Seminar paper

Recursive partitioning by conditional inference

Christoph Molnar

Supervisor:
Stephanie Möst

5. March 2013
Department of Statistics
University of Munich

Abstract

Recursive partitioning is a popular modeling tool for data analysis. Due to the resulting tree structure of the partitioned covariate space, the models are easy to interpret. Many of the existing partitioning algorithms suffer from a biased variable selection, meaning that variables with more possible values are selected more frequently.

This paper presents the work of Hothorn et al. [2006], who proposed and implemented an partitioning algorithm, which has an unbiased variable selection: conditional inference trees. All decisions are formulated as statistical hypotheses and tested with permutation tests. The proposed framework covers not only numeric and nominal regression, but arbitrary measurement scales of response and covariates, like ordinal or censored regression. This papers regards the algorithm in detail and illustrates it with two examples. Special emphasis is on the used test statistics.

Contents

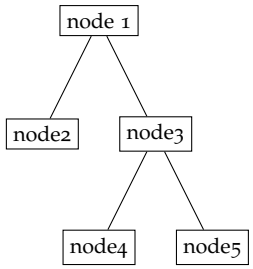
Recursive partitioning

Recursive partitioning is a powerful yet simple tool in predictive and explanatory statistics [1]. Models built by partitioning have the shape of decision trees, which makes the approach easy to understand for everyone, without having to understand the algorithm behind. The algorithm partitions the data to reconstruct the relationship between the response variable Y and the covariates X.

[1] For a detailed introduction to trees please read Friedman et al. [2001, page 305]

$Y = f(X)$

Partitioning can be done with many different approaches and therefore there is a great variety of algorithms to grow trees. They can be categorized into those which do regression, those which do classification and those which do both. Another characteristic is the number of splits per partition step. There are binary splits, which divide a partition into two new partitions and multiway splits which yield more than two partitions. There is even more variety in the philosophy of how to determine which variable to take for the next step and where to split it. Also the stopping criterion, when the tree is not grown any more, differs for the algorithms. One the they all have in common is the tree like structure. The presented algorithm is called "Recursive binary partitioning by conditional inference" [2]. It is an algorithm for recursive partitioning algorithm, with binary splits and all of it's split decisions are based on statistical hypothesis tests. The algorithm covers not only regression and classification, but even more complex scales like ordinal regression for example. Their creators are Torsten Hothorn, Kurt Hornik and Achim Zeileis. This work is mainly based on Hothorn et al. [2006].



[2] Will be called "Conditional tree algorithm" throughout this paper for simplicity

Motivation

Known issues

Recursive partitioning suffers from different problems.

CART (Classification And Regression Trees) by Breiman et al. [1984] is a popular and widely used partitioning algorithm. Let us take a closer look at the problems taking CART as an example and how the conditional trees approach solves them.

Overfitting

[3] Splits which separates only a few observations

If a tree is allowed to grow full length, pathological splits [3] can happen and if the covariate space is large enough we would end up with a tree, which contains only one observation in each terminal node. This tree would very likely be overfitting on the training data and deliver very bad results on new data. Approaches to avoid this problem are techniques called early stopping and pruning. Early stopping forces trees to stop growing when a criterion is not fulfilled. This criterion could be a minimum number of observations in a node. Pruning lets the tree grow at first and prunes the leafs back afterward. The CART algorithm uses both early stopping and pruning to avoid overfitting.

Heuristic approach, lack of statistical model

Connected to overfitting is the lack of a statistical concept behind the splits. Mingers [1987] mention, that the algorithm "[...] has no concept of statistical significance, and so cannot distinguish between a significant and an insignificant improvement in the information measure." In CART and many other algorithms, the next split is just an heuristic step, as the algorithm only searches for the next best split. The conditional trees algorithm measures the association and uses the covariate with the strongest association with the response variable.

Variable selection bias

Exhaustive search procedures as used by the CART algorithm tend to choose variables with more possible split points (variable selection

bias). This is a problem of multiple comparison. Covariates with many possible splits are searched more often for the best split. This problem was identified by different researchers, among them are the inventors of the CART algorithm Breiman et al. [1984, p. 42]

Restriction on possible measurement scales of Y and X

In the family of partitioning algorithm, the CART algorithm is one of the more powerful ones, as it can do regression for continuous Y as well as classification for categorical Y. Many other algorithm are restricted to one of the both tasks. Though, CART still lacks support for other scales of X and Y. Examples are: ordinal regression and censored data. Conditional trees offer a very general test statistic which can handle more cases than simple regression or classification.

The next chapter explains how the conditional trees algorithm is designed to overcome the mentioned problems and to offer an alternative approach.

Recursive partitioning by conditional inference

"Classic" recursive partitioning algorithms often have a kind of loss function, which is to be minimized with the next split. The optimization is done by an exhaustive search over all possible split points. After splitting the data, the steps are repeated for the new partitions until a stop criterion dictates to stop.

Summary of the conditional trees algorithm

Recursive partitioning with the conditional trees algorithm works different. In contrast to CART or other algorithms, variable selection and the search for the best split are strictly separated. Dependency between response and covariate is tested with hypothesis tests. To get rid of the problem of different scales for different covariates, the measurement for the association is the p-value. After choosing a covariate for the split, it is searched for the best split point. The stopping criterion is formulated in terms of statistical test theory as well: stop when the null-hypothesis of independence cannot be rejected any more.

All of the hypothesis testing is done by permutation tests. [4] When using permutation tests you are free to choose the test statistic. Some are better than others. It will depend on the scales of the response and the covariate and on how you want to formulate the problem. The test statistics used by the algorithm is derived from a framework developed by Strasser and Weber [1999]. They offer a very general formulation of a test statistic for permutation tests, which can handle arbitrary scales for response and covariate.

[4] For a quick introduction to permutation tests, please visit the appendix, page 27. A good idea of permutation tests is required to understand the recursive partitioning algorithm.

The following chapter explains the algorithm in detail.

Algorithm

In every partition, following steps are conducted[5] See also the original formulation in Hothorn et al. [2006, Chapter 2 and 3]

[5] Permutation test related steps are marked in red

1. **Stop criterion**
   - Test global null hypothesis H_0 of independence between Y and all X_j with $H_0 = \cap_{j=1}^{m} H_0^j$ and $H_0^j : D(\mathbf{Y}|X_j) = D(\mathbf{Y})$ (permutation tests for each X_j)
   - If H_0 not rejected (no significance for all X_j) $\Rightarrow$ Stop
2. **Variable selection**
   Select covariate X_{j*} with strongest association (smallest p-value)
3. **Best split point search**
   Search best split for X_{j*} (max. test statistic c) and partition data
4. **Repeat**
   Repeat steps 1.), 2.) and 3.) for both of the new partitions

The algorithm starts with the whole data set and tests if it should be split. If the null hypothesis is rejected the variable with the strongest association with the response is chosen and the partition is split into two new partitions. The steps will be repeated within both of the new partitions.[6] Covariates chosen for a split can be chosen again later (only in the case of a binary covariate it doesn't work). The tree stops to grow, if in all partitions the global null hypothesis of independence cannot be rejected (stop criterion). The next sections describe in detail how the single steps work and especially how permutation tests are applied.

[6] That's why it is called "recursive" partitioning"

The test statistic

All decisions in steps 1.) and 2.) of the algorithm are embedded in hypothesis tests. These are done with permutation tests (conditional inference). Also step 3.) makes use of the following test statistic, although nothing is tested. The test statistic from Hothorn et al. [2006, page 4]: [7]

$$\mathbf{T}_j(L_n, w) = vec\left(\sum_{i=1}^{n} w_i g_j(X_{ij}) h(Y_i, (Y_1, ..., Y_n))^T\right) \in \mathbb{R}^{p_j q}$$

is derived from Strasser and Weber [1999] and can be used to test if a response Y and a covariate X are independent. The test statistic depends on the learning sample L_n, weights w, influence function h and transformation g[8]. This test statistic is standardized before it is used. The test statistic T is not only standardized but is also mapped to a scalar value. In order to map the (possible) vector T to a scalar value, one obvious choice is to take the maximum of the standardized test statistic. This yields the following standardized linear test statistic:

$$c(\mathbf{t}, \mu, \Sigma) = \max_{k=1,...,pq} \left| \frac{(\mathbf{t} - \mu)_k}{\sqrt{(\Sigma)_{kk}}} \right|$$

The variable t is an actual value of the test statistic T, μ is the expected value of T under independence and $(\Sigma)_{kk}$ is the k-th diagonal entry of the covariance matrix[9]. Thus c is the test statistic which is calculated for each permutation. Extreme values for c for the observed data compared to the permutations will lead to the rejection of the null hypothesis of independence.

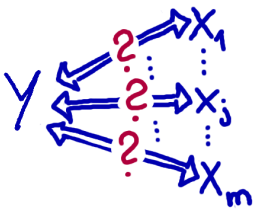
[7] This test statistic might look difficult in the first place, because it is a very general formulation. When looking at particular examples the formula becomes very friendly and seems more natural. This can be seen in the two included examples for regression 13 and classification 19

[8] read more on page 29

[9] see page 29 for the calculation of μ and Σ of T

Stop criteria and variable selection

The first step in each partition is to test if the partition should be split at all. This is formulated as a statistical hypothesis test with the global[10] null hypothesis of independence. It is composed of many (local) null hypotheses of independence between the response Y and each covariate X_j, which can be reformulated in terms of the



[10] Note that global means global for the partition but not for the whole tree

marginal distribution of Y. If Y and X_j are independent, the distribution of Y given X_j is the same as the marginal distribution of Y. Or more formally:

H_0 : The response Y is independent from all covariates $X_j, \quad j \in 1, \ldots, m$

$$H_0 = \cap_{j=1}^{m} H_0^j \text{ and } H_0^j : D(\mathbf{Y}|X_j) = D(\mathbf{Y})$$

One way to test for such a compound hypothesis is to test each of the m [11] hypothesis separately and to reject the global null hypothesis if Y is dependent of at least one of the covariates. To overcome the problem of multiple testing a p-value correction has to take place. The whole procedure:

[11] Let m be the number of covariates

1. Choose an influence function h depending on scale of Y
2. For each covariate X_j do the following:
   1) Choose an appropriate function g_j, which depends on the scale of the covariate X_j [12]
   2) Calculate the test statistic c_{j0} for the observed data.
   3) Permute the observations in the node
   4) Calculate c for all permutations
   5) Calculate the p-values[13] (number of test statistics c, where $|c| > |c_0|$
3. Correct p-value for multiple testing[14]
4. p-value $< \alpha$?[15] $\Rightarrow$ reject global H_0 and search variable for splitting else don't split.

[12] The functions h and g_j stays the same for each partition, so in theory it is enough to choose them once before the first split

[13] Each X_j gets an own p-value

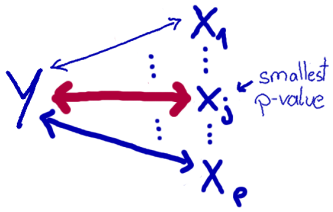
[14] Result is the p-value for the global test

[15] *alpha* is the a priori specified confidence level

Assuming the null hypothesis of independence was rejected, then the next step is to find the best variable for the split. While CART takes the variable which increases a criterion most (which leads to the variable selection bias), the conditional trees algorithm uses the p-value for variable selection. By using the p-values, the question of association strength is switched to statements of probability, where the original scales don't matter.

Thus the covariate with the smallest p-value is taken for the next split. Smallest p-value means the smallest probability of indepen-

dence between the chosen covariate and the response. From the procedure it should get clear, why variable selection and stop criteria are combined in one step. For testing the global null hypothesis every single p-value from each test is needed. These are also needed for the variable selection.



Splitting criteria

Steps 1) and 2) of the algorithm are completed. The covariate X_{j*} with the strongest association is chosen for the next split. Every covariate (which is not binary) has more than one possible split points. To determine where to split, a criterion which measures the goodness of the split has to be applied. The CART algorithm uses Gini for classification and sum of squares for regression. Both of the criteria could be used by the conditional tree algorithm as well, but the used approach is different. Because of the different types of possible regression- /classification - models (categorical, ordinal, numeric, censored, ...) a more general approach is suitable. The solution is again the test statistic derived from Strasser and Weber [1999]. A special case of the linear test statistic can be used, the formula is:

$$T_j^A(L_n, w) = vec\left(\sum_{i=1}^{n} w_i I(X_{j*i} \in A) \cdot h(Y_i, (Y_1, \ldots, Y_n))^T\right)$$

with A being a possible partition of the current observations and

$$I(X_{j*i} \in A) = \begin{cases} 1, & X_{j*i} \in A \\ 0 & X_{j*i} \notin A \end{cases}$$

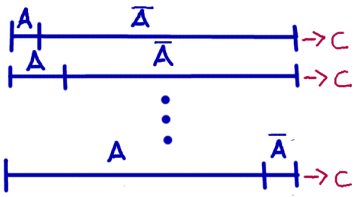
The difference to the test statistic for the association test is the transformation of X_{j*}. We only look at the different partitions of X_{j*}. Therefore the scale of X_{j*} is not of any interest anymore, but the partition which emerges by a certain split point. An appropriate function to capture only the difference in the partition, the transformation of X_{j*} is the indicator function. This results in the statistic

$$c_{max}(\mathbf{t}, \mu, \Sigma) = max_k \left| \frac{(\mathbf{t}^A - \mu)_k}{\sqrt{(\Sigma)_{kk}}} \right|$$

Note that the test statistic does not depend on the transformation g_{j*} of X_{j*}. The scale of the chosen covariate X_{j*} is regarded in another way: The possible partitions A and A^C depend on the scale of X_{j*}. For example if X_{j*} is categorical the different partitions A are combinations of the different categories. A continuous covariate will be searched for a split on the real line. Ordinal covariates will be ordered by categories and then searched for the split.

We search the partition A which maximizes c:

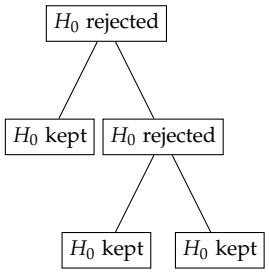
$$A^* = argmax_A c(t^A_{j*}, \mu^A_{j*}, \Sigma^A_{j*})$$



Maximizing c means, that we search for the partition with the strongest deviation of $h(Y)$ from what we would expect under independence between Y and X_{j*}. Additional stopping criteria like stopping, when the resulting partitions would become to small, can be implemented by restricting the searched split points.

Repeat

The three steps: stop criteria, variable selection and best split search are repeated until in every partition, which is at the end of the tree (so called terminal node) the null hypothesis of independence cannot be rejected anymore.



If there is no significant dependency of the response and any covariate, there would be no split at all and therefore every observation would in the same partition.

The user has to specify the significance level α, which is the significance level for each global null hypothesis. The value α can also be seen as a tuning parameter for the trees. Low values yield smaller and larger values larger trees, because a larger α leads to rejecting the null hypothesis more frequently and thus splitting more frequently.

Regression example: bodyfat

The first example is a continuous regression model, where both the response and the covariates are measured on a numeric scale. The model is illustrated with the *bodyfat* data set, which is available in the mboost package developed by Hothorn et al. [2012].

Y, X_j numeric, $\quad j = 1, \ldots, m$

Data set

The data set contains observations of 71 healthy women. The data contains body fat values, which are measured by DXA (Dual-energy X-ray absorptiometry), a method to determine the amount of body fat. Other variables in the data set are anthropometric measurements like the breadth of the knee, the waist circumference, the hip circumference etc.. The objective is to predict the body fat with the anthropometric measurements, because the DXA method is more expensive and not always available.

$n = 71$

Predict bodyfat with body measurements as input

Test statistic

Bodyfat measured by DXA as well as body measurements are numeric variables. Thus one possible choice for the influence function h and the transformation function $g_j, \quad j = 1, \ldots, m$ is the identity function, which means the variables will not be transformed at all.

Thus:

$$h = \mathbf{Y}_i \quad \text{and} \quad g_j = \mathbf{X}_j \quad j = 1, \ldots, m$$

This yields following not-standardized test statistic:[16]

$$\mathbf{T}_j(\mathcal{L}_n, \mathbf{w}) = \sum_{i=1}^{n} w_i \mathbf{X}_{ji} \mathbf{Y}_i = \sum_{i \in node} \mathbf{X}_{ji} \mathbf{Y}_i$$

[16] The formulation $\sum_{i \in node}$ means sum over all observations in the node (= partition). This is the same as the sum over all observations with additional weights, because only observations in the current node have weight $w = 1$, the others have weights $w = 0$

The next step is to standardize the test statistic:[17] [18]

$$c_{max}(\mathbf{t}, \mu, \Sigma) = \max_{k=1,\ldots,pq} \left| \frac{(t-\mu)_k}{\sqrt{(\Sigma)_{kk}}} \right| = \left| \frac{t-\mu}{\sqrt{\Sigma}} \right|$$

[17] Definitions of μ and Σ can be found on page 29

[18] $n_{node} := \sum_{i=1}^{n} w_i$ =Number of observation in node
$\bar{Y}_{node}$: Mean of Y in node
$\bar{X}_{j,node}$: Mean of X_j in node

$$\begin{aligned}
\mu_j &= \left(\sum_{i=1}^{n} w_i X_{ji} \right) \mathbb{E}_{node}(h) = n_{node} \cdot \bar{X}_{j,node} \bar{Y}_{node} \\
\Sigma &= \frac{n_{node}}{n_{node}-1} \mathbb{V}_{node}(h) \cdot \sum_{i=1}^{n} w_i X_{ji}^2 - \frac{1}{n_{node}-1} \mathbb{V}_{node}(h) n_{node}^2 \bar{X}_{j,node}^2 \\
&= \frac{1}{n_{node}} \sum_{i \in node} (Y_i - \bar{Y}_{node})^2 \sum_{i=1}^{n} X_{ji}^2 \\
&\quad - \frac{1}{n_{node}-1} \cdot \frac{1}{n_{node}} \sum_{i \in node} (Y_i - \bar{Y}_{node})^2 n_{node}^2 \bar{X}_{j,node}^2 \\
&= \frac{1}{n_{node}-1} \sum_{i \in node} (Y_i - \bar{Y}_{node})^2 \left(\sum_{i \in node} X_{ji}^2 - n_{node} \bar{X}_{j,node}^2 \right) \\
&= \frac{1}{n_{node}-1} \left(\sum_{i \in node} (Y_i - \bar{Y}_{node})^2 \right) \left(\sum_{i \in node} (X_{ji} - \bar{X}_{j,node})^2 \right)
\end{aligned}$$

Therefore:

$$c \propto \left| \frac{\sum_{i \in node} X_{ji} Y_i - n_{node} \bar{X}_{j,node} \bar{Y}_{node}}{\sqrt{\left(\sum_{i \in node} (Y_i - \bar{Y}_{node})^2 \right) \left(\sum_{i \in node} (X_{ji} - \bar{X}_{j,node})^2 \right)}} \right|$$

The linear test statistic is proportional to Pearson's correlation coefficient. This means, that the permutation test is testing if the correlation between Y and any X_j is different than zero. Thus by choosing the identity function for h and g_j the null hypothesis of independence between Y and X_j is formulated as "The correlation between Y and X_j is zero".

The next step is to calculate the test statistic (the Pearson's correlation coefficient multiplied with a constant) for the observation in the current partition (where $w \neq 0$). The response of the observations will be permuted and the test statistic calculated again. This will be done often enough to approximate the distribution of the test statistic for the sample. If the correlation coefficient of the original data is very extreme compared to the permuted test statistics, the p-value

will be very low.

The procedure of calculating the test statistic for the original data and the permutations is done for every covariate $X_j, \quad j \in 1, \ldots, m$ separately.

Test statistic for splitting criteria

For regression and the identity function for the influence function h, the statistic used to find the best split is the following: [19]

[19] n_A: Number of observations in partition A
$\bar{Y}_A$: Mean of Y in A

$$\mathbf{T}^A_{j*}(\mathcal{L}_n, \mathbf{w}) = \sum_{i=1}^{n} w_i I(X_{j*i} \in A) \cdot Y_i = \sum_{i: x_{j*i} \in A} Y_i = n_A \bar{Y}_A$$

$$\mu^A_{j*} = \sum_{i=1}^{n} w_i I(X_{j*i} \in A) \cdot \frac{1}{n_{node}} \sum_{i=1}^{n} w_i Y_i = n_A \bar{Y}_{node}$$

$$\begin{aligned}
\Sigma^A_{j*} &= \frac{n_{node}}{n_{node} - 1} \frac{1}{n_{node}} \sum_{i=1}^{n} w_i (Y_i - \bar{Y}_{node})^2 \cdot \sum_{i=1}^{n} w_i I(X_{j*i} \in A)^2 \\
&\quad - \frac{1}{n_{node} - 1} \frac{1}{n_{node}} \sum_{i=1}^{n} w_i (Y_i - \bar{Y}_{node})^2 \cdot \left(\sum_{i=1}^{n} w_i I(X_{j*i} \in A) \right)^2 \\
&= \frac{1}{n_{node} - 1} \sum_{i \in node} (Y_i - \bar{Y}_{node})^2 n_A \left(1 - \frac{n_A}{n_{node}} \right) \\
&= Var(Y_{node}) \cdot Var(Z)
\end{aligned}$$

with

$$Z \sim B(n_{node}, \pi = \frac{n_A}{n_{node}})$$

Can be interpreted as the probability that z observations would be assigned to A if the process of assigning would be random with probability $\frac{n_A}{n_{node}}$. Thus the standardized test statistic is:

$$c_{max}(\mathbf{t}^A_{j*}, \mu^A_{j*}, \Sigma^A_{j*}) = max_k \left| \frac{(\mathbf{t}^A - \mu)_k}{\sqrt{(\Sigma)_{kk}}} \right| = n_A \left| \frac{\bar{Y}_A - \bar{Y}_{node}}{\sqrt{Var(Y_{node}) \cdot Var(Z)}} \right|$$

The partition which maximizes the above expression will be chosen. Maximizing c_{max} means finding the partition, where the difference between the mean of Y in the partition and the mean of Y in the whole partition is large and the number of observations in A is large

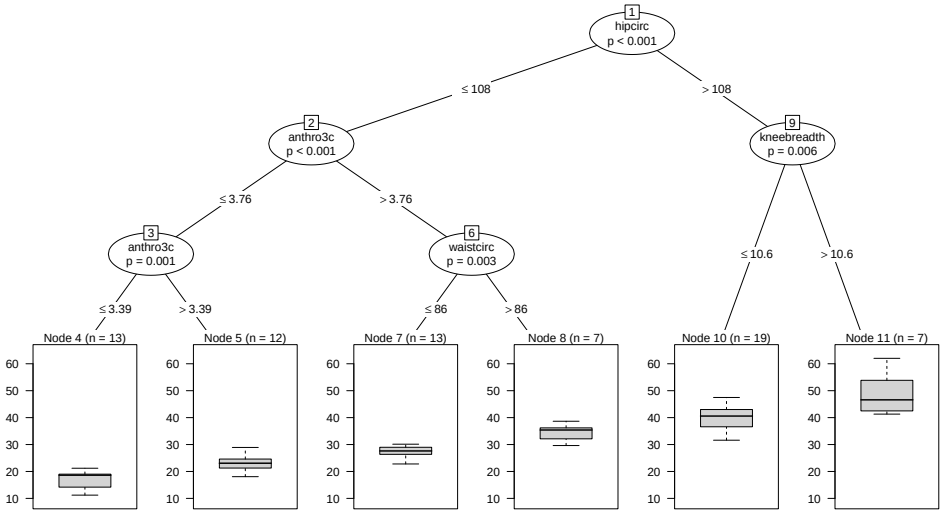


Figure 1: Conditional tree for bodyfat data

at the same time.

R-Code

The data set can be loaded from mboost (Hothorn et al. [2012]) package using the `data()` function:

```
data(bodyfat, package = "mboost")
```

The conditional tree algorithm is implemented in the party package (Hothorn et al. [2006]), which is available on `http://cran.r-project.org/`. The usage is similar to the `lm()`-function, with the formula interface. The formula `DEXfat ~ .` means that the tree should model the response DEXfat (bodyfat measurement) depending on all available covariates.

```
library("party")
## fit a conditional tree
cond_tree <- ctree(DEXfat ~ ., data = bodyfat)
plot(cond_tree)
```

The result is a tree (Figure 1) with six terminal nodes (five splits). The variable chosen for the first split is `hipcirc`, the circumference of the hip in cm. If it is bigger than 108cm the next measurement

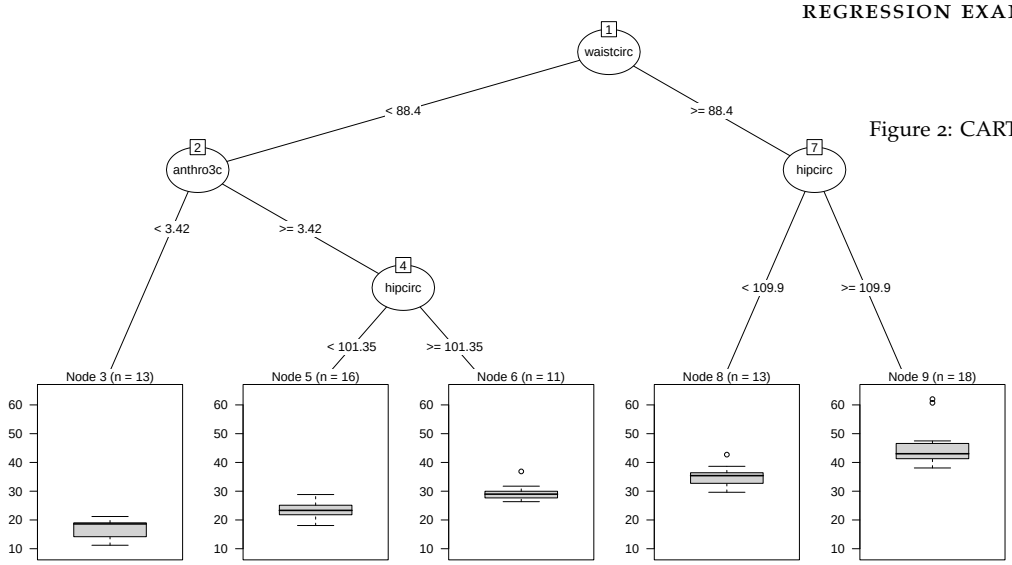


Figure 2: CART tree for bodyfat data

to look at is the breadth of the knee (`kneebreadth`). If the breadth is smaller than 10.6 cm the estimated is 39.7, which is equal to the mean in this terminal node. The interpretation for the other nodes is equivalent.

```
library("rpart")
library("partykit")

##
  ## Attaching package:  'partykit'
## The following object(s) are masked from 'package:party':
  ##
  ##    ctree, ctree_control, edge_simple, node_barplot,
node_boxplot,
  ##    node_inner, node_surv, node_terminal

cart <- rpart(DEXfat ~ ., data = bodyfat)
plot(as.party(cart))
```

The CART algorithm gives us a different tree (Figure 2). In comparison, CART uses the circumference of the waist for the first split while the conditional trees algorithm uses the same measurement of the hip. There is even a covariate (kneebreadth) which is used by the conditional trees algorithm, but not used by the CART algorithm. The resulting trees are structurally different.

Classification example: glaucoma

This is an example of a classification tree with the conditional trees algorithm. The response is binary and all the covariates are measured on a numeric scale. The data for this illustration is available in the `ipred` package (Peters and Hothorn [2012]).

Y binary, X_j numeric $\quad j = 1, \ldots, m$

Data set

The glaucoma data set contains eye laser scanning of both healthy persons and persons with glaucoma. Glaucoma is an eye disease which in worst case leads to blindness. If the person is healthy the response variable `Class` equals zero else if the person suffers from glaucoma it has the value one. The subject is to predict if a person has the glaucoma disease or not, based on different laser scanning measurements. Measured are different volumes and surfaces of the eye

n = 196

Predict glaucoma based on eye measurements

Test statistic

The test statistic is different to the one from the regression example. The transformation g for the covariates stays the same, while the influence function $h(Y)$ is different, as the scale of Y is different in the classification example. Instead of using the identity for h, the influence function is a vector with the same dimensionality as the number of categories, two-dimensional in the glaucoma example. The vector $h(Y_i)$ equals one at the position k when observation i is in the particular category, and zeros at the other positions. The glaucoma data set has two classes, glaucoma and normal. Thus if person i has

glaucoma, the vector $h(Y_i)$ is $(0,1)^T$.

$$h = e_J(\mathbf{Y}_i) = \begin{pmatrix} Y_{G,i} \\ Y_{N,i} \end{pmatrix} = \begin{cases} (1,0)^T & \text{Glaucoma} \\ (0,1)^T & \text{normal} \end{cases} \quad \text{and} \quad g(\mathbf{X}_{ji}) = \mathbf{X}_{ji}$$

This yields the following linear test statistic:[20]

$$\mathbf{T}_j(\mathcal{L}_n, \mathbf{w}) = vec\left(\sum_{i=1}^{n} w_i \mathbf{X}_{ji} e_J(\mathbf{Y}_i)^T\right) = \begin{pmatrix} n_G \cdot \bar{X}_{j,G} \\ n_N \cdot \bar{X}_{j,N} \end{pmatrix}$$

[20] n_N: Number of healthy persons in the node
n_G: Number of persons with glaucoma in node

The test statistic is the vector of the means of X_j in the categories glaucoma and normal, weighted by the number of observations in this category. Again we need mean and variance to standardize T_j:

$$\begin{aligned}
\mu_j &= \sum_{i \in node} X_{ji} \frac{1}{n_{node}} \begin{pmatrix} n_G \\ n_N \end{pmatrix} = \begin{pmatrix} n_G \cdot \bar{X}_{j,node} \\ n_N \cdot \bar{X}_{j,node} \end{pmatrix} \\
\Sigma_j &= \frac{n_{node}}{n_{node} - 1} V_{node}(h) \cdot \sum_{i \in node} X_{ji}^2 - \frac{1}{n_{node} - 1} \left(\sum_{i \in node} X_{ji}\right) \left(\sum_{i \in node} X_{ji}\right) \\
&= \frac{1}{n_{node} - 1} V_{node}(h) n_{node} \sum_{i \in node} \left(X_{ji} - \bar{X}_{j,node}\right)^2 \\
V_{node}(h) &= \frac{1}{n_{node}} \sum_{i \in node} \left(e_J(Y_i) - \begin{pmatrix} \frac{n_G}{n_{node}} \\ \frac{n_N}{n_{node}} \end{pmatrix}\right) \left(e_J(Y_i) - \begin{pmatrix} \frac{n_G}{n_{node}} \\ \frac{n_N}{n_{node}} \end{pmatrix}\right)^T \\
&= \frac{1}{n_{node}} \sum_{i \in node} \begin{pmatrix} Y_{G,i} - \frac{n_G}{n_{node}} \\ Y_{N,i} - \frac{n_N}{n_{node}} \end{pmatrix} \begin{pmatrix} Y_{G,i} - \frac{n_G}{n_{node}} \\ Y_{N,i} - \frac{n_N}{n_{node}} \end{pmatrix}^T \\
&= \frac{1}{n_{node}} \sum_{i \in node} \begin{pmatrix} Y_{G,i} - \bar{Y}_G \\ Y_{N,i} - \bar{Y}_N \end{pmatrix} \begin{pmatrix} Y_{G,i} - \bar{Y}_G \\ Y_{N,i} - \bar{Y}_N \end{pmatrix}^T \\
&= \frac{1}{n_{node}} \begin{pmatrix} \sum_{i \in node} (Y_{G,i} - \bar{Y}_G)^2 & \sum_{i \in node} (Y_{G,i} - \bar{Y}_G)(Y_{N,i} - \bar{Y}_N) \\ \sum_{i \in node} (Y_{G,i} - \bar{Y}_G)(Y_{N,i} - \bar{Y}_N) & \sum_{i \in node} (Y_{N,i} - \bar{Y}_N)^2 \end{pmatrix}
\end{aligned}$$

For the standardized test statistic only the diagonal elements of the covariance matrix are needed, because only the variance within the

classes are considered:

$$(\Sigma_j)_{kk} = \frac{1}{n_{node}-1}\sum_{i\in node}(Y_{Class}-\bar{Y}_{Class})^2\sum_{i\in node}(X_{ji}-\bar{X}_{ji})^2 \quad \text{Class} \in \{G,N\}$$
$$= \underbrace{n_{node}\bar{Y}_{Class}(1-\bar{Y}_{class})}_{\widehat{Var}(Y_{Class,node})}\underbrace{\frac{1}{n_{node}-1}\sum_{i\in node}\left(X_{ji}-\bar{X}_{j,node}\right)^2}_{\widehat{Var}(X_{j,node})}$$

And finally the standardized test statistic is denoted as:

$$c = \frac{n_{Class}\bar{X}_{j,Class} - n_{Class}\bar{X}_{j,node}}{\sqrt{(\Sigma)_{kk}}} = n_{Class}\frac{(\bar{X}_{j,Class}-\bar{X}_{j,node})}{\sqrt{\widehat{Var}(Y_{Class,node})\widehat{Var}(X_{j,node}))}}$$

This standardized test statistic has a very vivid interpretation. The numerator is the difference between the observed mean of the covariate X_j for all the observations in the node which are in the class we are looking at and the mean of X_j in the whole node. We expect $\bar{X}_{j,Class}$ to be very similar to $\bar{X}_{j,node}$ under the null hypothesis of independence of X_j and Y. Thus in case of independence between response and covariate the numerator should be very small. The denominator contains the root of the product of variance Y and variance X_j. It adjusts the difference. The whole term is weighted by n_{Class}.

Test statistic for splitting criteria

Finding the best split point differs to the regression example as well. The test statistic T^A_{j*} looks like this:

$$T^A_{j*}(\mathcal{L},w) = vec\left(\sum_{i\in node} I(X_{j*i}\in A)e_j(Y_i)^T\right) = \frac{1}{n_{node}}\sum_{i\in node} I(X_{j*i}\in A)\begin{pmatrix}Y_{G,i}\\ Y_{N,i}\end{pmatrix} =$$
$$= \frac{1}{n_{node}}\begin{pmatrix}\sum\limits_{i\in A} Y_{G,i}\\ \sum\limits_{i\in A} Y_{N,i}\end{pmatrix} = \frac{1}{n_{node}}\begin{pmatrix}\frac{1}{n_A}n_{G,A}\\ \frac{1}{n_A}n_{N,A}\end{pmatrix}$$

The mean and variance of this test statistic are:

$$\begin{aligned}
\mu_{j*} =& \mathbb{E}(T|S) = vec\left(\sum_{i \in node} I_A(X_{j*i})\mathbb{E}(e_J(Y)|S)\right) = \\
=& \left(\sum_{i \in node} I_A(X_{j*i})\right)\left(\frac{1}{n_{node}} \sum_{i \in node} \begin{pmatrix} Y_{G,i} \\ Y_{N,i} \end{pmatrix}\right) = \\
=& \frac{n_A}{n_{node}} \begin{pmatrix} n_G \\ n_N \end{pmatrix} \\
\Sigma_{j*} =& \frac{1}{n_{node} - 1} \mathbb{V}(h)(n_{node} n_A - n_A^2)
\end{aligned}$$

$$\Rightarrow (\Sigma_{j*})_{kk} = \frac{1}{n_{node} - 1} \frac{1}{n_{node}} \underbrace{(Y_{Class} - \bar{Y}_{Class})^2}_{= n_{node} \bar{Y}_{Class}(1 - \bar{Y}_{Class}) = \hat{\mathbb{V}}(Y_{Class})} (n_{node} n_A - n_A^2) = \hat{V}(Y_{Class}) \frac{n_A}{n_{node} - 1} \left(1 - \frac{n_A}{n_{node}}\right)$$

This yields following standardized test statistic c:

$$c_{max,Class} = \frac{\frac{1}{n_{node}} n_{class,A} - \frac{n_A}{n_{node}} n_{class}}{(\Sigma_j)_{class}} = \frac{\frac{1}{n_{node}}(n_{class,A} - \frac{n_A \cdot n_{class}}{n_{node}})}{\hat{\mathbb{V}}(Y_{class}) \frac{n_A}{n_{node} - 1}(1 - \frac{n_A}{n_{node}})} \quad \text{Class} \in \{G, N\}$$

The enumerator contains the difference between the actual number of observations with glaucoma (or normal) in partition A and the number of observations in A with glaucoma (or normal) under complete randomness.

R-Code

```
library("rpart")
library("party")
data("GlaucomaM", package = "ipred")
cond_tree <- ctree(Class ~ ., data = GlaucomaM)
classic_tree <- rpart(Class ~ ., data = GlaucomaM)
plot(cond_tree)
```

```
plot(as.party(classic_tree), cex = 1.5)
```

Like in the regression example, the tree grown with CART and conditional inference are structurally different. The measurement

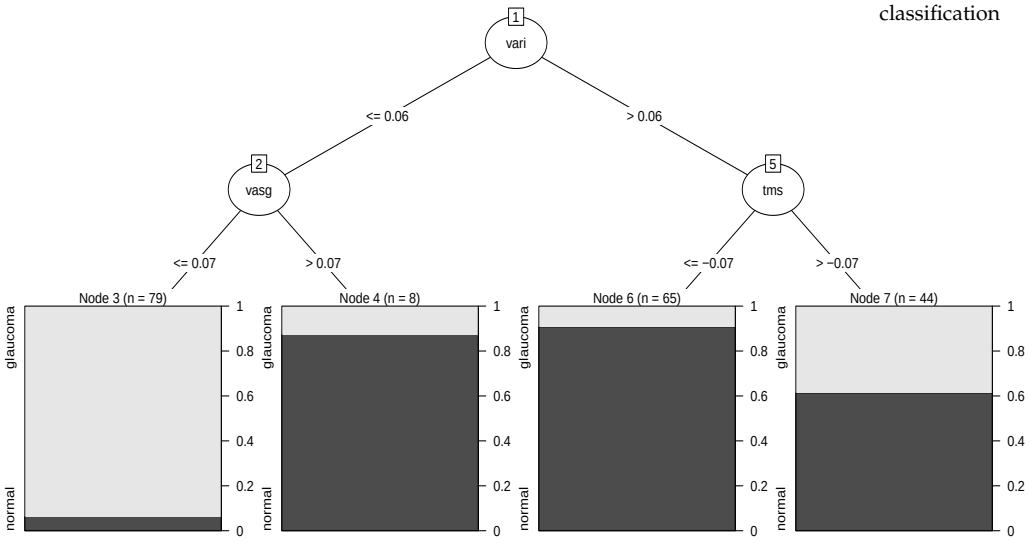


Figure 3: Conditional tree for Glaucoma classification

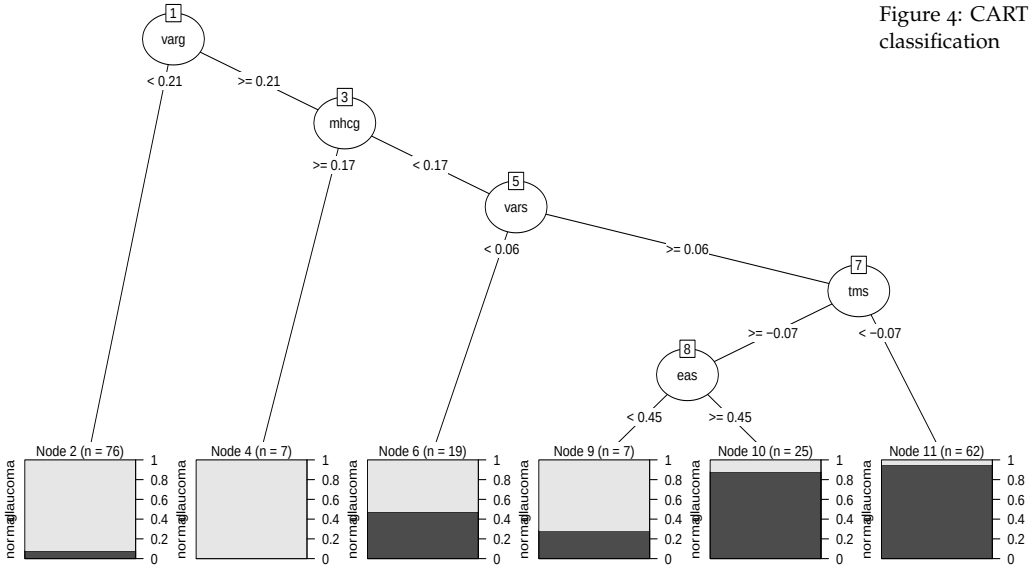


Figure 4: CART tree for Glaucoma classification

vari which is used for the first split in the conditional tree (Figure 3) is not even used for any split in the CART tree (Figure 4. Interestingly Hothorn et al. [2006] show, that the predictive accuracy of both approaches are very similar.

Summary

Conditional inference trees are a statistical approach to recursive partitioning. The underlying theory is embedded in a well defined framework of conditional inference (permutation tests). Unlike most other partitioning algorithms, variable selection and best split search are separated. The split criterion and the variable selection for the next split are formulated in terms of statistical hypothesis tests. The framework covers a wide range of different scales for Y and X, more than mentioned in this paper (e.g. ordinal regression, censored regression).

The CART algorithm finds application in a very powerful predictive algorithm called RandomForest. A RandomForest combines many trees in a randomized fashion, which often results in a very good prediction tool. Conditional inference trees can also be used to populate a RandomForest. This is implemented in the party package with the `cforest`-function.

appendix: Permutation tests

Permutation tests[21] are a special class of non-parametric tests. In contrast to parametric tests, there is no distribution assumption for the data. The philosophy of permutation tests is, that under the null hypothesis of independence, the pairs of (Y_i, X_i) are exchangeable. Exchangeability is not only the philosophy behind, but a requirement to perform a permutation test. That means, we can obtain the distribution of a test statistic (under the assumption of independence) by calculating the test statistic for all possible permutations of Y and X. The significance value p is obtained by locating the observed statistic in the distribution.

One problem of permutation tests is that the number of possible permutations grows very fast with a larger number n of observations. The number of permutations is $n!$ and the calculation time for all possible test statistics is for many problems unreasonably long. The trick here is to draw just a number of random permutation to approximate the distribution of the test statistic. This is called Monte Carlo method.

[21] Synonyms are: exact tests, randomization tests, re-randomization tests

appendix: Test statistic

Hothorn et al. [2006] use the following test statistic which is derived from Strasser and Weber [1999]:

$$\mathbf{T}_j(L_n, w) = vec\left(\sum_{i=1}^{n} w_i g_j(X_{ij}) h(Y_i, (Y_1, ..., Y_n))^T\right) \in \mathbb{R}^{p_j q}$$

It may look difficult in the first place, but it can be broken down to the following parts:

$vec()$ The core of the test statistic can be a matrix. In this case $vec()$ - operator vectorizes the matrix

$\sum_{i=1}^{n}$ The test statistic is a sum over all observations.

w Observations which are not in the current partition will get the weight $w = 0$ and otherwise $w = 1$. This ensures us, that only the data in the current node is in focus.

g_j A transformation of the j-th covariate X_j. Transformation depends on scale of the covariate.

h Influence function. Transformation of the response Y.

Expectation and variance of the test statistic are derived by the frame-

work from Strasser and Weber [1999]:

$$\mu_j = \mathbb{E}(\mathbf{T}_j(\mathcal{L}_n, \mathbf{w})|S(\mathcal{L}_n, \mathbf{w})) = vec\left(\left(\sum_{i=1}^{n} w_i g_j(X_{ji})\right)\mathbb{E}(h|S(\mathcal{L}_n, \mathbf{w}))^T\right)$$

$$\begin{aligned}\Sigma_j &= \mathbb{V}(\mathbf{T}_j(\mathcal{L}_n, \mathbf{w})|S(\mathcal{L}_n, \mathbf{w})) \\ &= \frac{n_{node}}{n_{node}-1}\mathbb{V}(h|S(\mathcal{L}_n, \mathbf{w})) \otimes \left(\sum_i w_i g_j(X_{ji}) \otimes w_i g_j(X_{ji})^T\right) \\ &- \frac{1}{n_{node}-1}\mathbb{V}(h|S(\mathcal{L}_n, \mathbf{w})) \otimes \left(\sum_i w_i g_j(X_{ji})\right) \otimes \left(\sum_i w_i g_j(X_{ji})\right)^T\end{aligned}$$

$$n_{node} = \sum_{i=1}^{n} w_i$$

$$\mathbb{E}(h|S(\mathcal{L}_n, \mathbf{w})) = n_{node}^{-1}\sum_i w_i h(\mathbf{Y}_i, (\mathbf{Y}_1, \ldots, \mathbf{Y}_n)) \in \mathbb{R}^q$$

$$\begin{aligned}\mathbb{V}(h|S(\mathcal{L}_n, \mathbf{w})) = n_{node}^{-1}\sum_i w_i (&h(\mathbf{Y}_i, (\mathbf{Y}_1, \ldots, \mathbf{Y}_n)) - \mathbb{E}(h|S(\mathcal{L}_n, \mathbf{w}))) \\ &(h(\mathbf{Y}_i, (\mathbf{Y}_1, \ldots, \mathbf{Y}_n)) - \mathbb{E}(h|S(\mathcal{L}_n, \mathbf{w})))^T\end{aligned}$$

For the permutation tests, the standardized version of the test statistic T is used:

$$c(\mathbf{t}, \mu, \Sigma) = \max_{k=1,\ldots,pq}\left|\frac{(\mathbf{t}-\mu)_k}{\sqrt{(\Sigma)_{kk}}}\right|$$

Bibliography

L. Breiman, J. Friedman, R. Ohlsen, and C. Stone. Classification and regression trees. *Wadsworth International Group*, 1984.

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer Series in Statistics, 2001.

T. Hothorn, K. Hornik, and A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006.

Torsten Hothorn, Peter Buehlmann, Thomas Kneib, Matthias Schmid, and Benjamin Hofner. *Model-Based Boosting*, 2012. URL `http://CRAN.R-project.org/package=mboost`. R package version 2.1-3.

John Mingers. Expert systems-rule induction with statistical data. *Journal of the operational research society*, pages 39–47, 1987.

Andrea Peters and Torsten Hothorn. *ipred: Improved Predictors*, 2012. URL `http://CRAN.R-project.org/package=ipred`. R package version 0.9-1.

H. Strasser and C. Weber. On the asymptotic theory of permutation statistics. *Report Series SFB "Adaptive Information Systems and Modelling in Economics and Management Science"*, 27., 1999.

Terry Therneau, Beth Atkinson, and Brian Ripley. *rpart: Recursive Partitioning*, 2012. URL `http://CRAN.R-project.org/package=rpart`. R package version 4.1-0.