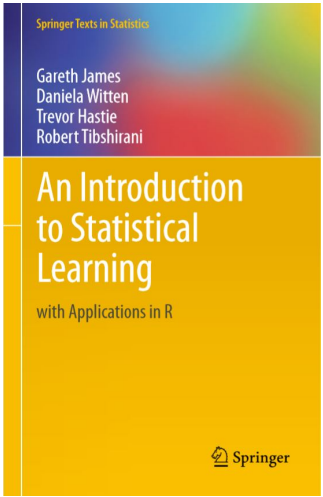
# 깊게 배우는 머신러닝

## Ch.7 Moving Beyond Linearity

훈 러닝 (Hun Learning)

January 14, 2020

# TEXTBOOK




- **An Introduction to Statistical Learning : with Applications in R**
- 목차:
  1. Intro
  2. Statistical Learning
  3. Linear Regression
  4. Classification
  5. Resampling Methods
  6. Linear Model Selection and Regularization
  7. Moving Beyond Linearity
  8. Tree-based Methods
  9. Support Vector Machines
  10. Unsupervised Learning

# MOVING BEYOND LINEARITY

- Linear Model은 즉 "Y와 X의 관계는 선형이다"라는 가정을 의미(선형 가정). 이 가정에 따른 대표적인 fitting 방법이 OLS.
- Linear Model의 장점은 이해하기 쉽다는 것. 그러나 실제로 두 변수의 관계가 완전히 선형인 경우가 얼마나 있겠나? Predictive Power가 달린다..
- 그래서 선형 가정을 유지하면서 Linear Model를 개선하는 방법이 OLS에 Penalty항을 추가한 Lasso와 Ridge, 각 변수를 압축하는 PCA 등이 있었다(6장).
- 이번 장에서는 선형 가정을 조금 완화하면서도 우리가 이해 가능한 모델을 만드는 방법에 대해 알아본다. → **Generalized Additive Models!**

# MOVING BEYOND LINEARITY

**대표적인 Non-linear 방법들**

- **Polynomial Regression:** 2차, 3차항을 넣어보자!
- **Step Functions:** 구간 별로 다른 상수를 fitting 해보자!
- **Regression Splines:** 구간 별로 다르게 1차, 2차, 3차항을 넣어보자!
  - Truncated Polynomial Basis
  - B-Spline Basis
  - Smoothing Splines
- **Local Regression:** 하나하나의 점마다 직선을 그어보자!
- **Generalized Additive Models:** 변수마다 다르게 fitting 해보자!

간단한 논의를 위해 설명변수가 하나($p = 1$)인 경우만 얘기해보자.

# POLYNOMIAL REGRESSION

**Polynomial Regression**

- 그냥 단순히 선형회귀식에 $x^2, x^3, ...x^d$ 항을 추가하는 방법. 엄밀히 말하면 d차항들에 대한 선형 모델이다.
- Regression:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i + \beta_3 x_i + \epsilon_i$$

- Classification (Binary):

$$P(y_i = 1|x_i) = \frac{\exp\left(\beta_0 + \beta_1 x_i + \beta_2 x_i + \beta_3 x_i + \epsilon_i\right)}{1 + \exp\left(\beta_0 + \beta_1 x_i + \beta_2 x_i + \beta_3 x_i + \epsilon_i\right)}$$

  (log-odd에 대하여 선형)
- 3차항보다는 더 나가지 않는다. 고차항일수록 boundary에서 불안정해지기 때문.

**Polynomial Regression**

나이가 많으면서 연봉이 높은 sample이 거의 없기 때문에 신뢰구간이 넓어진다. (why?)

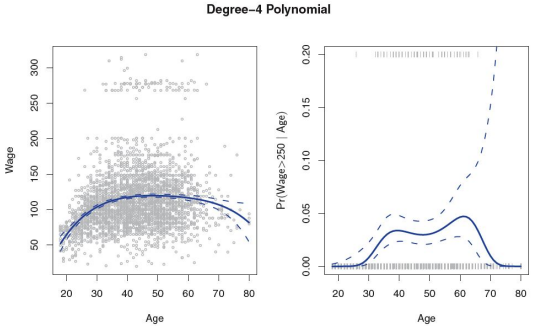



**FIGURE 7.1.** *The* `Wage` *data.* Left: *The solid blue curve is a degree-4 polynomial of* `wage` *(in thousands of dollars) as a function of* `age`*, fit by least squares. The dotted curves indicate an estimated 95 % confidence interval.* Right: *We model the binary event* `wage>250` *using logistic regression, again with a degree-4 polynomial. The fitted posterior probability of* `wage` *exceeding* $250,000 *is shown in blue, along with an estimated 95 % confidence interval.*

**Recall: Inference of mean response**

단순선형회귀에서 오차항의 가정이 $\epsilon \sim N(0, \sigma^2)$ 일때 다음과 같이 쓸 수 있다.

- Mean Response $E(Y_h)$: $X = X_h$로 주어졌을때 평균적으로 기대되는 Y의 값
- Point Estimator of $E(Y_h)$: $\hat{Y}_h = \hat{\beta}_0 + \hat{\beta}_1 X_h$

$$E(\hat{Y}_h) = \beta_0 + \beta_1 X_h$$

$$Var(\hat{Y}_h) = \sigma^2 [\frac{1}{n} + \frac{(X_h - \bar{X})^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2}] \textbf{ (Pointwise Variance)}$$

- $\hat{Y}_h$ 자체는 정규분포를 따르나 모수 $\sigma^2$를 추정치로 대체할 경우,

$$\frac{\hat{Y}_h - E(\hat{Y}_h)}{s^2(\hat{Y}_h)} \sim t(n-2)$$

- t 분포에 따라 $E(Y_h)$의 95% 신뢰구간은

$$\hat{Y}_h \pm s^2(\hat{Y}_h) \cdot t(0.975; n-2)$$

(Note: in MLR, $s^2(\hat{Y}_h) = \sigma^2 X_h^T (X^T X)^{-1} X_h$)

# STEP FUNCTIONS

**Step Functions**

- X를 cutpoint $(c_1, c_2, ...c_K)$에 따라 K+1 구간으로 나눠 각 구간 별로 상수를 fitting하자!

$$
\begin{aligned}
C_0(X) &= I(X < c_1) \\
C_1(X) &= I(c_1 \leq X < c_2) \\
C_2(X) &= I(c_2 \leq X < c_3) \\
&\vdots \\
C_{K-1}(X) &= I(c_{K-1} \leq X < c_K) \\
C_K(X) &= I(c_K \leq X)
\end{aligned}
$$

- Regrssion:

$$y_i = \beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + ... + \beta_K C_K(x_i) + \epsilon_i$$

- Classification(Binary)의 경우도 마찬가지.

## Step Functions

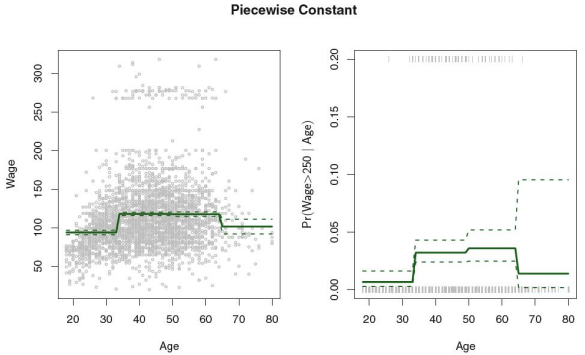



**FIGURE 7.2.** *The* `Wage` *data.* Left: *The solid curve displays the fitted value from a least squares regression of* `wage` *(in thousands of dollars) using step functions of* `age`*. The dotted curves indicate an estimated 95 % confidence interval.* Right: *We model the binary event* `wage>250` *using logistic regression, again using step functions of* `age`*. The fitted posterior probability of* `wage` *exceeding* $250,000 *is shown, along with an estimated 95 % confidence interval.*

# REGRESSION SPLINES

**스플라인?**

- 어떤 cutpoint (혹은 knot)를 기준으로 m+1개의 구간이 있을 때,
  1) 연속인 k차 "piecewise" 다항식으로,
  2) 각 knot에서 1차, 2차, ... k-1차 도함수까지 연속인 선을
  kth spline이라 말한다.[1] → **"구간별로 정의된 매끈한 다항식!"**

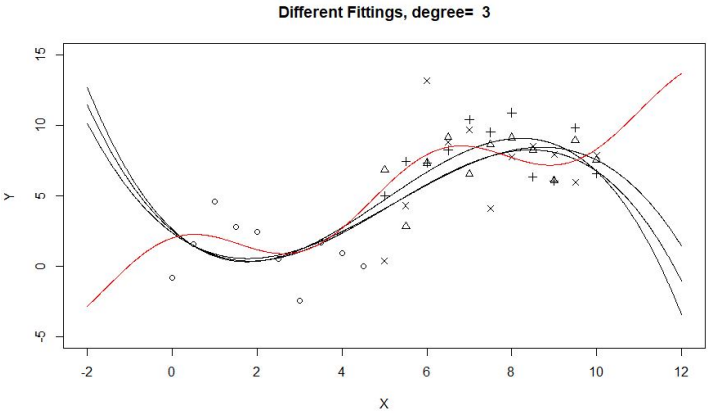
> Formally, a function $f : \mathbb{R} \to \mathbb{R}$ is a $k$th order spline with knot points at $t_1 < \ldots < t_m$, if
>
> - $f$ is a polynomial of degree $k$ on each of the intervals $(-\infty, t_1], [t_1, t_2], \ldots [t_m, \infty)$, and
> - $f^{(j)}$, the $j$th derivative of $f$, is continuous at $t_1, \ldots t_m$, for each $j = 0, 1, \ldots k-1$.

- 1차 spline은 linear spline. 그냥 구간별로 선을 그어 이은 것.
- 가장 대표적인 예는 3차인 cubic spline.
  3차부터는 사람의 눈으로 knot을 구별하지 못할 정도로 매끄럽다.
- 그냥 다항식으로 fitting하면 되지 왜 굳이 스플라인을 쓰는가?

[1] https://www.stat.cmu.edu/ ryantibs/advmethods/notes/smoothspline.pdf

# REGRESSION SPLINES

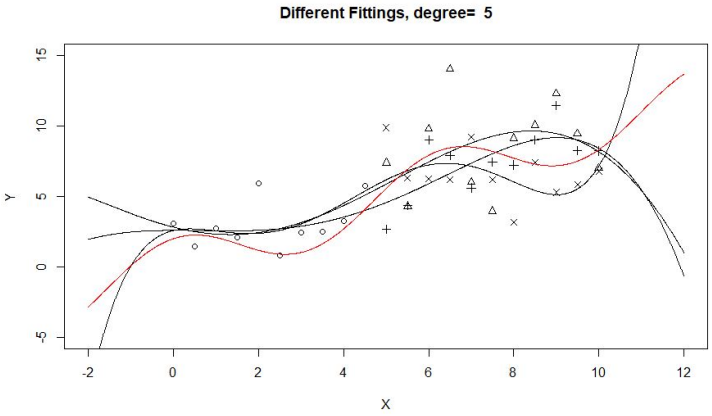
**스플라인?**
다항식의 차수가 높아질수록 데이터가 없는 boundary에서의 fitting이 불안정하다!




[2]R코드 출처 https://www.hds.utc.fr/ tdenoeux/dokuwiki/media/en/splines.pdf

# REGRESSION SPLINES

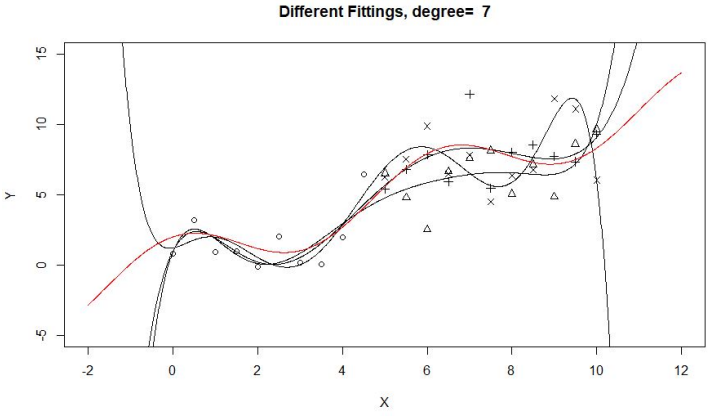
**스플라인?**
다항식의 차수가 높아질수록 데이터가 없는 boundary에서의 fitting이 불안정하다!




[2]R코드 출처 https://www.hds.utc.fr/ tdenoeux/dokuwiki/_media/en/splines.pdf

# REGRESSION SPLINES

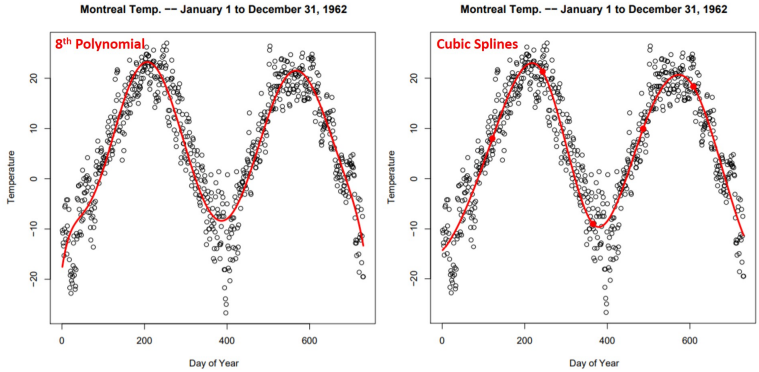
**스플라인?**
다항식의 차수가 높아질수록 데이터가 없는 boundary에서의 fitting이 불안정하다!




[2]R코드 출처 https://www.hds.utc.fr/ tdenoeux/dokuwiki/*media*/*en*/*splines.pdf*

**스플라인?**

Piecewise Polynomial로도 다항식만큼 유연한 fitting이 가능하다!




[3]이미지 출처 http://people.stat.sfu.ca/ cschwarz/Consulting/Trinity/Phase2/TrinityWorkshop/Workshop-material-Simon/Intro$_t$o$_s$plines/intro$_t$o$_s$plines$_n$otes.pdf

# REGRESSION SPLINES

**Spline: Truncated Polynomial Basis**

- 예컨대 3차 spline을 만들기 위해선 각 knot마다 2계 도함수까지는 연속이어야 한다. 이러한 식을 만드는 가장 직관적인 방법은 Truncated Polynomial을 사용하는 것!
- Truncated Polynomial of Degree D: $(x_+ = max\{x, 0\})$

$$(x - \psi_k)_+^D = \begin{cases} 0 & \text{if } x < \xi_k \\ (x - \xi_k)^D & \text{if } x \geq \xi_k \end{cases}$$

for K knots: $(\xi_1, \xi_2, ..., \xi_K)$

- K개의 knot에 대해 D차 spline을 fitting할 경우 회귀식은 다음과 같다.

$$\hat{y}_i = \hat{\beta}_0 + \sum_{d=1}^{D} \hat{\beta}_d x_i^d + \sum_{k=1}^{K} \hat{\beta}_k (x_i - \xi_k)_+^D = \sum_{j=1}^{D+1+K} \hat{\beta}_j g_j(x_i)$$

**Spline: Truncated Polynomial Basis**

- 이 경우 Design Matrix $\mathbb{G} \in \mathcal{R}^{n \times (D+K+1)}$은 다음과 같다.

$$\mathbb{G}_{ij} = g_j(x_i), \ \textit{for } i \in [n], \ j \in [D+K+1]$$

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^D & (x_1-\xi_1)_+^D & \cdots & (x_1-\xi_K)_+^D \\ 1 & x_2 & x_2^2 & \cdots & x_2^D & (x_2-\xi_1)_+^D & \cdots & (x_2-\xi_K)_+^D \\ 1 & x_3 & x_3^2 & \cdots & x_3^D & (x_3-\xi_1)_+^D & \cdots & (x_3-\xi_K)_+^D \\ & & & \vdots & & & & \\ 1 & x_n & x_n^2 & \cdots & x_n^D & (x_n-\xi_1)_+^D & \cdots & (x_n-\xi_K)_+^D \end{bmatrix}_3$$

- Design Matrix의 각 열 $g_j(x)$은 결국 **K개의 knot로 이뤄진 D차 piecewise 다항식이라는 연속함수공간(벡터스페이스)를 "truncated polynomial"로 span하는 기저**로 볼 수 있다.
- 이후는 OLS처럼 $\min_\beta \|Y - G\beta\|_2^2$, 이를 만족하는 $\hat{\beta}$는 $\hat{\beta} = (G^T G)^{-1} G^T Y$로 구할 수 있다.

[3] http://people.stat.sfu.ca/ cschwarz/Consulting/Trinity/Phase2/TrinityWorkshop/Workshop-handouts/TW-04-Intro-splines.pdf

# REGRESSION SPLINES

**Spline: Truncated Polynomial Basis**

- **Degree of Freedom of Spline:** K개의 knot에 대해 D차 spline을 할 경우의 자유도는 다음과 같이 계산한다.[4] (추정해야할 계수의 수로 이해하면 편하다.)

$$(D+1)(K+1) - DK = D + K + 1$$

(예컨대 5개의 knot로 6개의 구간을 나눠 3차 spline을 fitting할 때, 6개의 구간 각각에 대해 상수를 포함한 4개의 회귀계수를 추정해야하니 $(3+1) \times (5+1) = 24$이지만, 첫 구간을 제외한 나머지 5구간에서 추정할 계수는 오직 3차항의 계수이므로 $3 \times 5 = 15$를 빼 총 자유도는 9이다.)
**교재는 이 계산에서 상수항($\beta_0$)을 빼고 센다. 우리도 헷갈리니까 빼고 세자. 그러면 D+K.**

- 예컨대 구간을 나누지 않은 선형회귀(K=0, D=1)는 자유도 1, 구간을 두 개로 나눈 linear spline(K=1, D=1)은 자유도 2, 구간을 다섯 개로 나눈 cubic spline(K=4, D=3)은 7
- **차수가 정해졌을 때 자유도는 곧 knot의 개수를 의미한다.** 에컨대 cubic spline에서 knot가 없으면 자유도는 3, knot가 2개이면 자유도는 5 이런 식.

[4] https://www.hds.utc.fr/ tdenoeux/dokuwiki/*media/en/splines.pdf*

# REGRESSION SPLINES

**Spline: Truncated Polynomial Basis**

- (심화) Degree of freedom은 Spline의 Hat Matrix의 trace와 같다.
- For order Dth spline with given knots $\xi_i$ ($i \in [K]$),

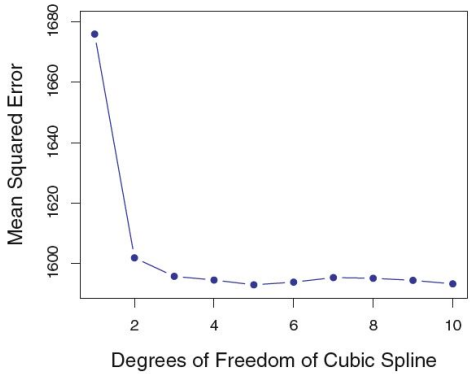
$$\hat{\beta} = (G_\xi^T G_\xi)^{-1} G_\xi^T Y$$
$$\hat{Y} = G_\xi \hat{\beta} = G_\xi (G_\xi^T G_\xi)^{-1} G_\xi^T Y = H_\xi Y$$
$$(H_\xi = G_\xi (G_\xi^T G_\xi)^{-1} G_\xi^T)$$
$$\vdots$$
$$df_{D,K} = trace(H_\xi) = D + K + 1$$

[4] https://www.hds.utc.fr/ tdenoeux/dokuwiki/_media/en/splines.pdf

# REGRESSION SPLINES

**Spline: Truncated Polynomial Basis**

- 자유도의 개념을 이해했으니 이제 **과연 몇개의 knot를 써야하는가**에 대해서 얘기해보자.
- 가장 손쉬운 방법은 여러 자유도($\propto$ knot의 개수)에 대해 k-fold CV를 구하는 것.
- 옆의 예시의 경우 df=4, 즉 1개의 knot로 두 구간 나누는게 제일 적당하다. 그 이상은 별 이득이 없는 것으로 보인다.
- 만일 K개의 knot가 적당하다면 어딜 기준으로 나눠야하는가? 이론적으로는 데이터 변동의 변곡점에 놔야 하겠으나, 그냥 **균일하게 놓는 것이** 일반적이다.

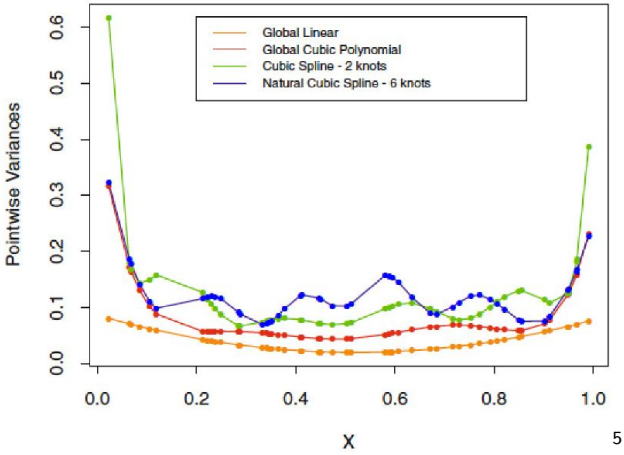


# REGRESSION SPLINES

**Natural Splines**

- D차 spline이 D차 다항식에 비해 유일하게 안 좋은 점은 바로 **Boundary 구간에서 분산이 크다는 것.** 왜 그럴까?
- 직관적으로 생각해보면, 전 구간에 걸쳐 모든 샘플을 사용해 D차 커브를 fitting하는 D차 다항식 fitting에 비해, D차 spline은 knot로 나뉜 각 구간 내의 제한된 샘플을 사용해 D차 커브를 fitting 한다.
- 때문에 데이터가 거의 없는 Boundary(처음과 마지막 knot의 밖 구간)에서는 제한된 샘플로 많은 계수를 추정해야 하므로, 그 구간에서의 추정치의 분포가 자유도가 더 낮은 t분포를 따르니 CI가 더 클 수 밖에 없다.
  (OLS에서 mean response가 t(n-p) 분포를 따르는 것을 기억하자.)
- 이와 같은 이유로 Boundary 구간에서 인위적으로 1차식이 되도록 조건을 가해 **추정 계수의 개수를 줄인 것이 Natural Spline**

**Natural Splines**




[5]

[5]https://www.hds.utc.fr/ tdenoeux/dokuwiki/media/en/splines.pdf

## Natural Splines

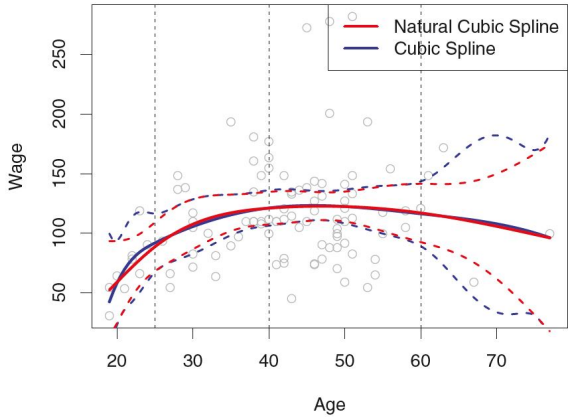
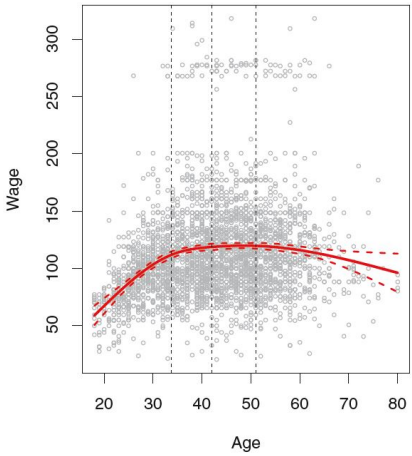



**FIGURE 7.4.** *A cubic spline and a natural cubic spline, with three knots, fit to a subset of the* `Wage` *data.*

**Natural Splines 자유도?**

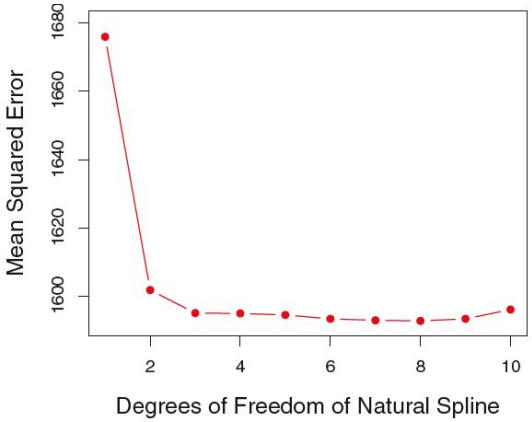
- 이 경우 자유도는 Boundary 구간에서의 제약만큼 감소한다. 예컨대 Cubic Spline의 경우 양 쪽 끝 구간에서 자유도 2씩을 빼 총 4가 빠진다.
- 때문에 knot 개수에 대한 해석도 다르다. Cubic Spline(D=3)에서 K=2이면 자유도는 $3+2=5$ 이지만, Natural Cubic Spline에 자유도는 $3+2-2\times 2=1$이다.
- 옆에 Natural Cubic Spline에서 **K=5이다!** 3이 아니다! 때문에 이 경우 자유도는 $3+5-4=4$!!



**Natural Splines 자유도?**

- Natural Cubic Spline에서 적정 knot의 개수를 알아보기 위해 df에 대해 k-fold CV를 했을 때, 적정 자유도가 3이 나온 경우.
  이때 knot의 개수는?
- $D + K - (2 \times 2) = 3$이므로, D=3이니 K=4.

(애초에 알아보기 쉽게 knot 개수로 표시하지 왜 사람 힘들게 자유도로 적어놨을까? 여러 방법에 걸쳐 자유도가 flexibility를 비교할 수 있는 공통의 척도이기 때문.)



# REGRESSION SPLINES

**Spline: B-spline Basis**

- Truncated Polynomial은 K개의 knot로 이뤄진 D차 piecewise 다항식이라는 연속함수공간(벡터스페이스)를 span하는 기저임을 보았다. 이 경우 Design Matrix는 다음과 같다.

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^D & (x_1-\xi_1)_+^D & \cdots & (x_1-\xi_K)_+^D \\ 1 & x_2 & x_2^2 & \cdots & x_2^D & (x_2-\xi_1)_+^D & \cdots & (x_2-\xi_K)_+^D \\ 1 & x_3 & x_3^2 & \cdots & x_3^D & (x_3-\xi_1)_+^D & \cdots & (x_3-\xi_K)_+^D \\ & & & \vdots & & & & \\ 1 & x_n & x_n^2 & \cdots & x_n^D & (x_n-\xi_1)_+^D & \cdots & (x_n-\xi_K)_+^D \end{bmatrix}$$

- Truncated Polynomial은 개념적으로 간단하나 **1) 각 열 간의 상관관계가 높은 점, 2) 차수가 올라갈수록 rounding으로 인한 오차 문제가 발생한다는 점 때문에 수치적으로 불안정하다.**
- 연속다항식의 함수공간에 여러 기저가 있는 것처럼, K knots Dth order Spline에도 다른 기저가 있다. 그 중 가장 효율적이고 알고리즘으로 구현하기 쉬운 것이 B-spline.

# REGRESSION SPLINES

**Spline: B-spline Basis**

- B-spline Basis를 구성하는 방법은 다음과 같다.[6] knots $\{t_j,\ j = 1, 2, ..., K\}$에서 3차 Cubic Spline을 span한다고 해보자. 먼저 knot를 다음과 같이 무한수열로 확장한다.

$$-\infty..., t_{-1}, t_0, t_1, ..., t_K, t_{K+1}, t_{K+2}, ...\infty$$

- 그 후 먼저 B-spline of order 0을 다음과 같이 정의한다.

$$B_j^0(x) = \begin{cases} 1, & \text{if } t_j \le x < t_{j+1} \\ 0, & \text{otherwise,} \end{cases}$$

- 그 후 j의 양 옆 ($j = 0, \pm 1, ...$)에 대해 그보다 높은 차수 $k \ge 1$의 B-spline은 다음과 같다.

$$B_j^k(x) = \frac{x - t_j}{t_{j+k} - t_j} B_j^{k-1}(x) + \frac{t_{j+k+1} - x}{t_{j+k+1} - t_{j+1}} B_{j+1}^{k-1}(x)$$

- 이렇게 정의된 $B_j^3(x)$, ($j \in [K + 4]$)들이 바로 Cubic Spline의 Basis를 구성한다!

[6] Peihua Qiu (2005). Image Processing and Jump Regression Analysis. Wiley

# REGRESSION SPLINES

**Spline: B-spline Basis**

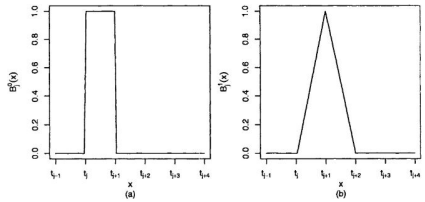
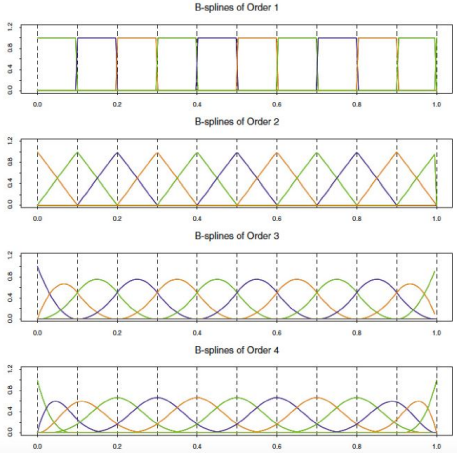



***Fig. 2.5*** Four B-splines when $t_j, t_{j+1}, t_{j+2}, t_{j+3}$, and $t_{j+4}$ are 0, 0.25, 0.5, 0.75, and 1.0. (a): $B_j^0(x)$. (b): $B_j^1(x)$. (c): $B_j^2(x)$. (d): $B_j^3(x)$.[6]

[6] Peihua Qiu (2005). Image Processing and Jump Regression Analysis. Wiley

# REGRESSION SPLINES

**Spline: B-spline Basis**




[7]

[7] https://www.hds.utc.fr/ tdenoeux/dokuwiki/media/en/splines.pdf

**Smoothing Splines**

- 그냥 Spline을 하면 내가 직접 CV 에러를 보면서 knot 개수를 정해줘야한다. 그러나 **모든 데이터를 하나의 knot으로 간주하면**이런 knot selection 문제를 피할 수 있다. 다만 이 경우 끔찍한 Overfitting이 발생하는 것이 문제.
- **목적함수에 Regularization term을 도입해 모든 데이터를 knot로 간주하면서 Overfitting을 방지하는 방법을 Smoothing Spline라고 한다. 결론부터 말하자면 모든 데이터를 통과하는 Natural Cubic Spline이 바로 Smoothing Spline이다.**
- 다음 목적함수를 최소화하는 함수 $f$를 Smoothing Spline이라고 한다.

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

앞의 항을 "loss", 뒤의 항을 "penalty"라고 한다. 2계도함수의 적분을 최소화한다는 것은 곧 함수의 기울기의 변동, 즉 roughness를 최소화한다는 것!

**Smoothing Splines**

- 이때 함수 f는 여러 basis (truncated polynomial 혹은 B-spline)의 선형 결합으로 이뤄졌다. 때문에 위의 조건식은 아래와 같이 다시 쓸 수 있다. (데이터 총 n개를 모두 knot로 취급하였으니 총 n개의 truncated polynomial과 그에 대한 계수의 합으로 볼 수 있음)

$$\textbf{Smoothing Spline } f(x_i) = \sum_{j=1}^{n} \beta_j g_j(x_i)$$

$$\textbf{Design Matrix } G_{ij} = g_j(x_i) = (x_i - x_j)_+^D$$

$$\text{where } G = \begin{bmatrix} (x_1 - x_1)_+^D & (x_1 - x_2)_+^D & \cdots & (x_1 - x_n)_+^D \\ (x_2 - x_1)_+^D & (x_2 - x_2)_+^D & \cdots & (x_2 - x_n)_+^D \\ \vdots & \vdots & \ddots & \vdots \\ (x_n - x_1)_+^D & (x_n - x_2)_+^D & \cdots & (x_n - x_n)_+^D \end{bmatrix} \in \mathcal{R}^{n \times n}$$

**Smoothing Splines**

- 이를 바탕으로 조건식을 다시 쓰면

$$\min_{\beta} \|Y - G\beta\|_2^2 + \lambda \beta^T \Omega \beta$$

- 이때 Ω는 다음과 같이 정의한다.

$$\Omega_{ij} = \int g_i''(t) g_j''(t) dt$$

- 간단한 미분을 해보면 이 식을 만족하는 $\hat{\beta}$는

$$\hat{\beta} = (G^T G + \lambda \Omega)^{-1} G^T Y$$

- 이를 통해 fitted된 Smoothing Spline의 식은

$$\textbf{Fitted Smoothing Spline } \hat{f}(x) = \sum_{j=1}^{n} \hat{\beta}_j g_j(x)$$

**Smoothing Splines**

- **Interior Knots:** 실제로는 n개의 데이터 모두로 하면 $G \in \mathcal{R}^{n \times n}$이 너무 크니 그보다 작은, 예컨대 $log(n)$에 비례하는 개수만큼 interior knots를 설정한다.
- **Smoothing Spline의 자유도:** 일반적인 Spline에서 hat matrix의 trace를 자유도로 정의한 것과 마찬가지로 정의한다. 이를 Effective Degree of Freedom이라 한다.

$$\begin{aligned}
\hat{\beta} &= (G^T G + \lambda\Omega)^{-1} G^T Y \\
\hat{Y} &= G\hat{\beta} = G(G^T G + \lambda\Omega)^{-1} G^T Y \\
(S_\lambda &= G(G^T G + \lambda\Omega)^{-1} G^T) \\
&\vdots \\
df_\lambda &= trace(S_\lambda)
\end{aligned}$$

**Smoothing Splines: Smoothing Parameter**

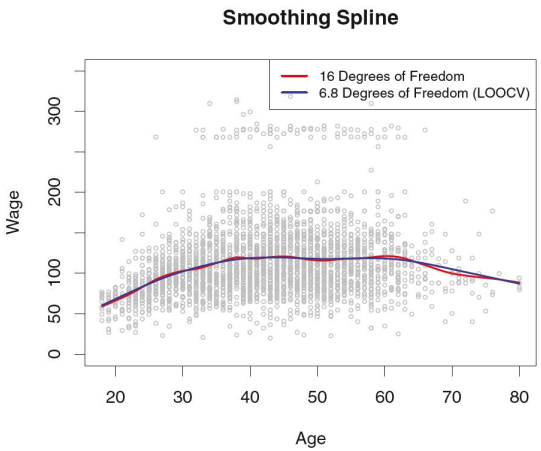
$$\min_{\beta} \|Y - G\beta\|_2^2 + \lambda\beta^T\Omega\beta$$

$$df_\lambda = trace(S_\lambda)$$

- Smoothing spline은 natural cubic spline(D=3)이지만 모든 점에서 knots를 가지기에 자유도가 굉장히 높을 것 같다. $\lambda$**의 조정에 따라 Effective df를 조정할 수 있다!**
- $\lambda$가 0으로 갈수록, 즉 제약을 적게 걸수록 fitting이 과적화되어 결국은 n개의 데이터를 한 치의 오차 없이 그대로 예측하게 된다. 이때 자유도는 데이터의 개수 n이 되어 (n개의 관측치를 설명하는데 모수가 n개?!) hat matrix $S_\lambda$는 그냥 $I_n$이 된다.
  (As $\lambda \to 0$, $df_\lambda \to n$, $S_\lambda \to I_n$)
- 반대로 $\lambda$가 무한대로 가면, 어떠한 굴곡도 없는 함수가 되어 사실상 회귀분석과 마찬가지가 된다.
  (As $\lambda \to \infty$, $df_\lambda \to 2$, $S_\lambda \to H$)

**Smoothing Splines: Smoothing Parameter**

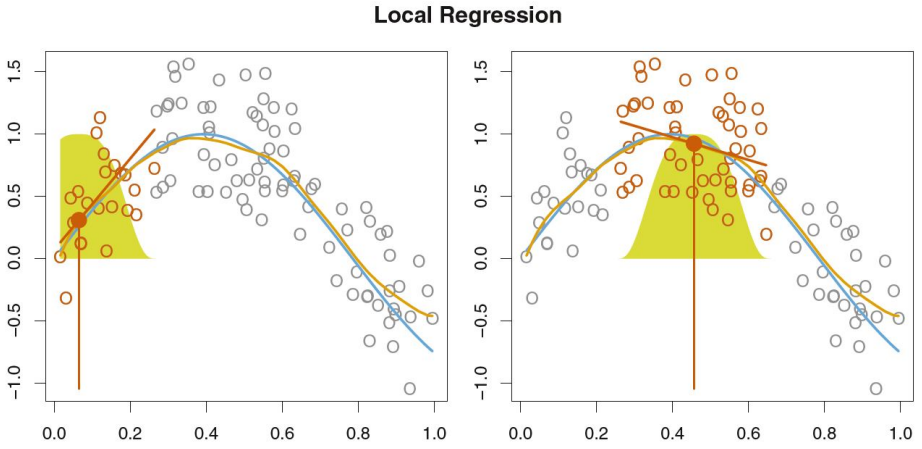
- $\lambda$를 어떻게 조정하느냐에 따라서 Effective df가 달라지므로 다음과 같이 $\lambda$를 결정할 수 있다.
  1. LOOCV나 k-fold CV 등이 최저인 $\lambda$와 그에 대응하는 자유도를 선택하거나
  2. 미리 정한 자유도에 따라 대응하는 $\lambda$를 결정한다.
- 옆의 경우 미리 16으로 정한 자유도에서의 fit과 LOOCV 기준 최저인 자유도 6.8에서의 fit이 거의 차이가 없으므로 6.8를 선택한다.
  (Natural cubic spline에서 자유도 6.8은 K=7.8, 즉 knots 약 8개에 대응한다.)



# LOCAL REGRESSION

**Local Regression**

X의 범위 내 모든 점에서 **가중치를 둔 회귀분석**을 하여 그은 직선을 이으면 결국 곡선이 되더라!



## Local Regression

**Algorithm 7.1** *Local Regression At $X = x_0$*

1. Gather the fraction $s = k/n$ of training points whose $x_i$ are closest to $x_0$.

2. Assign a weight $K_{i0} = K(x_i, x_0)$ to each point in this neighborhood, so that the point furthest from $x_0$ has weight zero, and the closest has the highest weight. All but these $k$ nearest neighbors get weight zero.

3. Fit a *weighted least squares regression* of the $y_i$ on the $x_i$ using the aforementioned weights, by finding $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize

$$\sum_{i=1}^{n} K_{i0}(y_i - \beta_0 - \beta_1 x_i)^2. \tag{7.14}$$

4. The fitted value at $x_0$ is given by $\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$.

# LOCAL REGRESSION

**Local Regression**

- Local Regression은 2번 단계에서 근처 점들에 비중($K_i$)을 어떻게 줄 것이냐, 3번 단계에서 1차, 2차, 3차 fitting을 할 것이냐에 따라서 다르지만, 가장 중요한 것은 **1번 단계에서 주변에 몇 번의 점을 볼 것인가(span)이다!** (KNN과 비슷)
- 주변에 많은 점을 볼수록 (span)이 넓을수록 더욱 평탄한 곡선을 기대할 수 있다.
  - **Varying Coefficients Model:** $X_1$외에 여러 predictor가 있는 경우에 적용해, 어떤 predictor에 한해 그 값에 따라 계수가 달라지는 모델을 만들 수 있다. (global in some, local in the others)
  - 나아가 두 변수에 대해 동시에 Local Regression을 할 수도 있지만, 변수가 3개 이상이면 근처에 있는 관측치를 찾기 어렵기 때문에 제한적.

# GENERALIZED ADDITIVE MODELS

**GAM for Regression**

각 변수마다 적절한 Linear, Non-linear 방법으로 fitting해, 그걸 다 더하는게 GAM!

- 일반적인 선형회귀 모형이 아래를 가정했다면

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \epsilon_i$$

- 이를 일반화한 GAM의 가정은 다음과 같다.

$$y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + ... + f_p(x_{ip}) + \epsilon_i$$

- **이때 $f_j(x_{ij})$를 Design Matrix로 나타낼 수만 있다면 형태에 제한이 없다.** 즉 한 변수에는 natural spline, 다른 변수는 step function, 또 다른 변수는 linear regerssion 를 해도 된다는 것!

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

**(어떤 방법을 쓰던 그에 따라 하나의 큰 Design Matrix를 만들고 나면 결국은 그냥 OLS!)**

## GAM for Regression

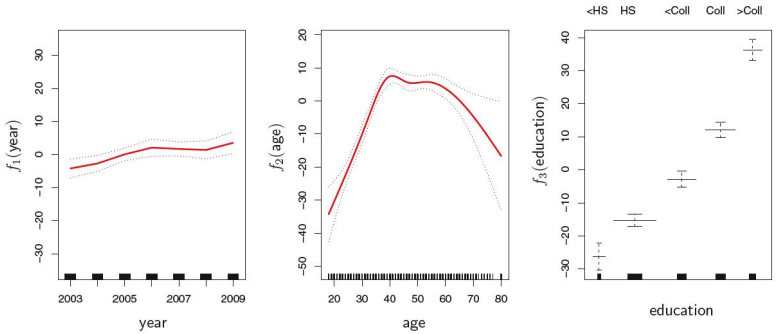



**FIGURE 7.11.** *For the* `Wage` *data, plots of the relationship between each feature and the response,* `wage`*, in the fitted model (7.16). Each plot displays the fitted function and pointwise standard errors. The first two functions are natural splines in* `year` *and* `age`*, with four and five degrees of freedom, respectively. The third function is a step function, fit to the qualitative variable* `education`*.*

# GENERALIZED ADDITIVE MODELS

**GAM for Regression**

Smoothing Spline의 경우 Design Matrix로 나타낼 수 있어도 Regularization Term 때문에 OLS 방법으로 fitting 할 수 없다. 이 경우 **Backfitting** 알고리즘을 사용.

**Backfitting Algorithm**[8]

1. (Initial Condition) p개의 변수에 대해 가정에 따라 "적절한" 모델 구색을 맞춰주고, $\beta_0 = \bar{y}$로 설정한다.
2. (Iteration) $for(i\ in\ 1 \sim p)$, $f_i(X)$를 제외한 다른 모든 $f_j(X)$를 가지고 Y에다가 갖다 빼면 Partial Residual이 나옴. 거기다가 대고 Regularization 텀에 맞춰 양껏 Smoothing 해줘서 $f_i(X)$를 업데이트.
3. 이 짓을 $\hat{Y}$의 값이 어느 정도 안정될 때까지 계 속 반복!

[8] http://ugrad.stat.ubc.ca/ nancy/526$_2$003/*projects*/*kazi*2.*pdf*

**GAM for Regression**

대부분의 경우 Backfitting을 열심히 해서 나온 Smoothing Spline이나 Natural Spline이나 거기서 거기.

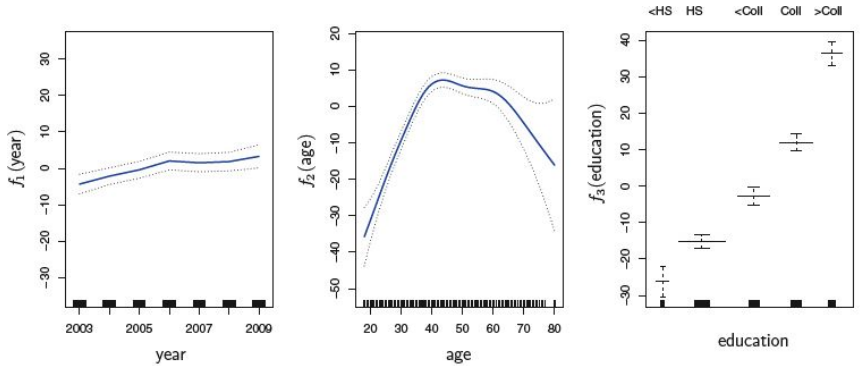



**FIGURE 7.12.** *Details are as in Figure 7.11, but now $f_1$ and $f_2$ are smoothing splines with four and five degrees of freedom, respectively.*

# GENERALIZED ADDITIVE MODELS

**GAM: Pros vs Cons**

- **장점:** Additive Model이므로 "다른 변수를 고정한 채" 한 변수의 변동으로 인한 Y의 변화를 직접 볼 수 있다.
- **단점:** Additive Model이므로 "여러 변수의 상호작용"을 볼 수 없다. 다만 Design Matrix에 interation term을 넣을 수는 있다.

GAM보다 더 일반적인 모델은 8장에서...(Random forest, Boosting, ...)