

Supervised Learning - Classification and Regression Trees (CART)

Mark Asch - IMU/VLP/CSU

2023

Program

1. Data Analysis

- (a) Introduction: the 4 identifiers of “big data” and “data science”
- (b) **Supervised learning methods:** regression—advanced, k-NN, linear classification methods, SVM, NN, **decision trees**.
- (c) Unsupervised learning methods: k-means, principal component analysis, clustering.

Introduction

- Tree-based methods have a **long history** in applied statistics, dating back to the 1960's and formalized by L. Breiman in the early 1980's.
- They are considered to be easily explainable, mainly because they reproduce, to some extent, the decision-making process of the human brain. In this sense, they can be considered as **expert systems**.
- They have recently been improved by the more modern techniques of **bagging** and **boosting**.
- Breiman's **random forest** (2001) remains one of the most simple, stable and robust methods for performing tree-based analysis.
- They can be used for both **regression** and **classification**
 - ⇒ The objective of any tree-based method is to **segment** the space of predictions into a number of simple regions.

- ⇒ Then, for a given observation, use the **average** or the mode of the training observations of the region in which the observation belongs as the prediction.
- ⇒ Basic decision trees are **simple** to compute, and very **simple** to interpret.
- ⇒ Unfortunately, this basic approach is **not robust**---see below---and we usually must resort to methods based on **multiple trees** that improve the predictive accuracy even though we then lose some of the interpretability.

A Simple Example

- The most easily understood example of a decision tree is one based on **baseball statistics**.
 - ⇒ We want to predict a player's salary based on the number of years and the number of hits.
 - ⇒ An extremely simple and understandable tree is obtained, and is shown in the Figure



Figure 1: Regression tree for predicting baseball players' salaries, based on their experience (Years) and their number of Hits.

Construction of a Regression Tree

The **decision tree** is constructed in two steps.

1. Divide the prediction space, containing all the values of the predictors X_1, \dots, X_p , into J distinct regions, or intervals, R_1, \dots, R_J .
2. To each observation, attribute the average of the response values of region R_j in which it falls.

The **regions** themselves are constructed following the sequence of basic steps:

1. Divide the space of predictors into hypercubes.
2. Find the cubes that minimize the **residual sum of squares**,

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

where \hat{y}_{R_j} is the average response of the training observations in cube j .

3. Use a recursive, binary split—top-down approach—where we choose X_j and a splitting value s that minimizes

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

with

$$R_1(j, s) = \{X | X_j < s\} \text{ and } R_2(j, s) = \{X | X_j \geq s\}$$

4. Continue subdividing the regions until no region contains more than N observations.
- **Prediction** : once the regions R_1, \dots, R_J created, we predict the response for a test observation by the average of the training observations in the region in which the test observation belongs.

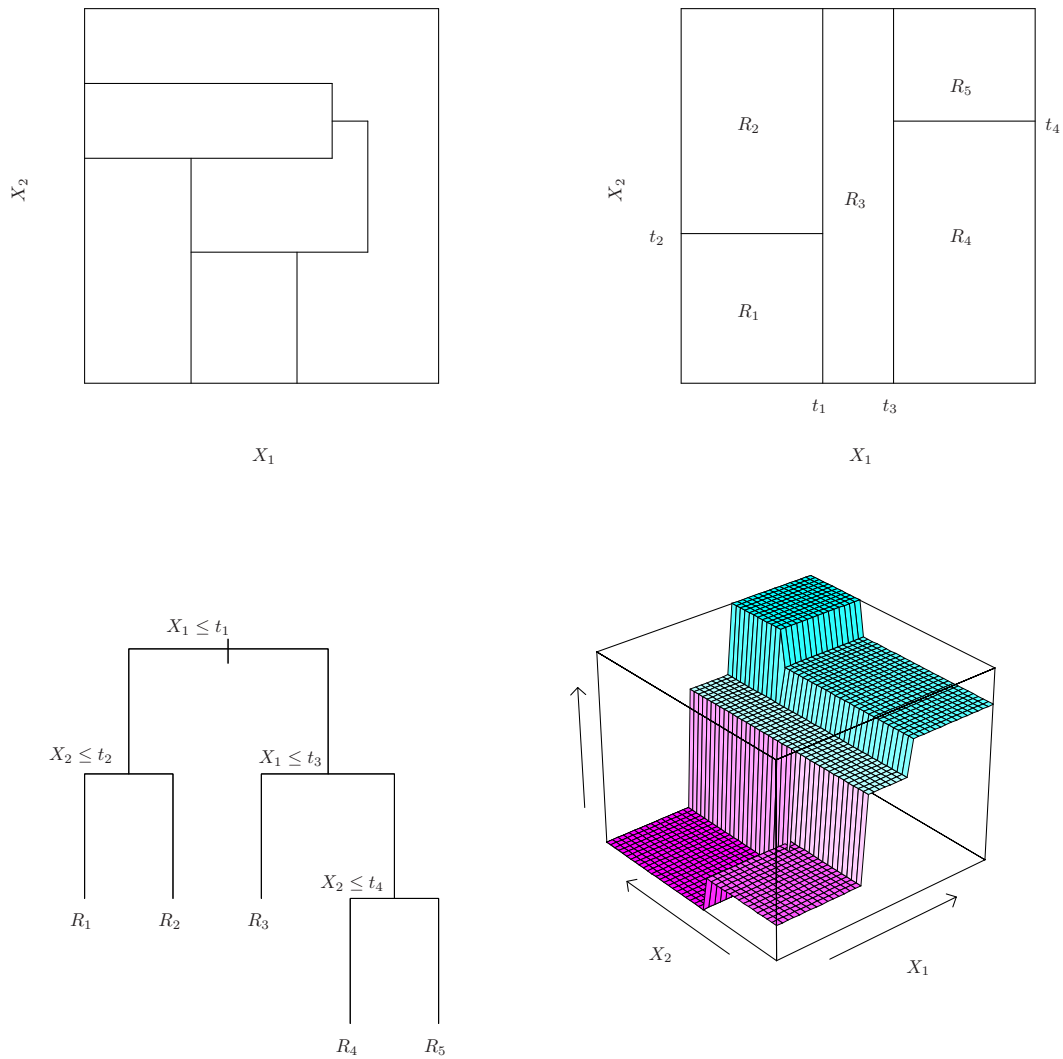


Figure 2: Recursive binary partition of 2 attributes into 5 regions.

Pruning Trees

- The procedure described above produces a good result for the training set, but has a tendency to **overfit**—see below.
- A smaller tree, with fewer branches, could produce a smaller variance and a better interpretation, at the cost of an increase in the **bias**.

⇒ This is the well-known bias-variance compromise/trade off.

- **Pruning** proceeds as follows:

1. Construct a tree with a large number of branches, T_0 .
2. Prune this tree, with a parameter α , to obtain a **sub-tree** that gives a smaller test error,

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

where

- (a) $|T|$ is the number of terminal nodes of tree T ,
 - (b) the region R_m corresponds to the terminal node m ,
and
 - (c) \hat{y}_{R_m} is the average of the training observations in R_m .
- The parameter α represents a **compromise** between the complexity of the sub-tree and its fit to the training data.
 - ⇒ It is the same penalization as is used in the LASSO regression .
 - ⇒ The value of α is selected by a validation set, or preferably by **cross-validation**

Classification Trees

- These are trees that predict a **qualitative response**, in the form of a category or class.
- The tree is constructed by recursive binary division, based on the classification error rate.
- In practice, the **GINI index** is used as a measure of total variance instead of the RSS, and is defined by

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

where \hat{p}_{mk} is the proportion of training observations in region m and class k .

Example: Cardiac data

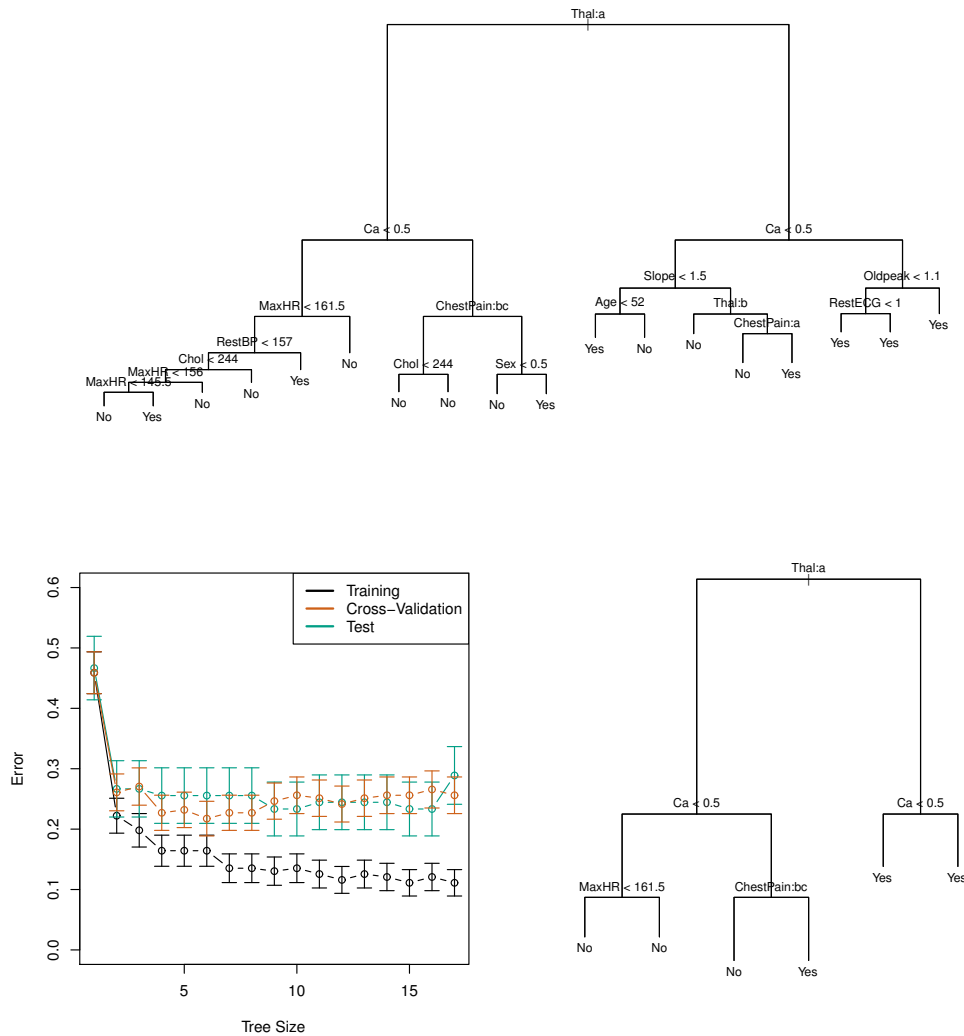


Figure 3: Tree without pruning; training, cross-validation and test errors; tree after pruning.

Trees vs. Linear Models

- Recall: in **linear regression** we assumed a model of the form

$$f(X) = \beta_0 + \sum_{j=1}^p \beta_j X_j.$$

- In **regression trees**, the model has the form

$$f(X) = \sum_{m=1}^M c_m \mathbf{1}_{(X \in R_m)}.$$

- Which model is better?
 - ⇒ If the relation between features and response is well approximated by a linear model, then SLR will be better than CART because it exploits the linear structure.
 - ⇒ If the relation is complex and nonlinear, then decision trees will be better.

Pros and Cons

- ✓ Easy to explain and interpret.
- ✓ Closer to the process of human decision-making.
- ✓ Nice graphical representation.
- ✗ Prediction accuracy is often lower.
- ✗ Lack of robustness.

Multiples Trees

To construct more powerful, tree-based prediction methods, we use of different approaches for aggregating trees:

1. Bagging.
2. Boosting.
3. Random Forests.

Bagging

- Decision trees have **high variances**.
 - ⇒ If we randomly split the data in two, we will find **different** trees for each half.
 - ⇒ A low variance method will produce the **same** result when applied to distinct subsets.

Definition 1 (Bagging). Bootstrap aggregation, or *bagging*, is a general procedure for variance reduction of a statistical learning method.

- Recall:
 - ⇒ For a set of n independent observations, Z_1, \dots, Z_n , each with variance σ^2 , the variance of the mean \bar{Z} is equal to σ^2/n .
 - ⇒ So, taking the average of such a set of observations, will always **reduce** the variance, and
 - ⇒ as a result **increase** the accuracy of a statistical learning method.

- We can create multiple training sets by using a **bootstrap** approach, based on **sampling with replacement**. The steps are:
 - ⇒ Generate B different training sets.
 - ⇒ Perform the learning to obtain $\hat{f}^{*b}(x)$ for sample b .
 - ⇒ Calculate the average over all the predictions (bagging),

$$\hat{f}_{\text{bag}} = \frac{1}{B} \sum_{b=1}^B \hat{f}^{*b}(x).$$

- Bagging is particularly efficient for decision trees, and usually uses a few **hundred** trees for the aggregation.

Out-of-Bag (OOB) estimation

- To estimate the test error, without cross-validation or validation sets, we can exploit the observations that are **never chosen** in the sampling with replacement.
 - ⇒ On average, each tree obtained from bagging only uses **two-thirds** of the observations in the dataset.
 - ⇒ The remaining one-third are called the **OOB observations**.
 - ⇒ So, to estimate the test error without having to resort to the usual validation sets or cross-validation, one can use these **unused** observations.
 - ⇒ We take the average, or the majority vote, to compute the prediction.
- An **OOB prediction** is finally obtained for each of the n observations, then we calculate an MSE, or an overall classification error.
- For B large enough, the OOB error is equivalent to the **LOOCV** error—see Lecture on Resampling—and can replace cross-validation when it becomes too expensive, in particular for **big data**.

Random Forests (RF)

- RF improves bagging by performing a **decorrelation** among the multiple trees.
 - ⇒ It is based on the fact that the average of decorrelated random variables will always produce a **lower variance** than an average of correlated variables.
- We proceed as for bagging:
 - ⇒ Construct a given number of trees from bootstrap samples.
 - ⇒ But, at each division/split, only a **random subsample** of m predictors ($m < p$) is chosen as candidates for the division.
 - ⇒ The division is based on one of these m predictors.
 - ⇒ A **new sample** is drawn at each division.
 - ⇒ Normally, $m \approx \sqrt{p}$, where p is the number of predictors.
 - ⇒ When $m = p$, we are in the case of **bagging**.

- During classical bagging, all the trees obtained will be **similar** because the strongest predictor will be chosen each time for the topmost split.
 - ⇒ This is not the case during RF, and the result is a decorrelation between the trees.
 - ⇒ This is particularly pertinent when we have a large number of **correlated predictors**.
- RF provides, thanks to the random choice of topmost splits, a **ranking** of the **importance**/influence of the explanatory variables. This can eventually be used as a basis for **model reduction**.

Algorithms for RF

- There are two recommended packages for RF in R.

⇒ The first is the `randomForest` library:

```
library(randomForest)
model <- randomForest(formula, data=...,
                      na.action=...,
                      ntree=..., mtry=...)
```

- The second is within the `caret` framework:

```
library(caret) model <- train(formula, data=...,
na.action=..., method=rf)
```

- In python, Random Forest is provided via the `RandomForestRegressor` and `RandomForestClassifier` classes.

```
from sklearn.ensemble import RandomForestRegressor
```

```
from sklearn.datasets import make_regression
X, y = make_regression(n_features=4,
                       n_informative=2,
                       random_state=0, shuffle=False)
regr = RandomForestRegressor(max_depth=2,
                             random_state=0)
regr.fit(X, y)
print(regr.predict([[0, 0, 0, 0]]))
```

Boosting

- This is another procedure that can be applied to decision trees.
- We fit, progressively on the basis of smaller trees, the global tree.
- Three tuning parameters are used for this:
 - ⇒ Number of trees, B , between 500 and 1000.
 - ⇒ Shrinkage parameter, $\lambda > 0$, typically 0.01 or 0.001.
 - ⇒ Number of splits in each tree, d , often with $d = 1$.
- It is by fitting to residues, slowly, that boosting improves \hat{f} in the regions where the performance is bad

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x),$$

where $\hat{f}^b(x)$ is fitted to residues, for $b = 1, 2, \dots, B$.

- AdaBoost is the basic boosting algorithm, using adaptive boosting.
- XGBoost is a highly efficient algorithm
<https://xgboost.readthedocs.io>
that can be used for very large data volumes (big data),
and is based on a gradient method for accelerating the
optimization.

Examples

1. A very comprehensive example for classifying edible status of mushrooms as a function of 22 morphological features - see [class-champignon.html](#)
2. Gradient Boosting for iris data (using caret) - see [boost-iris.html](#).
3. Gradient Boosting for baseball data - see [boostingHitters.html](#).

References

1. M. DeGroot, M. Schervish, *Probability and Statistics*, Addison Wesley, 2002.
2. Spiegel, Murray and Larry Stephens, *Schaum's Outline of Statistics*, 6th edition, McGraw Hill. 2017.
3. G. James, D. Witten, T. Hastie, R. Tibshirani. *An Introduction to Statistical Learning with Applications in R*. Springer. 2013.
4. T. Hastie, R. Tibshirani, J. Friedman. *The Elements of Statistical Learning*. Springer. 2009.
5. Rachel Schutt and Cathy O'Neil. *Doing Data Science*. O'Reilly. 2014.