Seoul National University
Dept. of Civil and Environmental Engineering

Instructor: Junho Song
junhosong@snu.ac.kr

M1586.002500 Information Engineering for CE Engineers
In-Class Material: Class 19
Moving Beyond Linearity (ISL Chapter 7)

1. "Smoothing" Splines

(a) Why "smoothing"?

In regression, we want to find some function, say $g(x)$, that fits the observed data well
→ we want RSS to be small

But, if we don't put any constraint on $g(x)$, then we can always make RSS zero simply by choosing g such that it interpolates all of the y_i.

→ Such a function would woefully overfit the data.

(b) Smoothing splines:

To ensure that g is smooth, one can find the function g minimizing

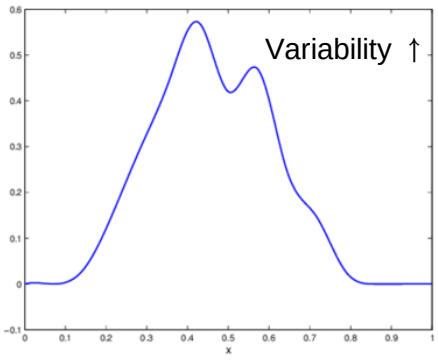
$$\sum_{i=1}^{n}\left(y_i - g(x_i)\right)^2 + \lambda \int g''(t)^2\, dt$$

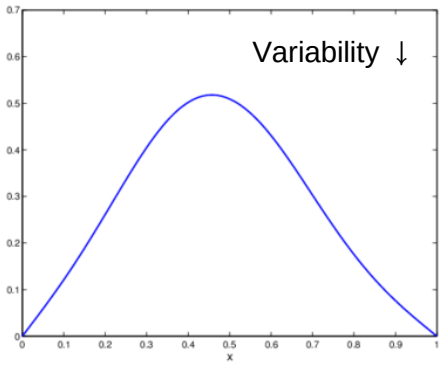
- λ: nonnegative tuning parameter
- g'': the second derivative of a function g

- $\sum_{i=1}^{n}\left(y_1 - g(x_i)\right)^2$: **Loss function** that encourages g to fit the data well
- $\lambda \int g''(t)^2\, dt$: **Penalty term** that discourages the variability in g.

Note: The _____ derivative of a function is a measure of its **roughness**

→ It is large if $g(t)$ is very wiggly near t, and it is close to zero otherwise.
→ A measure of the total change in the function $g'(t)$ over its entire range





→ **"Loss - Penalty" trade-off**

Instructor: Junho Song
junhosong@snu.ac.kr

(c) Tuning parameter λ

For an intermediate value of λ, g will approximate the traning observations but will be somewhat smooth.
→ **λ controls the bias-variance trade-off** of the smoothing spline.

$\lambda = 0$: g can be any function that interpolates the data well
$\lambda = \infty$: The simple least square line fit, since no second derivative can be tolerated

Example 1 (Penalty function in smoothing splines): Suppose that a curve $\hat{g}$ is computed to smoothly fit a set of n points using the following formula

$$\hat{g} = \underset{g}{\operatorname{argmin}} \left(\sum_{i=1}^{n} \left(y_i - g(x_i)\right)^2 + \lambda \int \left(g^{(m)}(x)\right)^2 dx \right)$$

where $g^{(m)}$ represents the m-th derivative of g (and $g^{(0)} = g$). Provide the sketches of $\hat{g}$ in each of the following scenarios.

a) $\boldsymbol{\lambda = \infty, m = 1}$
→ In this case, $\hat{g} = c$ because a large smoothing parameter forces $g^{(1)}(x) \to 0$

b) $\boldsymbol{\lambda = \infty, m = 3}$
→ In this case, $\hat{g} = cx^2 + dx + e$ because a large smoothing parameter forces $g^{(3)}(x) \to 0$

c) $\boldsymbol{\lambda = 0, m = 3}$
→ The penalty term doesn't play any role, so in this case $\hat{g}$ is the interpolating spline.

(d) Properties of $g(x)$

It has been shown that $g(x)$ minimizing $\sum_{i=1}^{n}\left(y_i - g(x_i)\right)^2 + \lambda \int g''(t)^2\, dt$ is a piecewise cubic polynomial with knots at the unique values of $x_1, \ldots, x_n$ → "Splines"

"Shrunken" version of the natural spline regression (cf. Ridge and Lasso)

(e) Optimal shrinkage: choosing the smoothing parameter λ

→ We can find the value of λ that makes the cross-validated RSS as small as possible

For example, the *leave-one-out* cross-validation error (LOOCV) can be computed for smoothing splines

$$\mathrm{RSS}_{cv}(\lambda) = \sum_{i=1}^{n} \left(y_i - \hat{g}_{\lambda}^{(-i)}(x_i)\right)^2$$

$\hat{g}_{\lambda}^{(-i)}(x_i)$ indicates the fitted value for the smoothing spline evaluated at x_i, where the fit uses all of the training observations except for the ith observation (x_i, y_i).

Instructor: Junho Song
junhosong@snu.ac.kr

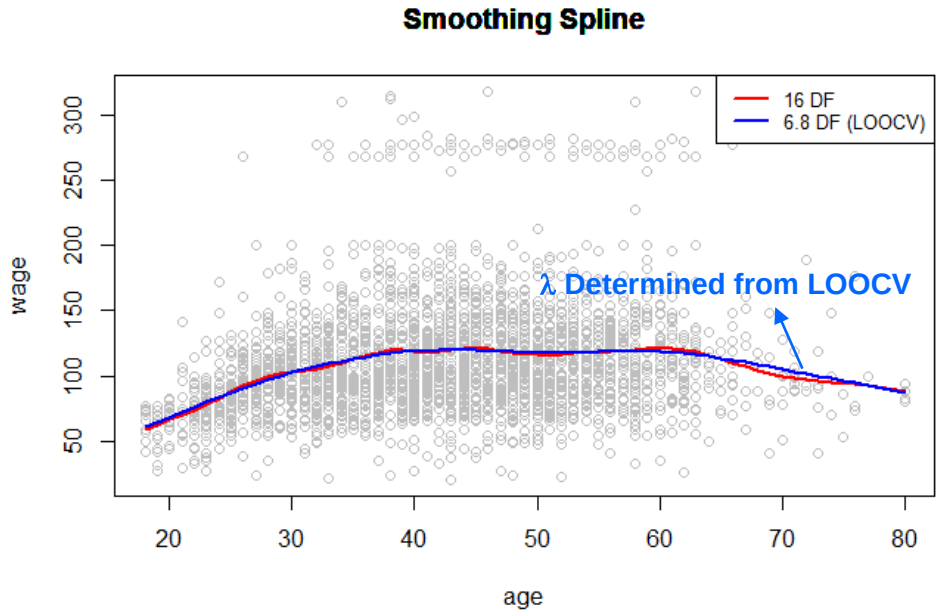
An efficient way to compute $\mathrm{RSS}_{cv}(\lambda)$:

$$\mathrm{RSS}_{cv}(\lambda) = \sum_{i=1}^{n} \left[\frac{y_i - \hat{g}_\lambda(x_i)}{1 - \{\mathbf{S}_\lambda\}_{ii}} \right]^2$$

where $\mathbf{S}_\lambda$ is the $n \times n$ matrix that relates the vectors of fitted values and the original response values in $\hat{\mathbf{g}}_\lambda = \mathbf{S}_\lambda \boldsymbol{y}$. The formula for $\mathbf{S}_\lambda$ is available.

(f) Effective degree of freedom, df_λ of smoothing spline

$$df_\lambda = \sum_{i=1}^{n} \{\mathbf{S}_\lambda\}_{ii}$$




```
library(ISLR) # ISLR library
library(splines) # splines library
attach(Wage)
agelims=range(age)
plot(age,wage,xlim=agelims,cex=.5,col="darkgrey") # scatter plot
title("Smoothing Spline")
fit = smooth.spline(age,wage,df=16) # smoothing splines with dof=16
fit2 = smooth.spline(age,wage,cv=TRUE) # smoothing splines with leave-one-
   out cross-validation
fit2$df
# In this case, the degree of freedom is 6.8
lines(fit,col="red",lwd=2) # smoothing splines line
lines(fit2,col="blue",lwd=2) # polynomial regression line
legend("topright",legend=c("16 DF","6.8
   DF"),col=c("red","blue"),lty=1,lwd=2,cex=.8)
# Red line: dof=16, blue line: dof=6.8 (for LOOCV)
```

Instructor: Junho Song
junhosong@snu.ac.kr

2. Generalized Additive Models (GAMs)

(a) Explore the problem of predicting Y on the basis of **several predictors, $X_1, \ldots, X_p$**.

→ This amounts to an extension of multiple linear regression.

→ GAMs can be applied with both **quantitative** and **qualitative** responses.

(b) GAMs for regression problems:

$$y_i = \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i$$
$$= \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon_i$$

→ It is **additive** model because f_j for each X_j are calculated separately, and then add together all of their contributions.

→ For example, take natural splines and consider the task of fitting the model

$$wage = \beta_0 + f_1(year) + f_2(age) + f_3(education) + \epsilon$$

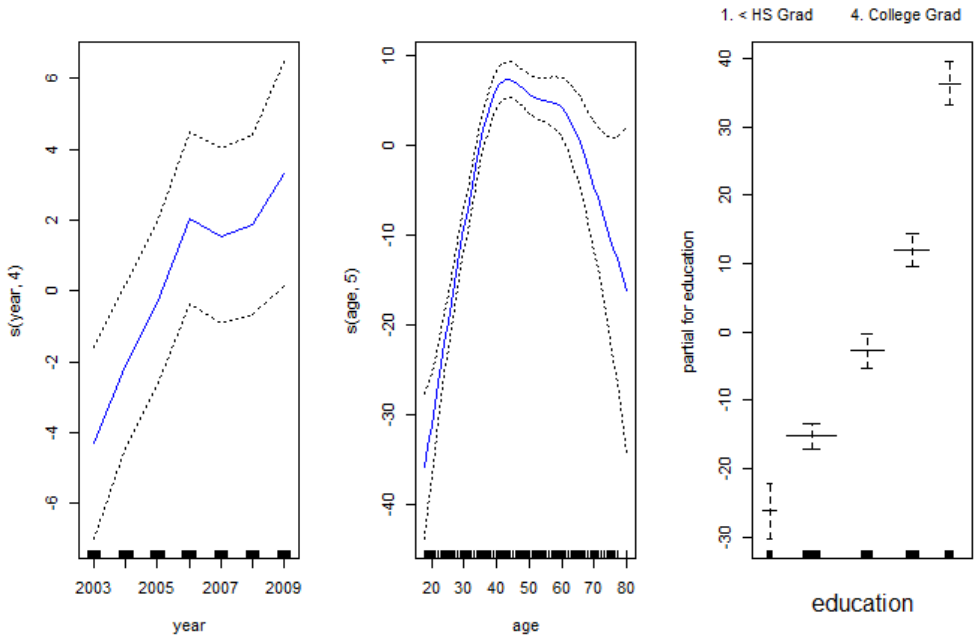
Here year and age are quantitative variables, and education is a qualitative variable with five levels.

→ For quantitative variables**,** fit the functions using smoothing splines with four and five degrees of freedom, respectively.
→ For qualitative variable, use a separate constant for each level, via the usual dummy variable approach (introduced in CM 07)

Note: The *gam()* library is useful for GAMs model

```
library(ISLR) # ISLR library
attach(Wage)
library(gam) # gam library
gam.m3 = gam(wage~s(year,4)+s(age,5)+education,data=Wage)
# Fit a Generalized Additive Models with smoothing spline using gam
   library
par(mfrow=c(1,3))
plot(gam.m3, se=TRUE,col="blue") # Plot fitted function and pointwise
   standard errors
```

Instructor: Junho Song
junhosong@snu.ac.kr



→ The left-hand pael indicates that holding age and education fixed, wage tends to increase slightly with year

Note: Because the GAMs model is addictive, we can still examine the effect of each X_j on Y individually while holding all of the other variables fixed.

(c) Pros and Cons of GAMs

GAMs model	Pros	GAMs fits a non-linear f_j to each X_j, so that we can automatically model non-linear relationship
		The non-linear fits can make more accurate predictions for the response Y
		Because the model is addictive, the effect of each X_j on Y can be examined individually
	Cons	Because the model is restricted to be additive, important interactions could be missed.

→ For "fully" general models, we have to look for even more flexible approaches such as random forests and boosting, described in next classes.

Seoul National University
Dept. of Civil and Environmental Engineering

Instructor: Junho Song
junhosong@snu.ac.kr

M1586.002500 Information Engineering for CE Engineers

In-Class Material: Class 20

Tree-Based Methods (ISL Chapter 8)

1. Decision Tree

(a) Goal: Regression or classification

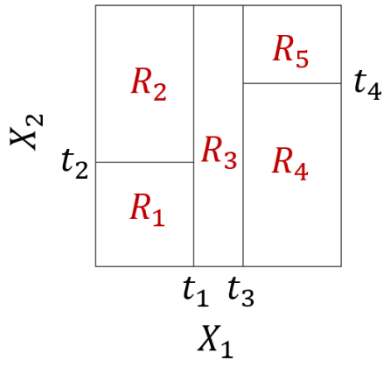
(b) Strategy: **Stratifying** or **segmenting** predictor space into multiple boxes (high-dimensional rectangles) R_m, $m = 1, \cdots, M$

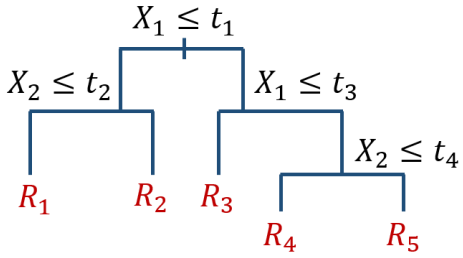
(c) Terminology:
- Predictors: input variables for prediction X_j, $j = 1, \cdots, p$
- Terminal nodes (leaves): the segmented regions R_m, $m = 1, \cdots, M$
- Internal nodes: the points where the predictor space is split

(d) The same prediction is made for every observation in a region R_m
- Regression tree: the **mean response** for the training observations within the *m*th box
- Classification tree: the **most commonly occurring** class of training observations

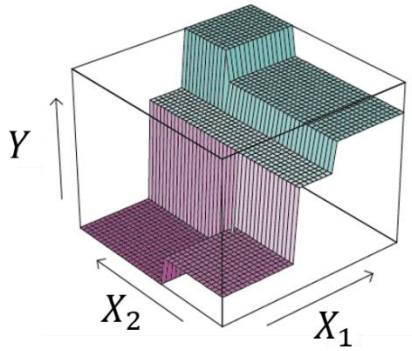
(e) Example: Five-region feature space (predictors X_1 and X_2; and response Y)

- Stratification of predictor space and decision tree





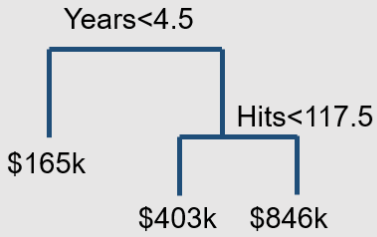
- Perspective plot of the prediction surface



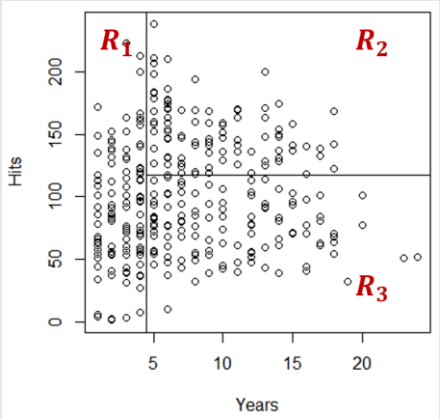
Instructor: Junho Song
junhosong@snu.ac.kr

Example 1 (Application of regression tree): Using the Hitters data set and a decision tree with 3 terminal nodes, predict a baseball player's Salary based on Years (the number of years that he has played in the major leagues) and Hits (the number of hits that he made in the previous year).

Using R package "tree," the tree is created as



which corresponds to the stratification of predictor space into four regions $R_1, \cdots, R_4$ as



Then, we can interpret the result as:

(1) Years is the most important factor in determining Salary; and
(2) Among players who have been in the major leagues for five or more years, Hit becomes influential on Salary.

```
library(tree)
library(ISLR)
attach(Hitters)

Hitters.data = Hitters[which(Hitters$Salary > 0),names(Hitters) %in%
   c("Salary","Years","Hits")]
# rows with positive salary only & columns with those names
Hitters.data["Salary"] = log(Hitters.data["Salary"]) # to have bell-shape

tree.hitters=tree(Salary~.,Hitters.data)
tree.hitters=prune.tree(tree.hitters,best=3)
# prune subtrees to have 3 leaves

plot(tree.hitters)
text(tree.hitters)
```

Instructor: Junho Song
junhosong@snu.ac.kr

2. Building a Decision Tree (1): Stratification

(a) Goal: Find boxes $R_1, \cdots, R_M$ that minimize the reference error measure

(b) Methodology: **Recursive Binary Splitting** which is characterized as

(1) Top-down: Begins at the top of the tree (i.e. a single region), then successively splits the predictor space

(2) Greedy: At each step of splitting, the best split is made without looking ahead

(c) Reference error measure

For **regression trees**, **residual sum of squares (RSS)** is used, i.e.

$$\sum_{m=1}^{M} \sum_{i \in R_m} \left(y_i - \hat{y}_{R_m}\right)^2$$

where y_i is *i*th observation's actual value of predictor; and $\hat{y}_{R_j}$ is the mean response for the training observations within the *m*th box

For **classification trees**, there are three alternative measures:

(1) **Classification error rate**

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

where $\hat{p}_{mk}$ is the proportion of training observations in the *m*th region that are from *k*th class.

However, it turns out that the measure is not sufficiently sensitive for tree-growing, and in practice the following two measures are preferable.

(2) **Gini index**

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

which measures the total variance across the *K* classes – small if all of the $\hat{p}_{mk}$'s are close to zero or one.

(3) **Entropy**

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

Will take on a value near zero if the $\hat{p}_{mk}$'s are all near zero or near one.

Both Gini index and entropy measure the node **purity** – a small value indicates that a node contains predominantly observations from a single class, being numerically quite similar. (**Practice**: Ex. 3 in Section 8.4 – comparison of the three indices)

Instructor: Junho Song
junhosong@snu.ac.kr

3. Building a Decision Tree (2): Tree Pruning

(a) Goal: Avoid overfitting due to too large tree

(b) Strategy: Grow a very large tree T_0, and then **prune** it back to obtain a **subtree** with optimal trade-off between variance (large tree) and bias (small tree)

(c) Methodology: **Cost complexity pruning** (weakest link pruning) in which for each nonnegative value of tuning parameter α, the subtree $T \subset T_0$ is identified to have smallest value of

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

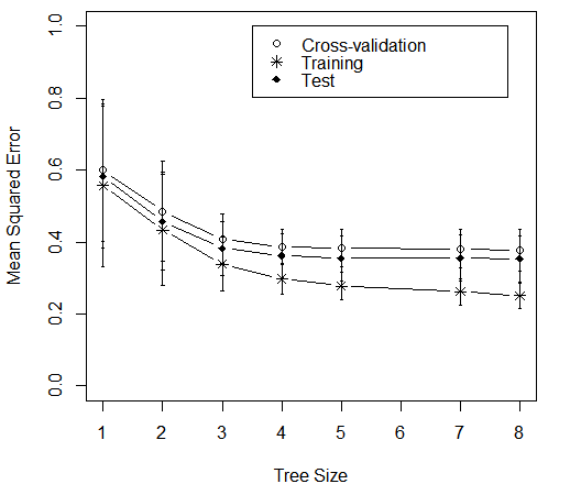
where $|T|$ is the number of terminal nodes in tree T

(d) Tuning parameter α controls the trade-off between variance and bias

(e) To select one among the subtrees identified for each value of α, a validation set or cross-validation can be used.

Example 2 (Pruning): Using the Hitters data set, check how the mean squared errors of (1) 5-fold cross-validation, (2) training data, and (3) test data change as the size of tree increases.

After 100 experiments, the mean squared errors and the corresponding standard deviations are obtained as



Although the training error constantly decreases as the tree size increases, the errors of cross-validation and test data stop decreasing after the tree size reaches 4. This implies that highly complex trees are likely to suffer from overfitting.

Instructor: Junho Song
junhosong@snu.ac.kr

```
library(tree)
library(ISLR)
attach(Hitters)

# select 150 data as training set
set.seed(10)
train = sample(1:nrow(Hitters.data),150)
Hitters.test = Hitters.data[-train,]
Hitters.train = Hitters.data[train,]

tree.hitters = tree(Salary~.,Hitters.data,subset=train)

# 5-fold cross-validation
cv.hitters = cv.tree(tree.hitters,FUN=prune.tree,K=5)
# pruning by training data
prune.hitters.train = prune.tree(tree.hitters)
# pruning by test data
prune.hitters.test = prune.tree(tree.hitters,newdata=Hitters.test)

# plot
plot(cv.hitters$size,cv.hitters$dev/nrow(Hitters.train),type="b",ylab=
'Mean Squared Error',xlab='Tree Size',ylim=c(0,1))
lines(prune.hitters.train$size,prune.hitters.train$dev/nrow(Hitters.train)
,type="b",pch=8)
lines(prune.hitters.test$size,prune.hitters.test$dev/nrow(Hitters.test)
,type="b",pch=18)
```

4. Building a Decision Tree (3): Summary

(a) 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

(b) Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.

(c) Use *K*-fold cross-validation to choose α, i.e. divide the training observations into *K* folds. For each $k = 1, \cdots, K$:

(1) Repeat Steps 1 and 2 on all but the *k*th fold of the training data.
(2) Evaluate the mean squared prediction error on the data in the left-out *k*th fold, as a function of α.

Average the results for each value of α, and pick α to minimize the average error.

(d) Return the subtree from Step 2 that corresponds to the chosen value of α.