Seminar paper

Conditional trees

Christoph Molnar

Supervisor:
Stephanie Möst

1. June 2012
Department of Statistics
University of Munich

Contents

Introduction

Recursive partitioning is a powerful yet simple tool in predictive and explanatory statistics. Models build by partitioning take on the form of decision trees, which makes the approach easy to understand for everyone without understanding the algorithm behind. The model partitiones the data to reconstruct the relationship

$$Y = f(X)$$

, where Y is called the response variable, which depends on a function f of the covariates matrix X. IMAGE OF TREE? Partitioning can be done with many different approaches and therefore the landscape of algorithms is very vivid. The differences of trees algorithms his the way trees are grown. They can be divided into those which can do regression, those which can do classification and those which can do both. Another characteristic is the number of split per partition step. There are binary splits, which divides the partition into two new partitions and multiway splits which yield more than two partitions. The variety gets big in the philosophy of how to determine which variable to take for the next step and where to split it. The point where the tree is not grown any more differs for the algorithms. Somewhere between all of those algorithms is the conditional trees framework.

Motivation

Problems of trees

Recursive partitioning suffers from different problems, some of which are already solved by some approaches. CART (Classification And Regression Trees) is a famous and widely used example of partitioning algorithms. Let us take a closer look at the problems with CART as example and how the conditional trees approach solves them.

Overfitting

If a tree is allowed to grow full length, pathological split could 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 some criterion is not fullfilled. This criterion could be a minimum number of observations in a node. Pruning let's the tree grow at first and prunes the leafs back afterwards. The CART algorithm uses both early stopping and pruning to avoid overfitting.

High variance

As the tree strongly depends on the first splits, different variables at for the first split can yield two structurally different trees. Therefore trees (also CART) are sensitive to variance in the data and resulting trees themselves have a **high variance**.

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.

Heuristic approach, lack of statistical model

The next split is just a heuristic, as the algorithm only searches for the next best split (like CART). Conditional trees algorithm measures

the association and uses the covariate with the strongest association with the response variable. The algorithm is embeded in a well-defined framework of hypothesis.

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 as well as classification. 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, just to name two.

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

Conditional trees

“Classic" 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.

Recursive partitioning with the conditional trees algorithm works different and can be summarized as follows:

In contrast to CART or other algorithms, variable selection and the search for the best split are strictly seperated. 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. Only then, after choosing, the covariate 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. For a quick introduction to permutation tests, please visit the appendix. LINK. A good idea of permutation tests is required to understand the recursive partitioning algorithm.

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 algorithm uses test statistics suggested by a framework developed by Strasser and Weber QUOTE. They offer a very general formulatin of a test statistic for permutation tests, which can handle arbitrary scales for response and covariate. AND THEY HAVE NICE ASYMPTOTICS?? The formula is presented in detail later.

The following chapter explains the algorithm in detail.

Algorithm

All decisions are embedded into hypothesis tests. The conditional trees algorithm uses permutation tests to test the hypothesis of independence between a covariate and the response. This will be described further in the single steps of the algorithm.

The algorithm:

Permutation test related steps are written 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 splitted. If the answer is positive the variable with the strongest association with the response is chosen and the data set will be split into two partitions. The steps will be repeated within both of the new partitions. Covariates chosen for a split can be chosen again later (only in the case of a bivariate covariate it doesn't make sense). First if in all partitions the global null hypothesis of independence cannot be rejected (Stop criterion) the tree does not grow any further and the algorithm stops.

The next Sections describe in detail how the single steps work and especially how permutation tests are applied.

The test statistic

All decisions of the algorithm are embeded in hypothesis tests. These are done with permutation tests (conditional inference).

Strasser and Weber [LINK] have formulated a very general test statistic, which can be used to do a permutation test if a response Y and a covariate X are independent.

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

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 and classification LINK. At this point you can accept the test statistic as it is or you can read more in the appendix LINK about it.

This test statistic is standardized before it is used. $c(\mathbf{t}, \mu, \Sigma) = \max_{k=1,...,pq} \left| \frac{(\mathbf{t}-^-)_k}{\sqrt{(\Sigma)_{kk}}} \right|$

Formulas for mean and variance of T is also in the appendix LINK. Thus c is the test statistic which is calculated for each permutation and extreme values for c for the observed data compared to the permutations will lead to rejecting the null hypothesis of independence.

Stop criteria (1) and variable selection (2)

In every partition the first step of the algorithm asks: "Split at all?". This is formulated in a proper Null Hypothesis, the global null hypothesis. Note that global means here global in this partition and not the whole tree.

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

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

Each hypothesis H_0^j is tested seperatly. Most simple approach would be to look at the m resulting p-values and to rejected the global null hypothesis of independence if one the p-values exceeds the predetermined significance level α (e.g. $\alpha = 0.05$). Though multiple comparison has to be considered and any multiple testing procedure can be used at this part of the algorithm to determine t he if H_0 can be rejected.

Thus response is tested for dependency with each covariate seperately. This is done with permutation tests and the test statistic presented above.

1. Choose an influence function $h(Y)$
2. For each covariate X_j choose an apropriate function $g(X_j)$, which depends on the scale of the covariate.
   (a) Calculate the test statistic $c_j 0$ for the observed data.
   (b) Permutate the observations in the node
   (c) Calculate c for all permutations
   (d) Calculate p-value (number of test statistics c, where $|c| > |c_0|$
3. correct p-value for multiple testing
4. p-value $< \alpha$? $\Rightarrow$ reject global H_0 and search variable for splitting else don't split.

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

Thus the covariate with the smallest p-value is taken for the next split. The smallest p-value means the smallest probability of independence with 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 (3)

Steps 1) and 2) of the algorithm are completed. The covariate with the strongest association is chosen for the next partition step. Every covariate (which is not binary) has more than one possible split. 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 approach is different. Because of the different types of possible regression-/classification - models (categorial, ordinal, numeric, censored, ...) are more general approach is suitable. Again the test statistic framework from Strasser and Weber [CITE] can be used. A special linear test statistic, of the same kind, which is used for the stop criterion and variable selection, can be used. The formula is:

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

The difference to the test statistic for the association test is the transformation of X. We only look at the different partitions of X. Therefore the scale of X 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 is the indicator function. It is defined as: $I(X_{ji} \in A) = \begin{cases} 1, & X_{ij} \in A \\ 0 & X_{ij} \notin A \end{cases}$, where A is one possible partition. 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 categorial 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 searched for splits where the two resulting partitions are a result of a inbetween the ordered categories.

We search the A which maximizes:

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

Additional stopping criteria like stopping when the resulting partitions would become to small can be implemented by restricting the searched split points.

Repeat(4)

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 independency cannot be rejected anymore. If there is no significant dependency of the response and any covariate, than there would be no split at all, every observation is 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 recejecting the null hypothesis more frequently and splitting more often.

Continuous 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* available in the mboost LINK package.

The data set

The data set contains observations of 71 healthy women. The measurements contain body fat, which is 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.

The 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 $g_j, \quad \forall j \in 1, \ldots, m$ is the identity function, which means the variables will not be transformed at all.

Thus:

$$h = \mathbf{Y}_i$$

$$g = \mathbf{X}_i$$

This yields following not-standardized test statistic:

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

The next step is to standardize the test statistic:

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

With LINK TO MU AND SIGMA FORMULA :

CORRECT INDICES

$$\mu_j = \sum_{i=1}^{n} X_{ij} \mathbb{E}(h|S) = n \cdot \bar{X}_j \bar{Y} \tag{1}$$

$$\Sigma = \frac{n_{node}}{n_{node}-1} \mathbb{V}(h) \cdot \sum_{i=1}^{n} X_{ij}^2 - \frac{1}{n_{node}-1} \mathbb{V}(h) n_{node}^2 \bar{X}_j^2 \tag{2}$$

$$= \frac{1}{n_{node}} \sum_{i=1}^{n} (Y_i - \bar{Y})^2 \sum_{i=1}^{n} X_{ij}^2 - \frac{1}{n_{node}-1} \cdot \frac{1}{n_{node}} \sum_{i=1}^{n} (Y_i - \bar{Y})^2 n_{node}^2 \bar{X}_j^2 \tag{3}$$

$$= \frac{1}{n_{node}-1} \sum_{i=1}^{n} (Y_i - \bar{Y})^2 (\sum_{i=1}^{n} X_{ij}^2 - n_{node} \bar{X}_j^2) \tag{4}$$

$$= \frac{1}{n_{node}-1} \left(\sum_{i=1}^{n} (Y_i - \bar{Y})^2 \right) \left(\sum_{i=1}^{n} (X_{ij} - \bar{X})^2 \right) \tag{5}$$

Therefore:

$$c \propto \left| \frac{\sum_{i:node} X_{ij} Y_i - n_{node} \bar{X}_j \bar{Y}}{\sqrt{\left(\sum_{i:node} (Y_i - \bar{Y})^2 \right) \left(\sum_{i:node} (X_{ij} - \bar{X}_j)^2 \right)}} \right|$$

The linear test statistic is proportional to the pearson correlation coefficient. This means, that the permutation test tests if the correlation between Y and any X_j is different than 0. 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 correlation coefficient multiplied with a constant) for the observation in the current partition (where $w \neq 0$). The response of the observations

will be permutated 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 permutated 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 $X_j, \quad j \in 1, \ldots, m$ seperatly.

R-Code

The data set for the example is available in the mboost LINK package. In addition to the response DEXfat it contains the nine body measurements.

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

##    age DEXfat waistcirc hipcirc elbowbreadth kneebreadth anthro3a anthro3b
## 47  57  41.68     100.0   112.0          7.1         9.4     4.42     4.95
## 48  65  43.29      99.5   116.5          6.5         8.9     4.63     5.01
## 49  59  35.41      96.0   108.5          6.2         8.9     4.12     4.74
## 50  58  22.79      72.0    96.5          6.1         9.2     4.03     4.48
## 51  60  36.42      89.5   100.5          7.1        10.0     4.24     4.68
## 52  61  24.13      83.5    97.0          6.5         8.8     3.55     4.06
##    anthro3c anthro4
## 47     4.50    6.13
## 48     4.48    6.37
## 49     4.60    5.82
## 50     3.91    5.66
## 51     4.15    5.91
## 52     3.64    5.14
```

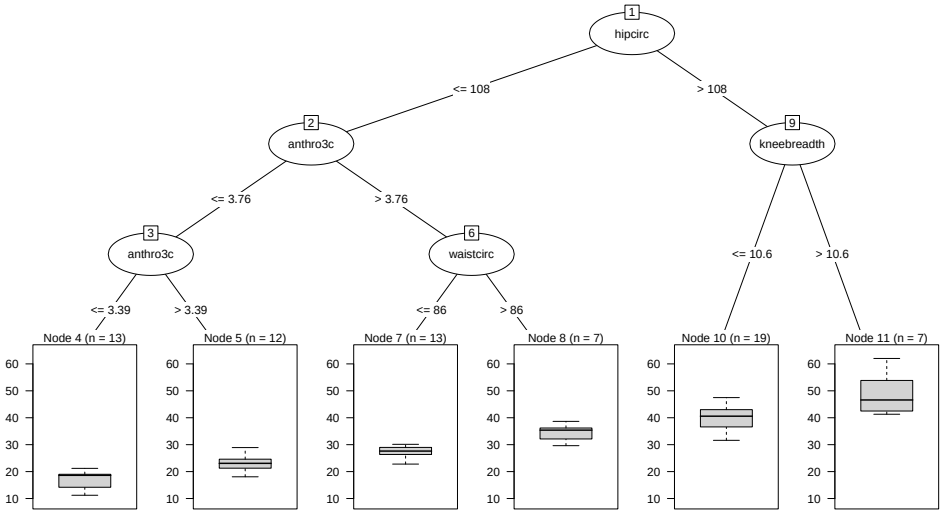
The conditional tree algorithm is implemented in the party package LINK, which is available on CRAN LINK. The usage is similiar to the `lm()`-function, with the formula interface. The formula `DEXfat` $\sim$ `.` means that the tree should modell the response DEXfat (bodyfat

measurement) depending on all available covariates.

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

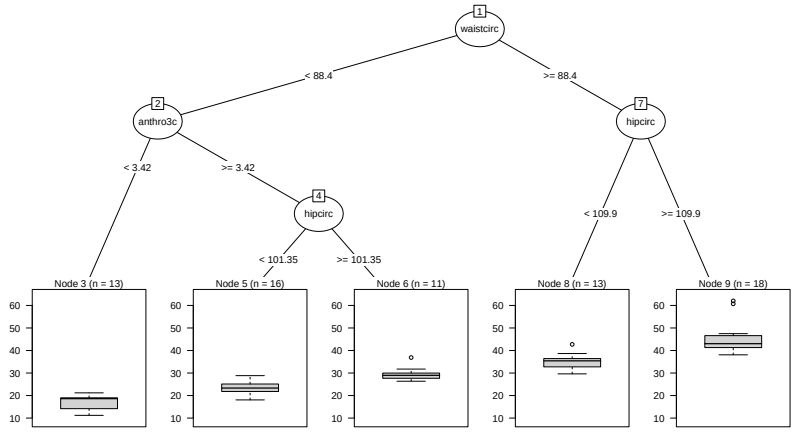
The result is a tree with six terminal nodes (five splits). The variable chosen for the first split is `hipcirc`, the circumference of the hip in cm. If it is bigger than 108cm the next measurement to look at is the breadth of the knee (`kneebreadth`). If the breadth is smaller than 10.6 cm the estimated is 39.7, which is equal to the mean in this terminal node.

```
plot(cond_tree)
```



The CART algorithm gives us a different tree.

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



For regression and the identitiy function for the influence function h, the test statistic is the following:

$$c_{max}(\mathbf{t}, \mu, \Sigma) = max_k \left| \frac{(\mathbf{t}^A - \mu)_k}{\sqrt{(\Sigma)_{kk}}} \right| \tag{6}$$

$$= \left| \frac{\sum_{i=1}^{n} w_i I(X_{ji} \in A) \cdot Y_i - \sum_{i=1}^{n} w_i I(X_{ji} \in A) \cdot n_{node}^{-1} \sum_{i=1}^{n} w_i Y_i}{\sqrt{\frac{n_{node}}{n_{node}-1} \frac{1}{n_{node}} \sum_{i=1}^{n} w_i (Y_i - \bar{Y}_{node})^2 \cdot \sum_{i=1}^{n} w_i I(X_{ji} \in A)^2 - \frac{1}{n_{node}-1} \frac{1}{n_{node}} \sum_{i=1}^{n} w_i (Y_i - \bar{Y}_{node})^2 \cdot \left(\sum_{i=1}^{n} w_i I(X_{ji} \in A) \right)^2}} \right| \tag{7}$$

$$= \left| \frac{\sum_{i: x_{ij} \in A} Y_i - n_A \bar{Y}_{node}}{\sqrt{\frac{1}{n_{node}-1} \sum_{i=1}^{n} (Y_i - \bar{Y}_{node})^2 n_A (1 - \frac{n_A}{n_{node}})}} \right| \tag{8}$$

$$= n_A \left| \cdot \frac{\bar{Y}_A - \bar{Y}_{node}}{\sqrt{\frac{1}{n_{node}-1} \sum_{i \in node} (Y_i - \bar{Y}_{node})^2 \cdot n_{node} (\frac{n_A}{n_{node}})(1 - (\frac{n_A}{n_{node}}))}} \right| \tag{9}$$

$$= n_A \left| \frac{\bar{Y}_A - \bar{Y}_{node}}{\sqrt{Var(Y) \cdot Var(Z)}} \right| \tag{10}$$

$$\tag{11}$$

with $Z \sim B(n_{node}, \pi = \frac{n_A}{n_{node}})$ Can be interpretated as the probability that z observations would be assigned to A if the process of aissigning would be random with probability $\frac{n_A}{n_{node}}$. Is maximal for $n_A = \frac{n_{node}}{2}$. The closer $\frac{n_A}{n_{node}}$ to 0.5 the bigger is c (assuming Y_A stays the same). Thus the test statistic favors bigger partitions. $n_{node} := \sum_{i=1}^{n} w_i \; n_{node}$: Number of observation in node $\bar{Y}_{node}$: Mean of Y

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

Additional stopping criteria like stopping when the resulting partitions would become to small can be implemented by restricting the searched split points.

Classification: Example glaucoma

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 illustration is available in the `ipred` package LINK.

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 can lead to blindness. If the person is healthy the response variable `Class` is zero else 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.

Test statistic

CHECK IF TRANSPOSITION T IS CORRECT

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 $h(Y)$ changes, 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 cateogry k, and zeros at the other positions. The Glaucoma data set knows two classes, Glaucoma and normal. Thus if observation i has Glaucoma

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

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

This yields the following linear test statistic:

$$\mathbf{T}_j(\mathcal{L}_n, \mathbf{w}) = vec\left(\sum_{i=1}^{n} w_i \mathbf{X}_{ij} e_J(\mathbf{Y}_i)^T\right) = \begin{pmatrix} n_{Glaucoma} \cdot \bar{X}_{j,Glaucoma} \\ n_{normal} \cdot \bar{X}_{j,normal} \end{pmatrix}$$

with mean and variance:

$$\mu_j = \sum_{i:node} X_{ji} \frac{1}{n_{node}} \begin{pmatrix} n_{Glaucoma} \\ n_{normal} \end{pmatrix} = \begin{pmatrix} n_{Glaucoma} \cdot \bar{X}_{j,node} \\ n_{normal} \cdot \bar{X}_{j,node} \end{pmatrix}$$

$$\Sigma = \frac{n_{node}}{n_{node}-1} V(h) \cdot \sum_{i:node} X_{ji}^2 - \frac{1}{n_{node}-1} \left(\sum_{i:node} X_{ij}\right)\left(\sum_{i:node} X_{ij}\right) = \frac{1}{n_{node}-1} V(h) n_{node} \sum_{i:node} \left(X_{ji} - \bar{X}_{j,node}\right)^2$$

$$\begin{aligned} V(h) &= \frac{1}{n_{node}} \sum_{i:node} \left(e_J(Y_i) - \begin{pmatrix} \frac{n_G}{n_{node}} \\ \frac{n_N}{n_{node}} \end{pmatrix}\right)\left(e_J(Y_i) - \begin{pmatrix} \frac{n_G}{n_{node}} \\ \frac{n_N}{n_{node}} \end{pmatrix}\right)^T \\ &= \frac{1}{n_{node}} \sum_{i:node} \begin{pmatrix} Y_G - \frac{n_G}{n_{node}} \\ Y_N - \frac{n_N}{n_{node}} \end{pmatrix}\begin{pmatrix} Y_G - \frac{n_G}{n_{node}} \\ Y_N - \frac{n_N}{n_{node}} \end{pmatrix}^T \\ &= \frac{1}{n_{node}} \sum_{i:node} \begin{pmatrix} Y_G - \bar{Y}_G \\ Y_N - \bar{Y}_N \end{pmatrix}\begin{pmatrix} Y_G - \bar{Y}_G \\ Y_N - \bar{Y}_N \end{pmatrix}^T \\ &= \frac{1}{n_{node}} \begin{pmatrix} (Y_G - \bar{Y}_G)^2 & (Y_G - \bar{Y}_G)(Y_N - \bar{Y}_N) \\ (Y_G - \bar{Y}_G)(Y_N - \bar{Y}_N) & (Y_N - \bar{Y}_N)^2 \end{pmatrix} \end{aligned}$$

Thus:

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

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.

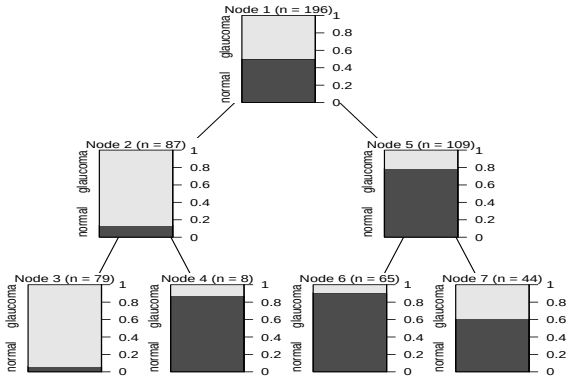
Thus

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

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_{node}.

R-Code

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



```
## Error:  trying to get slot "tree" from an object (class
"constparty") that is not an S4 object
```

```
rpart.plot(classic_tree, cex = 1.5)

## Error:  could not find function "rpart.plot"
```

Finding the best split point is a little different to classification. The test statistic T_J looks like this:

$$T_j(\mathcal{L}, w) = vec(\sum_{i:node} I_A(X_{j*i} e_J(Y_i)^T) = \frac{1}{n_{node}} \sum_{i:node} I_A(X_{j*i}) \begin{pmatrix} I_1(Y_i) \\ I_0(Y_i) \end{pmatrix} =$$
$$= \frac{1}{n_{node}} \begin{pmatrix} \sum_{i:A} I_1(y_i) \\ \sum_{i:A} I_0(y_i) \end{pmatrix} = \frac{1}{n_{node}} \begin{pmatrix} \frac{1}{n_A} n_{G,A} \\ \frac{1}{n_A} n_{N,A} \end{pmatrix}$$

The mean of this test statistic is:

$$\mu = \mathbb{E}(T|S) = vec((\sum_{i:node} I_A(X_{j*i}) \mathbb{E}(e_J(y_i)|S))) =$$
$$= (\sum_{i:node} I_A(Xj*i))(\frac{1}{n_{node}} \sum_{i:node} \begin{pmatrix} Y_{G,i} \\ Y_{N,i} \end{pmatrix}) =$$
$$= \frac{n_A}{n_{node}} \begin{pmatrix} n_G \\ n_N \end{pmatrix}$$

The variance is:

$$\Sigma_j = \frac{1}{n_{node} - 1} \mathbb{V}(h)(n_{node} n_A - n_A^2)$$

bla

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

This yields following standardized test statistic c:

$$c_{max} = \frac{\frac{1}{n_{node}} n_{class,A} - \frac{n_A}{n_{node}} n_{class}}{(\Sigma_j)_{class}} = \frac{\frac{1}{n_{node}} (n_{class,A} - \frac{n_A \cdot n_{class}}{n_{node}})}{\mathbb{V}(Y_{class}) \frac{n\hat{A}}{n_{node}-1} (1 - \frac{n_A}{n_{node}})}$$

Summary

Conditional inference trees are a statistical approach to recursive partitioning. The underlying theory is embedded in a well defined framework of conditional inference aka permutation tests. Unlike most other partitioning algorithms variable selection and best split search are seperated. The choices if to split anyway and if yes, which variable to use 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.

As CART trees find application in the RandomForest - algorithm, conditional trees can also be used for a special implementation of a RandomForest - algorithm. This is implemented as well in the party package LINK with the cforest function.

Permutation tests

Permutation tests are a special case of non-parametric tests. [QUOTE?] Synonyms are: exact tests, randomization tests, re-randomization 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 (which as well is required to perform a classic 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 permutation of Y and X. The classic example is the comparison of two groups of patients, which received either medication A or B (let's call the group label X). The success of the medication can be measured by a blood value Y. The question to answer is, whether the type of medication leads to different Y. This can be formulated as the a null hypothesis of independence between the group label X and the blood value Y. Thus more formally it is a test:

$$H_0 : \mu_A = \mu_B \quad \text{vs} \quad H_1 \mu_A \neq \mu_B$$

A natural choice for the test statistic is to replace μ_A and μ_B with their maximum-likelihood estimators: $T(X, Y) = \hat{\mu_A} - \hat{\mu_B} = \frac{1}{n} \sum_{i \in A} Y_i - \frac{1}{n} \sum_{i \in B} Y_i$. Values close to zero would indicate, that they might be independent. But how close to zero may the values be, so that we still stick with the null hypothesis of independence? To answer that, we calculate the distribution of the test statistic and reject the null hypothesis at the tails of the distribution, where the probability that the measured result can happen under H_0 undercuts a beforehand specified $\alpha \in [0, 1]$. PICS AND NUMBERS.

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

Asymptotics (Strasser and Weber)

Hothorn und Co QUOTE use the following test statistic which is derived from Strasser and Weber QUOTE

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

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

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

- $\sum$ The test statistic is a sum over all observations

- w I lied: Not all observations, because 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.

The test statistic has an expectation and variance, which are also

derived by the framework from Strasser and Weber QUOTE:

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

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

$$\mathbf{w}. = \sum_{i=1}^{n} w_i$$

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

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

Thus we can standardize the linear tests statistic: $c(\mathbf{t}, \mu, \Sigma) = \max_{k=1,\ldots,pq}\left|\frac{(\mathbf{t}-\mu)_k}{\sqrt{(\Sigma)_{kk}}}\right|$

List of Figures