Lecture 4: Rule-based classification and regression

Felix Held, Mathematical Sciences

MSA220/MVE440 Statistical Learning for Big Data

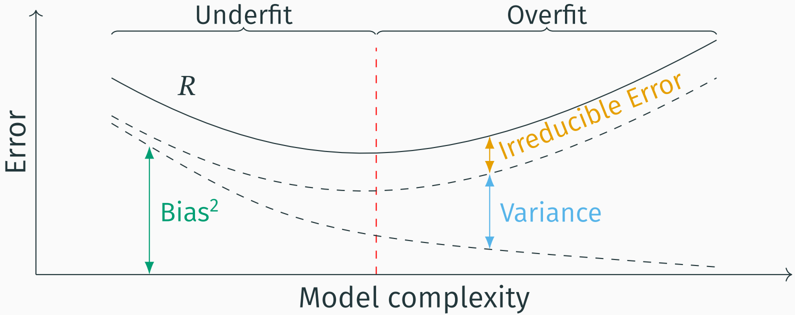
1st April 2019

CHALMERS UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG

Amendment: Bias-Variance Tradeoff

Bias-Variance Decomposition

$$\begin{aligned} R &= \mathbb{E}_{p(\mathcal{T},\mathbf{x},y)}\left[(y-\hat{f}(\mathbf{x}))^2\right] && \text{Total expected prediction error} \\ &= \sigma^2 && \text{Irreducible Error} \\ &+ \mathbb{E}_{p(\mathbf{x})}\left[\left(f(\mathbf{x})-\mathbb{E}_{p(\mathcal{T})}\left[\hat{f}(\mathbf{x})\right]\right)^2\right] && \text{Bias}^2 \text{ averaged over } \mathbf{x} \\ &+ \mathbb{E}_{p(\mathbf{x})}\left[\mathrm{Var}_{p(\mathcal{T})}\left[\hat{f}(\mathbf{x})\right]\right] && \text{Variance of } \hat{f} \text{ averaged over } \mathbf{x} \end{aligned}$$



Observations

- Irreducible error cannot be changed
- Bias and variance of $\widehat{f}$ are sample-size dependent
  - For a consistent estimator $\widehat{f}$

    $$\mathbb{E}_{p(\mathcal{T})}[\widehat{f}(x)] \to f(x)$$

    for increasing sample size
  - In many cases:

    $$\mathrm{Var}_{p(\mathcal{T})}(\widehat{f}(x)) \to 0$$

    for increasing sample size
  - **Caution:** Theoretical guarantees are often dependent on the number of variables p staying fixed and increasing n. Might not be fulfilled in reality.

Amendment: Leave-One-Out Cross-validation (LOOCV)

Cross-validation with $c = n$ is called **leave-one-out cross-validation**.

- Popular because explicit formulas (or approximations) exist for many special cases (e.g. regularized regression)
- Uses the most data for training possible
- More variable than c-fold CV for $c < n$ since only one data point is used for testing and the training sets are very similar
- In praxis: Try out different values for c. Be cautious if results vary drastically with c. Maybe the underlying model assumptions are not appropriate.

Classification and Partitions

Classification and Partitions

A classification algorithm constructs a partition of feature space and assigns a class to each.

- kNN creates local neighbourhoods in feature space and assigns a class in each
- Logistic regression divides feature space implicitly by modelling $p(i|\mathbf{x})$ and determines decision boundaries through Bayes' rule
- Discriminant analysis creates an explicit model of the feature space conditional on the class. It models $p(\mathbf{x}, i)$ by assuming that $p(\mathbf{x}|i)$ is a normal distribution and either estimates $p(i)$ from data or through prior knowledge.

New point-of-view: Rectangular Partitioning

Idea: Create an **explicit partition** by dividing feature space into **rectangular regions** and assign a **constant conditional mean** (regression) or **constant conditional class probability** (classification) to each region.

Given regions R_m for $m = 1, \dots, M$, a classification rule for classes $i \in \{1, \dots, K\}$ is

$$\hat{c}(\mathbf{x}) = \underset{1 \leq i \leq K}{\arg\max} \sum_{m=1}^{M} \mathbb{1}(\mathbf{x} \in R_m) \left(\sum_{\mathbf{x}_l \in R_m} \mathbb{1}(i_l = i) \right)$$

and a regression function is given by

$$\widehat{f}(\mathbf{x}) = \sum_{m=1}^{M} \left(\frac{1}{|R_m|} \sum_{\mathbf{x}_l \in R_m} y_l \right) \mathbb{1}(\mathbf{x} \in R_m)$$

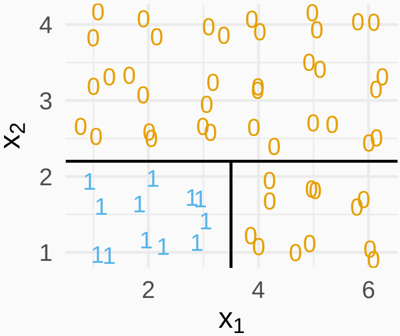
(Derivations are similar to kNN with regions instead of neighbourhoods.)

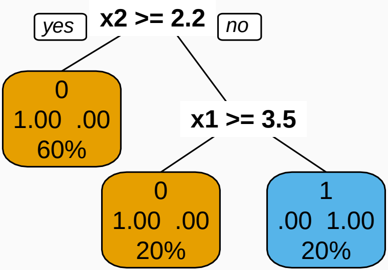
Classification and Regression Trees (CART)

- **Complexity of partitioning:**

  Arbitrary Partition > Rectangular Partition > Partition from a sequence of binary splits

- Classification and Regression Trees create a **sequence of binary axis-parallel splits** in order to **reduce variability** of values/classes in each region





CART: Tree building/growing

1. Start with all data in a **root node**
2. Binary splitting
   2.1 Consider each feature $x_{\cdot j}$ for $j = 1, \dots, p$. Choose a **threshold** t_j (for continuous features) or a **partition of the feature categories** (for categorical features) that results in the greatest improvement in **node purity**:

   $$\{i_l : x_{lj} > t_j\} \quad \text{and} \quad \{i_l : x_{lj} \leq t_j\}$$

   2.2 Choose the feature j that led to the best splitting of the data and create a new **child node** for each subset
3. Repeat Step 2 on all child nodes until the tree reaches a **stopping criterion**

All nodes without descendents are called **leaf nodes**. The sequence of splits preceding them defines the regions R_m.

Measures of node purity

Use

$$\widehat{\pi}_{im} = \frac{1}{|R_m|} \sum_{\mathbf{x}_l \in R_m} \mathbb{1}(i_l = i)$$

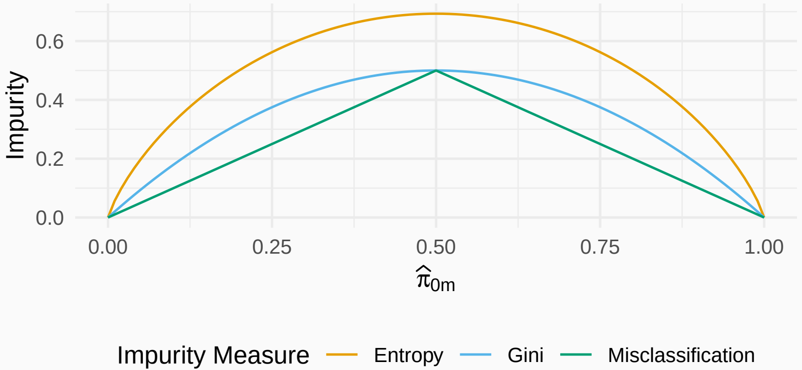
- Three common measures to determine impurity in a region R_m are (for classification trees)

$$\begin{aligned} \text{Misclassification error:} \quad & 1 - \max_i \widehat{\pi}_{im} \\ \text{Gini impurity:} \quad & \textstyle\sum_{i=1}^{K} \widehat{\pi}_{im}(1 - \widehat{\pi}_{im}) \\ \text{Entropy/deviance:} \quad & -\textstyle\sum_{i=1}^{K} \widehat{\pi}_{im} \log \widehat{\pi}_{im} \end{aligned}$$

- All criteria are zero when only one class is present and maximal when all classes are equally common.
- For regression trees the decrease in mean squared error after a split can be used as an impurity measure.

Node impurity in two class case

Example for a two-class problem ($i = 0$ or 1). $\hat{\pi}_{0m}$ is the empirical frequency of class 0 in a region R_m.



Only gini impurity and entropy are used in practice (averaging problems for misclassification error).

Stopping criteria

- Minimum size of leaf nodes (e.g. 5 samples per leaf node)
- Minimum decrease in impurity (e.g. cutoff at 1%)
- Maximum tree depth, i.e. number of splits (e.g. maximum 30 splits from root node)
- Maximum number of leaf nodes

Running CART until one of these criteria is fulfilled generates a **max tree**.

Summary of CART

- **Pro:** Outcome is easily interpretable
- **Pro:** Can easily handle missing data
- **Neutral:** Only suitable for axis-parallel decision boundaries
- **Con:** Features with more potential splits have a higher chance of being picked
- **Con:** Prone to overfitting/unstable (only the best feature is used for splitting and which is best might change with small changes of the data)

How can overfitting be avoided?

- **Tuning of stopping criteria:** These can easily lead to early stopping since a weak split might lead to a strong split later
- **Pruning:** Build a max tree first. Then reduce its size by collapsing internal nodes. This can be more effective since weak splits are allowed during tree building. ("The silly certainty of hindsight")
- **Ensemble methods:** Examples are bagging, boosting, stacking, ...

A note on pruning

- A common strategy is **cost-complexity pruning**.
- For a given $\alpha > 0$ and a tree T its cost-complexity is defined as

$$C_\alpha(T) = \underbrace{\sum_{R_m \in T} \left(\frac{1}{|R_m|} \sum_{\mathbf{x}_l \in R_m} \mathbb{1}(i_l \neq \hat{c}(\mathbf{x})) \right)}_{\text{Cost}} + \underbrace{\alpha |T|}_{\text{Complexity}}$$

  where $(i_l, \mathbf{x}_l)$ is the training data, $\hat{c}$ the CART classification rule and $|T|$ is the number of leaf nodes/regions defined by the tree.
- It can be shown that successive subtrees T_k of the max tree $T_{\max}$ can be found such that each tree T_k minimizes $C_{\alpha_k}(T_k)$ where $\alpha_1 \geq \cdots \geq \alpha_J$
- The tree with the lowest cost-complexity is chosen

Re-cap of the bootstrap and variance reduction

The Bootstrap – A short recapitulation (I)

Given a sample $x_i, i = 1, \dots, n$ from an underlying population estimate a statistic θ by $\hat{\theta} = \hat{\theta}(x_1, \dots, x_n)$.

What is the uncertainty of $\hat{\theta}$?

Solution: Find confidence intervals (CIs) quantifying the variability of $\hat{\theta}$.

Computation:

- Through theoretical results (e.g. linear models) if distributional assumptions fulfilled
- Linearisation for more complex models (e.g. nonlinear or generalized linear models)
- Nonparametric approaches using the data (e.g. **bootstrap**)

All of these approaches require fairly large sample sizes.

Nonparametric bootstrap

Given a sample $x_1, \dots, x_n$ **bootstrapping** performs for $b = 1, \dots, B$

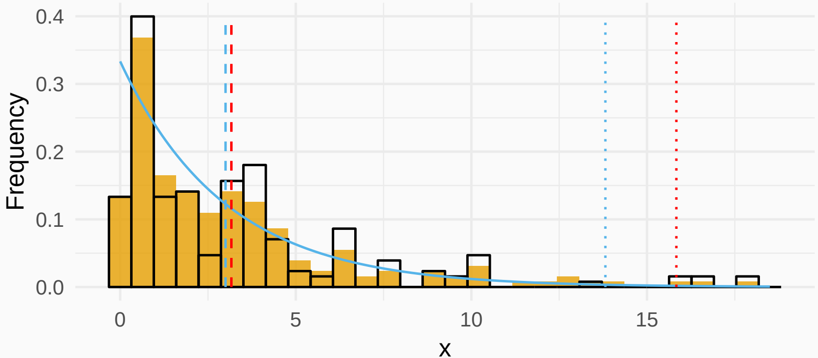
1. Sample $\tilde{x}_1, \dots, \tilde{x}_n$ with replacement from original sample
2. Calculate $\hat{\theta}_b(\tilde{x}_1, \dots, \tilde{x}_n)$

- B should be large (in the 1000–10000s)
- The distribution of $\hat{\theta}_b$ approximates the sampling distribution of $\hat{\theta}$
- The bootstrap makes **exactly one strong assumption**:

  The data is discrete and values not seen in the data are impossible.[1]

[1] Check out this blog post!

CI for statistics of an exponential random variable



Data (n = 200) simulated from $x \sim \mathrm{Exp}(1/3)$, i.e. $\mathbb{E}_{p(x)}[x] = 3$

- Orange histogram shows original sample
- Blue line is the true density
- Black outlined histogram shows a bootstrapped sample
- Vertical lines are the mean of x (dashed) and the 99% quantile (dotted) [red = empirical, blue = theoretical]

CI calculation: Normal approximation and percentile method

1. **Normal approximation:** Set $\overline{\theta} = \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}_b$ and estimate the standard error of $\hat{\theta}$ as

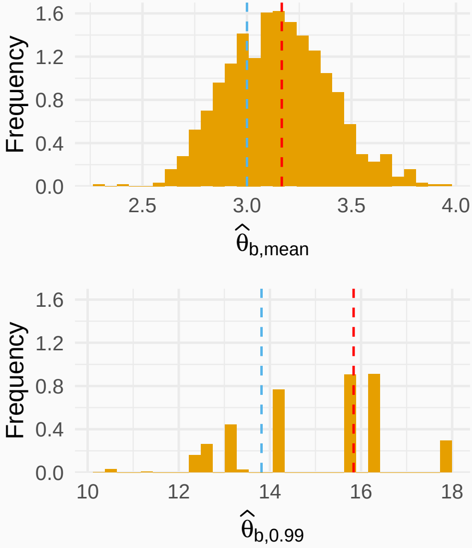
$$\widehat{\sigma}_{se} = \sqrt{\frac{\sum_{b=1}^{B}(\hat{\theta}_b - \overline{\theta})^2}{B-1}}$$

   Assume the distribution of $\hat{\theta}$ is approximately $N(\hat{\theta}, \widehat{\sigma}_{se})$ giving CI

$$\hat{\theta} \pm z_{1-\alpha/2}\widehat{\sigma}_{se}$$

2. **Percentile/quantile method:** Take the α and $\alpha/2$ quantiles of the bootstrap estimates $\hat{\theta}_b$ as boundaries of CI

CI calculation: Applied to example



For the mean value, normal approximation assumption seems reasonable

95% CIs

Normal Approx. (2.68, 3.65)

Perc. Method (2.71, 3.67)

For the quantile, bootstrapping requires much larger n and shows high uncertainty

Based on $B = 1000$ bootstrap samples

Modifications to nonparametric bootstrap

- Different sampling strategies. Some examples:
  - m-out-of-n bootstrap: Draw $m < n$ samples without replacement
  - Draw from a smooth density estimate of the data
  - Draw from a parametric distribution fitted to the original data
- Normal approximation doesn't always apply and percentile method is unstable for complicated statistics. Example of alternative
  - Bootstrap-t: Instead of normal quantiles, estimate quantiles from
    $$\frac{\hat{\theta}_b - \hat{\theta}}{\widehat{\sigma}_b}$$
    where $\widehat{\sigma}_b$ is an estimate of the standard error
  - Many other alternatives exist ...

Limitations of the bootstrap

- Number of samples needs to be quite large
- Extreme values (minimum, maximum very small or large quantiles) can be hard to estimate since they might not even appear in data
- Many basic CI estimation algorithms assume that the bootstrap distribution is approximately normal (often not the case in reality)

Bootstrap aggregation (bagging)

1. Given a training sample $(y_l, \mathbf{x}_l)$ or $(i_l, \mathbf{x}_l)$, we want to fit a predictive model $\widehat{f}(\mathbf{x})$
2. For $b = 1, \dots, B$, form bootstrap samples of the training data and fit the model, resulting in $\widehat{f}_b(\mathbf{x})$
3. Define

$$\widehat{f}_{\text{bag}}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^{B} \widehat{f}_b(\mathbf{x})$$

   where $\widehat{f}_b(\mathbf{x})$ is a continuous value for a regression problem or a vector of class probabilities for a classification problem

Majority vote can be used for classification problems instead of averaging

Bagging and variance reduction

- Bagging using averages approximates

$$f_{\text{ag}}(\mathbf{x}) = \mathbb{E}_{p(\mathcal{T})}\left[\widehat{f}(\mathbf{x})\right]$$

- For the conditional expected error in squared error loss

$$\mathbb{E}_{p(\mathcal{T},y|\mathbf{x})}[(y - \widehat{f}(\mathbf{x}))^2] \geq \mathbb{E}_{p(\mathcal{T},y|\mathbf{x})}[(y - f_{\text{ag}}(\mathbf{x}))^2]$$

- Some notes:
  - Remember the graphs of kNN from last lecture: Noisy individually, more stable (less variable) on average
  - Bagging shows no effect on linear models

Correlation and bagged variance

Recall: For identically distributed (i.d.) random variables x_i, $i = 1, \dots, n$

$$\mathrm{Var}\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) = \frac{1-\rho}{n}\sigma^2 + \rho\sigma^2$$

where $\rho \in [0, 1)$ is the (positive) pairwise correlation coefficient and σ^2 is the variance of each x_i.

- Bootstrap samples are correlated and increase total variance
- Decreasing correlation between bootstrap samples would decrease the variance of a bagging estimate

Random Forests

Random Forests

1. Given a training sample with p features, do for $b = 1, \dots, B$
    1.1 Draw a bootstrap sample of size n from training data (with replacement)
    1.2 Grow a tree T_b until each node reaches minimal node size $n_{\min}$
        1.2.1 Randomly select m variables from the p available
        1.2.2 Find best splitting variable among these m
        1.2.3 Split the node
2. For a new $\mathbf{x}$ predict

Regression: $\widehat{f}_{rb}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} T_b(\mathbf{x})$

Classification: Majority vote at $\mathbf{x}$ across trees

Note: Step 1.2.1 leads to less correlation between trees built on bootstrapped data.

Variable importance

1. **Impurity index:** Splitting on a feature leads to a reduction of node impurity. Summing all improvements over all trees per feature gives a measure for variable importance
2. **Out-of-bag error**
    - During bootstrapping for large enough n, each sample has a chance of about 63% to be selected
    - For bagging the remaining samples are **out-of-bag**.
    - These out-of-bag samples for tree T_b can be used as a test set for that particular tree, since they were not used during training. Resulting in test error E_0
    - Permute variable j in the out-of-bag samples and calculate test error again $E_1^{(j)}$
    - The increase in error

      $$E_1^{(j)} - E_0 \geq 0$$

      serves as an importance measure for variable j

Take-home message

- Direct partitioning of feature space is a complex task
- Simplifications in form of binary splits resulting in tree models work well
- High interpretability of CART, but also high variability
- Random Forests tackles variance reduction though bagging and random selection of splitting features