Penalized B-spline

Gayoung Moon

2025-06-06

School of Mathematics, Statistics and Data Science
Sungshin Women's University

Outline

What Is Penalized B-spline

- Penalized B-spline(P-spline) is a spline method that adds a penalty term to B-spline to create a smoother curve.

- P-spline can solve the problems that occur in B-spline regression:
  - The problem sensitive to number or location of knots.
  - The problem that the model is too flexible and vibrates.

The Function of P-spline

- The following is the function of P-spline:

$$\hat{y} = \sum_{j=1}^{K} \beta_j B_j(x)$$

and we minimize the equation that follows as:

$$\begin{aligned}&\min_{\beta}\{\|y - B\beta\|^2 + \lambda\|D^{(d)}\beta\|^2\}\\ =&\min_{\beta}\left\{\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{K}\beta_j B_j(x_i)\right)^2 + \lambda\sum_{j=1}^{K}(\Delta^d\beta_j)^2\right\}.\end{aligned}$$

The Coefficient of P-spline

- We can also compute the coefficient of P-spline $\hat{\beta}$ like:

$$\hat{\beta} = \left(B^\top B + \lambda D^{(d)\top} D^{(d)}\right)^{-1} B^\top y.$$

- y: vector of the responses that are observed,
  B: design matrix based on B-spline basis,
  β: vector of B-spline coefficients,
  λ: smoothing parameter,
  $D^{(d)}$: difference matrix that computes the d-th order difference of β.

Outline

P-spline Package

- P-spline package is a package that receives the number of knots, adjusts their positions, and performs penalized b-spline regression.

function.	explain
knots	Function of knots sequence using the quantiles of the data
bspline_basis	B-spline basis function and design matrix of B-spline basis
fit_pspline	Fitting P-spline model
predict_pspline	Predicting the values of y for given x based on P-spline
plot_pspline	Drawing a plot of P-spline

The Variables Using in Package

variable	explain
x	Numeric vector of the data
new__x	Numeric vector expressing a grid of evaluation points
grid__x	Numeric vector with a grid of evaluation points
x__values	Numeric vector of x (input)
y__values	Numeric vector of y (response)
model	Function that fit P-spline with given data points
interior__knots	Vector of interior knots
dimension	Number of the basis function
degree	Degree of spline(default= 3)
lambda	Smoothing parameter (default = 1e-2)
diff__order	Order of the difference penalty (default = 2)

Computing Knots Using quantile

```r
knots_quantile <- function(x, dimension, degree = 3){
  dimension = max(dimension, degree + 1)
  num_interior = dimension - degree - 1

  if (num_interior > 0){
    probs <- (1:num_interior)/(num_interior + 1)
    interior_knots <- quantile(x, probs, type = 1)
  }
  else{
    interior_knots <- numeric(0)
  }

  return(interior_knots)
}
```

Adding Boundary Knots to Interior Knots

```r
# Add boundary knots to interior knots.

add_boundary_knots <- function(x, interior_knots, degree = 3, tiny = 1e-5){
  knots<- c(rep(min(x) - tiny, degree + 1),
            interior_knots, rep(max(x) + tiny,
            degree + 1))
  return(knots)
}
```

Definition B-spline Basis Function

```r
# B-spline basis function

b_spline <- function(x, knots, degree, i){
  # Base case: 0th degree
  if (degree == 0){
    return(ifelse(knots[i] <= x & x < knots[i + 1], 1, 0))
  }

  # Recursive case: degree > 0
  B_i_d1 <- b_spline(x, knots, degree - 1, i)
  B_i1_d1 <- b_spline(x, knots, degree - 1, i + 1)

  denom1 <- knots[i + degree] - knots[i]
  denom2 <- knots[i + degree + 1] - knots[i + 1]

  term1 <- if (denom1 == 0) 0 else ((x - knots[i])/denom1)*B_i_d1
  term2 <- if (denom2 == 0) 0 else ((knots[i + degree + 1] - x)/denom2)*B_i1_d1

  return(term1 + term2)
}
```

Definition Design Matrix of B-spline Basis

```r
# Create design matrix of B-spline basis

create_design_matrix <- function(x_values, knots, degree){
  n <- length(x_values) # number of the data
  num_basis <- length(knots) - degree - 1 # number of the basis function

  design_matrix <- matrix(0, nrow = n, ncol = num_basis)

  for (j in 1:num_basis){
    for (i in 1:n){
      design_matrix[i, j] <- b_spline(x_values[i], knots, degree, j)
    }
  }

  return(design_matrix)
}
```

Fitting P-spline Model

```r
fit_pspline <- function(x_values, y_values,
                        interior_knots, degree = 3,
                        lambda = 0.01, diff_order = 2){

  knots <- add_boundary_knots(x_values, interior_knots, degree)

  G <- create_design_matrix(x_values, knots, degree)
  # Design Matrix of B-spline

  num_basis <- ncol(G)
  D <- diff(diag(num_basis), differences = diff_order)

  P <- t(D) %*% D # Penalty Matrix

  beta_hat <- solve(t(G) %*% G + lambda * P) %*% t(G) %*% y_values

  return(list(beta = beta_hat, knots = knots,
              degree = degree, lambda = lambda,
              penalty = P))
}
```

Predicting the Values of Data Based on P-spline

```r
predict_pspline <- function(model, new_x){
  G_new = create_design_matrix(new_x, model$knots, model$degree)
  y_pred = G_new %*% model$beta
  return(y_pred)
}
```

Plotting P-spline with Scatter Plots

```r
library(ggplot2)

plot_pspline <- function(x_values, y_values, model, grid_x, num_knots, lambda,
        point_size, line_size){
  y_pred = predict_pspline(model, grid_x)
  data_plot = data.frame(x = x_values, y = y_values)
  spline_plot = data.frame(x = grid_x, y = y_pred)

  ggplot() +
    geom_point(data = data_plot, aes(x, y),
               color = "blue", size= point_size) +
    geom_line(data = spline_plot, aes(x, y),
              color = "red", linewidth = line_size) +
    labs(title =
        sprintf("Fitted P-spline Regression (knots= %d, lambda= %.2f)",
        num_knots, lambda), x = "x", y = "y") +
    xlim(c(min(x_values), max(x_values)))
}
```

Outline

Generating Data set

```r
set.seed(924)
n= 100
x_values= sort(runif(n, 0, 1))
y_values= sin(2*pi*x_values) + cos(4*pi*x_values) + rnorm(n, sd= 0.2)
```

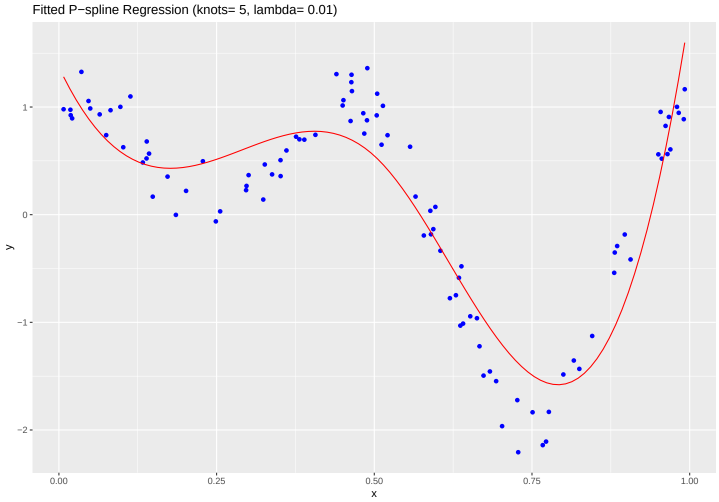
Using P-spline Package

```r
# devtools::install_github('Ga-young-Moon/Penalized-B-spline')
library(Pspline)

# spline degree specification
degree= 3

# knot generation
num_interior= 5
interior_knots= knots_quantile(x_values, num_interior)

# setting the value of lambda
# and the difference penalty order
lambda= 1e-2
diff_order= 2

# model fitting
model= fit_pspline(x_values, y_values, interior_knots,
                   degree, lambda, diff_order)
```

```
##
```

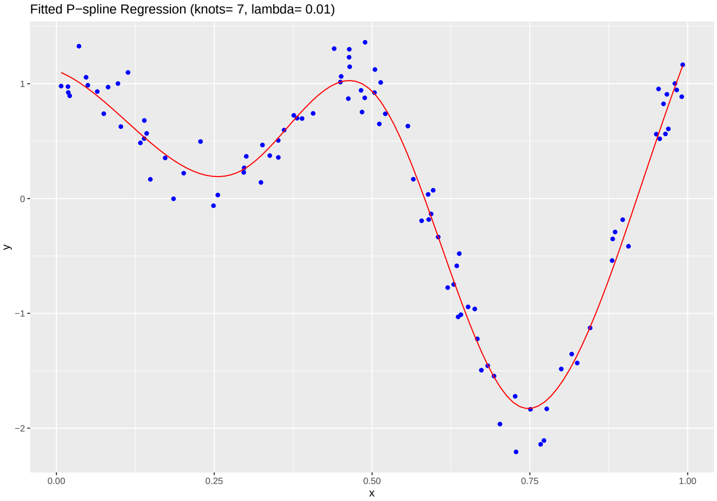
Drawing the Plot of Data Point

```
grid_x = seq(min(x_values), max(x_values), length.out = 100)
plot_pspline(x_values, y_values, model, grid_x, 5, 1e-2, 1.5, 0.5)
```



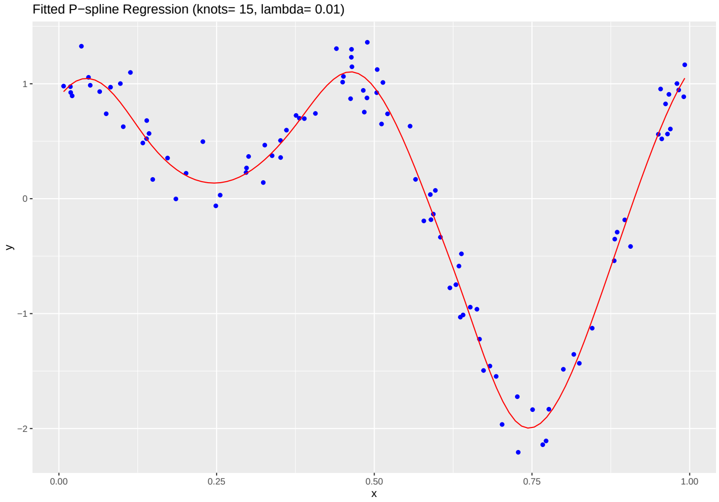
Plot Shape According to Number of Knots (Knots= 7)

- From the plot above, we can see that when there are 5 knots, the spline curve does not follow the data very well.
  $\rightarrow$ Therefore, we need to check the plot when the number of knots is increased.
- If the number of knots is 7:



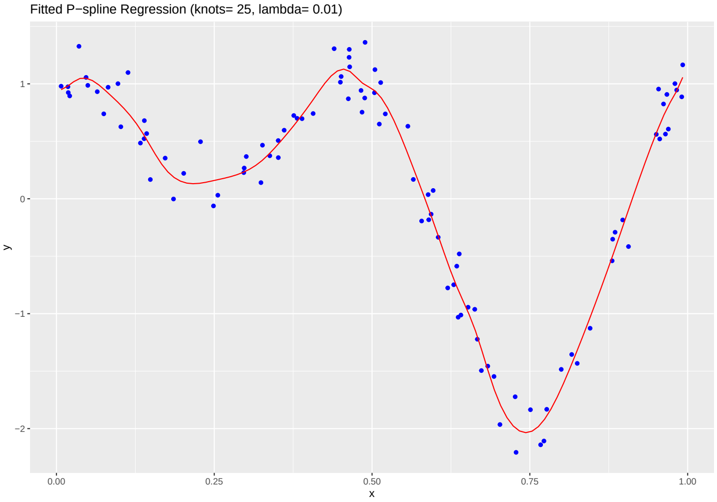
Plot Shape According to Number of Knots (Knots= 15)

- If the number of knots is 15:



Plot Shape According to Number of Knots (Knots= 25)

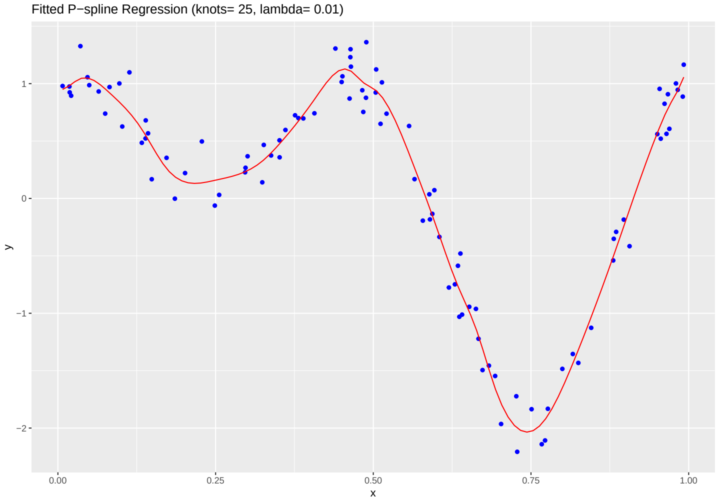
- If the number of knots is 25:



- As the number of knots increases, we can see that the spline model becomes more flexible.

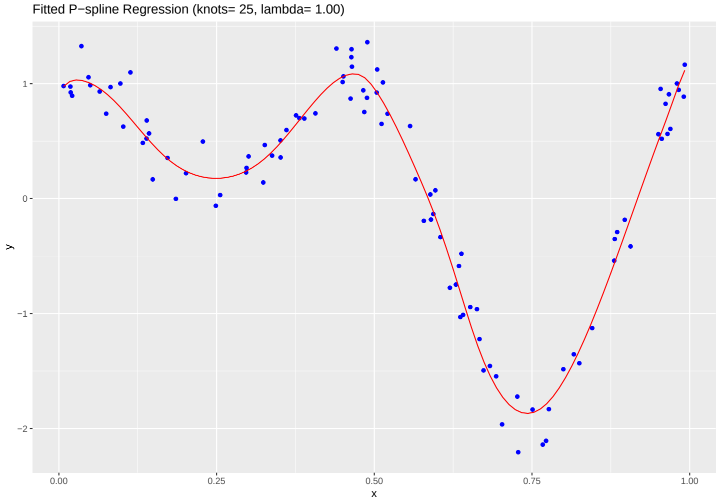
Plot Shape According to Value of λ ($\lambda = 0.01$)

- Then, we can also see how the plot changes depending on λ value.



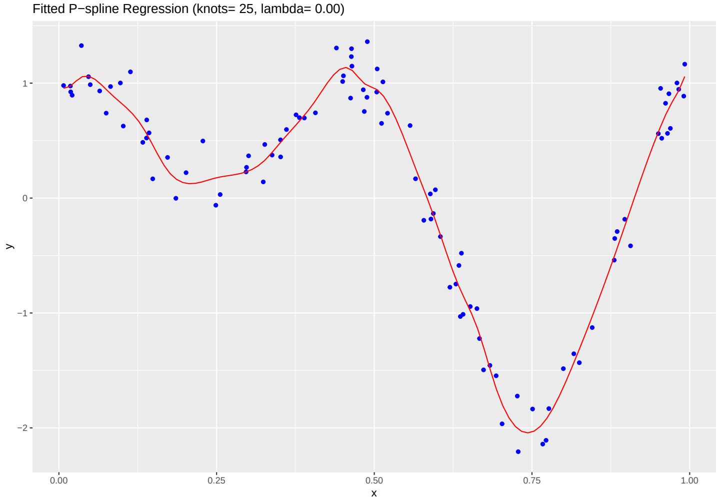
Plot Shape According to Value of λ ($\lambda = 1$)

- If λ= 1:



Plot Shape According to Value of λ ($\lambda = 0.0001$)

- If λ= 0.0001:



- The smaller λ value, the better the curve follows the data.

Q & A

Q & A

Thank you :)