Penalize Regression R Package

Penalize Regression R Package

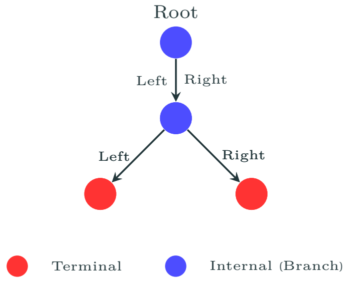
JuHyun Kang

June 5, 2025

The Three Sisters of Newton
School of Mathematics, Statistics and Data Science
Sungshin Women's University

Decision Trees

- A decision tree is a predictive model that splits the input space into regions using simple decision rules.
- It partitions the data based on feature values to minimize prediction error.
- A decision tress consists of vertices and branches.
  - Branch nodes : Vertices that branch left and right
  - Terminal nodes : Vertices do not branch



Decision Tree Regression in Population

- The observed data $(x_1, y_1), \cdots, (x_i, y_i), \cdots, (x_N, y_N) \in \mathbb{R}^p \times \mathbb{R}$
- The terminal regions $R_1, \cdots, R_j, \cdots, R_m$
- When the joint probability density function is $f_{XY}(x, y)$, we construct a rule such that

$$x_i \in R_j \;\Rightarrow\; \hat{y}_i = \bar{y}_j \tag{1}$$

with

$$\bar{y}_j := \mathbb{E}[Y \mid R_j] = \frac{\int_{-\infty}^{\infty} \int_{R_j} y \cdot f_{XY}(x, y) dx dy}{\int_{-\infty}^{\infty} \int_{R_j} f_{XY}(x, y) dx dy} \tag{2}$$

$$\int_{-\infty}^{\infty} \sum_{j=1}^{m} \int_{R_j} (y - \bar{y}_j)^2 f(x, y) dx dy. \tag{3}$$

Decision Tree Regression in Samples

- The simultaneous density function f_{XY} is unknown and should be estimated from the samples.
- If the size of the region R_j is n_j, then one might replace (1), (2) and (3) by

$$x \in R_j \;\Rightarrow\; \hat{y}_i = \bar{y}_j$$

$$\bar{y}_j := \frac{1}{n_j} \sum_{i:\, x_i \in R_j} y_i,$$

and

$$\sum_{j=1}^{m} \sum_{i:\, x_i \in R_j} (y_i - \bar{y}_j)^2, \tag{4}$$

where $\sum_{i:\, x_i \in R_j}$ sums over i such that $x_i \in R_j$.

Practical Considerations in Decision Tree Construction

- It is computationally infeasible to find the optimal tree due to the large number of variable and split combinations.
- A greedy approach is used: at each node, the best variable and threshold are chosen based only on the current data, without considering future splits.
- The threshold is usually chosen from existing sample values in the node.

How to Split in Decision Tree

- Suppose that a branch node contains samples x_i with $i \in S$, and by the sample size S, where S is a subset of $\{1, 2, \cdots, N\}$.
- We divide subset $\{x_k | k \in S\}$ of $\{x_1, ..., x_N\}$ into $\{x_k | x_k < x_{i,j}, k \in S\}$ and $\{k | x_k \geq x_{i,j}, k \in S\}$ by specifying $i \in S$ and $j = 1, \cdots, p$.
- Although this approach does not take overfitting into consideration, we may select i, j that minimize

$$\sum_{k:\, x_{k,j} < x_{i,j}} (y_i - \bar{y}_{i,j}^L)^2 + \sum_{k:\, x_{k,j} \geq x_{i,j}} (y_i - \bar{y}_{i,j}^R)^2$$

where n_R and n_L are the numbers of k such that $x_{k,j} < x_{i,j}$.

Define $\bar{y}_{i,j}^L = \frac{1}{n_L} \sum_{k:\, x_{k,j} < x_{i,j}} y_k$ and $\bar{y}_{i,j}^R = \frac{1}{n_R} \sum_{k:\, x_{k,j} \geq x_{i,j}} y_k$.

Squared Loss Function in R

```r
sq.loss <- function(y) {
  y.bar <- mean(y)
  sum((y - y.bar)^2)
}
```

- We construct the following procedure for regression, where we apply a loss measure sq.loss function to the argument branch function.
- The procedure chooses the i, j that maximize the decrease before and after branching, assuming that each $k \in S$ proceeds to the left and right depending on $x_{k,j} < x_{i,j}$ and $x_{k,j} \geq x_{i,j}$.

Brach Function with Decision Tree Regression in R

```r
branch <- function(x, y, f, S, m = ncol(x)) {
  n <- length(S); p <- ncol(x)
  if (n == 0) return(NULL)
  best.score <- Inf; best.info <- NULL

  for (j in 1:p) {
    for (i in S) {
      split_point <- x[i, j]
      left <- S[x[S, j] < split_point]
      right <- S[x[S, j] >= split_point]
      L <- f(y[left]); R <- f(y[right]); score <- L + R

      if (score < best.score) {
        best.score <- score
        best.info <- list(i = i, j = j, left = left, right = right,
                          score = best.score, left.score = L,
                          right.score = R)
      }}}
  return(best.info)}
```

Why Use CV in Decision Tree Regression

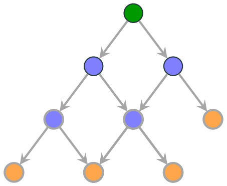
- When fitting a decision tree by minimizing the impurity criterion, the error becomes smaller as the tree becomes more complex (larger m), reaching zero error when each region contains only one sample ($m = N$)
- This leads to overfitting, where the model perfectly fits the training data but fails to generalize to unseen data.
- A typical way to explicit avoid overfitting is to obtain $m \geq 1$ and $R_1, R_2, \cdots, R_m$ that minimize

$$\sum_{j=1}^{m} \sum_{i:\, x_i \in R_j} (y_i - \bar{y}_j)^2 + \alpha m,$$

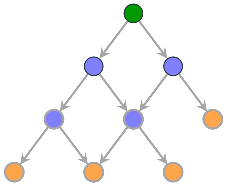
where the value of $\alpha > 0$ can be obtained via CV.

Visualization of Decision Tree with α

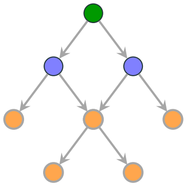
$\alpha = 0$



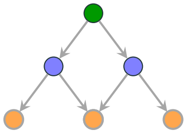
$\alpha = 0.1$



$\alpha = 0.5$



$\alpha = 2$



When to Stop Branching in Decision Trees

1. Node Sample Size Below Threshold:
   Stop branching when the node sample size is less than a minimum threshold, $n_{\min}$.
2. No Samples in Either Group:
   Stop branching if, after splitting, no samples are left in either the left or right group.
3. Insufficient Decrease in Impurity:
   Stop branching if the decrease in impurity before and after the split is smaller than a pre-defined threshold, α. This is because the objective function has the form "loss sum $+ \alpha \cdot m$", so a split is considered meaningful only if the decrease in loss exceeds α.

Decision Tree Function and Stack-Based Tree Construction in R

```r
# Define a decision tree function with stopping rules
dt <- function(x, y, f = "sq.loss", alpha = 0, n.min = 1, m = ncol(x)) {

  g <- switch(f, # Choose impurity function based on user input
              "sq.loss" = sq.loss, "mis.match" = mis.match,
              "gini" = gini, entropy)
  n <- length(y)

  stack <- list(list(parent = 0, set = 1:n, score = g(y))) # Initialize stack
  vertex <- list()
  k <- 0  # node index counter

  # Start tree construction using stack (top-down approach)
  while (length(stack) > 0) {
    r <- length(stack)
    node <- stack[[r]]
    stack <- stack[-r]  # pop node from stack
    k <- k + 1  # create new node ID
    res <- branch(x, y, g, node$set, m) # find best split
```

Stopping Conditions and Terminal Node Creation

```r
    # Check stopping criteria
    if (is.null(res) ||
        (node$score - res$score) < alpha ||
        length(node$set) < n.min ||
        length(res$left) == 0 ||
        length(res$right) == 0) {

      # Create terminal node if any stopping condition is met
      vertex[[k]] <- list(
        parent = node$parent,
        j = 0,
        set = node$set
      )
```

Internal Node Creation and Stack Update

```
    } else {
      # Create internal node and store split rule
      vertex[[k]] <- list(
        parent = node$parent,
        set = node$set,
        th = x[res$i, res$j],  # split threshold
        j = res$j              # split variable (column index)
      )

      # Push right and left children to the stack
      stack[[r]] <- list( parent = k, set = res$right,
                          score = res$right.score)
      stack[[r + 1]] <- list( parent = k, set = res$left,
                              score = res$left.score)
    }
  }
```

Build Tree Structure and Assign Terminal Predictions

```r
  r <- length(vertex) # Initialize child node pointers
  for (h in 1:r) {vertex[[h]]$left <- 0; vertex[[h]]$right <- 0}

  for (h in r:2) { # Connect parent and child nodes
    pa <- vertex[[h]]$parent
    if (vertex[[pa]]$right == 0) {vertex[[pa]]$right <- h}
    else {vertex[[pa]]$left <- h}
  }
  # Assign predicted values for terminal nodes
  mode <- function(y) names(sort(table(y), decreasing = TRUE))[1]
  g_center <- if (f == "sq.loss") mean else mode

  for (h in 1:r) {
    if (vertex[[h]]$j == 0) {
      vertex[[h]]$center <- g_center(y[vertex[[h]]$set])
    }
  }
  return(vertex)
}
```

Boston Dataset

Variable	**Description**
CRIM	Per capita crime rate by town
ZN	Proportion of residential land zoned for lots over 25,000 sq.ft.
INDUS	Proportion of non-retail business acres per town
CHAS	Charles River dummy variable (1 if tract bounds river; 0 otherwise)
NOX	Nitric oxides concentration (parts per 10 million)
RM	Average number of rooms per dwelling
AGE	Proportion of owner-occupied units built before 1940
DIS	Weighted distances to five Boston employment centers
RAD	Index of accessibility to radial highways
TAX	Full-value property-tax rate per $10,000
PTRATIO	Pupil-teacher ratio by town
B	$1000 \times (\mathrm{Bk} - 0.63)^2$, where Bk is the proportion of Black residents
LSTAT	Percentage of lower status population
MEDV	Median value of owner-occupied homes (in $1000s)

Boston Housing Dataset Variable Descriptions

Example of Boston Data

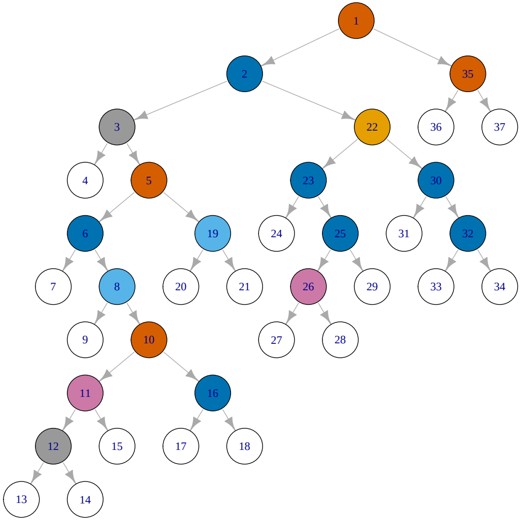
```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
##   medv
## 1 24.0
## 2 21.6
## 3 34.7
## 4 33.4
## 5 36.2
## 6 28.7
```

Graphical Representation of the Decision Tree in R

```r
library(MASS)
library(igraph)
x = as.matrix(Boston[,1:13])
y = as.vector(Boston[,14])
vertex = dt(x,y,n.min=50)

# Graph Output
r = length(vertex)
col = array(dim = r)
edge.list = matrix(nrow=r, ncol=2)
for (h in 1:r)
{
  col[h] = vertex[[h]]$j
  edge.list[h,] = c(vertex[[h]]$parent, h)
}
edge.list = edge.list[-1,]
g= graph_from_edgelist(edge.list)
V(g)$color=col
plot(g, layout = layout.reingold.tilford(g, root=1))
```

Graphical Representation of the Decision Tree in R (contd)



Node Information with Decision Tree Regression in R

```r
#Table Output
VAR = NULL
TH = NULL
for (h in 1:r) {
  if (vertex[[h]]$j != 0) {
    VAR = c(VAR, vertex[[h]]$j)
    TH = c(TH, vertex[[h]]$th)
  }
}
head(cbind(VAR, TH), 3)
```

```
##      VAR     TH
## [1,]   6  6.943
## [2,]  13 14.430
## [3,]   8  1.413
```

Prediction Function with Decision Tree Regression in R

```r
value <- function(u, vertex) {
  r <- 1

  # Search through the tree following the given conditions
  while (vertex[[r]]$j != 0) {
    if (u[vertex[[r]]$j] < vertex[[r]]$th) {
      r <- vertex[[r]]$left
    } else {
      r <- vertex[[r]]$right
    }
  }
  return(vertex[[r]]$center) # center value of the final node
}
```

CV Design for Decision Tree Regression Optimization

- Dataset: Boston Housing dataset (initially using N = 100 data points)
- Objective: Find the optimal regularization parameter α within the range $0 \leq \alpha \leq 1.5$
- CV Method: 10-fold cross-validation
- CV Metric: Mean Squared Error (MSE) for performance evaluation
- Goal: Identify the optimal value of α that minimizes cross-validation error
- Model: Decision Tree Regression with regularization (using α to control tree complexity)

Optimizing Decision Tree Regression via Cross-Validation in R

```r
# Using inital 100 sample in Boston dataset
df <- Boston[1:100, ]
n <- nrow(df)
p <- ncol(df)

# Split into input variables (x) and target variable (y)
x <- as.matrix(df[, 1:13])
y <- as.vector(df[, 14])

# # Generate a sequence of alpha candidates
alpha_seq <- seq(0, 1.5, 0.1)
fold_size <- floor(n / 10)  # 10-fold CV

cv_errors <- numeric(length(alpha_seq))  # For storing the results
```

Optimizing Decision Tree Regression via Cross-Validation in R (contd)

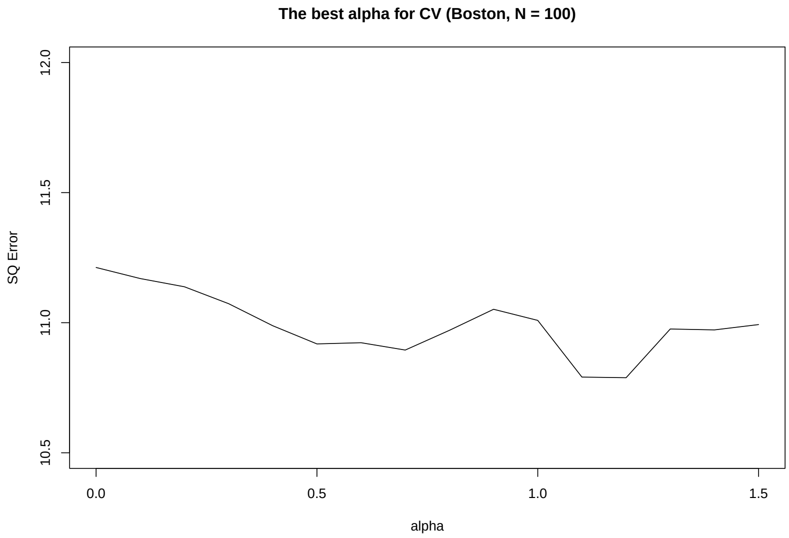
```r
# 10-fold cross-validation for each alpha value
for (i in seq_along(alpha_seq)) {
  alpha <- alpha_seq[i]
  total_error <- 0

  for (h in 1:10) { # Generate fold index
    test_idx <- ((h - 1) * fold_size + 1):(h * fold_size)
    train_idx <- setdiff(1:n, test_idx)

    # Train decision tree model
    vertex <- dt(x[train_idx, ], y[train_idx], alpha = alpha)

    for (t in test_idx) { # Predict and accumulate squared errors
      prediction <- value(x[t, ], vertex)
      total_error <- total_error + (y[t] - prediction)^2
    }
  }
  cv_errors[i] <- total_error / n # Mean squared errors
}
```

Cross-Validation Results by Decision Tree Regression



Q & A

Q & A

Thank you :)