\left\|\vec{\beta}\right\|^2 + C\sum_{i=1}^{n}\xi_i$

The solution for ß has the form $\beta = \sum_{i=1}^{N} \alpha_i y_i x_i$

Transformation into Feature Space H

Example: $\Phi(x, y) = x^2 + y^2$

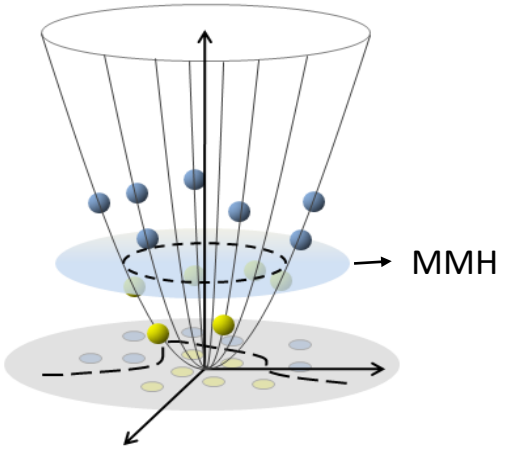
Input Space X



Kernel Function

$\Phi(x, y) = x^2 + y^2$

Feature Space H



Non linear separable in low dimensions

linear separable in higher dimensions

Kernel function

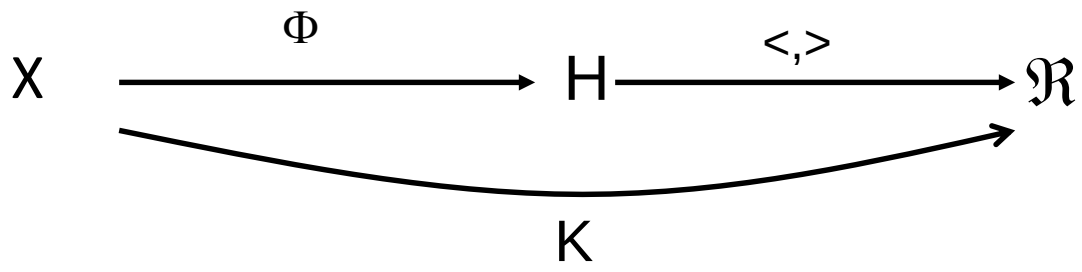
Recall:

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \left\langle \vec{x}_i, \vec{x}_j \right\rangle$$

$$f(\vec{x}) = \sum_{i=1}^{m} \alpha_i y_i \left\langle \vec{x}_i, \vec{x} \right\rangle + \beta_0$$

SVM depends on x only via scalar product $\left\langle \vec{x}, \overrightarrow{x'} \right\rangle$.

Replace transformation Φ and scalar product with kernel function $K\left(\overrightarrow{x}, \overrightarrow{x'}\right) = \left\langle \Phi(\overrightarrow{x}), \Phi(\overrightarrow{x'}) \right\rangle$, because transformation is very complex and computational.



Kernel Trick

The feature space is very large. Mapping the examples into the characteristic space and then calculating the scalar products between the transformed examples is very inefficient.

A core function that delivers the same result directly applied to the examples would be efficient!

$$K\left(\overrightarrow{x}, \overrightarrow{x'}\right) = \left\langle \Phi(\overrightarrow{x}), \Phi(\overrightarrow{x'}) \right\rangle$$

$$\Phi: X \rightarrow \mathcal{H}$$

The scalar product of the vectors in characteristic space $\mathcal{H}$ corresponds to the value of the core function above the examples.

What functions make $K\left(\overrightarrow{x}, \overrightarrow{x'}\right) = \left\langle \Phi(\overrightarrow{x}), \Phi(\overrightarrow{x'}) \right\rangle$ true?

Why is this a trick?

We don't need to know what the feature space really looks like.

We just need the kernel function as a measure of similarity.

Another black box:

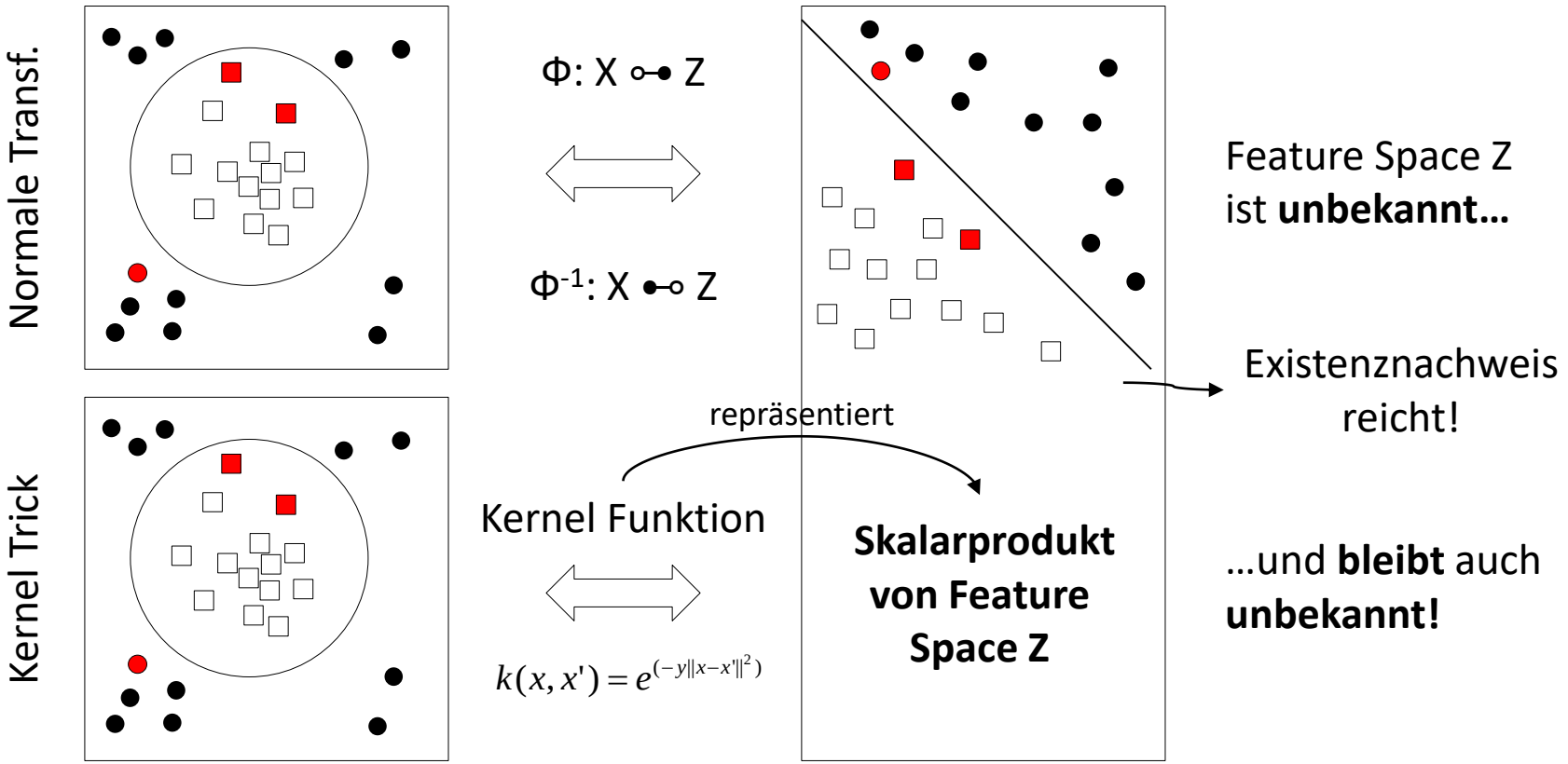
We don't know what happens in the kernel, we are only interested in the result.

Nevertheless, we have the geometric interpretation of the separating hyperplane:

SVMs are more transparent than artificial neural networks, for example.

Der Kernel-Trick

Transformation ohne zu transformieren



Summary: Kernel

- Kernel functions calculate the scalar product of the observations in a characteristic space without actually mapping it into the characteristic space. $k(x,x') = \Phi(x) * \Phi(x')$
- Polycore and RBF core as examples.
- The kernel trick: k(x,x') can only be calculated from x*x'.
- A function $X \times X$, which can be represented for all x_i in X as a positively definite gram matrix with symmetrical function k, is called kernel function.
- The mercer condition checks if the function is a kernel function, i.e. if the matrix is positive.

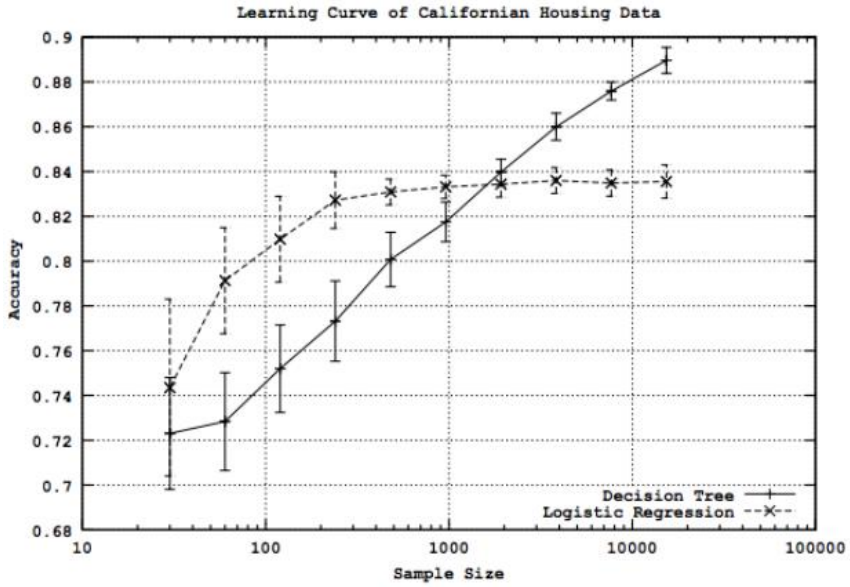
Model Evaluation

Prof. Dr. Stephan Trahasch

Offenburg University of Applied Sciences

Learning Curves

given training/test set partition
  for each sample size s on learning curve
   (optionally repeat n times)
   - randomly select s instances from training set
   - learn model
   - evaluate model on test set to determine accuracy a
   - plot(s, a) or (s, avg.accuracy and error bars)



Methods of Estimation

- Holdout
  - Reserve 2/3 for training and 1/3 for testing
- Stratification
  - oversampling vs undersampling
- Random subsampling
  - Repeated holdout
- Cross validation
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n
- Bootstrap
  - Sampling with replacement

Confusion Matrix for a binary classifier

Let model M be a classifier for 2 classes: {Yes, No}

Predicted Class \ True Class	Yes	No
Yes	TP = True Positive	FP = False Positive
No	FN = False Negative	TN = True Negative

$$\text{Accuracy} = \frac{TP + TN}{P + N}$$

$$\text{Error rate} = \frac{FP + FN}{P + N}$$

Sensitivity

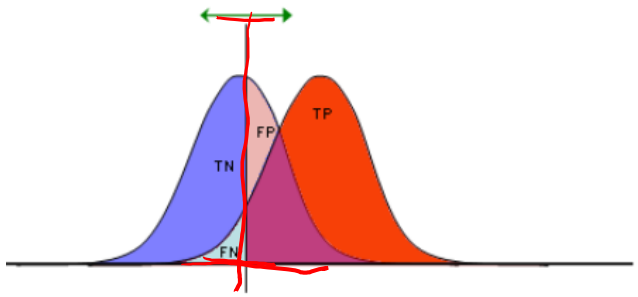
$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

Specificity $\frac{TN}{TN + FP}$

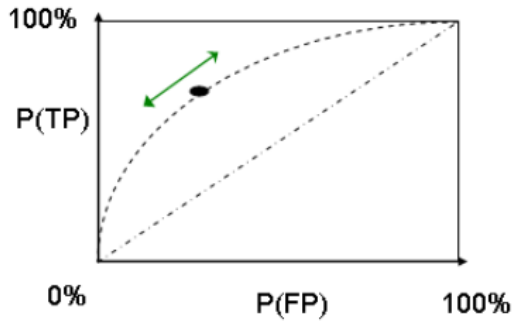
ROC Analysis – Receiver Operating Characteristic

Origin Signal theory: 2 signal sources.
To which source does
a received signal belong?



TP	FP
FN	TN
1	1

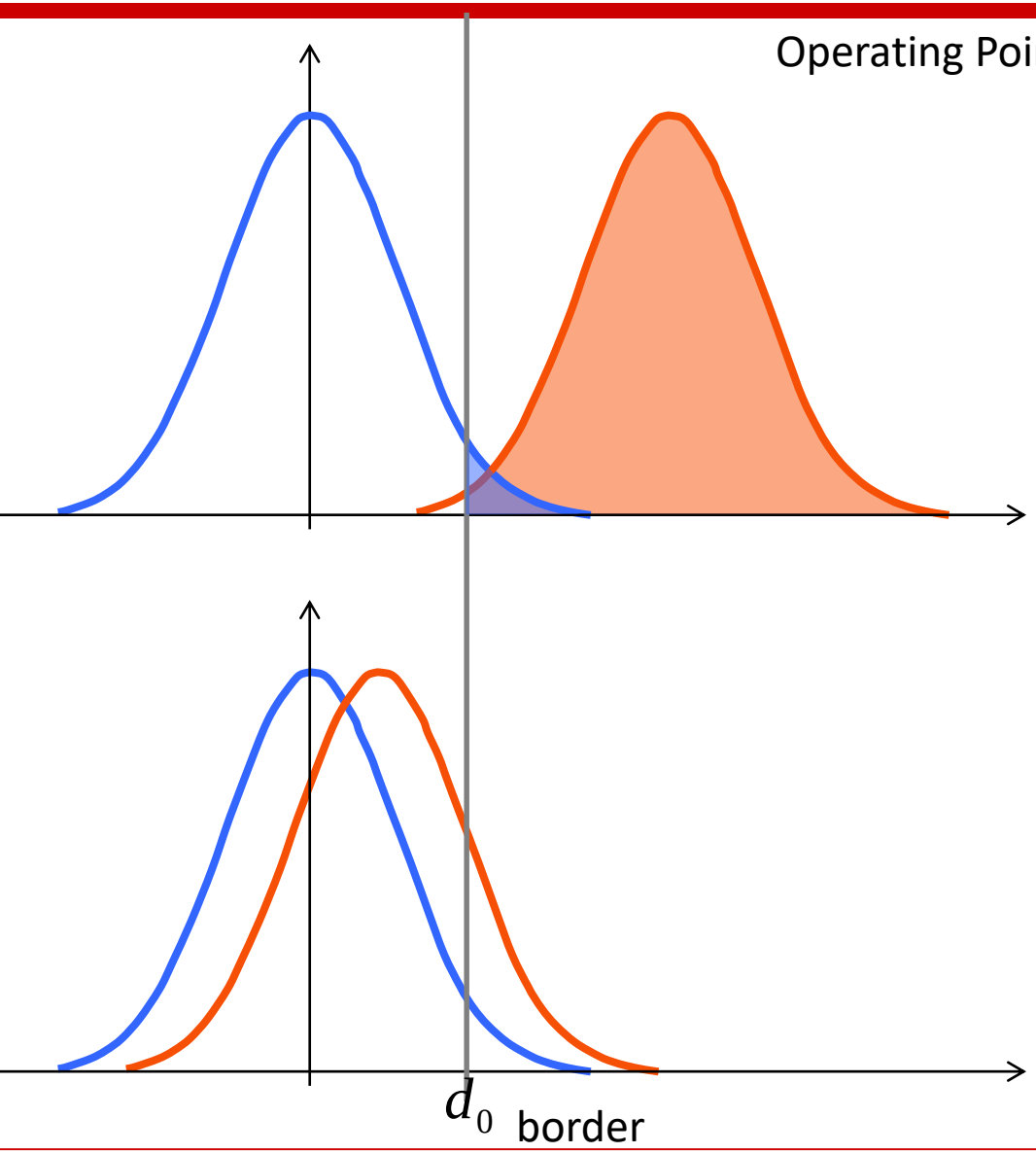
Objective:
What is the best method for
varying parameter values?

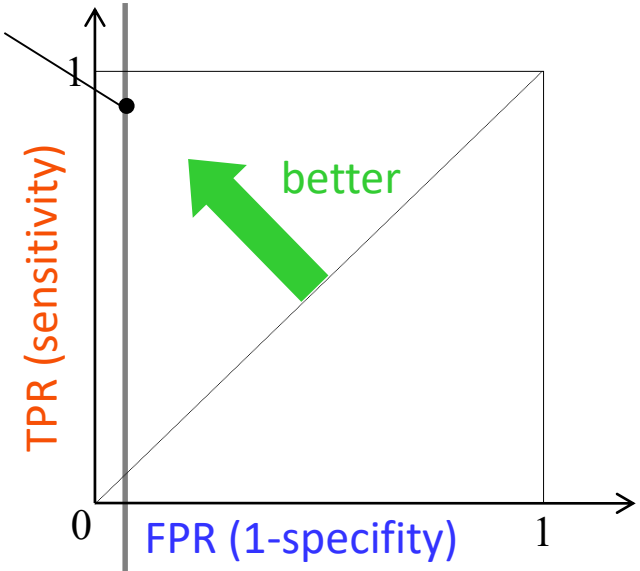


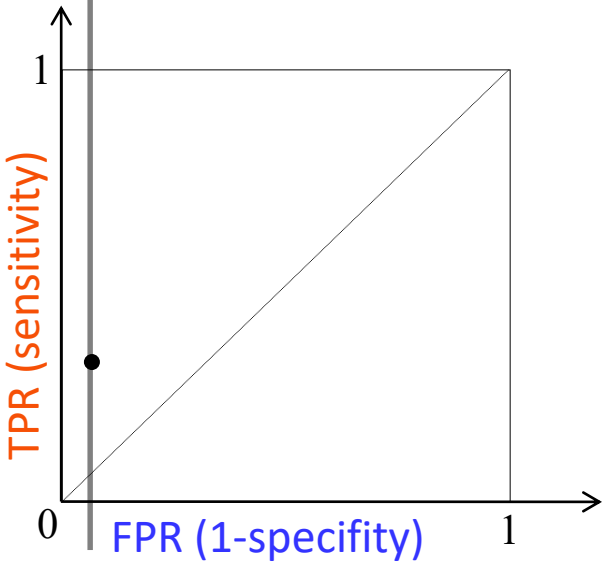
Method:

- Visualization as ROC curve
- fpr and tpr for each binary classifier
- x-axis: false positive rate fpr
- y-axis: true positive rate tpr

ROC Curve


Operating Point
d_0
border

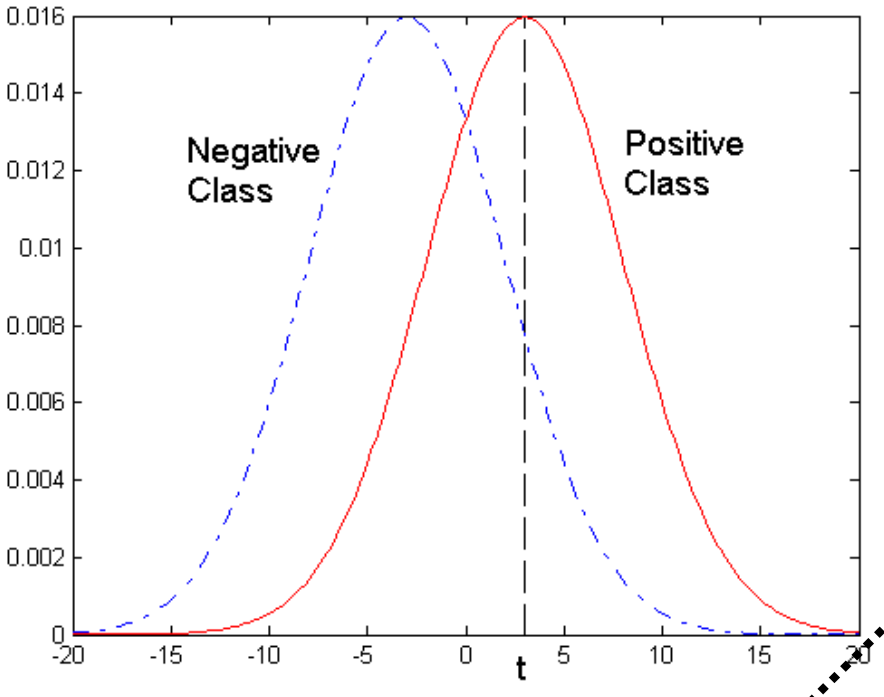

1
better
TPR (sensitivity)
0
FPR (1-specifity)
1

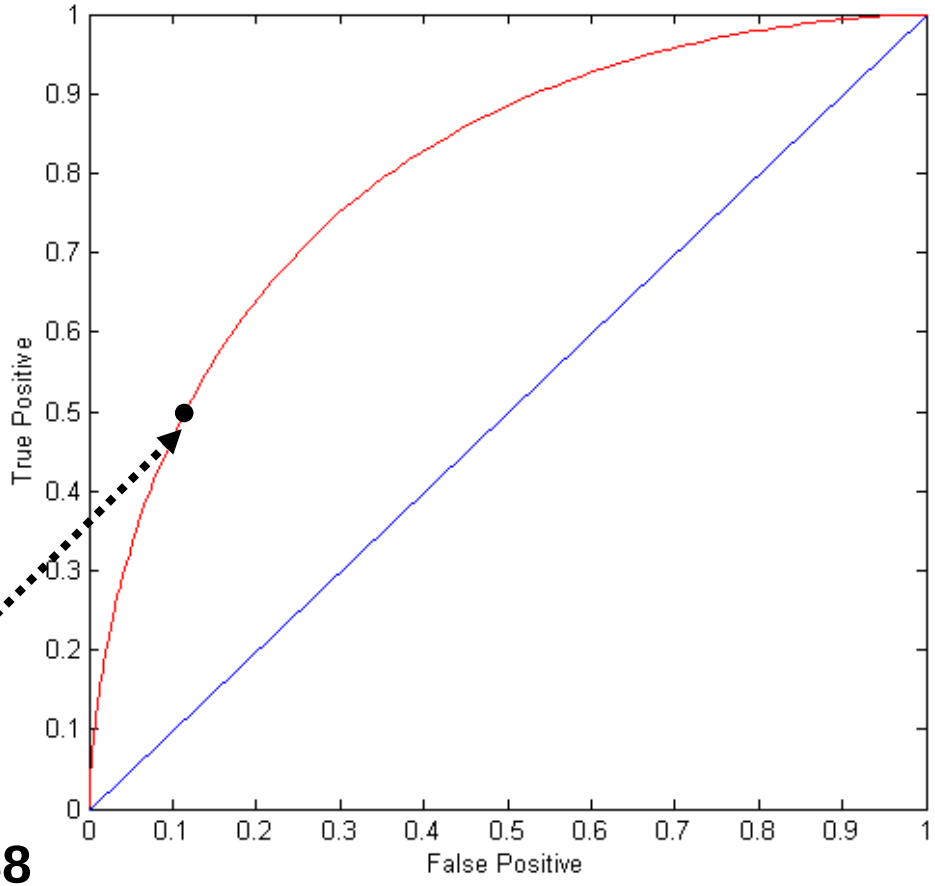

1
TPR (sensitivity)
0
FPR (1-specifity)
1

ROC Curve

1-dimensional data set containing 2 classes (positive and negative)

any points located at x > t is classified as positive

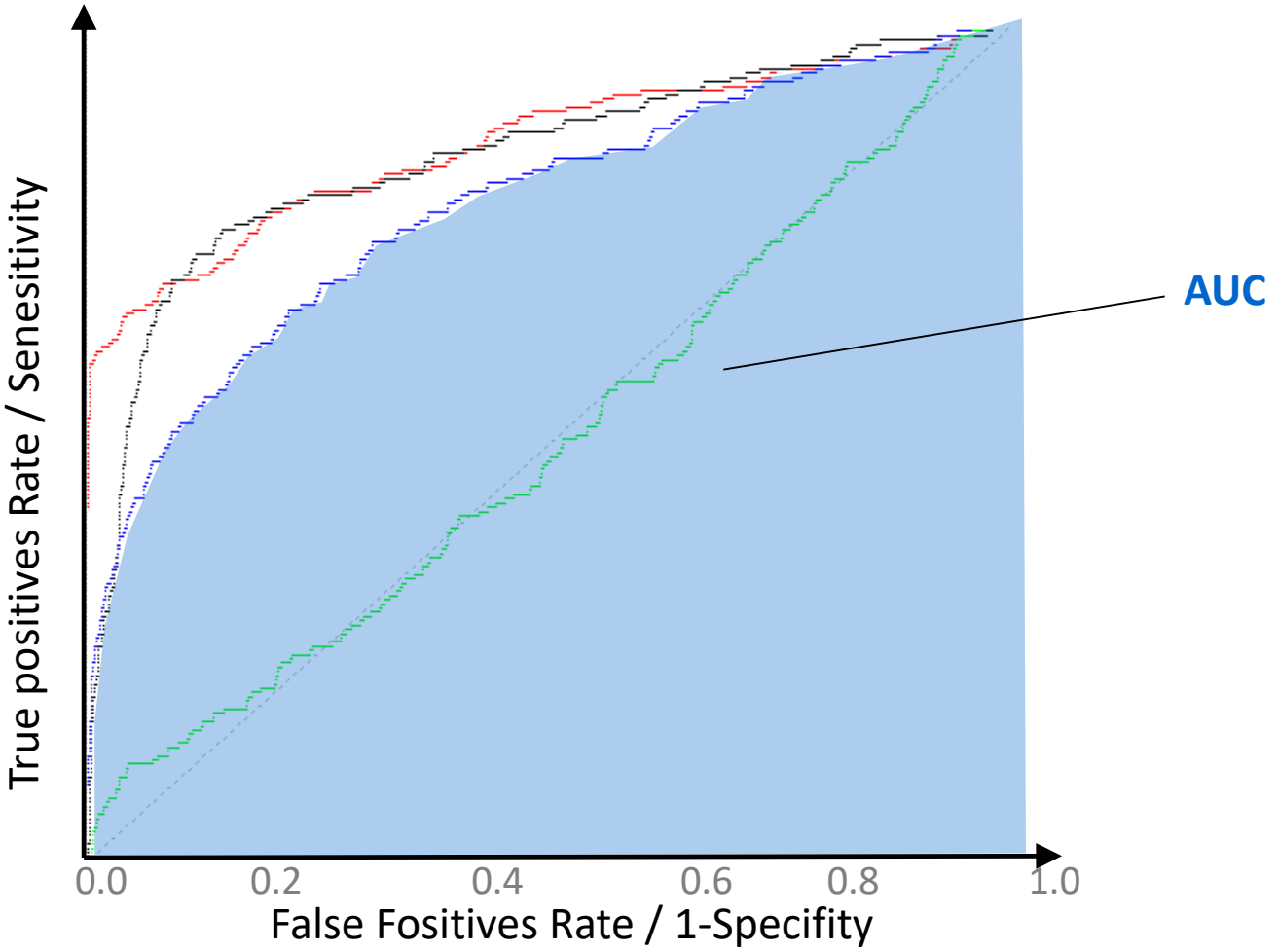




At threshold t:

TP=0.5, FN=0.5, FP=0.12, FN=0.88

ROC Curve



Integral (area under curve = AUC)

Perfect classiciation = 1 , random = 0.5

kNN, Curse of Dimensionality and PCA

Prof. Dr. Stephan Trahasch

Offenburg University of Applied Sciences

Main Idea of kNN

Find the k-nearest neighbors to a new data instance z

- rank the feature vectors according to Euclidean distance
- select the k vectors which are have smallest distance to z

Classification

- ranking yields k feature vectors and a set of k class labels
- pick the class label which is most common in this set ("voting")
- classify z as belonging to this class

"Training" is trivial: just use training data as a lookup table, and search to classify a new datum

→ "Lazy Learner"

Basic Considerations

- k = number of neighbours considered
- Decision set the set of k-next neighbours considered for classification
- Decision rule How do you determine the class of the object to be classified from the classes of the decision set?
- Distance function defines the similarity for pairs of objects

Normalization methods

- This odd prediction is caused by features taking different ranges of values, this is equivalent to features having different variances.
- We can adjust for this using normalization.

Normalization methods

z-transformation: $\forall x \in X: \quad z_i = \frac{x_i - \bar{x}}{s}$ bzw. $z_i = \frac{x_i - \mu}{\sigma}$

Normalization to interval [0, 1]: $\forall x \in X: \mathrm{x}' = \frac{x - \min(x)}{\max(x) - \min(x)}$

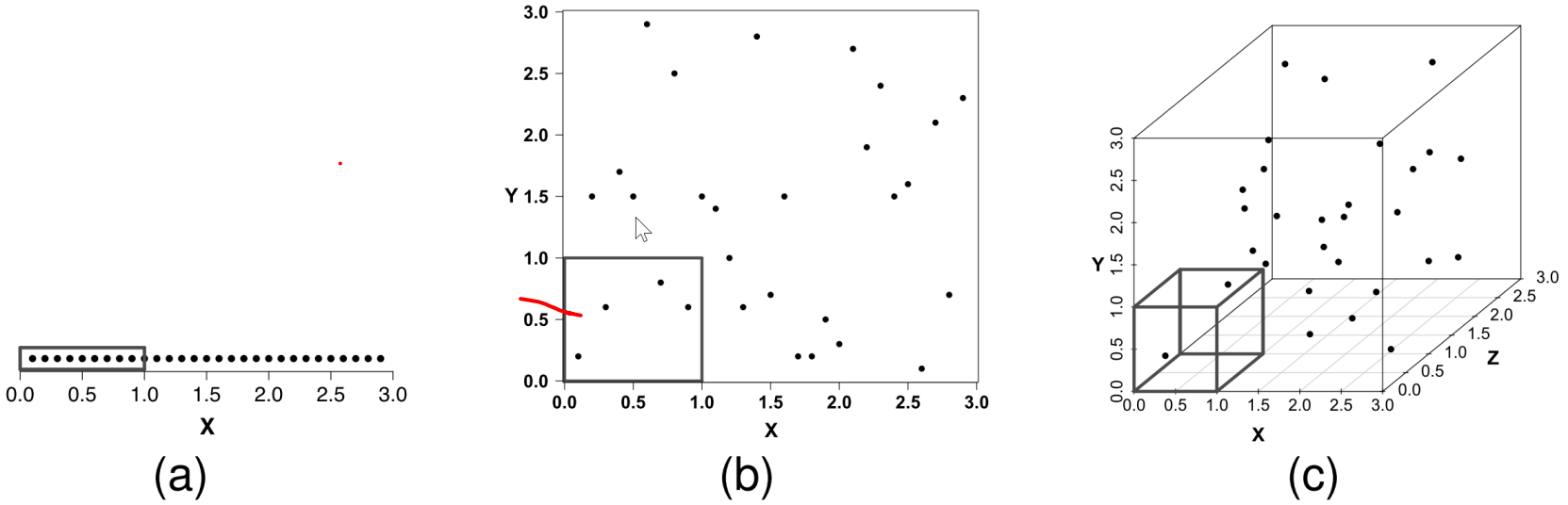
In general: $\mathrm{x}' = \frac{x - \min(x)}{\max(x) - \min(x)} * (\mathrm{high} - \mathrm{low}) + \mathrm{low}$

Curse of Dimensionality (Bellman, 1961)

10^2 data points in the unit interval have a distance of 0.01;

Equal distribution in a 10-dimensional unit cube with a distance of 0.01 requires 10^{2*10} data points!

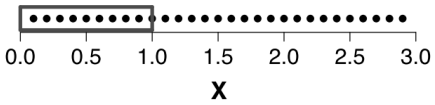
If 100 examples at p = 1 result in a dense space, you have to collect 100^{10} examples for the same density at p = 10.



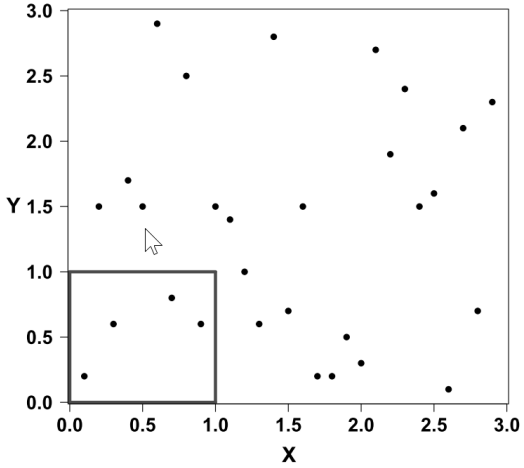
Across figures (a), (b) and (c) the density of the marked unit hypercubes decreases as the number of dimensions increases.

Source: Pattern recognition and machine learning

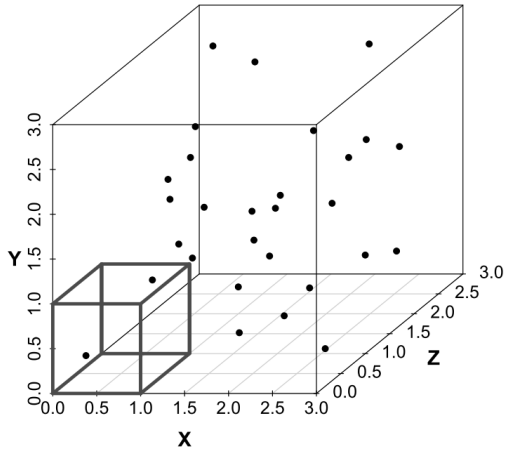
Curse of Dimensionality



(a)

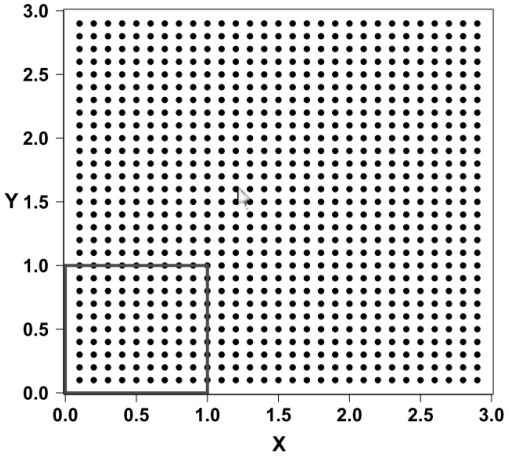


(b)

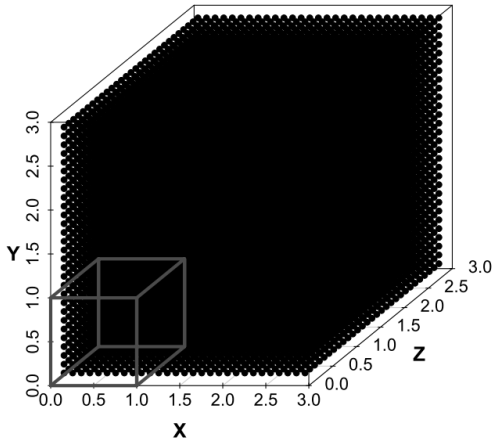


(c)

Figures (d) and (e) illustrate the cost we must incur if we wish to maintain the density of the instances in the feature space as the dimensionality of the feature space increases.



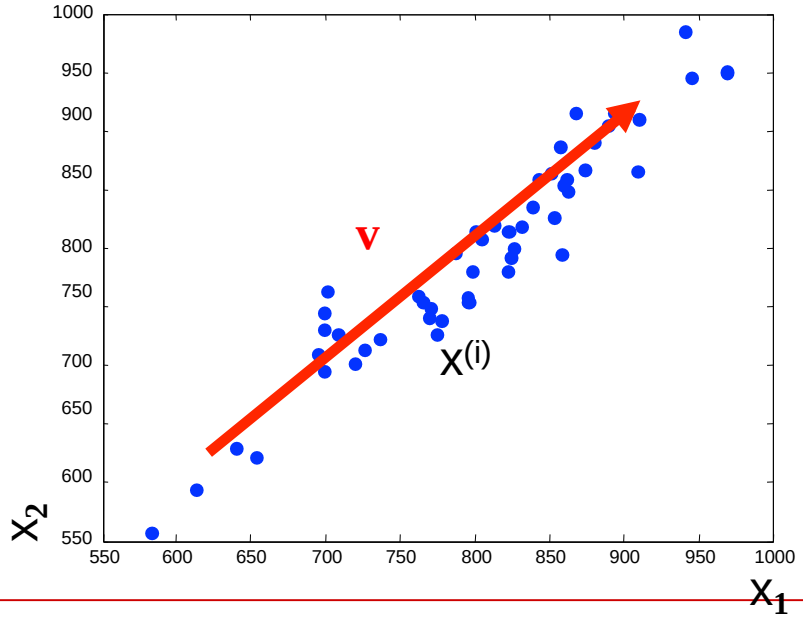
(d)

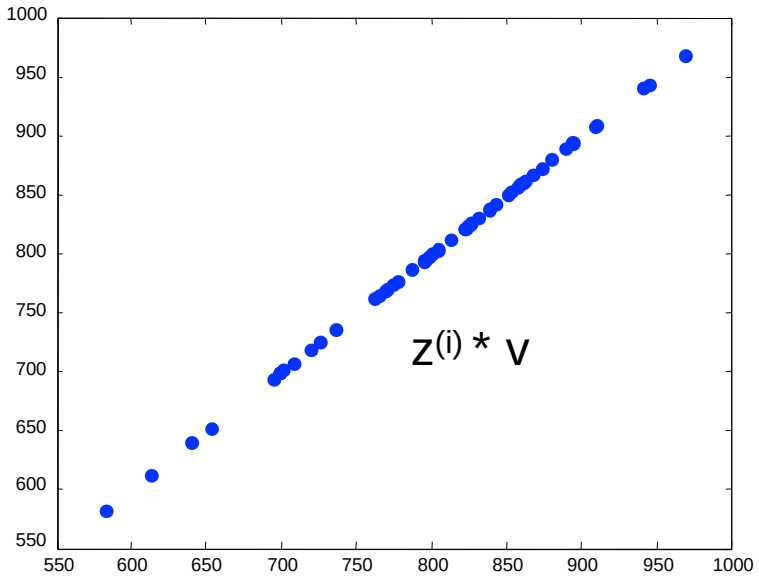


(e)

Dimensionality reduction

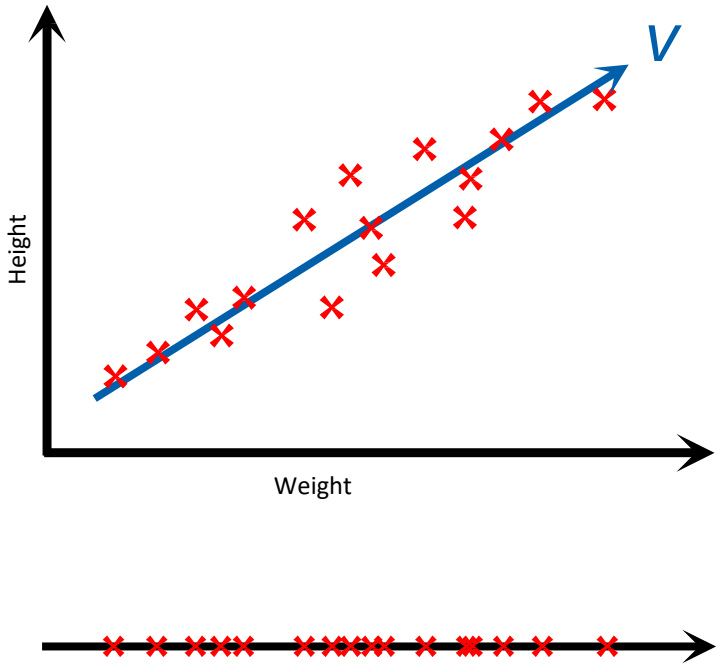
- Example: data with two real values (x_1, x_2)
- We'd like to describe each point using only one value z_1
- We'll communicate a "model" to convert: $(x_1, x_2) \sim f(z_1)$
  Example: linear function $f(z)$: $(x_1, x_2) = z * v * (v_1, v_2)$
- v is the same for all data points (communicate once)
- z tells us the closest point on v to the original point (x_1, x_2)





Dimensionality reduction

- PCA is most commonly used dimensionality reduction method
- projects the data on k orthogonal bases vectors v that minimize the projection error



Example:

- original 2D dataset containing features weight and height
- projection on vector v

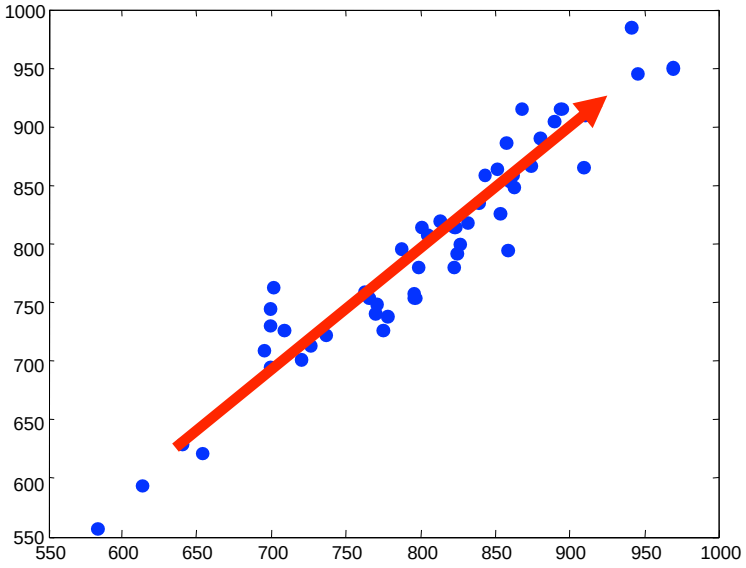
Principal Components Analysis

What is the vector that would most closely reconstruct X?

$$\min_{a,v} \sum_i \left(x^{(i)} - a^{(i)} \cdot v\right)^2$$

- Given v: $a^{(i)}$ is the projection of each point $x^{(i)}$ onto v
- v chosen to minimize the residual variance
- Equivalently, v is the direction of maximum variance
- Extensions to best two dimensions:

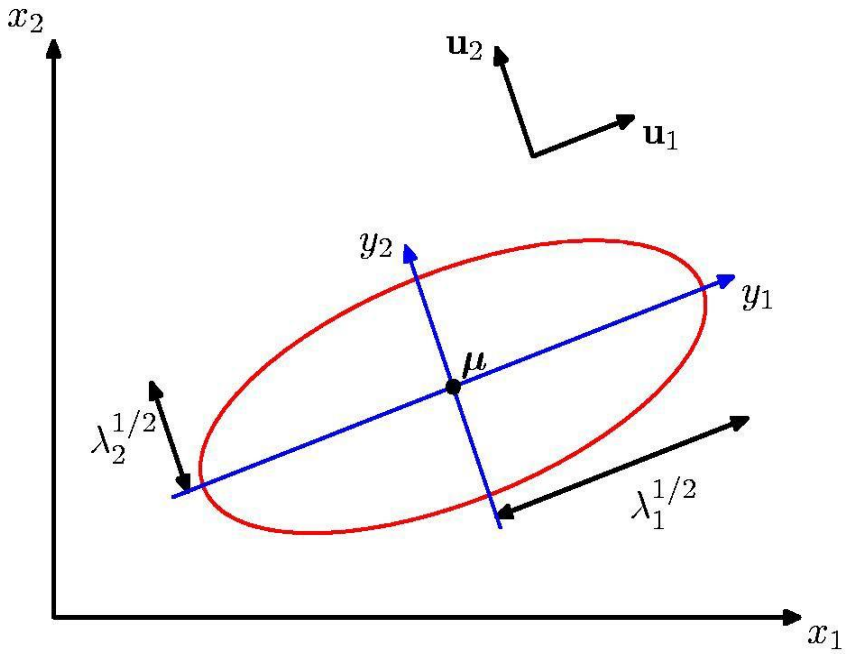
$$x_i = a_i v + b_i w + m$$



Geometry of Gaussian Distribution

$$\Delta^2 = (x - \mu)^T \Sigma^{-1} (x - \mu)$$

$$\Sigma = \mathrm{U} D U^T$$



Oval shows constant Δ^2 value…

Feature Selection

Sascha Niro, Prof. Dr. Stephan Trahasch

Offenburg University of Applied Sciences

Feature Selection goals

- Improve prediction performance (defy curse of dimensionality)
- Facilitate data visualization and data understanding
- Speed up model building process
- Improve understanding of the underlying process
- Reduce storage requirements

Feature Selection: Ranking/Filter Method

Algorithm:

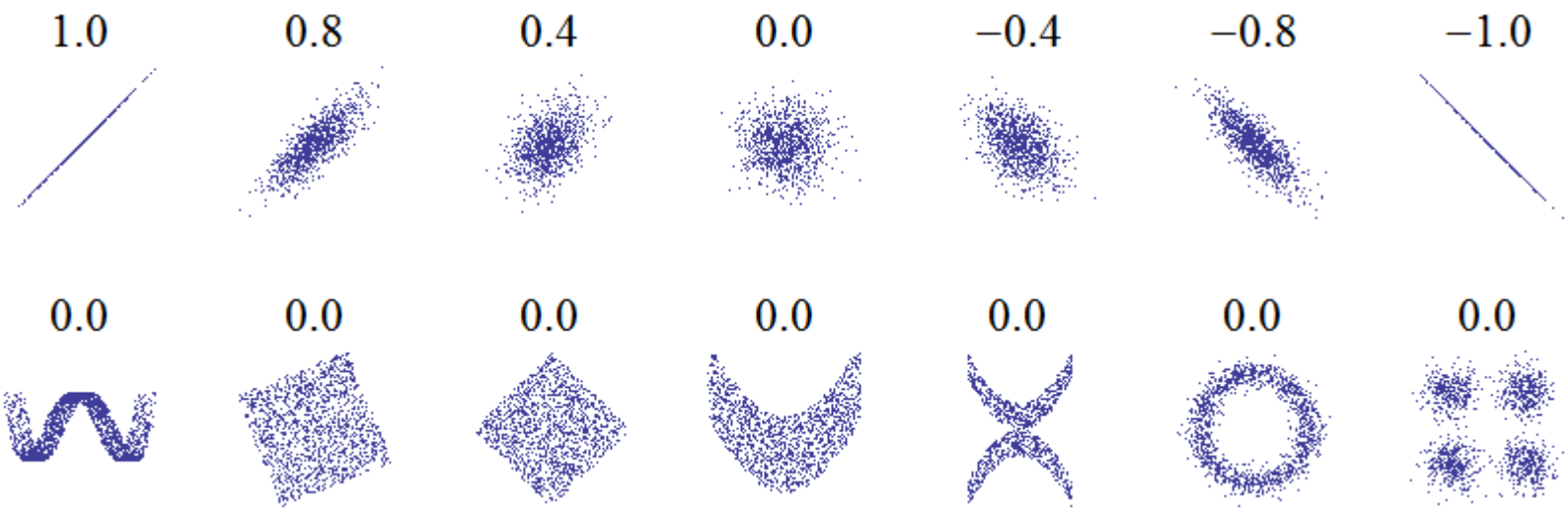
- Calculate the score of every feature x_i against the target variable y
- using a scoring criteria $S(i)$
- Sort the features by that score
- The rank is a used as a measure of variable importance

Rank	Feature	Score
1	B	10
2	A	5
3	C	4

Feature Selection: Ranking - Correlation

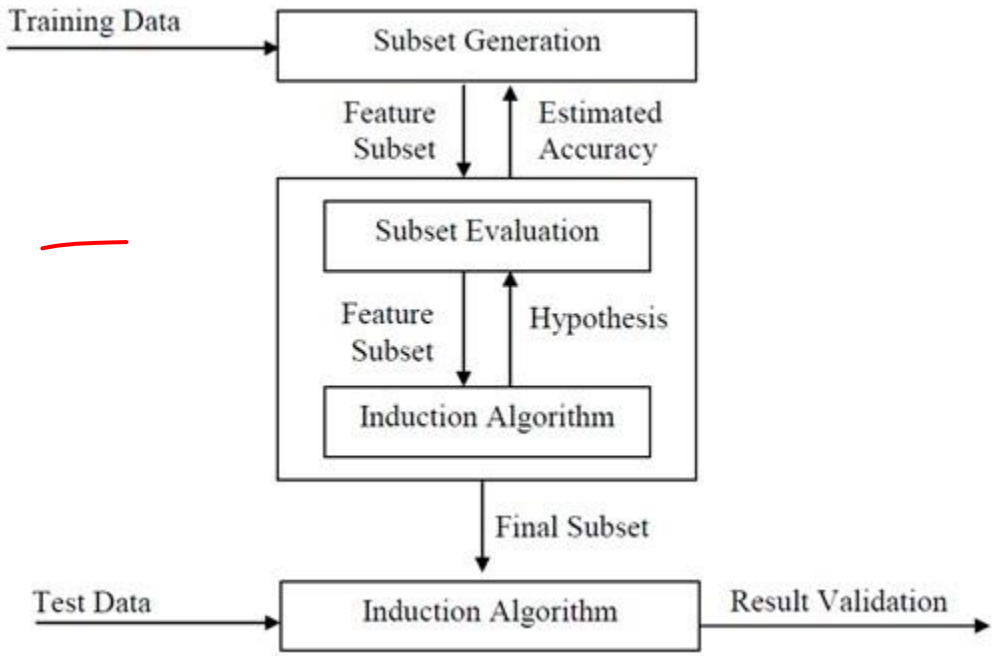
Criterion: Pearson correlation coefficient

$$Cor(x_i, y) = \frac{cov(x_i, y)}{\sqrt{var(x_i)\,var(y)}} = \frac{\sum_{i=1}^{m} (x_i - x)(y_i - y)}{\sqrt{\sum_{i=1}^{m} (x_i - x)^2}\sqrt{\sum_{i=1}^{m} (y_i - y)^2}}$$



Wrapper Method

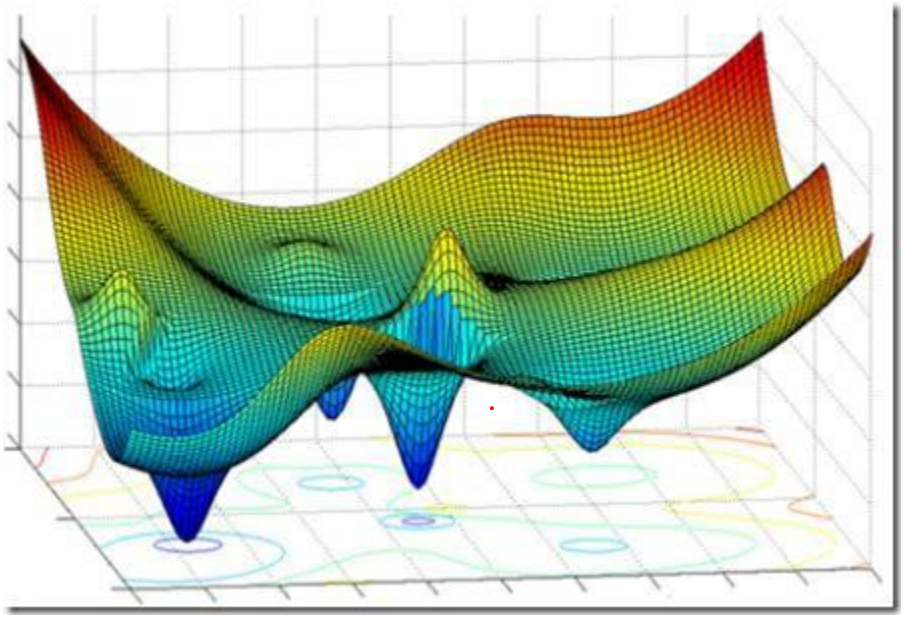
Use the prediction performance to assess the relative usefulness of subsets of variables while performing a search.



Wrapper Method

To be defined:

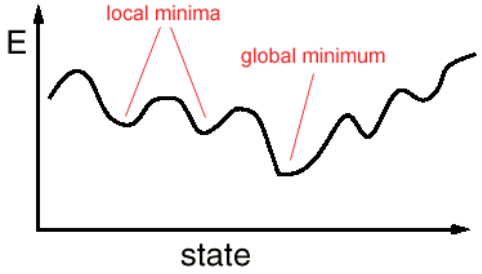
1. Search method
2. How to assess the prediction performance to guide the search and halt it
3. Which predictor to use

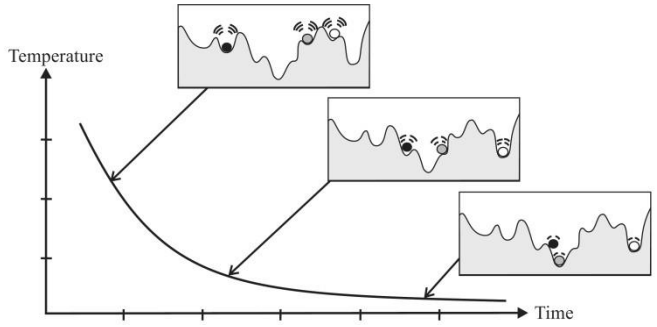


Wrapper Method – Search method

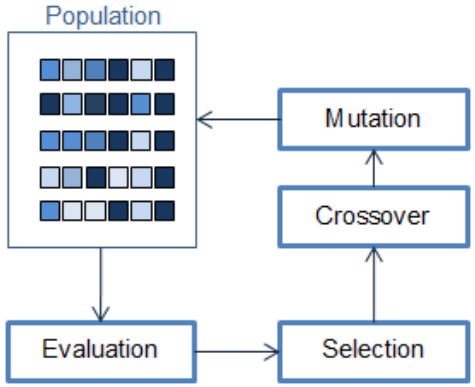
- Exhaustive Search (try all possible feature combinations)?

- Simulated Annealing





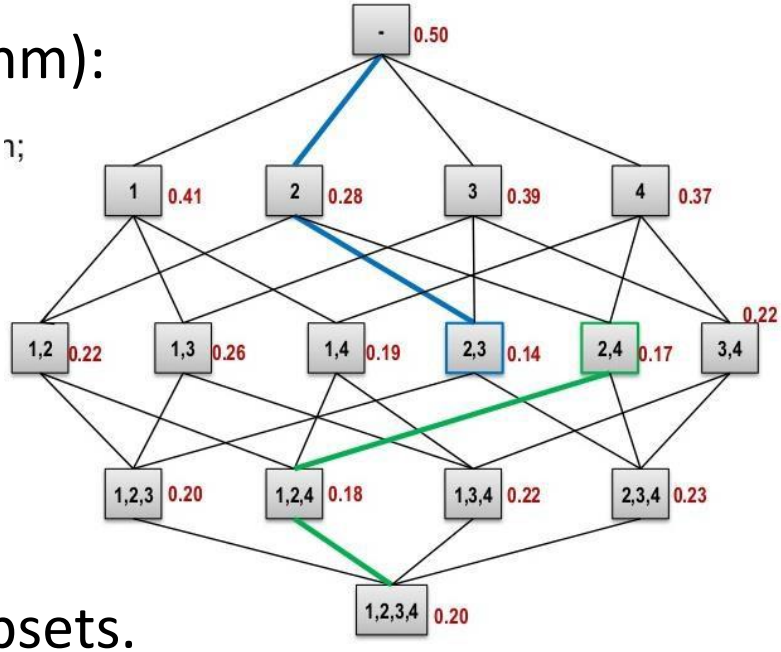
- Genetic algorithms



Wrapper Method – Search method

- Too intensive searching can lead to overfitting
- Simpler and more efficient strategies should be used such as greedy algorithms

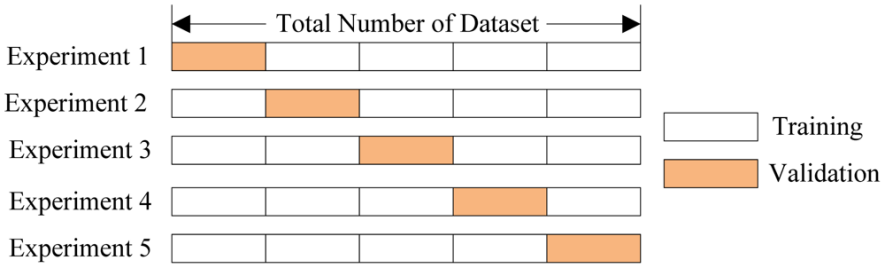
- Backward elemination (greedy algorithm):
  Starts with all features,
  least promising ones are
  progressively elimated.



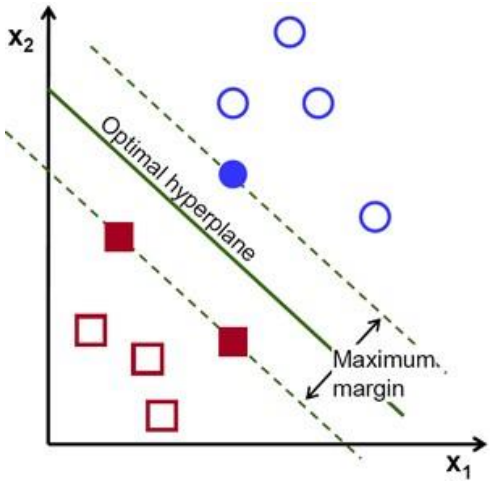
- Forward selection (greedy algorithm):
  Variables are progressively
  incorporated into larger and larger subsets.

Wrapper Method – Performance assessement, Predictor

- Performance Measure:
  - Accuracy on a validation set
  - Cross Validation



- Predictors:
  - Decision tree
  - Naive Bayes
  - Least-square linear predictors
  - Support Vector Machines



SVM

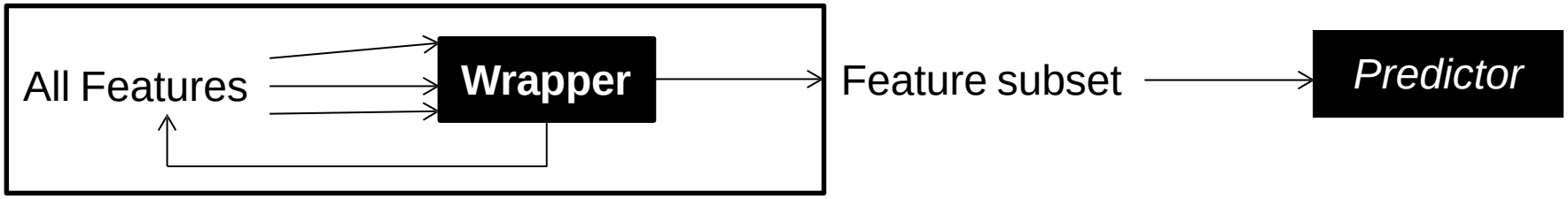
Wrapper Summary

Advantages

- Considers the relationships between variables
- Leads to good feature subsets

Disadvantages

- Computationally expensive
- Brute force method
- Danger of overfitting
- Variance of the feature subsets (Solution: Bootstrapping)



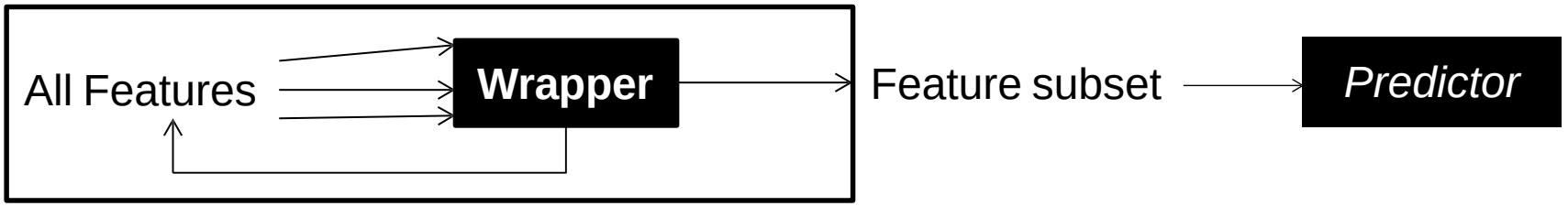
Embedded Method

- Feature selection is part of the learning algorithm
- Examples:
  - Decision trees (CART)
  - Random Forest
- Universal and simple approach
- Makes better use of the available data
- More efficient than Wrapper methods

Feature Selection: Summary

- Filter/Ranking:
  Computationally efficient, leads to redundant feature subsets

- Wrapper:
  Computationally more demanding but takes relationship between features into consideration and leads to non redundant feature subsets



- Embedded:

Applied Time Series Analysis
Time Series in R and Plots

Prof. Dr. Stephan Trahasch

Source: OTexts.org/fpp2/.

Introduction: What is a Time Series?

- A time series is a set of observations x_t , where each of the observations was made at a specific time t.
- the set of times T is discrete and finite
- observations were made at fixed time intervals
- continuous and irregularly spaced time series are not covered

Rationale behind time series analysis:

The rationale in time series analysis is to understand the past of a series, and to be able to predict the future well.

Time Series Patterns

- **Trend**

  pattern exists when there is a long-term increase or decrease in the data.
- **Seasonal**

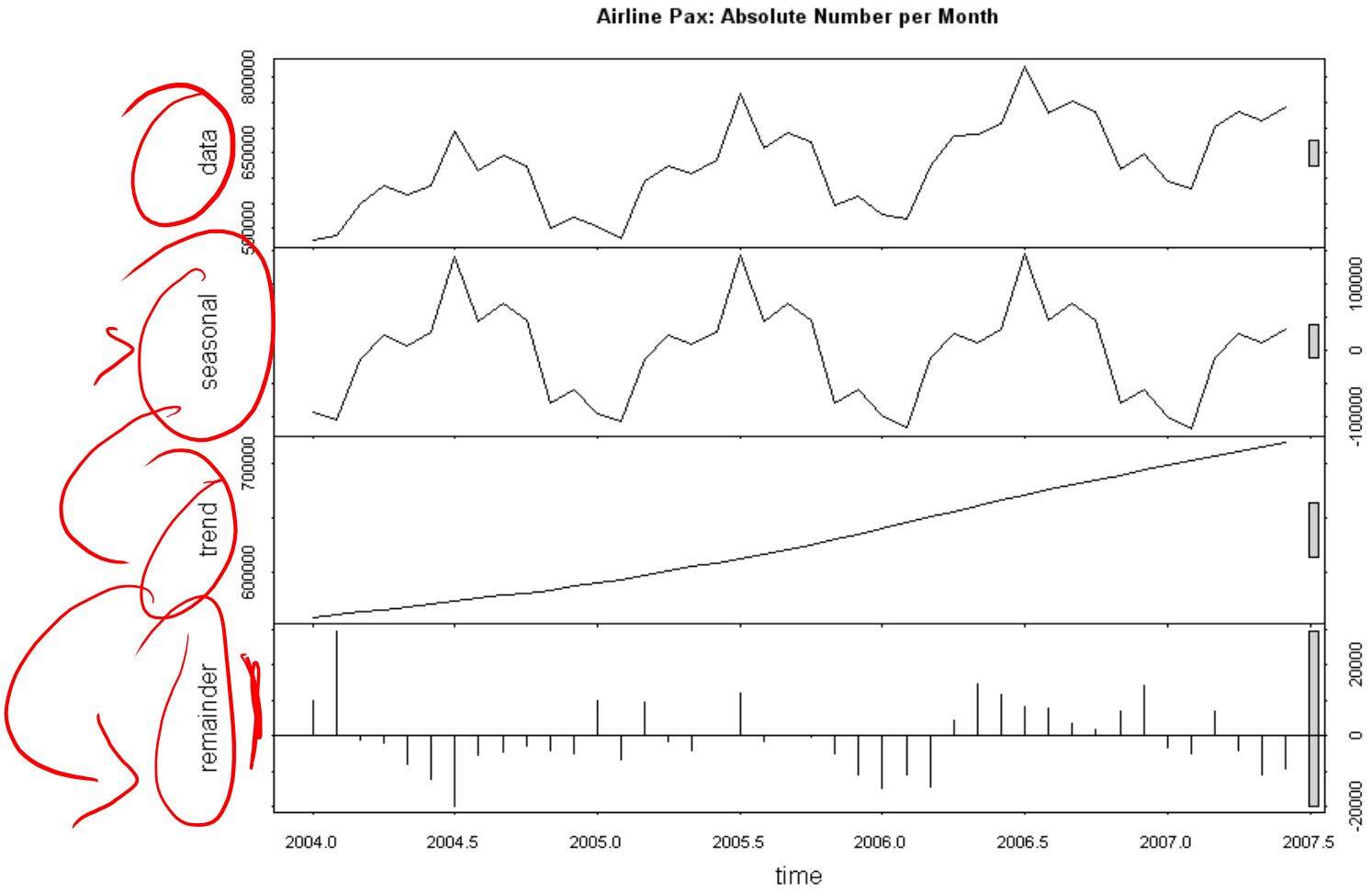
  pattern exists when a series is influenced by seasonal factors (e.g., the quarter of the year, the month, or day of the week).
- **Cyclic**

  pattern exists when data exhibit rises and falls that are (duration usually of at least 2 years).

Seasonal or cyclic?

- seasonal pattern constant length;
  cyclic pattern variable length
- average length of cycle longer than length of seasonal pattern
- magnitude of cycle more variable than magnitude of seasonal pattern

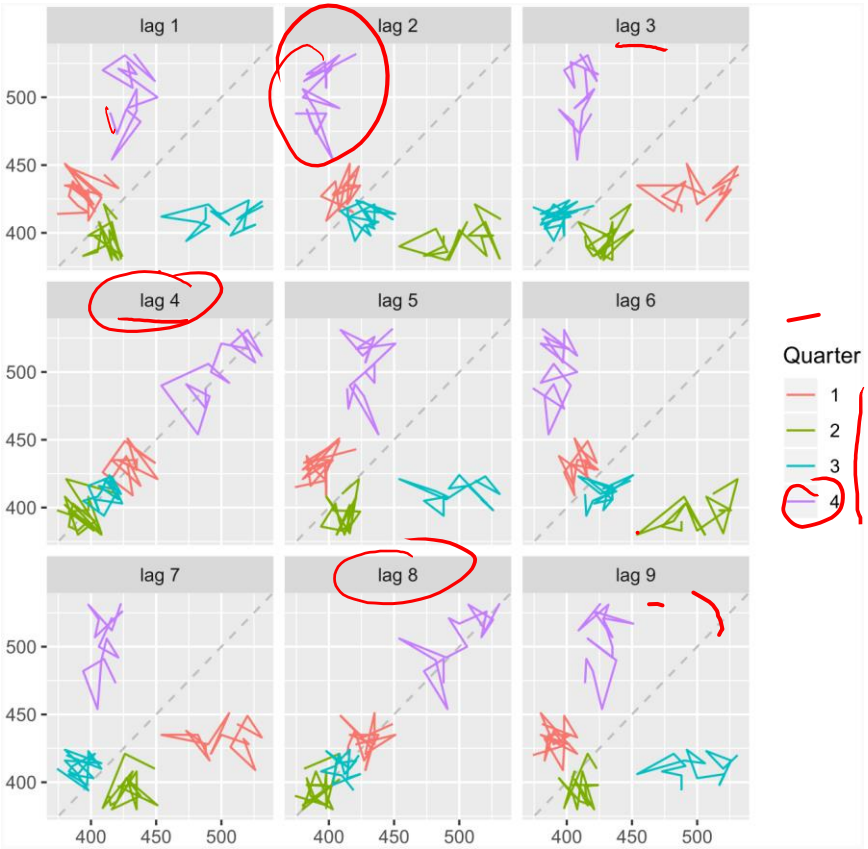
The timing of peaks and troughs is predictable with seasonal data, but unpredictable in the long term with cyclic data.


Airline Pax: Absolute Number per Month
data
seasonal
trend
remainder
500000
600000
650000
700000
800000
-100000
0
100000
-20000
0
20000
2004.0
2004.5
2005.0
2005.5
2006.0
2006.5
2007.0
2007.5
time

Lag plots and autocorrelation

Example: Beer production

```
beer <- window(ausbeer, start=1992)
gglagplot(beer)
```



Lagged scatterplots

Each graph shows y_t plotted against y_{t-k} for different values of k.

The autocorrelations are the correlations associated with these scatterplots.

Autocorrelation

Covariance and **correlation**:

measure extent of **linear relationship** between two variables (y and x).

Autocovariance and **autocorrelation**:

measure linear relationship between **lagged values** of a time series y

We measure the relationship between:

- y_t and y_{t-1}
- y_t and y_{t-2}
- y_t and y_{t-3}
- etc.

Autocorrelation

We denote the **sample autocovariance** at lag k by c_k.

$$c_k = \frac{1}{T} \sum_{t=k+1}^{T} (y_t - \bar{y})(y_{t-k} - \bar{y})$$

and the **sample autocorrelation** at lag k by r_k.

$$r_k = \frac{c_k}{c_0}$$

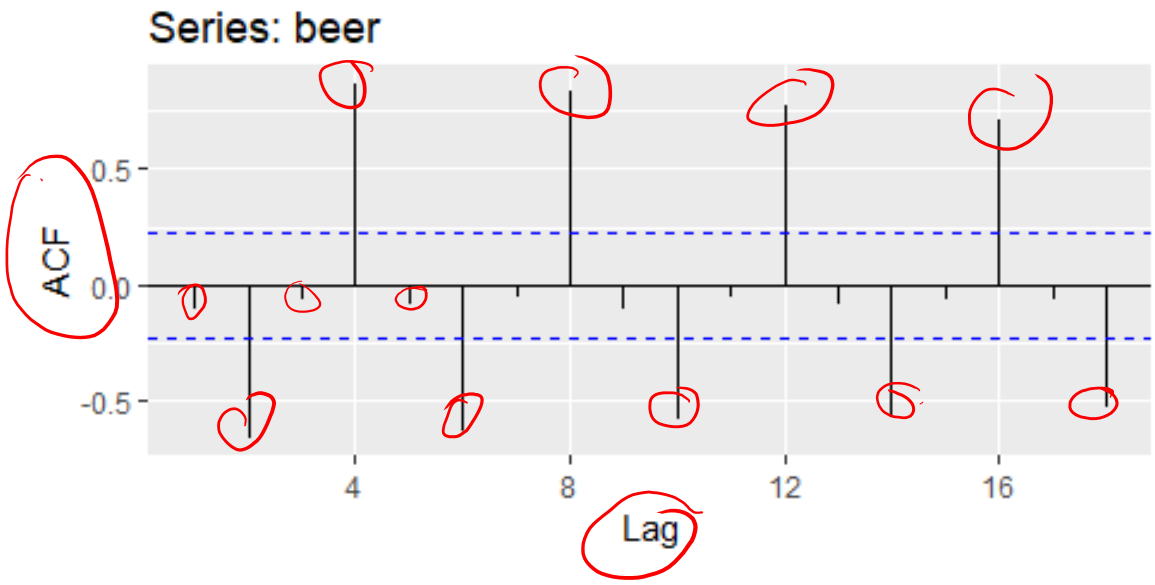
- r_1 indicates how successive values of y relate to each other
- r_2 indicates how y values two periods apart relate to each other
- r_k is the same as the sample correlation between y_t and y_{t-k}.

Autocorrelation

Results for first 9 lags for beer data:

r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9
-0.102	-0.657	-0.060	0.869	-0.089	-0.635	-0.054	0.832	-0.108

```
ggAcf(beer)
```



Autocorrelation

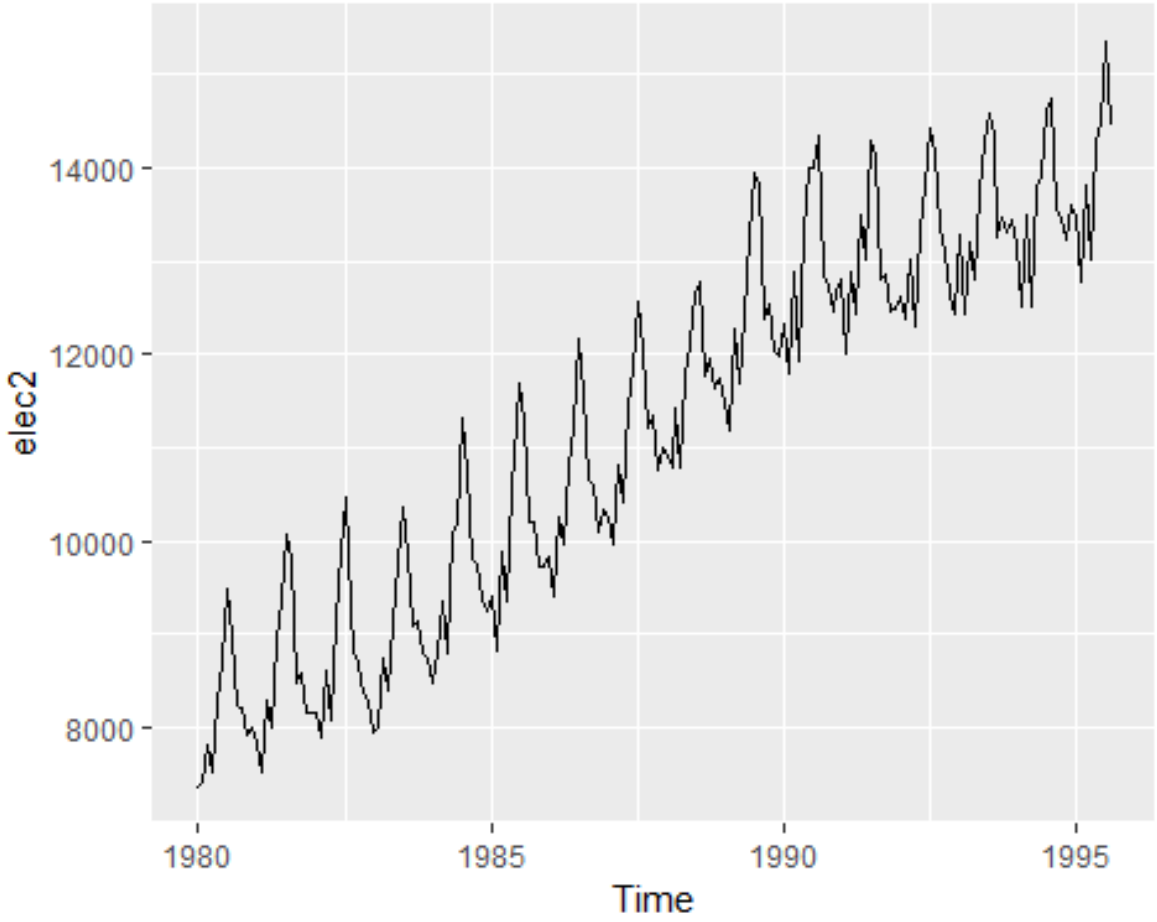
- r_4 higher than for the other lags. This is due to **the seasonal pattern in the data**: the peaks tend to be **4 quarters** apart and the troughs tend to be **2 quarters** apart.
- r_2 is more negative than for the other lags because troughs tend to be 2 quarters behind peaks.
- Together, the autocorrelations at lags 1, 2,… , make up the or ACF.
- The plot is known as a **correlogram**

Trend and seasonality in ACF plots

- When data have a trend, the autocorrelations for small lags tend to be large and positive.
- When data are seasonal, the autocorrelations will be larger at the seasonal lags (i.e., at multiples of the seasonal frequency)
- When data are trended and seasonal, you see a combination of these effects.

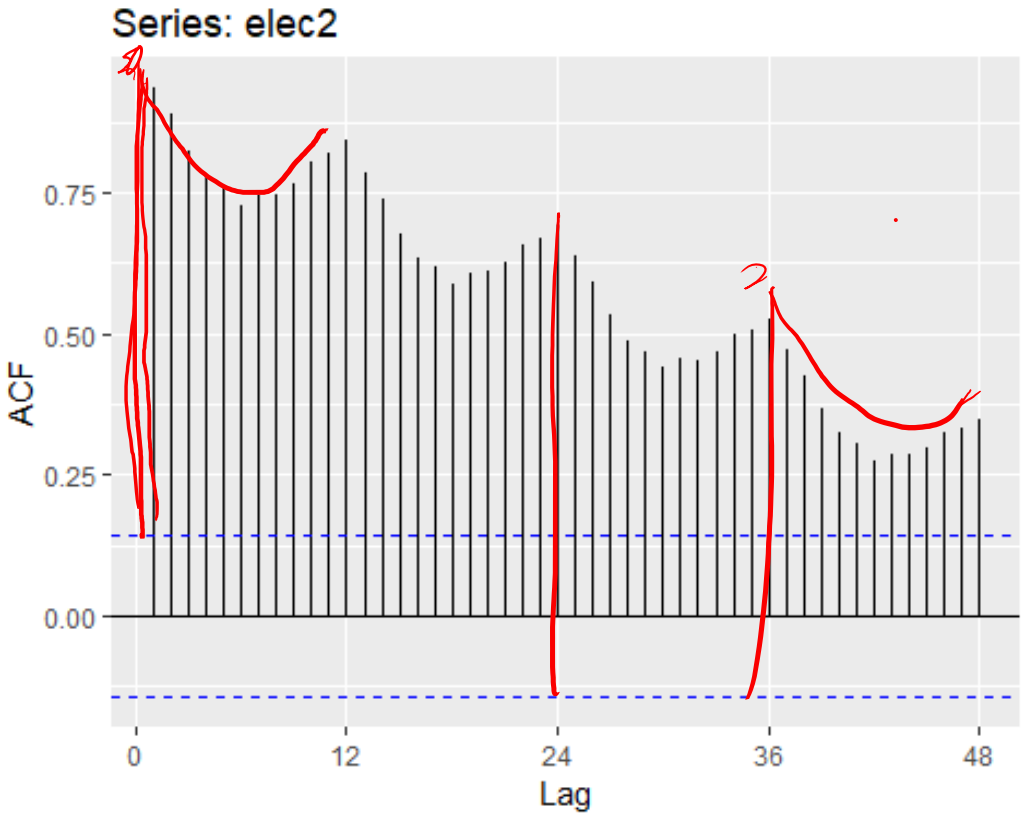
Aus monthly electricity production

```
elec2 <- window(elec, start=1980)
autoplot(elec2)
```



Aus monthly electricity production

```
ggAcf(elec2, lag.max=48)
```



Time plot shows clear trend and seasonality.
The same features are reflected in the ACF.
The slowly decaying ACF indicates trend.
The ACF peaks at lags 12, 24, 36, , indicate seasonality of length 12.

Example: Pigs slaughtered

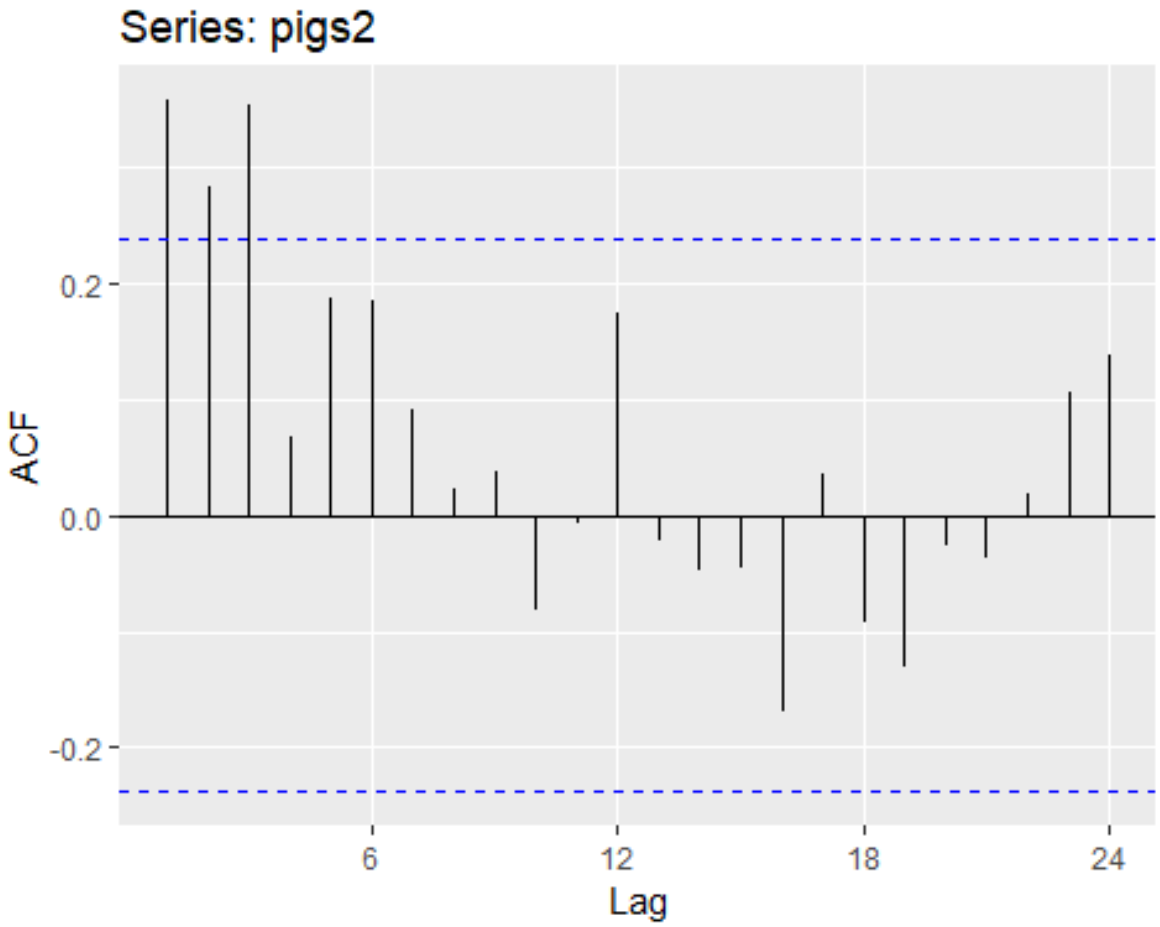
Monthly total number of pigs slaughtered in the state of Victoria, Australia, from January 1990 through August 1995. (Source: Australian Bureau of Statistics.)

- Difficult to detect pattern in time plot.
- ACF shows some significant autocorrelation at lags 1, 2, and 3.
- r_{12} relatively large although not significant. This may indicate some slight seasonality.

These show the series is **not a white noise series**.

Example: Pigs slaughtered

```
ggAcf(pigs2)
```



Applied Time Series Analysis
ARIMA

Prof. Dr. Stephan Trahasch

Offenburg University of Applied Sciences

Stochastic Model for Time Series

A time series process is a set $\{X_t, t \in T\}$ of random variables, where T is the set of times. Each of the random variables $X_t, t \in T$ has a univariate probability distribution F_t.

$P[X_1 \leq x_1, \dots, X_t \leq x_t]$ for all t and $x_1, \dots, x_t$

- If we exclusively consider time series processes with equidistant time intervals, we can enumerate $\{T = 1,2,3, \dots\}$
- An observed time series is a realization of $X = (X_1, \dots, X_n)$ and is denoted with small letters as $\mathrm{x} = (x_1, \dots, x_n)$.
- We have a multivariate distribution, but only 1 observation (i.e. 1 realization from this distribution) is available. In order to perform "statistics", we require some additional structure.

Strict Stationarity

For being able to do statistics with time series, we require that the series "doesn't change its probabilistic character" over time. Shifting the time axis does not affect the distribution.

A time series $\{X_t, t \in T\}$ is **strictly stationary**, if the joint distribution of the random vector $\{X_t, \dots, X_{t+h}\}$ is equal to the one of $\{X_s, \dots, X_{s+h}\}$ for all combinations of t, s and h.

$X_t \sim F$	all X_t are identically distributed
$E[X_t] = \mu$	all X_t have identical expected value
$Var(X_t) = \sigma^2$	all X_t have identical variance
$Cov(X_t, X_{t+h}) = \gamma_h$	the autocov depends only on the lag h

A stationary series is:
roughly horizontal, constant variance, no patterns predictable in the long-term

Stationarity

It is impossible to „prove“ the theoretical concept of stationarity from data. We can only search for evidence in favor or against it.

However, with strict stationarity, even finding evidence only is too difficult. We thus resort to the concept of weak stationarity.

A time series $\{X_t, t \in T\}$ is said to be **weakly stationary**, if

$$E[X_t, t \in T] = \mu$$

$$Cov(X_t, X_{t+h}) = \gamma_h \text{ for all lags h}$$

$$Var(X_t) = \sigma^2.$$

Note that weak stationarity is sufficient for „practical purposes“.

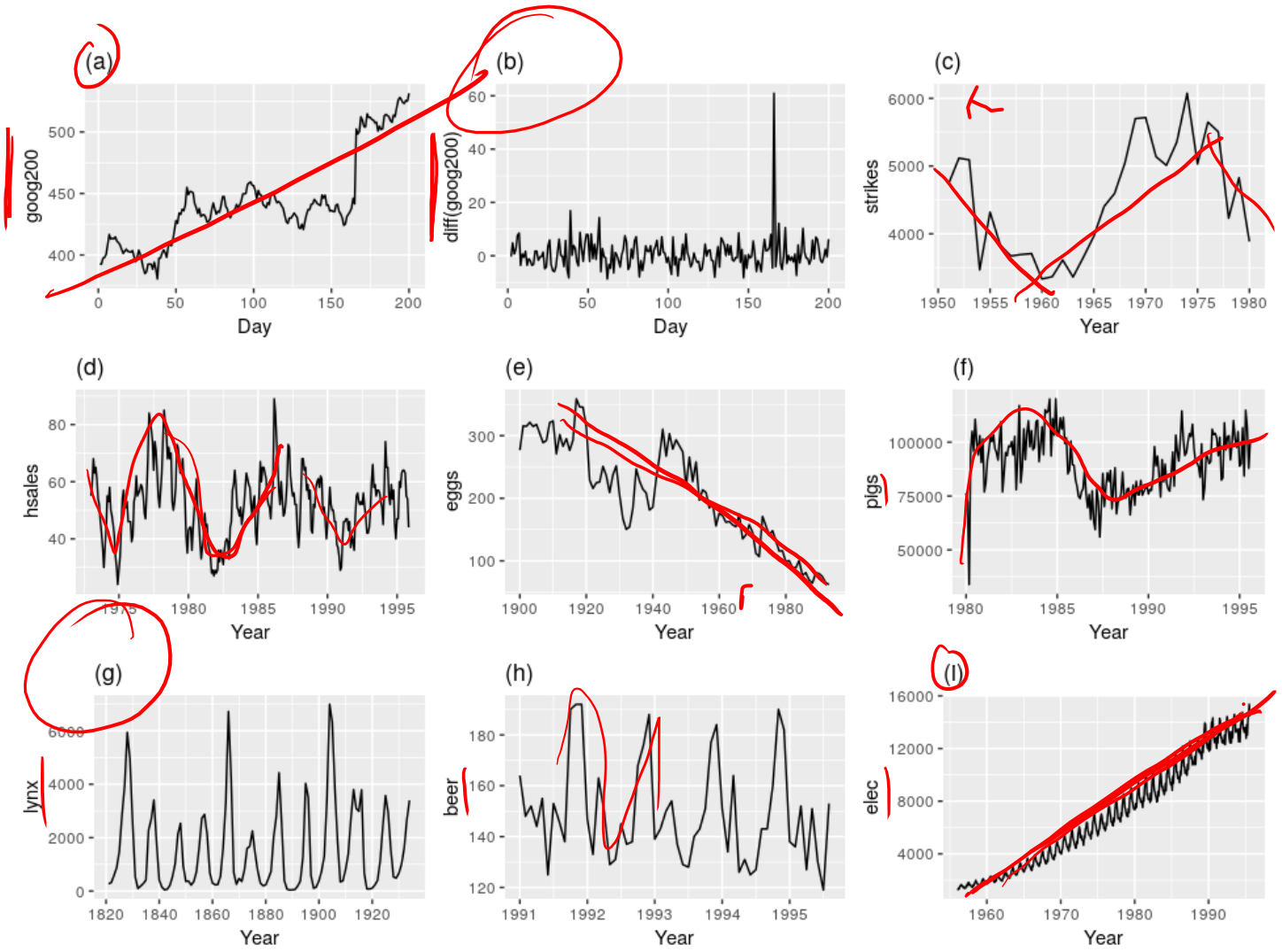
Evidence for Non-Stationarity

- Trend, i.e. non-constant expected value
- Seasonality, i.e. deterministic, periodical oscillations
- Non-constant variance, i.e. multiplicative error
- Non-constant dependency structure

Remark:

Note that some periodical oscillations, as for example in the lynx trappings data, can be stochastic and thus, the underlying process is assumed to be stationary. However, the boundary between the two is fuzzy.

Stationarity?



Stationary

If $\{y_t\}$ is a stationary time series, then for all s, the distribution of $(y_t, \dots, y_{t+s})$ does not depend on t.

Transformations help to stabilize the variance.

For ARIMA modelling, we also need to stabilize the mean.

Strategies for Detecting Non-Stationarity

- Time series plot
  - non-constant expected value (trend/seasonal effect)
  - changes in the dependency structure
  - non-constant variance
- Correlogram
  - non-constant expected value (trend/seasonal effect)
  - changes in the dependency structure

A (sometimes) useful trick, especially when working with the correlogram, is to split up the series in two or more parts, and producing plots for each of the pieces separately.

Non-stationarity in the mean

Identifying non-stationary series

- Time plot
- The ACF of stationary data drops to zero relatively quickly
- The ACF of non-stationary data decreases slowly.
- For non-stationary data, the value of r_1 is often large and positive.

Differencing

- Differencing helps to stabilize the mean.
- The differenced series is the between each observation in the original series: $y'_t = y_t - y_{t-1}$.
- The differenced series will have only $T - 1$ values since it is not possible to calculate a difference y_1' for the first observation.

Second-order differencing

Occasionally the differenced data will not appear stationary and it may be necessary to difference the data a second time:

$$\begin{aligned} y_t'' &= y_t' - y_{t-1}' \\ &= (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) \\ &= y_t - 2y_{t-1} + y_{t-2} \end{aligned}$$

- y_t'' will have $T - 2$ values.
- In practice, it is almost never necessary to go beyond second-order differences.

Seasonal differencing

A seasonal difference is the difference between an observation and the corresponding observation from the previous year.

$$y'_t = y_t - y_{t-m}$$

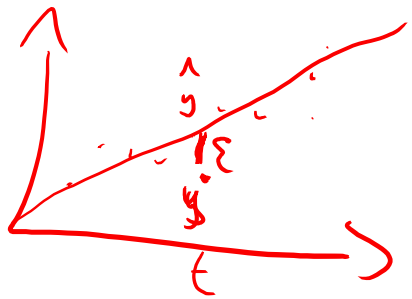
where $m =$ number of seasons.

- For monthly data $m = 12$.
- For quarterly data $m = 4$.

There is a wealth of time series models

- AR autoregressive model
- MA moving average model
- ARMA combination of AR & MA
- ARIMA non-stationary ARMAs
- SARIMA seasonal ARIMAs

…

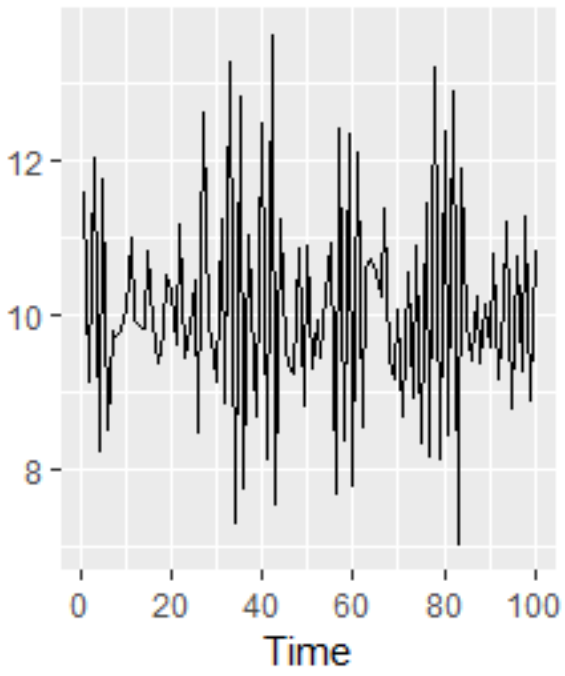


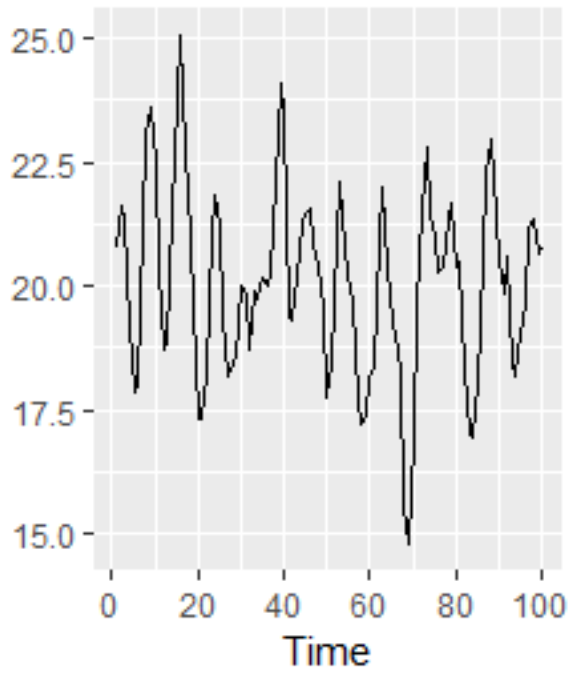
We start by discussing autoregressive models. They are perhaps the simplest and most intuitive time series models that exist.

Autoregressive models

$$y_t = c + \phi y_{t-1} + \cdots + \phi_p y_{t-p} + \varepsilon_t$$

Where ε_t is white noise. This is a multiple regression with lagged values of y_t as predictors.





The simplest model is the AR(1)-model

$$y_t = c + \phi y_{t-1} + \varepsilon_t$$

- When $\phi_1 = 0$,y_t is **equivalent to WN**
- When $\phi_1 = 1$ and $c = 0$, y_t is **equivalent to a RW**
- When $\phi_1 = 1$ and $c \neq 0$, y_t is **equivalent to a RW with drift**
- When $\phi_1 < 0$,y_t tends to **oscillate between positive and negative values**.

Stationarity conditions

We normally restrict autoregressive models to stationary data, and then some constraints on the values of the parameters are required.

General condition for stationarity:

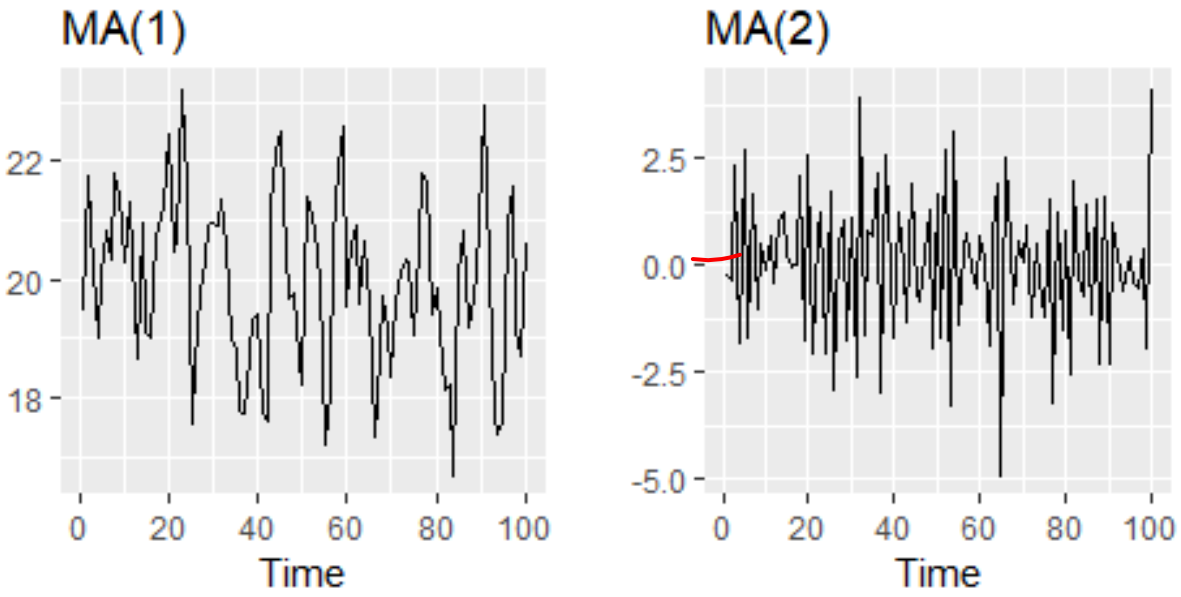
Complex roots of $1 - \phi_1 z - \phi_2 z^2 - \cdots - \phi_p z^p$ lie outside the unit circle on the complex plane.

- For $p = 1$: $-1 < \phi_1 < 1$.
- For $p = 2$: $-1 < \phi_2 < 1 \qquad \phi_2 + \phi_1 < 1 \qquad \phi_2 - \phi_1 < 1$.
- More complicated conditions hold for $p \geq 3$.
- Estimation software takes care of this.

2. Moving Average (MA) models

$$y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} \ldots + \theta_q \epsilon_{t-q}$$

where ϵ_t is white noise. This is a multiple regression with past errors as predictors. Don't confuse this with moving average smoothing!



ARIMA models

Autoregressive Moving Average models:

$$y_t = c + \phi y_{t-1} + \cdots + \phi_p y_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} \ldots + \theta_q \epsilon_{t-q} + \varepsilon_t$$

- Predictors include both **lagged values of y_t and lagged errors.**
- Conditions on coefficients ensure stationarity.
- Conditions on coefficients ensure invertibility.

Autoregressive Integrated Moving Average models

- Combine ARMA model with **differencing**.
- $(1 - B)^d y_t$ follows an ARMA model.

ARIMA models

auto.arima()

Autoregressive Integrated Moving Average models

ARIMA(p, d, q) model

AR: p = order of the autoregressive part

I: d = degree of first differencing involved

MA: q = order of the moving average part.

- White noise model: ARIMA(0,0,0)
- Random walk: ARIMA(0,1,0) with no constant
- Random walk with drift: ARIMA(0,1,0) with constant
- AR(p): ARIMA(p,0,0)
- MA(q): ARIMA(0,0,q)

Backshift notation for ARIMA

ARMA model:

$$y_t = c + \phi_1 B y_t + \cdots + \phi_p B^p y_t + \varepsilon_t + \theta_1 B \varepsilon_t + \cdots + \theta_q B^q \varepsilon_t$$

or $$(1 - \phi_1 B - \cdots - \phi_p B^p) y_t = c + (1 + \theta_1 B + \cdots + \theta_q B^q) \varepsilon_t$$

ARIMA(1,1,1) model:

$$(1 - \phi_1 B)\ (1 - B) y_t = c + (1 + \theta_1 B) \varepsilon_t$$

AR(1) — First difference — MA(1)

Written out:

$$y_t = c + y_{t-1} + \phi_1 y_{t-1} - \phi_1 y_{t-2} + \theta_1 \varepsilon_{t-1} + \varepsilon_t$$

Understanding ARIMA models

The constant c has an important effect on the long-term forecasts obtained from these models.

1. If $c = 0$ and $d = 0$, the long-term forecasts will go to zero.
2. If $c = 0$ and $d = 1$, the long-term forecasts will go to a non-zero constant.
3. If $c = 0$ and $d = 2$, the long-term forecasts will follow a straight line.
4. If $c \neq 0$ and $d = 0$, the long-term forecasts will go to the mean of the data.
5. If $c \neq 0$ and $d = 1$, the long-term forecasts will follow a straight line.
6. If $c \neq 0$ and $d = 2$, the long-term forecasts will follow a quadratic trend.