Supervised Learning I – CART

Big Data Analysis I

Christoph Kern[1] Frauke Kreuter Marcel Neunhoeffer Sebastian Sternberg

March 25, 2019

[1]c.kern@uni-mannheim.de

Table of contents

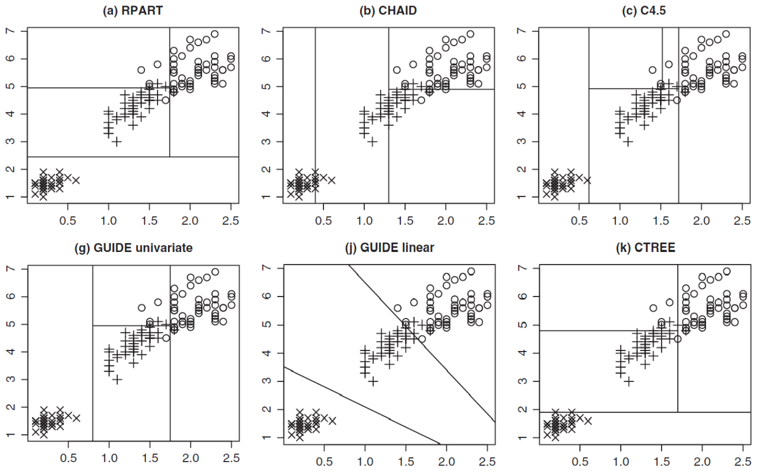
Introduction

Introduction

Decision Trees

- Data-driven approach for relating X and Y
- Popular and (somewhat) easy to interpret
- Important building block (base learner) for ensemble methods
- Many different tree building algorithms exist (Zhang & Singer 2010, Loh 2014)
    - Focus on interaction detection, prediction, parameter instability...

Figure 1: Decision Tree Algorithms



Loh (2014)

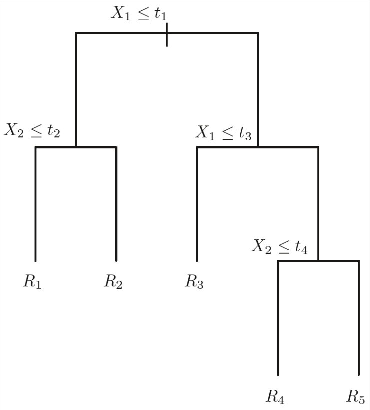
Classification and Regression Trees (CART)

Classification and Regression Trees (CART)

Classification and Regression Trees (CART)

- Approach for partitioning the predictor space into smaller subregions via “recursive binary splitting”
- Results in a “top-down” tree structure with...
  - Internal nodes within the tree
  - Terminal nodes as endpoints
- Can be applied to regression and classification problems

Figure 2: A small tree



James et al. (2013)

Tree growing

Growing a regression tree

Define pairs of regions for all $X_1, X_2, ..., X_p$ predictors and cutpoints c

$$\tau_L(j,c) = \{X|X_j < c\} \text{ and } \tau_R(j,c) = \{X|X_j \geq c\}$$

Find split s which maximizes the reduction in RSS

$$\Delta RSS(s,\tau) = RSS(\tau) - RSS(\tau_L) - RSS(\tau_R)$$

$$RSS(\tau) = \sum_{i \in \tau}(y_i - \hat{y})^2$$

with $\hat{y}$ being the mean of y in node τ

Tree growing

Growing a **regression tree**

Define pairs of regions for all $X_1, X_2, ..., X_p$ predictors and cutpoints c

$$\tau_L(j, c) = \{X | X_j < c\} \text{ and } \tau_R(j, c) = \{X | X_j \geq c\}$$

Find split s which maximizes the reduction in RSS

$$\Delta RSS(s, \tau) = RSS(\tau) - RSS(\tau_L) - RSS(\tau_R)$$

$$RSS(\tau) = \sum_{i \in \tau} (y_i - \hat{y})^2$$

with $\hat{y}$ being the mean of y in node τ

Tree growing

Growing a **classification tree**

Define pairs of regions for all $X_1, X_2, ..., X_p$ predictors and cutpoints c

$$\tau_L(j, c) = \{X | X_j < c\} \text{ and } \tau_R(j, c) = \{X | X_j \geq c\}$$

Find split s which maximizes the reduction in node impurity

$$\Delta I(s, \tau) = I(\tau) - p(\tau_L) I(\tau_L) - p(\tau_R) I(\tau_R)$$

Impurity measures

$$I_{Gini}(\tau) = \sum_{k=1}^{K} \hat{p}_k (1 - \hat{p}_k)$$

$$I_{entropy}(\tau) = -\sum_{k=1}^{K} \hat{p}_k \log \hat{p}_k$$

with $\hat{p}_k$ being the proportion of observations from class k in node τ

Tree growing

Growing a **classification tree**

Define pairs of regions for all $X_1, X_2, ..., X_p$ predictors and cutpoints c

$$\tau_L(j,c) = \{X|X_j < c\} \text{ and } \tau_R(j,c) = \{X|X_j \geq c\}$$

Find split s which maximizes the reduction in node impurity

$$\Delta I(s,\tau) = I(\tau) - p(\tau_L)I(\tau_L) - p(\tau_R)I(\tau_R)$$

Impurity measures

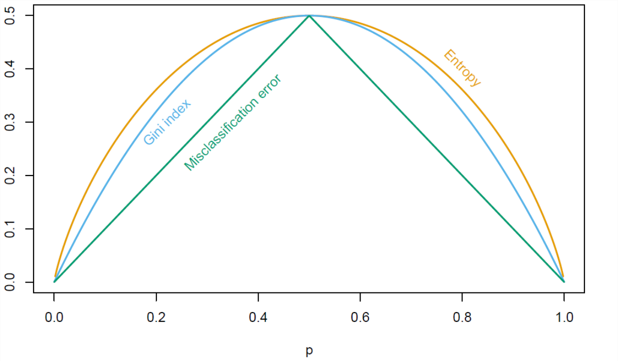
$$I_{Gini}(\tau) = \sum_{k=1}^{K} \hat{p}_k(1-\hat{p}_k)$$

$$I_{entropy}(\tau) = -\sum_{k=1}^{K} \hat{p}_k \log \hat{p}_k$$

with $\hat{p}_k$ being the proportion of observations from class k in node τ

Tree growing

Figure 3: Misclassification error, Gini index & entropy (scaled)



Hastie et al. (2009)

Tree growing

Algorithm 1: Tree growing process

```
Define stopping criteria;
Assign training data to root node;
if stopping criterion is reached then
    end splitting;
else
    find the optimal split point;
    split node into two subnodes at this split point;
    for each node of the current tree do
        continue tree growing process;
    end
end
```

Tree structure

A given tree

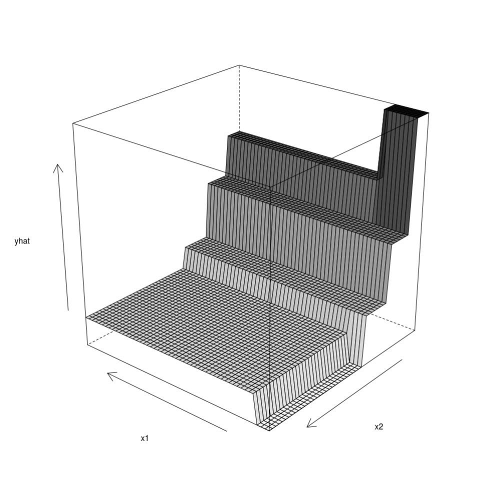
$$\mathcal{T} = \sum_{m=1}^{M} \gamma_m \cdot 1_{(i \in \tau_m)}$$

consists of a set of $m = 1, 2, \ldots, M$ nodes which can be used for prediction by...

- Regression
  - ...using the mean of y for training observations in τ_m
- Classification
  - ...going with the majority class in τ_m

$\rightarrow$ Prediction surface: Block-wise relationship between features and outcome

Figure 4: Tree prediction surface (example)



Missings

- Create a new category for missing values
- Use surrogate splits
    1. Choose best (primary) predictor based on complete cases
    2. Search for surrogate variables which mimic the chosen split
    3. Use surrogates if values for primary predictor are missing

Costs

$$\mathsf{L} = \begin{pmatrix} 0 & L_{fp} \\ L_{fn} & 0 \end{pmatrix}$$

- Typically $L_{fp} = L_{fn} = 1$
- Misclassifications can be weighted differently
    - Modification of loss-matrix through weights / modified Gini index

Tree structure

Missings

- Create a new category for missing values
- Use surrogate splits
  1. Choose best (primary) predictor based on complete cases
  2. Search for surrogate variables which mimic the chosen split
  3. Use surrogates if values for primary predictor are missing

Costs

$$\mathsf{L} = \begin{pmatrix} 0 & L_{fp} \\ L_{fn} & 0 \end{pmatrix}$$

- Typically $L_{fp} = L_{fn} = 1$
- Misclassifications can be weighted differently
  - Modification of loss-matrix through weights / modified Gini index

Tree pruning

Stopping rules

- Minimum number of cases in terminal nodes
- Decrease in impurity exceeds some threshold

$\rightarrow$ However, worthless splits can be followed by good splits

Cost complexity pruning

$$R_\alpha(\mathcal{T}) = R(\mathcal{T}) + \alpha|\mathcal{T}|$$

- Find the best subtree by balancing quality $R(\mathcal{T})$ and complexity $|\mathcal{T}|$
- α controls the penalty on the number of terminal nodes
- α can be chosen through CV

Tree pruning

Stopping rules

- Minimum number of cases in terminal nodes
- Decrease in impurity exceeds some threshold

$\rightarrow$ However, worthless splits can be followed by good splits

Cost complexity pruning

$$R_\alpha(\mathcal{T}) = R(\mathcal{T}) + \alpha|\mathcal{T}|$$

- Find the best subtree by balancing quality $R(\mathcal{T})$ and complexity $|\mathcal{T}|$
- α controls the penalty on the number of terminal nodes
- α can be chosen through CV

Summary

Summary

- Divide-and-conquer strategy that splits the data into subgroups
- Surface from decision trees is a non-smooth step function
- No need to specify the functional form in advance (unlike regression)
- Non-linearities and interactions are handled automatically
- Limitations: Instability(!), competition among correlated predictors, biased variable selection

Software Resources

Resources for R

- Basic CART implementation: `tree`
- Standard package to build CARTs: `rpart`
  - Includes build-in Cross-Validation and `prune` function
- Unified infrastructure for tree representation: `partykit`

References

References

Hastie, T., Tibshirani, R., Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY: Springer.

James, G., Witten, D., Hastie, T., Tibshirani, R. (2013). *An Introduction to Statistical Learning*. New York, NY: Springer.

Loh, W.-Y. (2014). Fifty Years of Classification and Regression Trees. *International Statistical Review 82*(3), 329–348.

Zhang, H., Singer, B. (2010). *Recursive Partitioning and Applications*. New York, NY: Springer.