CSE276C - Interpolation and Approximation

Henrik I. Christensen

Computer Science and Engineering
University of California, San Diego
http://cri.ucsd.edu

October 2024

Outline

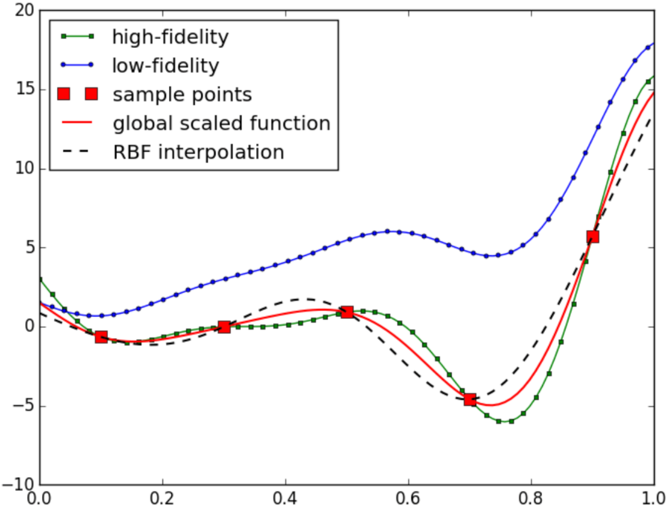
Material

- Numerical Recipes: Chapter 3
- Math for ML: Chapter 9

Objective

- How can we find an approximation / interpolation based on a set of data points?
- Model Based
  - We have domain knowledge that can be used:
  - Battery recharge
  - Dynamic Model of Drive System
  - Material properties for grasping
- Data Driven
  - All we have is the data (and possible constraints)
  - Driving in traffic, Painting, ...

Example



high-fidelity
low-fidelity
sample points
global scaled function
RBF interpolation
20
15
10
5
0
−5
−10
0.0
0.2
0.4
0.6
0.8
1.0

Weierstrass Approximation Theorem

Weierstrass Approx. Theorem

If f is a continuous function on the finite closed interval $[a, b]$ then for every $\epsilon > 0$ there is a polynomial $p(x)$ (whose degree and coefficients depend on ϵ) such that

$$\max_{x \in [a,b]} |f(x) - p(x)| < \epsilon$$

- This is wonderful right?
- He does not prescribe a strategy to derive $p(x)$!

Outline

Linear interpolation

- Lets start with a single variable case
  - We have a set $D = (x_i, f(x_i))\ i \in \{0, .., n\}$
- Connecting adjacent points by line segment

$$p(x) = f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i)$$

$$x \in [x_i, x_{i+1}]$$

- consider it a baseline for other approaches

Lagrange interpolation

- Could we fit an n'th order polynomial through n+1 data points: $(x_i, y_i)\ i \in \{0, .., n\}$
- Could be done recursively or in a batch form.
- Batch solution is estimating n+1 coefficient using n+1 simultaneous equations
- Interpolation polynomial

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

- For each data point we have the equation

$$y_i = a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_n x_i^n$$

- in matrix form

Lagrange interpolation (cont)

- In matrix form we have

$$\begin{pmatrix} 1 & x_0 & x_0^2 & x_0^3 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & x_1^3 & \dots & x_1^n \\ & & & \vdots & & \\ 1 & x_n & x_n^2 & x_n^3 & \dots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

- or

$$V\ x = y$$

  where **V** is referred to as a Vandermonde matrix.
- Unfortunately the system is frequently poorly conditioned

Lagrange polynominal interpolation

- Consider the n^{th} degree polynomial factored
- The classic Lagrange formula

$$\begin{aligned} p(x) \quad = \quad & \frac{(x-x_1)(x-x_2)\ldots(x-x_n)}{(x_0-x_1)(x_0-x_2)\ldots(x_0-x_n)}y_0+ \\ & \frac{(x-x_0)(x-x_2)\ldots(x-x_n)}{(x_1-x_0)(x_1-x_2)\ldots(x_1-x_n)}y_1+ \\ & \ldots \\ & \frac{(x-x_0)(x-x_2)\ldots(x-x_{n-1})}{(x_n-x_0)(x_n-x_1)\ldots(x_n-x_{n-1})}y_n+ \end{aligned}$$

- or

$$y_k L_k(x_k) = y_k L_k(x)$$

$$L_k(x) = \prod_{\substack{i=0 \\ i \neq j}}^{n} \frac{x - x_i}{x_k - x_i}$$

note

$$L_k(x_i) = \delta_{ik} = \begin{cases} 1 & k = i \\ 0 & i \neq k \end{cases}$$

Lagrange polynominal interpolation (cont)

- The resulting polynomial is

$$p(x) = \sum_{k=0}^{n} y_k L_k(x)$$

- A polynomial that passed through all the data points

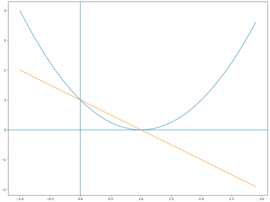
LPI - Example

- Lets try to show this for
$$f(x) = (x-1)^2$$
- Assume we have two data points (0,1) and (1,0).
- This results in $a_0 = 1$ and $a_1 = 0$.
- As $a_1 = 0$ we only have to consider

$$L_0(x) = \frac{x - x_1}{x_1 - x_0} = \frac{x-1}{0-1} = -x + 1$$

or

$$p(x) = -x + 1$$



LPI - Example (cont)

- Lets add an additional data point (-1, 4)

$$\begin{array}{ll} x_0 = 0 & a_0 = 1 \\ x_1 = 1 & a_1 = 0 \\ x_2 = -1 & a_2 = 4 \end{array}$$

So

$$\begin{array}{rcl} L_0(x) & = & \frac{x-x_1}{x_0-x_1}\frac{x-x_2}{x_0-x_2} = -(x-1)(x+1) \\ L_1(x) & = & \frac{x-x_0}{x_1-x_0}\frac{x-x_2}{x_1-x_2} = \text{Don't care} \\ L_2(x) & = & \frac{x-x_0}{x_2-x_0}\frac{x-x_1}{x_2-x_1} = \frac{1}{2}x(x-1) \end{array}$$

LPI - Example (cont)

- Putting it all together

$$
\begin{aligned}
p(x) &= a_0 L_0(x) + a_1 L_1(x) + a_2 L_2(x) \\
&= -(x-1)(x+1) + 2x(x-1) \\
&= (x-1)(-x-1+2x) \\
&= (x-1)(x-1) = (x-1)^2
\end{aligned}
$$

- The approximation is exact
- For large data-sets Lagrange can be a challenge
- Meandering between data-points can become significant

Outline

Cubic spline interpolation

- Smoothing w. constraints
- Limiting higher order gradients (say acceleration, curvature, ...)

$$\begin{aligned} f'''' &= 0 \\ f''' &= c_1 \\ f'' &= c_1 x + c_2 \\ f' &= \frac{c_1}{2} x^2 + c_2 x + c_3 \\ f &= \frac{c_1}{6} x^3 + \frac{c_2}{2} x^2 + c_3 x + c_4 \end{aligned}$$

Setting it up

- Assume you have tabulated values $y_i = y(x_i)$ for $i = 0 \ldots n-1$
- With linear interpolation we can do

$$y = Ay_j + By_{j+1}$$

for a point between x_j and x_{j+1} where

$$A = \frac{x_{j+1} - x}{x_{j+1} - x_j} \quad B = 1 - A = \frac{x - x_j}{x_{j+1} - x_j}$$

so think of them as special cases of Lagrange

- if we further assume we have access to values of y'' we can do a cubic expansion

Cubic interpolation

- We can expand the interpolation

$$y = Ay_j + By_{j+1} + Cy_j^{''} + Dy_{j+1}^{''}$$

where A and B are as defined earlier.

$$C = \frac{1}{6}(A^3 - A)(x_{j+1} - x_j)^2 \qquad D = \frac{1}{6}(B^3 - B)(x_{j+1} - x_j)^2$$

- If you differentiate (see NR sec 3.3) you get

$$\frac{d^2y}{dx^2} = Ay_j^{''} + By_{j+1}^{''}$$

which translate into the tabulated values at x_j and x_{j+1}.

- The advantage of cubic is that only neighboring points are used in estimation. A tridiagonal matrix can be used for the computations.

Outline

What about multi-variate interpolation?

- Does this generalize to multiple dimensions?
- We frequently have multi-dimensional data in robotics
  - Image data, Lidar, radar, ...
- What if we had an m-dimensional Cartesian mesh of data points?

$$f(\vec{x}) = f(x_{1i}, x_{2j}, x_{3k}, \ldots, x_{mq})$$

- For linear interpolation the generalization is straight forward

Bilinear interpolation

- Consider

$$y_{ij} = y(x_{1i}, x_{2j})$$

- with point intervals $[x_{1i}, x_{1(i+1)}]$ and $[x_{2j}, x_{2(j+1)}]$
- values for ij

$$\begin{aligned} y_0 &= y_{ij} \\ y_1 &= y_{(i+1)j} \\ y_2 &= y_{(i+1)(j+1)} \\ y_3 &= y_{i(j+1)} \end{aligned}$$

Bilinear interpolation (cont)

- The bilinear interpolation is the simplest
- use

$$
\begin{aligned}
t &= \frac{x_1 - x_{1i}}{x_{1(i+1)} - x_{1i}} \\
u &= \frac{x_2 - x_{2j}}{x_{2(j+1)} - x_{2j}}
\end{aligned}
$$

- then the interpolation is:

$$y(x_1, x_2) = (1-t)(1-u)y_0 + t(1-u)y_1 + tuy_2 + (1-t)uy_3$$

- For a fair sized grid this generates “good” solutions.

Outline

Kringing interpolation

- What if we consider the data generation by a stochastic process?
- Could we generate a maximum likelihood (ML) estimate?
- The data is a vector of samples from the process and we can compute the probability density estimate and parameters such as the mean
- Sometimes termed Gaussian Process Regression
- More generally we are trying to estimate

  $$f(x) = \sum_{i=0}^{N} w_i \phi_i(x) = \vec{w}\Phi(\vec{x})$$

  where w are weights and ϕ is a basis function.
- We can define a loss function

  $$L(f, y)$$

- The expected loss is then

  $$E[L] = \int \int L(f, y(x)) p(x, w) dx dw$$

- Our goal is now to minimize the $E[L]$, i.e. minimum loss or best fit

  $$f = E(y|x)$$

Basis functions

- We have multiple choices for basis functions
- Sometimes domain knowledge can provide suggestions
- Polynomial basis functions

$$\phi_i(x) = x_i$$

- Gaussian basis functions

$$\phi_i(x) = e^{-\frac{(x-x_i)^2}{2s}}$$

s controls scale / coverage

- Sigmoid basis functions

$$\phi_i(x) = \sigma\left(\frac{x-x_i}{s}\right)$$

where $\sigma(a) = \frac{1}{1+e^{-a}}$

Kringing interpolation - Gaussian Mixture

- For the Gaussian mixture we can use

$$p(f_i|x_i, w_i, \beta) = N(f_i|y(x_i), w_i, \beta)$$

- so that

$$p(f|X, w, \beta) = \prod_{i=0}^{n} N(f_i|w^T\phi(x_i), \beta^{-1})$$

or

$$lnp() = \frac{n}{2}ln(\beta) - \frac{n}{2}ln(2\pi) - \beta E_D(w)$$

where

$$E_D(w) = \frac{1}{2}\sum_{i=0}^{n}(y_i - w_i^T\phi(x_i))^2$$

The sum of squared errors

LSQ solution

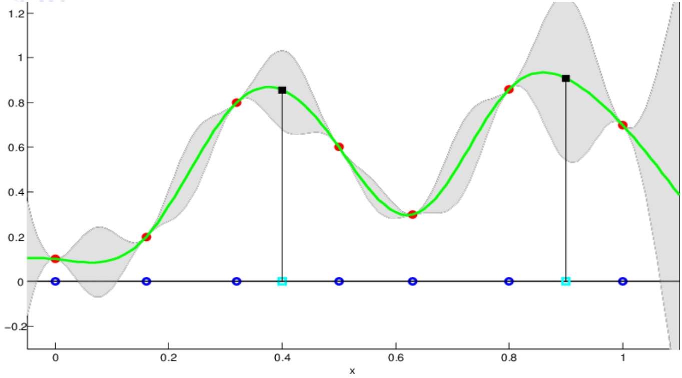
- We can compute

$$w_{ML} = (\Phi^T \Phi)^{-1} \Phi^T \vec{y}$$

where

$$\Phi = \begin{pmatrix} \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_n(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \dots & \phi_n(x_2) \\ & \vdots & & \\ \phi_1(x_n) & \phi_2(x_n) & \dots & \phi_n(x_n) \end{pmatrix}$$

Kringing example



Regularized Kringing

- We can use a regularized LSQ if we want to control the variation in w.
- Consider a revised error function

$$E' = E_D(w) + \lambda E_w(w)$$

say

$$E' = \frac{1}{2}\sum_i (y_i - w^T\phi(x_i))^2 + \frac{\lambda}{2} w^T w$$

which is minimized by

$$w = (\lambda + \Phi^T\Phi)^{-1}\Phi^T\vec{y}$$

as an example of how you can tweak the optimization / approximation

Outline

Summary

- Model based and data driven interpolation / approximation
- Basic Methods (Linear)
- Spline based interpolation
- Uni- and Multi-Variate Approaches
- Stochastic Models
- Next time functional interpolation & approximation