Cubic Spline Function Result

Koki Kitamori

2022 年 11 月 20 日

1

A)

cubic regression spline で CSV で与えられたデータ点を，以下に記載するじ Julia 言語によるプログラムで近似した．赤色の線が各領域での近似曲線を繋ぎ合わせたスプライン関数である．求める手順を以下に示す．

まず，与えられたデータから行列 X，ベクトル $\boldsymbol{y}$ を作る．c_k は k 番目の knot の位置である．

$$\mathbf{X} = \begin{pmatrix} h_0(x_1) & h_1(x_1) & \cdots & h_{K+3}(x_1) \\ \vdots & \vdots & & \vdots \\ h_0(x_k) & h_1(x_k) & \cdots & h_{K+3}(x_k) \\ \vdots & \vdots & & \vdots \\ h_0(x_n) & h_1(x_n) & \cdots & h_{K+3}(x_n) \end{pmatrix}, \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_k \\ \vdots \\ y_n \end{pmatrix} \tag{1}$$

ここで，$h_0(x) = 1, h_1(x) = x, h_2(x) = x^2, h_3(x) = x^3, h_{k+3}(x) = (x - c_k)_+^3, k = 1, 2, \ldots, K$ である．次に，

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\boldsymbol{y} \tag{2}$$

により，$\hat{\boldsymbol{\beta}}$ を求めることができる．最後に次の計算をすれば，3 次スプライン関数が求まる．

$$f(x) = \hat{\beta}_0 h_0(x) + \hat{\beta}_1 h_1(x) + \cdots + \hat{\beta}_{K+3} h_{K+3}(x) = \sum_{k=0}^{K+3} \hat{\beta}_k h_k(x) \tag{3}$$

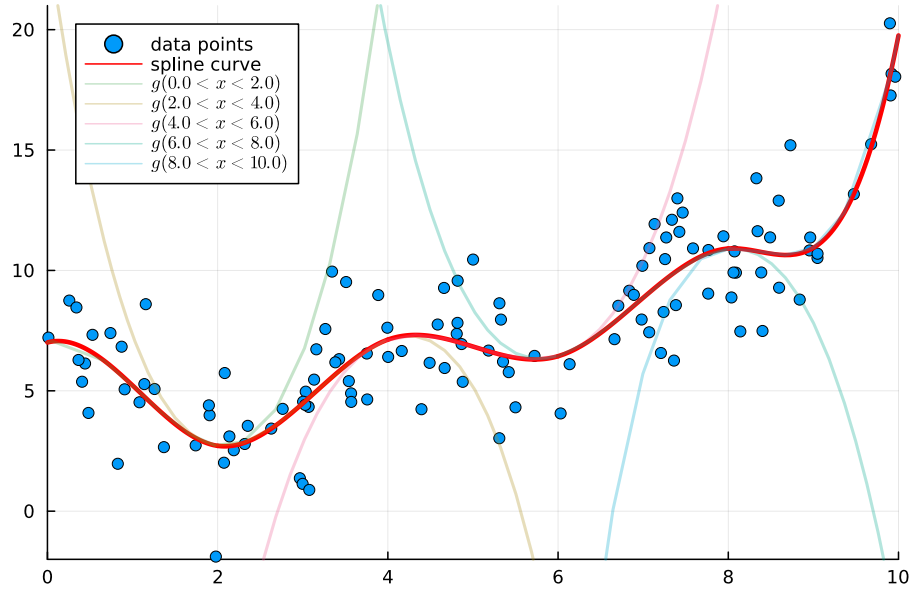

図 1: 4 knots cubic regression spline

```
# load library
using CSV, Plots, DataFrames, Polynomials, LinearAlgebra

# define cubic spline function and return x and fitted polynomial function
function cubic_regression_spline(x, y, K, k)
    X = [j ≤ 3 ? x[i]^j : max(0, (x[i] - k[j-3])^3) for i in eachindex(x), j in 0:K+3]
    # calculate linear regression
    β = (X' * X) \ X' * y
    # declare Polynomial array
    g = Polynomial[Polynomial(β [1:4])]
    # calculate spline function for all cases
    for i in 1:K
        push!(g, g[end] + β [i+4] * Polynomial([-k[i]^3, 3k[i]^2, -3k[i], 1]))
    end
    return X, g
end

# define function in otder to plot the spline function
function plot_spline(x, y, K, g, st)
    scatter(x, y, label="data points", legend=:topleft)
    for i in 1:K+1
        i == 1 ? plot!(g[i], label="spline curve", xlims=(st * (i - 1), st * i), c="red", lw=3) :
        plot!(g[i], label="", xlims=(st * (i - 1), st * i), c="red", lw=3)
        plot!(g[i], label="\$g($(round(st * (i - 1), digits=1))<x<$(round(st * i, digits=1)))\$",
            xlims=(0, 10), ylims=(-2, 21), xaxis=0:2:10, alpha=0.3, lw=2)
    end
    savefig("$K.pdf")
end


# ------------------------------------------------------------------------------------------------- #
# load data
data = Matrix(CSV.read("TrainingDataForAssingment5.csv", DataFrame))[:, 2:3]
x, y = data[:, 1], data[:, 2]
scatter(x, y, label="data points", legend=:topleft)

K = 4 # number of knots
k = [2, 4, 6, 8] # knots
X, g = cubic_regression_spline(x, y, K, k)
plot_spline(x, y, K, g, 2)
```

B

A) で示した手順に基づき，プログラムを作成・実行することにより求めた $f(x)$ は次のようになった．cubic regression spline とその関数の描画は今回のレポートで実行する回数が非常に多いため関数化し，cubic_regression_spline, plot_spline という 2 つの関数を作成している．cubic_regression_spline により，行列 $\mathbf{X}$ と $\hat{\boldsymbol{\beta}}$ を求めスプライン関数での回帰を行った．

$$
f(x) = \begin{cases}
7.008 + 0.946x - 3.912x^2 + 1.183x^3 \quad (0 \leq x \leq 2) \\
23.783 - 24.217x + 8.669x^2 - 0.914x^3 \quad (2 \leq x \leq 4) \\
-82.527 + 55.516x - 11.264x^2 + 0.747x^3 \quad (4 \leq x \leq 6) \\
249.853 - 110.674x + 16.434x^2 - 0.792x^3 \quad (6 \leq x \leq 8) \\
-1344.032 + 487.032x - 58.280x^2 + 2.321x^3 \quad (8 \leq x \leq 10) \\
0 \quad (otherwise)
\end{cases} \tag{4}
$$

```
# C) Leave on out cross validation using hat matrix (magic foemula)
function calc_CVLOO_magic(X, x, g, y, st)
    diag_H = diag(X * ((X' * X) \ X'))
    f = [g[Int(ceil(i / st))](i) for i in x]
    CV_LOO = sum([((y[i] - f[i]) / (1 - diag_H[i]))^2 for i in 1:length(y)]) / length(y)
    println(CV_LOO)
end
calc_CVLOO_magic(X, x, g, y, 2)

# ------------------------------------------------------------------------------------------------ #
# D) Leave on out cross validation without magic formula
CV_LOO_2 = 0.0
for iter in 1:length(y)
    # remove the first element of x and y
    x_, y_ = popfirst!(x), popfirst!(y)
    # fit by cubic spline function
    local X, g = cubic_regression_spline(x, y, K, k)
    # calculate CVLOO for each element
    global CV_LOO_2 += (y_ - g[Int(ceil(x_ / 2))](x_))^2
    # append x_ and y_ as the last element of x and y respectively
    push!(x, x_)
    push!(y, y_)
end
println(CV_LOO_2 / length(y))

# ------------------------------------------------------------------------------------------------ #
# use CVLOO to determine the number of knots among 1,2,...,15
for K_ in 1:15
    # calculate uniform knots
    k_ = [10i / (K_ + 1) for i in 1:K_]
    # calculate distance between 2 konts
    st = 10 / (K_ + 1)
    local X, g = cubic_regression_spline(x, y, K_, k_)
    plot_spline(x, y, K_, g, st)
    calc_CVLOO_magic(X, x, g, y, st)
end
```

C

次式 magic formula を用いて $\mathrm{CV}_{\mathrm{LOO}}$ を求めた.

$$\mathrm{CV}_{\mathrm{LOO}} = \frac{1}{n}\sum_{i=1}^{n}\left\{\frac{y_i - \hat{f}(x_i)}{(1-h_{ii})}\right\}^2 \qquad \text{(magic formula)}$$

$$\text{where} h_{ii} \text{ is the } i \text{ th diagonal element of } \mathbf{H} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T \tag{5}$$

プログラムではまず，対角成分を計算しそれを配列化している．そして，与えられたデータ点に対して予測値を求めている．最後に magic formula を用いて計算している.

以上の手順で求めた結果，$\underline{\mathrm{CV}_{\mathrm{LOO}} = 3.7273}$となった.

D

1. ベクトル $\boldsymbol{x}$ とベクトル $\boldsymbol{y}$ の先頭要素を取り除き，x_-, y_- という変数にそれぞれ格納しておく.
2. cubic_regression_spline 関数を実行し，取り除いた要素に対して $\left(y_i - \hat{f}_{-i}\left(x_i\right)\right)^2$ を求める.
3. それをすべてのデータ点について繰り返す.
4. 誤差平方和を求め，要素数で割る.

以上の手順で求めた結果，$\underline{\mathrm{CV}_{\mathrm{LOO}} = 3.7273}$となった.

2

$K = 1, ..., 15$ としたときの $\mathrm{CV}_{\mathrm{LOO}}$ の値は次の表のように変化した．また，その変化の様子を次の図 2 に示した．

K	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\mathrm{CV}_{\mathrm{LOO}}$	4.375	4.366	4.411	3.727	3.959	3.789	3.844	4.152	4.392	4.649	4.688	5.269	5.767	6.121	5.927

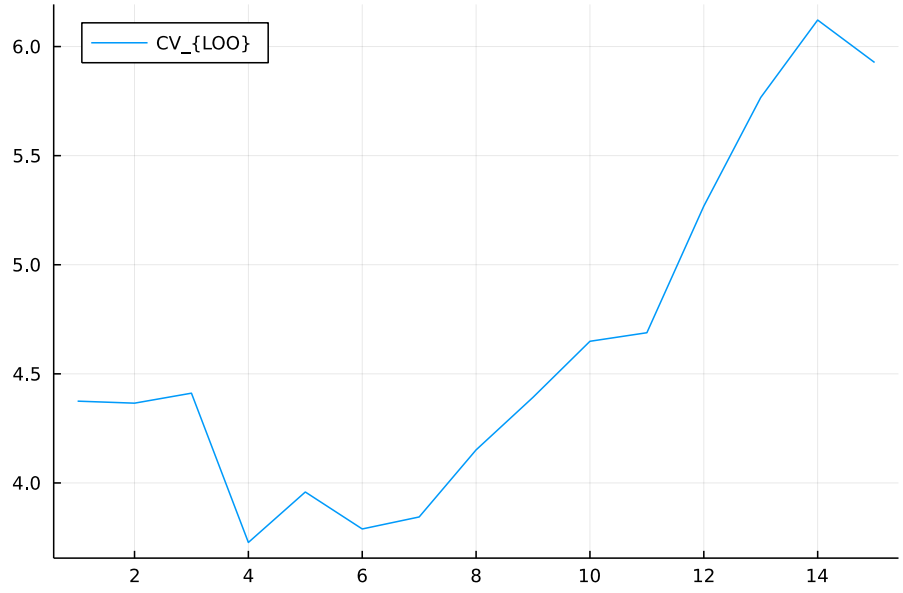

図 2: $\mathrm{CV}_{\mathrm{LOO}}$

図 2 より，knot 数が 4 のときに最も値が小さくなっていることから 4 が最適であろうと考えた．

3

多項式の場合は学習誤差は次の式で表されるため，データから計算可能なので単調に減少する．

$$\frac{1}{n}\sum_{i=1}^{n}\left(\hat{f}\left(\boldsymbol{x}_i\right) - y_i\right)^2, (\hat{f}(x) \text{ は学習データから推定された回帰係数}) \tag{6}$$

スプラインの場合は，次の式を小さくすることを目標にしているため，多項式回帰とは違い 2 乗誤差の値が単調にならない．

$$\text{モデル:} \quad y_i = f\left(x_i\right) + \varepsilon_i, \quad \varepsilon_i \text{ はノイズ}$$

$$\sum_{i=1}^{n}\left(y_i - f\left(x_i\right)\right)^2 \tag{7}$$