Caijun Qin

11/19/2021

1. For smoothing splines, the effective degrees of freedom moves in opposite direction to such that as → ∞, → 2. However, computing the coefficients for a smoothing spline with = 0 maximizes to the greatest possible value with the spline only using RSS for OLS to determine coefficients. Since p covariates exist for linear regression, the df without any constraints or penalties maximize to p.

2.

1. Since = ∞, ̂ essentially focuses on minimizing the term, ∫ [ ( )]2 . This leads to a flat spline with estimated values falling on the zero (0) line.
2. Since = ∞, ̂ essentially focuses on minimizing the term, ∫ [ ′( )]2 . This leads to a spline that tries to minimize the absolute value of slope at each x-value, while also minimizing the RSS (the first term in arg min argument). The optimal spline would look like a flat line (minimizing absolute slope to zero (0)) falling on the average of true y-values (minimizing RSS).
3. Since = ∞, ̂ essentially focuses on minimizing the term, ∫ [ ′′( )]2 . This leads to a flat spline that attempts to have low curvature, while minimizing RSS. The spline would be very smooth with no rapid changes of slope direction. Since a straight line has the a 2nd derivative of zero (smoothest possible), the spline would be effectually linear regression. The predicted y-values of f would be computed the same as linear regression, = ( )−1 .
4. Since = 0, ̂ has no penalty term effectually regardless of m. The spline would go through each training data point exactly and be overfitted to the training data. This may look like a very squiggly curved line (think Lagrangian interpolating polynomial with many points).

HW5

Caijun Qin

11/19/2021

Install packages



r = getOption("repos")

r["CRAN"] = "http://cran.us.r-project.org"

options(repos = r)

packages <- c('gam', 'randomForest', 'tree', 'Metrics')

install.packages(packages)



* Installing packages into 'D:/Users/qcaij/OneDrive - University of Florida/DESKTOP-R7MUAPV/DATA/Users/qcaij/Doc uments/R/win-library/4.1'
* (as 'lib' is unspecified)
* package 'gam' successfully unpacked and MD5 sums checked
* package 'randomForest' successfully unpacked and MD5 sums checked
* package 'tree' successfully unpacked and MD5 sums checked
* package 'Metrics' successfully unpacked and MD5 sums checked



##

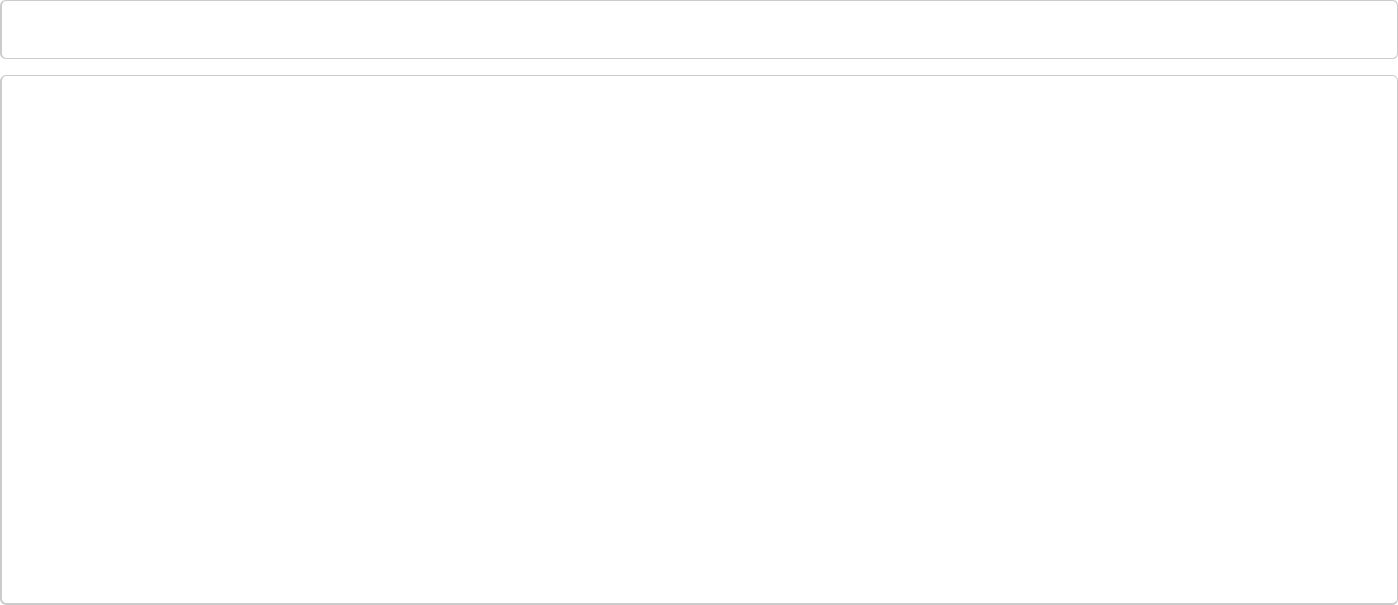
* The downloaded binary packages are in
* C:\Users\qcaij\AppData\Local\Temp\RtmpOcg2Lz\downloaded\_packages



lapply(packages, **library**, character.only = TRUE)



* Loading required package: splines
* Loading required package: foreach
* Loaded gam 1.20
* randomForest 4.6-14
* Type rfNews() to see new features/changes/bug fixes.
* [[1]]



## [1] "gam" "foreach" "splines" "stats" "graphics" "grDevices"

* [7] "utils""datasets" "methods" "base"
* [[2]]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ## | [1] | "randomForest" | "gam" | "foreach" | "splines" | "stats" |
| ## | [6] | "graphics" | "grDevices" | "utils" | "datasets" | "methods" |

* [11] "base"
* [[3]]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ## | [1] | "tree" | "randomForest" | "gam" | "foreach" | "splines" |
| ## | [6] | "stats" | "graphics" | "grDevices" | "utils" | "datasets" |
| ## [11] | | "methods" | "base" |  |  |  |
| ## |  |  |  |  |  |  |
| ## [[4]] | |  |  |  |  |  |
| ## | [1] | "Metrics" | "tree" | "randomForest" "gam" | | "foreach" |
| ## | [6] | "splines" | "stats" | "graphics" | "grDevices" | "utils" |
| ## [11] | | "datasets" | "methods" | "base" |  |  |

Load data



data.train <- read.csv(file = './Data/Problem3train.csv', header = TRUE)

data.test <- read.csv(file = './Data/Problem3train.csv', header = TRUE)

Question 3a

tree.obj <- tree(

formula = y ~ .,

data = data.train,

split = 'deviance'

)

tree.obj



* node), split, n, deviance, yval
* \* denotes terminal node
* 1) root 153 912.600 -0.38520
* 2) x2 < 2.05946 147 593.200 -0.11440
* 4) x2 < -1.85081 5 90.290 5.71300 \*
* 5) x2 > -1.85081 142 327.200 -0.31960
* 10) x2 < 1.33595 121 246.000 -0.06526
* 20) x3 < 0.122611 65 115.200 -0.60140

|  |  |
| --- | --- |
| ## | 40) x2 < -1.43456 6 2.400 1.24900 \* |

* 41) x2 > -1.43456 59 90.160 -0.78970
* 82) x5 < -0.399483 21 28.780 -1.39300

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ## | 164) | x6 | < | -0.922876 | 8 | 4.027 -0.27140 \* |
| ## | 165) | x6 | > | -0.922876 | 13 | 8.491 -2.08300 \* |

* 83) x5 > -0.399483 38 49.500 -0.45610
* 166) x8 < 0.14313 23 16.590 0.07244 \*
* 167) x8 > 0.14313 15 16.630 -1.26700 \*
* 21) x3 > 0.122611 56 90.380 0.55710
* 42) x4 < 1.29418 49 74.770 0.40380 \*

|  |  |
| --- | --- |
| ## | 43) x4 > 1.29418 7 6.398 1.63000 \* |

* 11) x2 > 1.33595 21 28.280 -1.78500 \*
* 3) x2 > 2.05946 6 44.370 -7.02100 \*



tree.cv <- tree::cv.tree(object = tree.obj, FUN = prune.tree)

tree.cv$dev

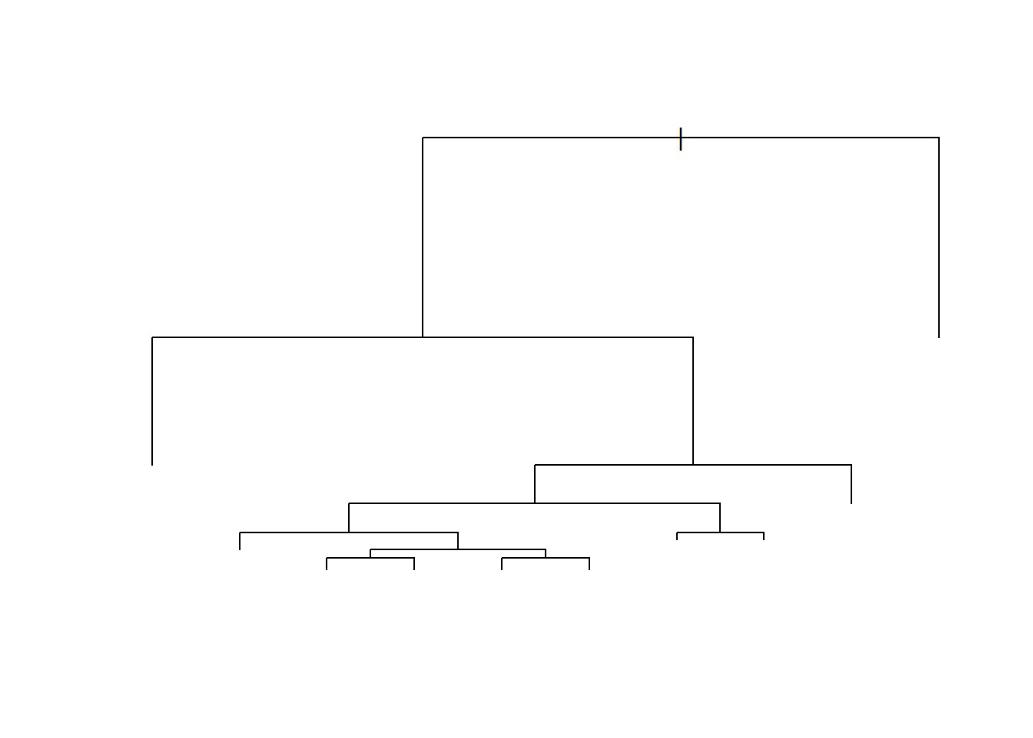


* [1] 611.4880 611.7997 561.0164 539.8354 590.4065 597.5310 833.0016 934.5437 tree.cv$size *# size of 5 corresponds to smallest deviance*
* [1]109654321



tree.pruned <- prune.tree(tree = tree.obj, k = 5)

plot(tree.pruned)



tree.pred <- predict(object = tree.pruned, newdata = data.test) tree.mse <- Metrics::mse(actual = data.test$y, predicted = tree.pred) tree.mse



## [1] 1.910123

The prediction error (MSE) for decision tree on test data set is 1.910123.

Question 3b



rf.obj <- randomForest::randomForest(

formula = y ~ .,

data = data.train,

mtry = ncol(x = data.train) - 1,

ntree = 1000

)

rf.pred <- predict(object = rf.obj, newdata = data.test)

rf.mse <- Metrics::mse(actual = data.test$y, predicted = rf.pred)

rf.mse



## [1] 0.4875422



rf.tree.diff <- tree.mse - rf.mse

rf.tree.diff



## [1] 1.422581

The prediction error (MSE) for bagging (random forest) on test data set is 0.4404094, outperforming a single decision tree by 1.469714. Question 3c



gam.obj <- gam::gam(

formula = y ~ s(x1, 4) + s(x2, 4) + s(x3, 4) + s(x4, 4) + s(x5, 4) + s(x6, 4) + s(x7, 4) + s(x8, 4) + s(x9, 4)

* s(x10, 4),

data = data.train

)

gam.pred <- predict(object = gam.obj, newdata = data.test)

gam.mse <- Metrics::mse(actual = data.test$y, predicted = gam.pred)

gam.mse



## [1] 0.9799886

The prediction error (MSE) for GAM on test data set is 0.9799886.