Predictive Modeling

Decision Trees

Mirko Birbaumer

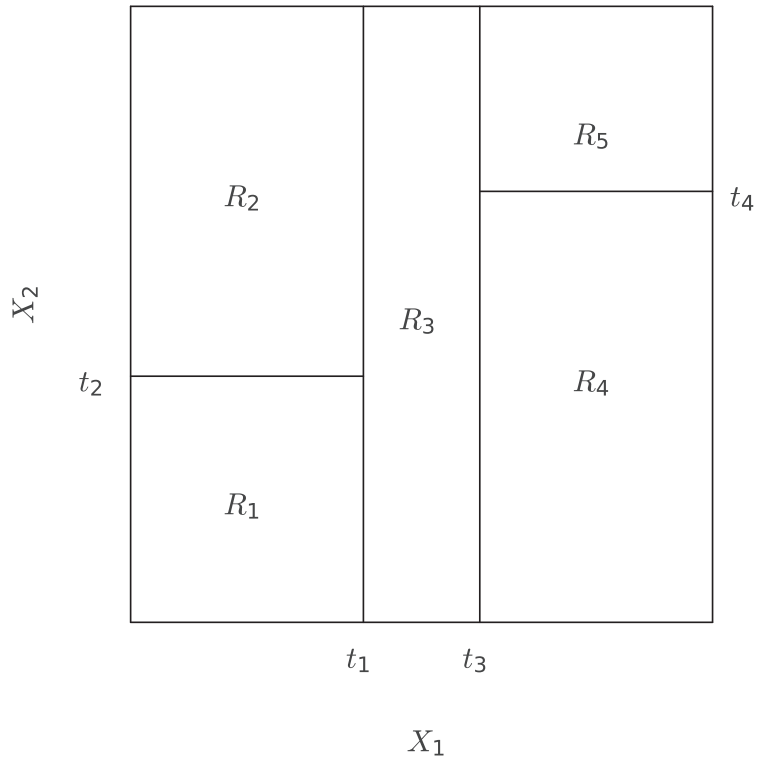
HSLU T&A

1. Classification Trees
2. Node Purity
3. Making Predictions
4. Tree Pruning
5. Decision Trees vs. Logistic Regression

Decision Trees

- **Tree-based** methods can be applied for *classification* and *regression*
- A classification tree aims at **segmenting** or **stratifying** the predictor space into a number of simple regions
- Each region is associated with a unique **class level** of the response

Decision Trees - Stratification of Predictor Space



Decision Trees

- For a new observation the **predicted class** equals the class level associated with the region containing the observation

- Since the set of splitting rules used to **segment** the predictor space can be summarized in a tree, these types of approaches are called **decision-tree** methods

Pros and Cons of Decision Trees

- Tree-based methods are simple and useful for interpretation (such as logistic regression, remember `Default` example)
- However, they typically are not competitive with the best supervised learning methods in terms of predictive accuracy
- Hence, we also discuss **bagging** and **random forests**. These methods grow multiple trees which are then combined to yield a single consensus prediction (*ensemble methods*)
- Combining a large number of trees can often result in dramatic improvements in prediction accuracy, at the loss of some interpretation

Example Stratification of Predictor Space : `Heart`

- We study the `Heart` data set
- These data contain a binary outcome `AHD` for 303 patients who presented with chest pain
- The level `Yes` indicates the presence of a heart disease based on an angiographic test, while `No` means no heart disease
- There are 13 predictors including `Age`, `Sex`, `Chol` (cholesterol measurement), and other heart and lung function measurements.

Example : `Heart`

	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng
1	1	typical	145	233	1	2	150	0
2	1	asymptomatic	160	286	0	2	108	1
3	1	asymptomatic	120	229	0	2	129	1
4	1	nonanginal	130	250	0	0	187	0
5	0	nontypical	130	204	0	2	172	0
6	1	nontypical	120	236	0	0	178	0

- **Challenge** : *predict* the presence of the heart disease by means of the predictor values only and without performing the angiographic test
- See example `1.1` in the `Decision Trees` chapter

Binary Splitting

- Idea behind a **decision tree** is to *recursively* partition the predictor variable space into disjoint regions, where each region is as *pure* as possible with respect to the labels of the response variable

- These regions are referred to as **terminal nodes** or **leaves**.

Binary Splitting

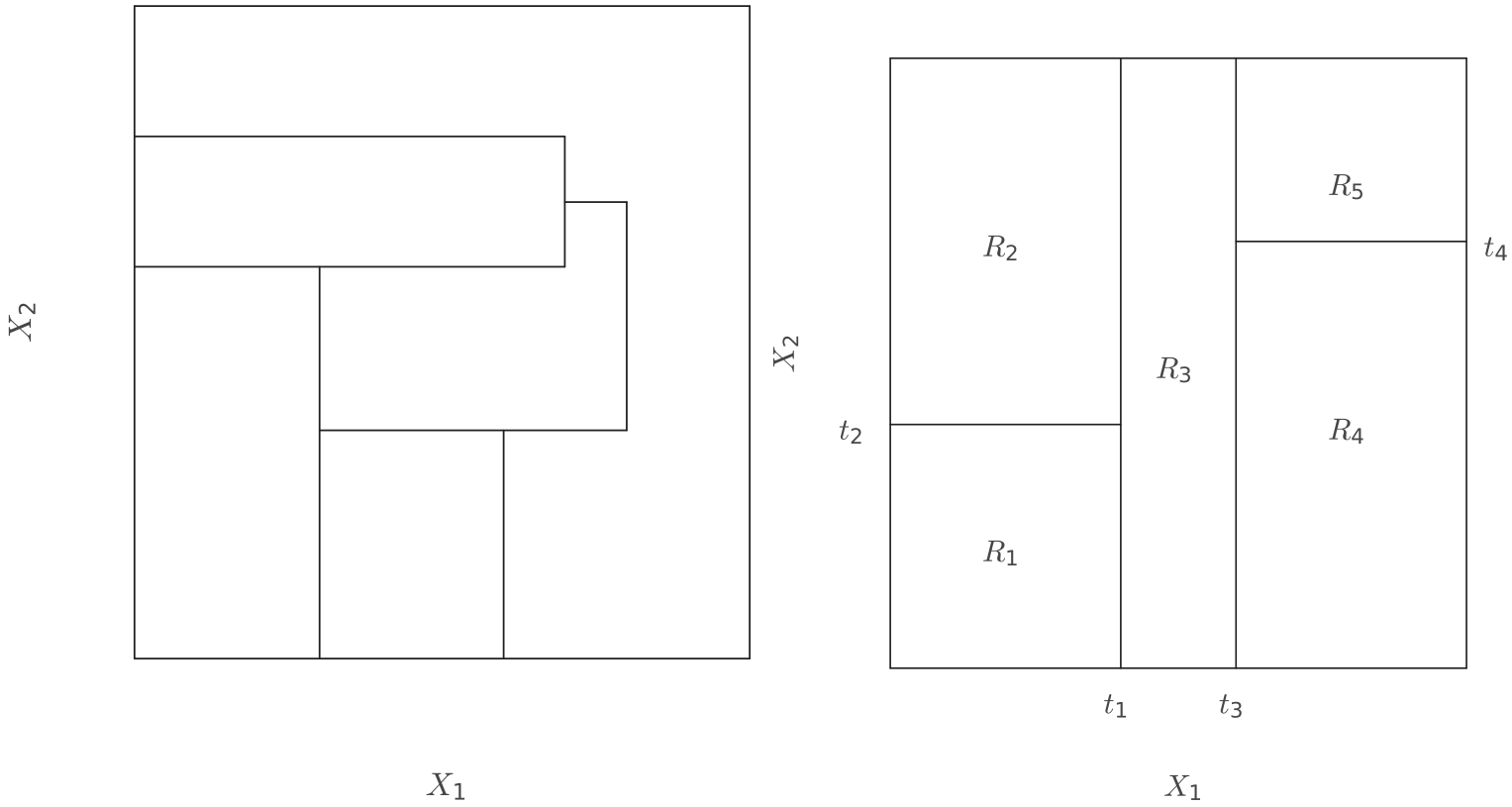
- Assume we are given the **training set** $(\overrightarrow{x}_1, y_1), \ldots, (\overrightarrow{x}_n, y_n)$ where $\overrightarrow{x}_i = x_{i1}, \ldots, x_{ip}$ are the **predictor values** and y_i is a **qualitative response** (not necessarily binary).
  1. Init the set of regions $\mathcal{R} = \{R\}$ by the predictor domain R (e.g. $\mathbb{R}^p$)
  2. Choose the optimal region R in $\mathcal{R}$ and the optimal predictor X_i such that a **binary split** of R with respect to X_i

$$R_1 = \{x \in \mathbb{R} \mid x_i > t\} \quad \text{and} \quad R_2 = \{x \in \mathbb{R} \mid x_i \leq t\} \tag{1}$$

     gives the *highest gain in purity* (for some threshold t)
  3. Replace R in $\mathcal{R}$ by R_1 and R_2 and return to 2.

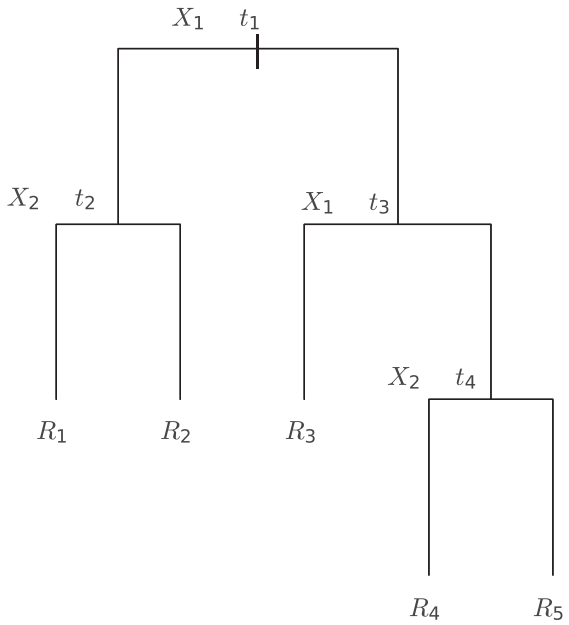
- The iteration is stopped if the current splitting fullfills a pre-defined stopping criterion (e.g. gain in purity is below some pre-defined threshold).

Examples : Binary Split



- *Left figure* : Predictor space segmentation which **cannot** be generated by binary splitting
- *Right figure* : First, variable X_1 is split with parameter t_1. For $X_1 \leq t_1$, variable X_2 is split at t_2 and so on $\rightarrow$ **Decision Tree**

Decision Trees



- R_1, R_2, R_3, R_4, and R_5 are known as **terminal nodes** or **leaves**
- Trees are typically drawn *upside down*, in the sense that the leaves are at the bottom of the tree
- Points along the tree where the predictor space is split are referred to as **internal nodes** ($X_1 = t_1$ and $X_1 = t_3$)

Example : Decision Tree for `Heart` data

- See examples `1.2` and `1.3` of the `Decision Trees` chapter
- Some regions in the partition contain either significantly more `AHD = No` cases or vice versa
- However, each region contains both species
- **Node purity** refers to a quantitative measure of how the most frequent occuring class in a region dominates the other classes

Node purity - Classification Error Rate

- Assume that the response Y has K levels and that we have built a tree T with M terminal nodes

- **Classification error rate** : fraction of training samples in a region that do not belong to the most common class
  - $$E_m(T) = 1 - \max_k \widehat{p}_{mk}$$
  - $\widehat{p}_{mk}$ is the proportion of the training data in region m that is from level k
  - However, classification error is not sufficiently sensitive to tree-growing

Node purity - Gini Index

- **Gini index** : measure of total variance across the K classes :
  - $$G_m(T) = \sum_{k=1}^{K} \widehat{p}_{mk}(1 - \widehat{p}_{mk})$$
  - Gini index takes on a small value if all the p_{mk} are close to zero or one
  - Gini index is referred to as a node **purity** measure : a small value indicates that a node contains predominantly observations from the same class

Example : Node Purity for `Heart` Data

- An alternative to Gini index is **Cross-entropy** :

$$D_m(T) = -\sum_{k=1}^{K} \widehat{p}_{mk} \log(\widehat{p}_{mk})$$

- It turns out that the Gini index and the cross-entropy are very similar numerically

- Please solve exercise `2` in the exercise sheet and check example `2.1` in the `Decision Trees` chapter

Pruning a Tree

- Growing a complex tree, i.e. a tree with small terminal nodes, will produce good predictions on the training data, but is likely to **overfit** the data, leading to poor test set performance. **Why?**

- See example `3.1` of the `Decision Trees` chapter

Pruning a Tree

- A smaller tree with fewer splits (that is, with fewer regions $R_1, R_2, \ldots, R_j$) might lead to lower variance and better interpretation at the cost of a little bias

- One way to overcome this problem is to grow the tree only so long as the decrease in the purity measure due to each split exceeds some (high) threshold

- This strategy will result in smaller trees, but is too short-sighted since a seemingly worthless split early on in the tree might be followed by a very good split, i.e. a split that leads to a large increase of node purity.

Pruning a Tree

- **Tree pruning** amounts to first grow a very large tree and successively **prune** it back until we find a good **subtree**

- **Cost Complexity Pruning** - also known as **weakest link pruning** is used to do this

Pruning a Tree : Cost Complexity Pruning

- We first compute a very large tree T_0

- We consider a sequence of subtrees T_α indexed by a nonnegative tuning parameter α : for each value of α there corresponds a subtree $T \subset T_0$ which minimizes the **cost-complexity function**:

$$R_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T|, \quad \alpha \geq 0 \tag{2}$$

where

  - $|T|$ denotes the number of terminal nodes in the tree T
  - m ranges over all terminal nodes
  - For each terminal node m, the number N_m denotes the number of elements in the node
  - $Q_m(T)$ is an arbitrary node purity measure (e.g. the Gini index $G_m(t)$)

Pruning a Tree : Cost Complexity Pruning

- For each given value of α we compute a T_α such that $R_\alpha(T)$ is minimized
- Here α is a tuning parameter that controls the trade-off between **complexity** and **fit to the training data**
- A **large** value of α **penalizes** the number of terminal nodes of the tree and hence the resulting tree T will have few terminal nodes, but potentially will fit poorly to the training data
- Vice versa, a **small** value of α puts all emphasis on data fit and a complex tree with many terminal nodes will result. In the extreme case that $\alpha = 0$, we will have $T = T_0$.

Cost Complexity Pruning : How to compute T_α?

- For each α it can be shown that there is a unique smallest subtree T_α that minimizes $R_\alpha(T)$
- To compute T_α **weakest link cutting** is used
- It amounts to successively collapse the **internal node** that leads to the **smallest increase** in $R(T)$ (weakest link) until the single-noded tree (root) is reached
- The sequence of trees generated by this procedure can be shown to contain T_α

How to find the optimal α?

Cost-complexity pruning

1. Use recursive binary splitting to grow a very large tree T_0 and set $\alpha = 0$
2. Increase α and for each value compute the subtree T_α that minimizes the cost-complexity function $R_\alpha(T)$
3. Stop if T_α is the root node

How to find the optimal α?

Optimal pruning by cross-validation

Divide the data set into K folds. For $k = 1, \ldots, K$

1. Grow a large tree T_0 based on all but the k-th fold of the data set.
2. Perform cost-complexity pruning and obtain a sequence of optimal subtrees T_α as a function of α. Compute the classification error rate on the k-th fold as a function of α.

For each value of α average the classification error rate over the K folds. Choose α such that the error rate is minimal.

Example : Check example `4.2` of the `Decision Trees` chapter

Decision Trees versus Logistic Regression

- Repetition : **Logistic regression** models the *probability* of classes given the predictor values. The classification is carried out by thresholding:

$$P(Y = 1|X_1, \ldots, X_p) = \frac{e^{\beta_0 + \sum_{i=1}^{p} \beta_i X_i}}{1 + e^{\beta_0 + \sum_{i=1}^{p} \beta_i X_i}}$$

- For an observed predictor value $(x_1, \ldots, x_p)$ the response $\widehat{y}$ is predicted by checking whether the right hand side above is larger ($\widehat{y} = 1$) or smaller ($\widehat{y} = 0$) than a threshold, say 0.5 :

$$\frac{e^{\beta_0 + \sum_{i=1}^{p} \beta_i x_i}}{1 + e^{\beta_0 + \sum_{i=1}^{p} \beta_i x_i}} \geq \frac{1}{2}$$

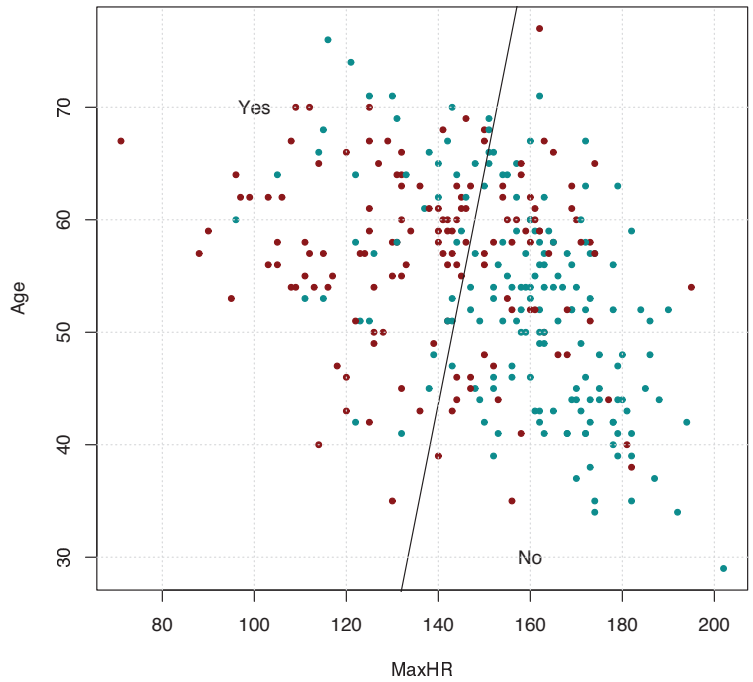
After some basic rearrangements this gives

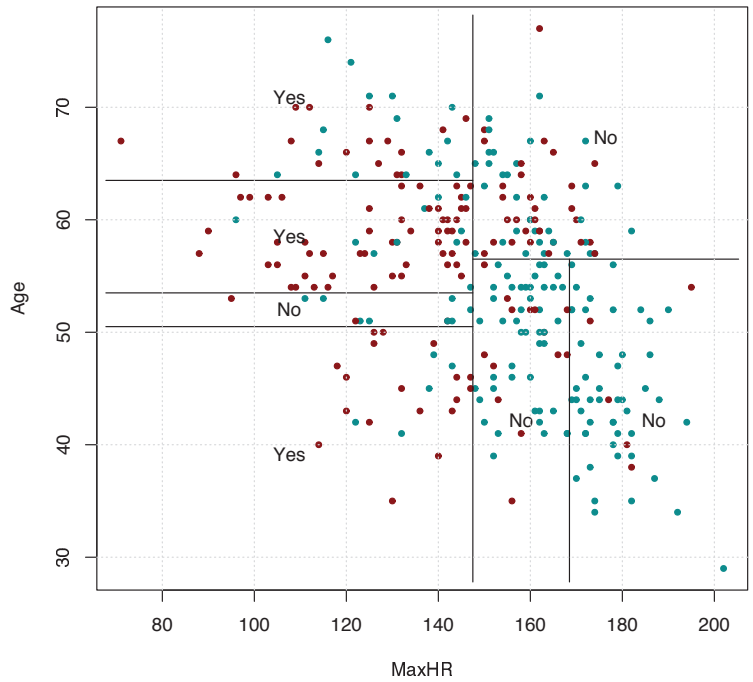
$$\beta_0 + \sum_{i=1}^{p} \beta_i x_i \geq 0.$$

Decision Trees versus Logistic Regression

- Put differently, logistic regression amounts to partition the predictor space in two half spaces by a hyperplane.

- In constrast, a decision tree partitions the feature space into axis parallel boxes, please check example `5.1` of the **`Decision Trees`** chapter





Pros and Cons

- *Advantages* :
  - Trees are very simple to explain to people. In fact, they are even easier to explain than linear regression!
  - Some people think that decision-trees more closely mirror human decision-making than do regression and classification approaches
  - Trees can be displayed graphically, and are easily interpreted even by non-experts
  - Trees can easily handle qualitative predictors without the need to create dummy variables
- *Disadvantages*:
  - Unfortunately, trees do not have the same level of predictive accuracy as some of the other regression and classification approaches

However, by **aggregating** many decision trees, the predictive performance of trees can be substantially improved