Biostatistics 615 - Statistical Computing

Topic 4 Interpolation and Extrapolation

Hyun Min Kang

Topic Overview

1. Linear interpolation and extrapolation
2. Polynomial interpolation and piecewise interpolation
3. Natural cubic spline

Topic Overview

1. Linear interpolation and extrapolation
2. Polynomial interpolation and piecewise interpolation
3. Natural cubic spline

Interpolation and Extrapolation

Given a set of data points: $\{(x_i, y_i)\}_{i=1}^n$ where $x_i, y_i \in \mathbb{R}$.

- **Interpolation**: find the values of "missing data" in between known values.
- **Extrapolation**: extend the model outside of known data to "predict" new values for which no measure could have been taken.
- **Curve fitting**: find a function $f(x)$ defined on $\mathbb{R}$ such that

$$y_i = \hat{f}(x_i).$$

- It depends on the model assumptions on the association between the outcome variable and predictors
  - Linear function
  - Polynomial function
  - Piecewise linear
  - Smoother curve

Linear Interpolation

- Start with only two observed points: (x_1, y_1) and (x_2, y_2).
- Find a linear function

$$y = mx + b$$

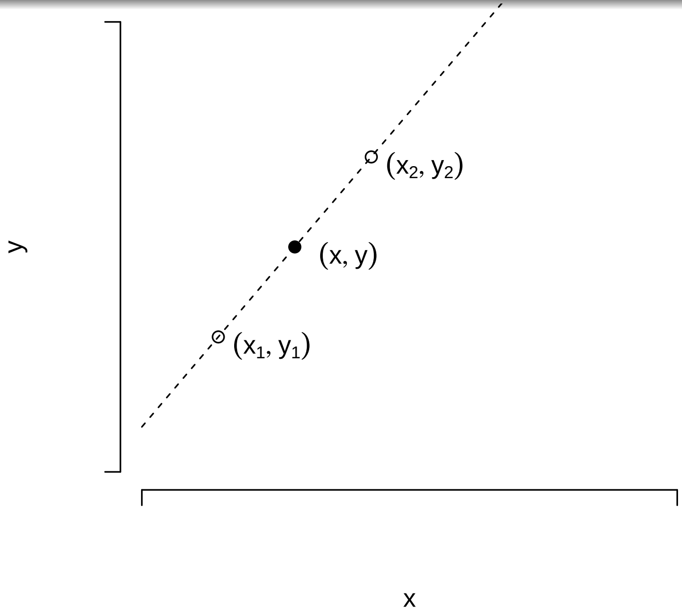
such that it passes the two points, that is,

$$\begin{cases} y_1 & = mx_1 + b \\ y_2 & = mx_2 + b \end{cases}$$

- The solution is

$$\begin{cases} m & = \frac{y_2 - y_1}{x_2 - x_1} \\ b & = y_1 - mx_1 = y_2 - mx_2 \end{cases}$$

Linear Interpolation based on two points



Hands-on Session

Visit https://bit.ly/615top04r

- Implementing linear interpolation

Potential problems with linear interpolation

- True relationship between x and y are not necessarily linear
- More observation points are usually available
- When $x_1 = x_2$ or $|x_1 - x_2|$ is very small. This approach is problematic.

Topic Overview

1. Linear interpolation and extrapolation
2. Polynomial interpolation and piecewise interpolation
3. Natural cubic spline

Higher-Order Polynomial Interpolation

- Given two data points, a line, a polynomial of degree one, will always pass exactly through the two points, provided the two values of x are different.
- For any three points, it is usually that no such line is available, unless those points lie precisely on the same line, but a quadratic function, a polynomial of degree two, will always fit the three data points exactly.
- In general, a polynomial of degree $n-1$ is necessary and sufficient to precisely fit the n data points.

Higher-Order Polynomial Interpolation

- Given a set of ordered pairs, $\{(x_i, y_i)\}_{i=1}^{n}$, the interpolating function $p_{n-1}(x)$ must meet the requirement,

$$p_{n-1}(x_i) = y_i,$$

for all $i = 1, \ldots, n$.

- The interpolating function is a polynomial of order $n-1$:

$$p_{n-1}(x) = \sum_{j=0}^{n-1} \beta_j x^j,$$

where β_j's are the polynomial's coefficients.

$$\sum_{j=0}^{n-1} \beta_j x_i^j = y_i,$$

for each of the i interpolating points.

Higher Order Polynomial Interpolation

In matrix form

$$\begin{pmatrix} 1 & x_1 & \dots & x_1^{n-1} \\ 1 & x_2 & \dots & x_2^{n-1} \\ \dots & \dots & \dots & \dots \\ 1 & x_n & \dots & x_n^{n-1} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \dots \\ \beta_{n-1} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix},$$

In the matrix form, we have

$$\boldsymbol{V}(\boldsymbol{x})\boldsymbol{\beta} = \boldsymbol{y}$$

where

- The matrix $\boldsymbol{V}(\boldsymbol{x})$ is called the Vandermonde matrix and it contains successive columns of $\boldsymbol{x}$ to the ith power, where $i = 0, \dots, n-1$.
- We can solve the above linear system to obtain $\boldsymbol{\beta}$ for the polynomial interpolation.

Hands-on Session

Visit https://bit.ly/615top04r

- Higher-order Polynomial Interpolation

Approximation Errors

- Similar to the linear interpolation, the higher-order polynomial interpolation may have large numerical errors when x_i and $x_{i'}$ are too close for any $i \neq i' \in \{1, \ldots, n\}$. Or the Vandermonde matrix $\boldsymbol{V}(\boldsymbol{x})$ is close to a singular matrix.
- In addition to numerical errors, the higher-order polynomial interpolation also has the approximation errors. Suppose the true relationship between y and x is $y = f(x)$, where $f(x)$ is a smooth function whose the nth derivative exists. Then the approximation error is

$$p_{n-1}(x) - f(x) = \frac{\prod_{i=1}^{n}(x - x_i)}{n!} f^{(n)}(x)$$

Hands-on Session

Visit `https://bit.ly/615top04r`

- Approximation errors in polynomial interpolation

Limitations of higher-order polynomial interpolation

- The computational complexity is $O(n^3)$. It is computationally challenging when n is large.
- Diverge from true values at the endpoints in many cases, especially when n is large.
- While the higher-degreed polynomial pass through all observed points, it may fluctuate wildly between two points, a pattern known as Runge's phenomenon.

Hands-on Session

Visit https://bit.ly/615top04r

- Runge's phenomenon

Piecewise Interpolation

- The true function may be complex and better approximated using two or more interpolations.
- Piecewise interpolation offers a solution: to use lower degreed polynomials for each part of a curve because we are able to efficiently fit those polynomials to approximately analyze the underlying data.

Piecewise Linear Interpolation

- A linear approach to interpolating between points: draw lines between points.
- The process for piecewise linear interpolation uses the same process as our linear interpolation model, except it repeats it for each consecutive pair of data points along the x-axis.
- Computational complexity is $O(n)$.

Hands-on Session

Visit https://bit.ly/615top04r

- Piecewise linear interpolation
- Revisiting Runge's phenomenon

Topic Overview

1. Linear interpolation and extrapolation
2. Polynomial interpolation and piecewise interpolation
3. Natural cubic spline

Cubic Spline Interpolation

Revisiting Piecewise Linear Interpolation

- If using a first-degree polynomial, a line over multiple intervals is an improvement over a single interpolating line.
- However, piecewise linear interpolation does not produce differentiable (or smooth) function at the joins.
- What if we use a higher-degree polynomial?

Cubic Spline Interpolation : A potential solution

- It constructs a differentiable, integrable and smooth curve, despite the joins.
- As each individual section is represented by a cubic curve, i.e. **polynomial of degree three**.

Cubic Spline Setup

- Denote by $\{(x_i, y_i)\}_{i=1}^{n}$ the data points to interpolate, where we assume that $x_i \leq x_{i+1}$ for $i = 1, \ldots, n-1$.
- For n data points, we construct $n-1$ interpolating cubic polynomials on $[x_1, x_2], \ldots, [x_{n-1}, x_n]$.
- Let $S_i(x)$ be a cubic polynomial function representing the curve over the domain $[x_i, x_{i+1}]$.
- We assume that

$$
\begin{aligned}
S_i(x) &= \sum_{j=0}^{3} a_{j,i}(x - x_i)^j \\
&= a_{0,i} + a_{1,i}(x - x_i) + a_{2,i}(x - x_i)^2 + a_{3,i}(x - x_i)^3
\end{aligned}
$$

- Then we have

$$
\begin{aligned}
S_i'(x) &= a_{1,i} + 2a_{2,i}(x - x_i) + 3a_{3,i}(x - x_i)^2 \\
S_i''(x) &= 2a_{2,i} + 6a_{3,i}(x - x_i)
\end{aligned}
$$

Equations to construct cubic splines

- The set of $\{S_i(x)\}$ has a total of $4(n-1)$ parameters.
- Continuity $(2n-2)$: for $i = 1, \ldots, n-1$,

$$S_i(x_i) = y_i, \quad (1)$$
$$S_i(x_{i+1}) = y_{i+1}. \quad (2)$$

- Smoothness $(2n-4)$: for $i = 1, \ldots, n-2$,

$$S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}), \quad (3)$$
$$S_i''(x_{i+1}) = S_{i+1}''(x_{i+1}). \quad (4)$$

- So far we establish $4n-6$ equations, we need to two more constraints to solve all the parameters.
- Boundary conditions: $S_1''(x_1) = 0$ and $S_{n-1}''(x_n) = 0$. We refer to this type of spline as the **Natural Cubic Spline**.

Summary - equations to construct cubic splines

$$
\begin{aligned}
S_i(x) &= a_{0,i} + a_{1,i}(x - x_i) + a_{2,i}(x - x_i)^2 + a_{3,i}(x - x_i)^3 \\
S_i'(x) &= a_{1,i} + 2a_{2,i}(x - x_i) + 3a_{3,i}(x - x_i)^2 \\
S_i''(x) &= 2a_{2,i} + 6a_{3,i}(x - x_i)
\end{aligned}
$$

Equations for natural cubic spline

$$
\begin{aligned}
S_i(x_i) &= y_i & i &= 1, \ldots, n-1 \\
S_i(x_{i+1}) &= y_{i+1} & i &= 1, \ldots, n-1 \\
S_i'(x_{i+1}) &= S_{i+1}'(x_{i+1}) & i &= 1, \ldots, n-2 \\
S_i''(x_{i+1}) &= S_{i+1}''(x_{i+1}) & i &= 1, \ldots, n-2 \\
S_1''(x_1) &= 0 \\
S_{n-1}''(x_n) &= 0
\end{aligned}
$$

Solve the natural cubic spline - Definitions

- For $i = 2, \ldots, n$, let

$$h_i = x_i - x_{i-1}, \quad \delta_i = \frac{y_i - y_{i-1}}{h_i}.$$

- For $i = 2, \ldots, n-1$, let

$$g_i = h_{i+1} + h_i, \quad v_i = \delta_{i+1} - \delta_i.$$

Solve the natural cubic spline - Continuity

- From $S_i(x_i) = y_i, i = 1, \ldots, n-1$,

$$a_{0,i} = y_i$$

- From $S_i(x_{i+1}) = y_{i+1}, i = 1, \ldots, n-1$,

$$a_{0,i} + a_{1,i}h_{i+1} + a_{2,i}h_{i+1}^2 + a_{3,i}h_{i+1}^3 = y_{i+1}$$

- Then for $i = 1, \ldots, n-1$,

$$a_{1,i} + a_{2,i}h_{i+1} + a_{3,i}h_{i+1}^2 = \frac{y_{i+1} - y_i}{h_{i+1}} = \delta_{i+1}. \tag{5}$$

Solve the natural cubic spline - Smoothness

- From $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}), i = 1, \ldots, n-2$,

$$a_{1,i+1} = a_{1,i} + 2a_{2,i}h_{i+1} + 3a_{3,i}h_{i+1}^2. \quad (6)$$

- From $S_i''(x_{i+1}) = S_{i+1}''(x_{i+1}), i = 1, \ldots, n-2$,

$$a_{2,i+1} = a_{2,i} + 3a_{3,i}h_{i+1}. \quad (7)$$

- From (7), for $i = 1, \ldots, n-2$,

$$a_{3,i} = \frac{a_{2,i+1} - a_{2,i}}{3h_{i+1}}. \quad (8)$$

- From (6), for $i = 1, \ldots, n-2$,

$$a_{1,i+1} - a_{1,i} = (a_{2,i} + a_{2,i+1})h_{i+1}$$

Solve the natural cubic spline - Putting Together

- From (5), for $i = 1, \ldots, n-2$,

$$3a_{1,i} + (2a_{2,i} + a_{2,i+1})h_{i+1} = 3\delta_{i+1} \tag{9}$$

- Then for $i = 2, \ldots, n-1$

$$3a_{1,i-1} + (2a_{2,i-1} + a_{2,i})h_i = 3\delta_i \tag{10}$$

- For $i = 2, \ldots, n-2$, we get this key equation from (9)-(10):

$$h_i a_{2,i-1} + 2g_i a_{2,i} + h_{i+1} a_{2,i+1} = 3v_i \tag{11}$$

Remember:
$a_{1,i+1} - a_{1,i} = (a_{2,i} + a_{2,i+1})h_{i+1}$, $g_i = h_{i+1} + h_i$, $v_i = \delta_{i+1} - \delta_i$.

Boundary conditions for natural cubic splines

- By $S_1''(x_1) = 0$, we have $a_{2,1} = 0$, so from (11),

$$2g_2 a_{2,2} + h_3 a_{2,3} = 3v_2$$

- By $S_{n-1}''(x_n) = 0$, we have $a_{2,n-1} + 3a_{3,n-1}h_n = 0$
- From (5),

$$3a_{1,n-1} + 2a_{2,n-1}h_n = 3\delta_n$$

- Also,

$$3a_{1,n-2} + (2a_{2,n-2} + a_{2,n-1})h_{n-1} = 3\delta_{n-1}$$

$$a_{1,n-1} - a_{1,n-2} = (a_{2,n-2} + a_{2,n-1})h_{n-1}$$

- Therefore, we also have

$$h_{n-1}a_{2,n-2} + 2g_{n-1}a_{2,n-1} = 3v_{n-1}$$

Thus

$$\begin{pmatrix} 2g_2 & h_3 & 0 & \ldots & \ldots & \ldots \\ h_3 & 2g_3 & h_4 & 0 & \ldots & \ldots \\ 0 & h_4 & 2g_4 & h_5 & 0 & \ldots \\ \ldots & 0 & \ldots & \ldots & \ldots & 0 \\ \ldots & \ldots & 0 & h_{n-2} & 2g_{n-2} & h_{n-1} \\ \ldots & \ldots & \ldots & 0 & h_{n-1} & 2g_{n-1} \end{pmatrix} \begin{pmatrix} a_{2,2} \\ a_{2,3} \\ \ldots \\ \ldots \\ a_{2,n-2} \\ a_{2,n-1} \end{pmatrix} = 3 \begin{pmatrix} v_2 \\ v_3 \\ \ldots \\ \ldots \\ v_{n-2} \\ v_{n-1} \end{pmatrix}$$

This implies that we can apply tridiagonal matrix algorithm to obtain $a_{2,i}$ for $i = 2, \ldots, n-1$.

Defining the remaining variables

Recall that $a_{2,1} = 0$, and also $a_{2,n} = 0$ holds.
Then for $i = 1, \ldots, n-1$, from (8) and (9), we have

$$a_{3,i} = \frac{a_{2,i+1} - a_{2,i}}{3h_{i+1}},$$
$$a_{1,i} = \delta_{i+1} - \frac{2a_{2,i} + a_{2,i+1}}{3} h_{i+1},$$

Recall that

$$a_{0,i} = y_i$$

This completes that the natural cubic spline fitting.

Tridiagonal Matrix Algorithm

For linear system $A\mathbf{x} = \mathbf{b}$, A is a general tridiagonal matrix

$$A = \begin{pmatrix} d_1 & u_1 & 0 & ... & ... & \\ l_1 & d_2 & u_2 & 0 & \dots & ... \\ 0 & l_2 & d_3 & u_3 & 0 & \dots \\ \dots & 0 & \dots & \dots & \dots & u_{n-1} \\ \dots & \dots & \dots & 0 & l_{n-1} & d_n \end{pmatrix}$$

which is determined by three vectors $\mathbf{d} = (d_1, \dots, d_n)$, $\mathbf{u} = (u_1, \dots, u_{n-1})$ and $\mathbf{l} = (l_1, \dots, l_{n-1})$.

What would happen if we solve this linear system with typical linear algebraic solution?

Can you think of more efficient way to solve this equation?

Forward Sweep: Step 1

$$\begin{pmatrix} d_1 & u_1 & 0 & ... & ... & ... \\ l_1 & d_2 & u_2 & 0 & \ldots & ... \\ 0 & l_2 & d_3 & u_3 & 0 & \ldots \\ \ldots & 0 & \ldots & \ldots & \ldots & u_{n-1} \\ \ldots & \ldots & \ldots & 0 & l_{n-1} & d_n \end{pmatrix} \boldsymbol{x} = \begin{pmatrix} b_1 \\ b_2 \\ \ldots \\ b_{n-1} \\ b_n \end{pmatrix}$$

$$\rightarrow \begin{pmatrix} 1 & u_1/d_1 & 0 & ... & ... & ... \\ l_1 & d_2 & u_2 & 0 & \ldots & ... \\ 0 & l_2 & d_3 & u_3 & 0 & \ldots \\ \ldots & 0 & \ldots & \ldots & \ldots & u_{n-1} \\ \ldots & \ldots & \ldots & 0 & l_{n-1} & d_n \end{pmatrix} \boldsymbol{x} = \begin{pmatrix} b_1/d_1 \\ b_2 \\ \ldots \\ b_{n-1} \\ b_n \end{pmatrix}$$

Set $u_1^* = u_1/d_1$ and $b_1^* = b_1/d_1$.

Forward Sweep: Step 2

$$\rightarrow \begin{pmatrix} 1 & u_1^* & 0 & ... & ... & ... \\ 0 & d_2 - u_1^* * l_1 & u_2 & 0 & \dots & ... \\ 0 & l_2 & d_3 & u_3 & 0 & \dots \\ \dots & 0 & \dots & \dots & \dots & u_{n-1} \\ \dots & \dots & \dots & 0 & l_{n-1} & d_n \end{pmatrix} \mathbf{x} = \begin{pmatrix} b_1^* \\ b_2 - b_1^* * l_1 \\ \dots \\ b_{n-1} \\ b_n \end{pmatrix}$$

$$\rightarrow \begin{pmatrix} 1 & u_1^* & 0 & ... & ... & ... \\ 0 & 1 & \frac{u_2}{d_2 - u_1^* * l_1} & 0 & \dots & ... \\ 0 & l_2 & d_3 & u_3 & 0 & \dots \\ \dots & 0 & \dots & \dots & \dots & u_{n-1} \\ \dots & \dots & \dots & 0 & l_{n-1} & d_n \end{pmatrix} \mathbf{x} = \begin{pmatrix} b_1^* \\ \frac{b_2 - b_1^* * l_1}{d_2 - u_1^* * l_1} \\ b_3 \\ \dots \\ b_n \end{pmatrix}$$

Set $u_2^* = \frac{u_2}{d_2 - u_1^* * l_1}$ and $b_2^* = \frac{b_2 - b_1^* * l_1}{d_2 - u_1^* * l_1}$.

Forward Sweep: Step i

For $i = 2, \ldots, n-1$, set

$$u_i^* = \frac{u_i}{d_i - u_{i-1}^* * l_{i-1}},$$
$$b_i^* = \frac{b_i - b_{i-1}^* * l_{i-1}}{d_i - u_{i-1}^* * l_{i-1}}.$$

And

$$b_n^* = \frac{b_n - b_{n-1}^* * l_{n-1}}{d_n - u_{n-1}^* * l_{n-1}}$$

Backward Sweep:

$$\begin{pmatrix} 1 & u_1^* & 0 & \dots & \dots & \dots \\ 0 & 1 & u_2^* & 0 & \dots & \dots \\ 0 & 0 & 1 & u_3^* & 0 & \dots \\ \dots & 0 & \dots & \dots & \dots & u_{n-1}^* \\ \dots & \dots & \dots & 0 & 0 & 1 \end{pmatrix} \boldsymbol{x} = \begin{pmatrix} b_1^* \\ b_2^* \\ \dots \\ b_{n-1}^* \\ b_n^* \end{pmatrix}$$

Then

$$x_n = b_n^*$$

For $i = n-1, n-2, \dots, 1$,

$$x_i = b_i^* - u_i^* * x_{i+1}$$

Hands-on Session

Visit https://bit.ly/615top04r

- Implementing natural cubic spline
- Testing natural cubic spline
- Revisiting Runge's phenomenon

Discussion

- What are the advantages and disadvantages of each interpolation method?
  - Linear interpolation
  - Higher-order polynomial interpolation
  - Piecewise linear interpolation
  - Natural cubic spline
- Can you explain the time complexity of computing natural cubic spline?
- Why is natural cubic spline useful? If we implement similar splines with quadratic (2nd-order) or quartic (4th-order) functions, what are the advantages and disadvantages?