STAT 380: Classification Technique: Tree Based Methods

UAEU

Outlook of the unit

- **Prediction and Classification Approaches**
  - Classification Techniques
    - Logistic regression
    - Discriminant analysis
  - Evaluating Performance of a Classification Technique
  - **Tree-based methods: Decision trees**
    - Classification trees
    - Regression trees

Tree-based methods: Decision trees

Prediction and classification approaches - A short overview

- Linear regression, logistic regression and multinomial logistic regression are common methods for prediction and classification.
- These methods relate a response variable to a set of explanatory variables - parametric nature.
- Linear relationships are being considered, which means that the effect on the response of a change in the explanatory variable from x to $x+1$ is the same for any value x.
- With regression/classification trees, the models become truly nonparametric and they allow for very flexible representations.

Tree-based methods: Regression/classification trees

- Here we describe tree-based methods for regression and classification.
- These involve stratifying or segmenting the predictor space into a number of simple regions.
- Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as decision-tree methods.
- Decision trees can be applied to both regression and classification problems.

Pros and Cons: Regression/classification trees

+ Tree-based methods are simple and useful for interpretation.

- However they typically are not competitive with the best supervised learning approaches in terms of prediction accuracy.

+ Hence we also discuss bagging, random forests (and boosting). These methods grow multiple trees which are then combined to yield a single consensus prediction.

+/- Combining a large number of trees can often result in dramatic improvements in prediction accuracy, at the expense of some loss interpretation.

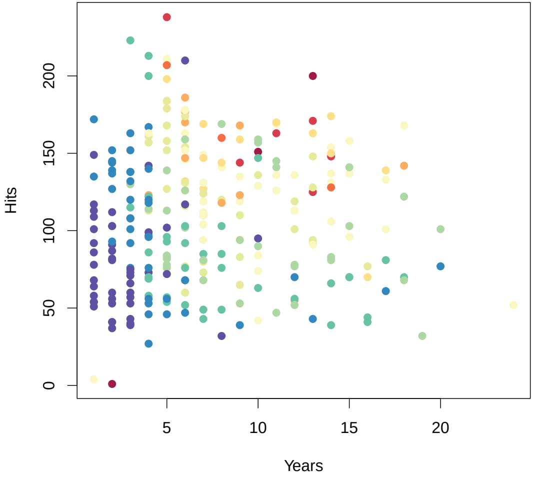
Predicting baseball players' salaries using regression trees

In order to motivate regression trees, we begin with a simple example.
Example:

- We use the *Hitters* data set (R-package: ISLR) to predict a baseball player's salary based on the variable *years* and *hits*.
- Years is the number of years that he has played in the major leagues.
- Hits is the number of hits that he made in the previous year.
- We log-transform salary so that its distribution has a more symmetric shape (salary is measured in thousands of dollars).

Predicting baseball players' salaries using regression trees

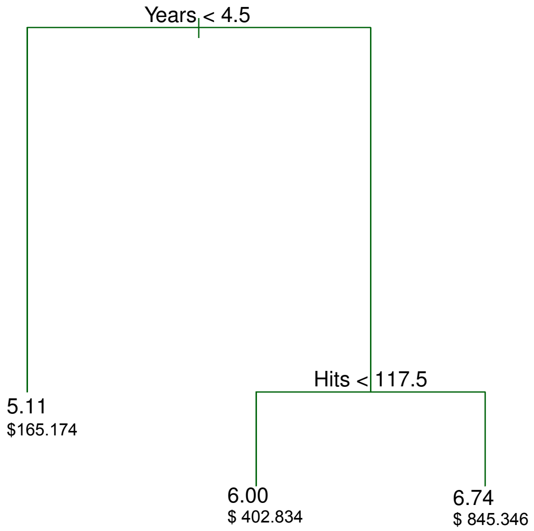
How would you stratify the data?



Salary is color-coded from low (blue, green) to high (yellow, red)

Predicting baseball players' salaries using regression trees

How would you stratify the data?

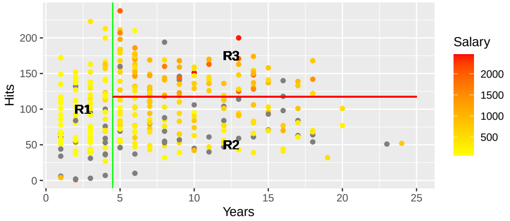


Predicting baseball players' salaries using regression trees

- A regression tree for predicting the log salary based on the number of years (played in the major leagues) and the number of hits (made in the previous year).
- At a given internal node, the label (of the form $X_j < t_k$) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to $X_j \geq t_k$.
- The split at the top of the tree results in two large branches. The left-hand branch corresponds to $Years < 4.5$, and the right-hand branch corresponds to $Years \geq 4.5$.
- The tree has two internal nodes (each decision defines a node) and three leaves/ terminal nodes. The number in each leaf is the mean of the response for the observations that fall there.

Predicting baseball players' salaries using regression trees

Overall, the tree stratifies or segments the players into three regions (terminal nodes or leaves) of predictor space:



$R_1 = \{X | Years < 4.5\}$, $R_2 = \{X | Years \geq 4.5, Hits < 117.5\}$,
$R_3 = \{X | Years \geq 4.5, Hits \geq 117.5\}$

Terminology for trees

- In keeping with the tree analogy, the regions R_1, R_2, and R_3 are known as *terminal nodes*.
- Decision trees are typically drawn upside down, in the sense that the leaves are at the bottom of the tree.
- The points along the tree where the predictor space is split are referred to as *internal nodes*.
- In the hitters tree, the two internal nodes are indicated by the text $\mathit{Years} < 4.5$ and $\mathit{Hits} < 117.5$.

Interpretation of results

- *Years* is the most important factor in determining *Salary*, and players with less experience earn lower salaries than more experienced players.
- Given that a player is less experienced, the number of *Hits* that he made in the previous year seems to play little role in his *Salary*.
- Among players who have been in the major leagues for five or more years, the number of *Hits* made in the previous year does affect *Salary*, and players who made more *Hits* last year tend to have higher salaries.

- Surely an over-simplification, but compared to a regression model, it is easy to display, interpret and explain.

Regression trees

Building regression trees: Segmentation of the space

The process of building a regression tree consists of mainly two steps:

1. We divide the predictor space - that is, the set of possible values for $X_1, X_2, ..., X_p$ - into J distinct and non-overlapping regions, $R_1, R_2, ..., R_J$.
2. For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.

Comment to step 1): The regions could have any shape. However, we choose to divide the predictor space into high-dimensional rectangles, or boxes, for simplicity and for ease of interpretation of the resulting predictive model.

Building regression trees: Segmentation of the space

How should we divide the predictor space?
The goal is to find boxes $R_1, R_2, ..., R_J$ that minimize residual sum of squares

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j} \right)^2,$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within R_j.

- Unfortunately, it is computationally infeasible to consider every possible partition of the feature space into J boxes.
- For this reason, we take a top-down approach that is known as recursive binary splitting.
- The best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

Building regression trees: Segmentation of the space

- We first select (both) the predictor X_j and the cutpoint s such that splitting the predictor space into the regions $R_1(j,s) = \{X | X_j < s\}$ and $R_2(j,s) = \{X | X_j \geq s\}$ leads to the greatest possible reduction in RSS.
- We minimize the equation:

$$\min_{j,s} \left[\sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \right]$$

- Next, we repeat the process, looking for the best X_j and cutpoint s to split the data further to minimize the RSS within each of the resulting regions. We split one of the two previously identified regions. We now have three regions.

Building regression trees: Segmentation of the space

- We split one of these three regions further to minimize the *RSS*.
- The process continues until a stopping criterion is reached; for instance, we may continue until no region contains more than five observations.

Prediction

We predict the response for a given test (validation) observation using the mean of the training observations in the region to which that test observation belongs.

Tree Pruning

- The process described above may produce good predictions on the training set, but is likely to overfit the data, leading to poor test (validation) set performance.
- A smaller tree with fewer splits (that is, fewer regions $R_1, ..., R_J$) might lead to lower variance and better interpretation at the cost of a little bias.
- One strategy is to grow a very large tree T_0, and then prune it back in order to obtain a subtree - *Cost complexity pruning*.

Tree Pruning - Cost complexity pruning

For each value of α there corresponds a subtree $T \subset T_0$ such that

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} \left(y_i - \hat{y}_{R_m}\right)^2 + \alpha |T|$$

is as small as possible.

- $|T|$ indicates the number of leaves/ terminal nodes.
- The tuning parameter α controls a trade-off between the subtree's complexity and its fit to the training data.
- When $\alpha = 0$, then the subtree T will simply equal T_0, the formula just measures the training error.
- As α increases, there is a price to pay for having a tree with many terminal nodes.
- We select an optimal value $\hat{\alpha}$ using cross-validation.
- We then return to the full data set and obtain the subtree corresponding to $\hat{\alpha}$.

Summary: Constructing a regression tree

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, ..., K$:
   - Build a regression tree (for a given α) on all $K - 1$th folds of the training data.
   - Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

   Average the results for each value of α, and pick α to minimize the average error.
4. Return the subtree from Step 2 that corresponds to the chosen value of α.

Prediction and classification methods: Example in R

Regression trees in R: Boston housing data

The `rpart()`-command constructs a regression tree and returns the results as an object.

```r
# Loading libraries and the data
library(rpart)
library(partykit)

# Run a regression tree using the rpart command
fit<-rpart(formula, data, method, minsplit, cp, subset)
```

- `minsplit` specifies min. number of units that must exist in a node in order for a split to be attempted.
- `cp` defines the complexity parameter α in the pruning of the tree.
- `subset` specifies the units in the training data set from the available data.

Regression trees in R: Boston housing data

Grow a general regression tree with
Crime rate + River 1/0 + Number of rooms + Teacher/Pupil ratio

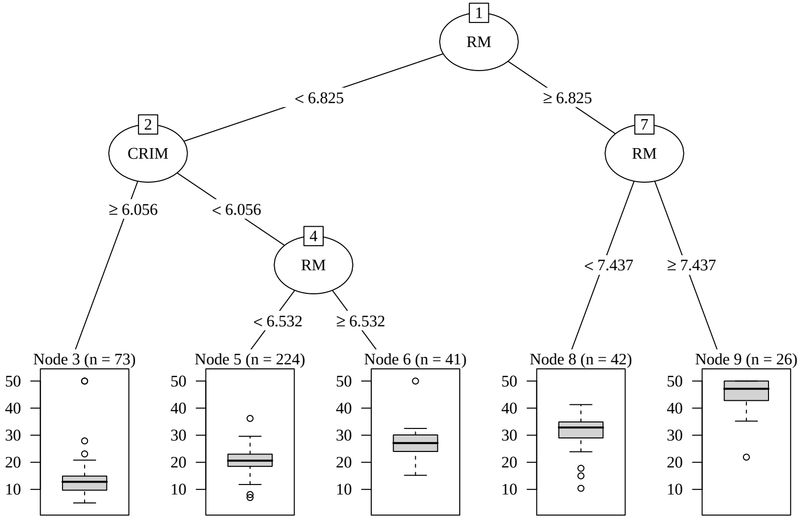
```
# Change complexity parameter alpha to 0 - full tree
> fit<-rpart(MEDV~CRIM+CHAS+RM+PTRATIO, data=Daten,
   minsplit=10, cp=0, subset=inTrain)
# Display the results of cross-validation
> printcp(fit)
Variables actually used in tree construction:
[1] CHAS    CRIM    PTRATIO RM
Root node error: 34771/406 = 85.643
          CP nsplit rel error  xerror     xstd
1  0.45424798      0   1.00000 1.00491 0.092344
2  0.10154523      1   0.54575 0.59452 0.063519
3  0.09118098      2   0.44421 0.50018 0.070210
.  ..........      .   ....... ....... ........
57 0.00014848     74   0.13146 0.43197 0.075392
58 0.00013117     75   0.13131 0.43179 0.075389
59 0.00000000     76   0.13118 0.43207 0.075387
```

Regression trees in R: Boston housing data

The `prune()`-command prunes the regression tree. The tree can be visualized by `plot()`-command.

```
> # Visualize cross-validation results
> plotcp(fit)
>
> # Prune the tree
> fit$cptable[which.min(fit$cptable[,"xerror"]),"CP"]
[1] 0.02605373
> pfit<- prune(fit, cp=0.02605373) # from cptable
>
> # Plot the pruned regression tree
> plot(as.party(pfit))
```

Regression trees in R: Boston housing data



Regression trees in R: Boston housing data

Compare the performance of the pruned tree with the full tree on the validation data.

```
# Predictions for the validation data
> pred_v_tree<-predict(fit,newdata=vali)
> pred_v_ptree<-predict(pfit,newdata=vali)

# Evaluate performance
> accuracy(pred_v_tree,vali$MEDV)
              ME   RMSE   MAE    MPE    MAPE
Test set -0.6629 5.8775 4.001 -9.270 21.3263
> accuracy(pred_v_ptree,vali$MEDV)
              ME   RMSE   MAE    MPE    MAPE
Test set -0.6485 5.6786 3.881 -9.676 20.3017
```

Classification trees

Classification trees - Some first comments

- A **classification tree** is very similar to a regression tree, except that it is classification used to predict a **qualitative response** rather than a quantitative one.
- We predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs.
- We are often interested not only in the **class prediction** corresponding to a particular terminal node region, but also in the **class proportions** among the training observations that fall into that region.

If we define model prediction based on the most commonly occurring class of training observations in a given region, the classification error rate is simply the fraction of the training observations that do not belong to the most common class :

$$\text{classification error rate} = 1 - \max_k \hat{p}_{m,k}$$

Building classification trees

- We use recursive binary **splitting** to grow a classification tree.
- In the classification setting, *RSS* cannot be used as a criterion for making the binary splits.
- A natural alternative to *RSS* is the classification error rate:

  $$E = 1 - \max_k(\hat{p}_{mk}),$$

  where $\hat{p}_{mk}$ represents the proportion of training observations in the m-th region that are from the k-th class.
- The **classification error rate** is simply the fraction of the training observations in that region that do not belong to the most common class.

Building classification trees - Two other measures

- The classification error is not sufficiently sensitive for tree-growing.
- The Gini index is defined by:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

  and measures the variation across the K classes (Gini is small when all $\hat{p}_{mk}$ are close to zero or one).
- Gini index is a measure of *node purity* - a small value indicates that a node contains predominantly observations from a single class.
- An alternative to the Gini index is cross-entropy

$$D = -\sum_{k=1}^{K} \underbrace{\hat{p}_{mk} \log(\hat{p}_{mk})}_{\leq 0}.$$

- The cross-entropy takes a value near zero if the $\hat{p}_{mk}$ are all near zero or near one (small values indicate that nodes are pure).

When to Stop

If tree growing is never stopped, it can result in perfectly pure nodes of one observation each. This is likely to overfit the training data and give poor prediction on the test set.

Strategy to overcome too complex models:

- stop if decrease in impurity is smaller than a threshold
- grow a very large tree and prune it back ('cut branches')

Classification trees in R: Boston housing data

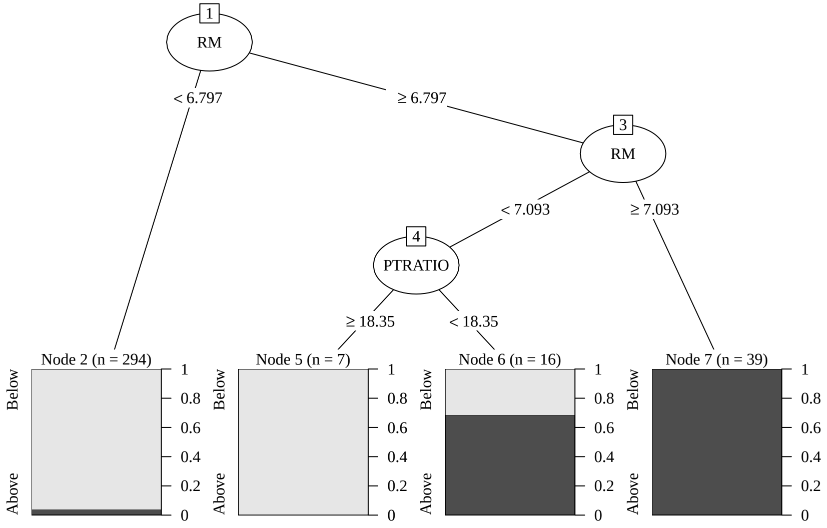
Classification trees can be fitted by using the same commands.

```
# Run a classification tree using the rpart command
> fit<-rpart(MEDV_Fac~CRIM+CHAS+RM+PTRATIO, data=Daten,
   method="class", minsplit=5, cp=0, subset=inTrain)

> # Prune the tree
> fit$cptable[which.min(fit$cptable[,"xerror"]),"CP"]
[1] 0.03225806
> pfit<- prune(fit, cp=0.03225806) # from cptable

> # Plot the pruned classification tree
> plot(as.party(pfit))
```

Classification trees in R: Boston housing data



Classification trees in R: Boston housing data

Compare the performance of the pruned tree with the full tree on the validation data.

```
# Predictions for the validation data
> pred_v_tree<-predict(fit,newdata=vali,type="class")
> pred_v_ptree<-predict(pfit,newdata=vali,type="class")

# Evaluate performance
> confusionMatrix(pred_v_tree, vali$MEDV_Fac)
Prediction Below Above
     Below   120     6
     Above     8    16
               Accuracy : 0.9067
> confusionMatrix(pred_v_ptree, vali$MEDV_Fac)
Prediction Below Above
     Below   122     3
     Above     6    19
               Accuracy : 0.94
```

Advantages and disadvantages of trees

+ Trees are very easy to explain to people.
+ Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches.
+ Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).

- Trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches.
- The decision trees suffer from high variance.

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