Function Approximation and Interpolation

Computational Economics

Dietmar Maringer

WWZ, University of Basel

Spring 2012

Approximation

two basic problems

- **interpolation problem:**
  find a function $\hat{f}$ that approximates f
- **function equation problem:**
  given an operator T, find a function f such that $Tf = 0$
  (equivalently, in the functional fixed-point problem: $f = Tf$)

Interpolation Principles

- often, $\hat{f}$ is a linear combination of basis functions $\phi_j(x)$:

$$\hat{f}(x) = \sum_{j=1}^{J} c_j \phi_j(x)$$

- typically, $\hat{f}$ is found using interpolation points, e.g.,

$$\begin{aligned} x_i &= a + \tfrac{i-1}{n-1}(b-a) && \text{evenly spaced} \\ x_i &= \tfrac{a+b}{2} + \tfrac{b-a}{2}\cos\left(\tfrac{n-i+0.5}{n}\pi\right) && \text{“Chebychev nodes”} \end{aligned}$$

- for $(n-1)$th order polynomial, use n interpolation points
- numerical procedure: find basis coefficients c_j that fit some given *interpolation points* for x and $f(x)$:

$$\sum_{j=1}^{J} c_j \phi_j(x_i) - f(x_i) = \underbrace{e_i}_{\to \min!} \Longrightarrow \mathbf{c} = (\Phi'\Phi)^{-1}\Phi' f_x$$

Polynomial Interpolation

basic idea

- **Weierstrass theorem:** for any $\varepsilon > 0$, there exists a polynomial p s.t.

$$||f - p||_\infty \equiv \sup_{x \in [a,b]} \left| f(x) - p(x) \right| < \varepsilon$$

- nth order polynomial for (perfect) fit through $J = (n+1)$ points

monomial basis

- using simple power functions $x^0, x^1, x^2, \ldots$: $\phi_j(x) = x^{j-1}$
- Vandermonde matrix Φ with for nth order polynomial ($J = n+1$):

$$\Phi = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_J & x_J^2 & \cdots & x_J^n \end{bmatrix}, \quad c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_J \end{bmatrix}, \quad f_x = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_J) \end{bmatrix}$$

- coefficients: $c = (\Phi'\Phi)^{-1}\Phi' f_x$
- problem: with increasing n, Vandermonde matrix, $[\Phi_{ij}] = [x_i^{j-1}]$, is ill-conditioned

Polynomial Interpolation

Chebychev polynomial basis

- Chebychev polynomials: for $z = 2\frac{x-a}{b-a} - 1$ (hence: $z \in [-1, 1]$)

$$\phi_j(x) = T_{j-1}(z)$$

where

$$T_0(z) = 1, \quad T_1(z) = z, \quad T_2(z) = 2z^2 - 1, \quad \ldots \quad T_k(z) = 2zT_{k-1}(z) - T_{k-2}(z)$$

- has favourable properties

Runge's phenomenon

- when using high order polynomials, $\hat{f}$ can become oscillating
- hence: increasing the order of the polynomial does not necessarily improve the approximation, but can have undesired side-effects

Piecewise Approximation

basic idea

- function $f(x)$ with bounds $x \in [a, b]$
- split range $[a, b]$ into p pieces $a = x_1 < x_2 < \cdots < x_{p+1} = b$
- approximate piecewise & splice pieces together

"nearest neighbour"

- find interpolation point x_i that is nearest to x
- use that interpolation point's function value as an estimate:

$$\hat{f}(x) = f(x_i)$$

"piecewise linear"

- find interpolation points such that $x \in [x_i, x_{i+1}]$
- compute function values $f_i = f(x_i)$ and $f_{i+1} = f(x_{i+1})$
- estimate by making a linear combination:

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

(Piecewise Polynomial) Splines

basic idea

- piecewise polynomial interpolation scheme of order k:

$$\hat{f}(x) = w_0 + w_1(x - x_i) + w_2(x - x_i)^2 + \cdots + w_k(x - x_i)^k$$

- ideally:
  - $\hat{f}(x_i) = f(x_i)$
  - derivatives exist
- special case: $k = 1$ for piecewise linear approximation
- typical cases: $k = 2$ (quadratic splines), $k = 3$ (cubic splines)

in `Matlab`

- `interp1(xi,fxi,x,`*`method`*`)` where
  - `xi` and `fxi` are vectors with x_is and $f(x_i)$s
  - `x` is the vector of point for which to compute $\hat{f}(x)$
  - *`method`* is the interpolation method;
    includes `'nearest'`, `'linear'` (default), `'spline'` (for cubic spline)