Approximation Methods

Dr. Kenneth L. Judd

September 5, 2011

Approximation Methods

- General Objective: Given data about $f(x)$ construct simpler $g(x)$ approximating $f(x)$.
- Questions:
  - What data should be produced and used?
  - What family of “simpler” functions should be used?
  - What notion of approximation do we use?
- Comparisons with statistical regression
  - Both approximate an unknown function and use a finite amount of data
  - Statistical data is noisy but we assume data errors are small
  - Nature produces data for statistical analysis but we produce the data in function approximation

Interpolation Methods

- Interpolation: find $g(x)$ from an n-dimensional family of functions to exactly fit n data points
- Lagrange polynomial interpolation
  - Data: $(x_i, y_i), i = 1, .., n.$
  - Objective: Find a polynomial of degree $n-1$, $p_n(x)$, which agrees with the data, i.e.,

$$y_i = f(x_i),\ \ i = 1, .., n$$

  - Result: If the x_i are distinct, there is a unique interpolating polynomial

- Does $p_n(x)$ converge to $f(x)$ as we use more points?
  - No! Consider

$$f(x) = \frac{1}{1+x^2}$$
$$x_i = -5, -4, \ldots, 3, 4, 5$$

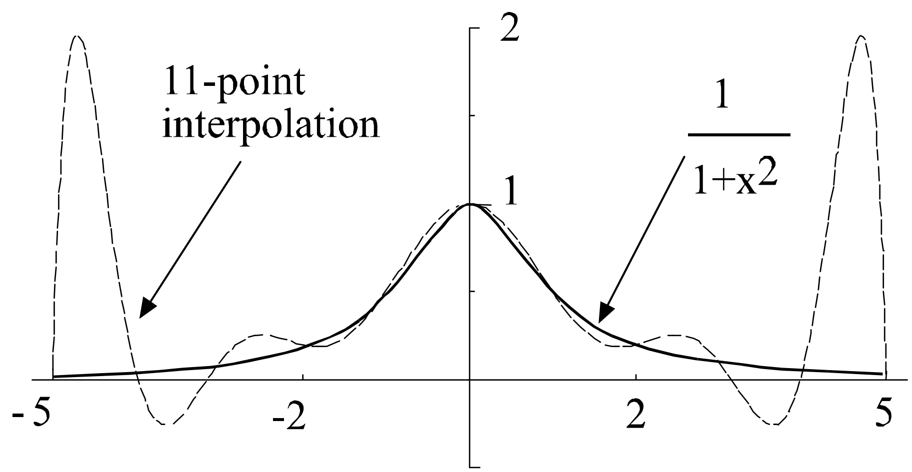


Figure 1:

  - Why does this fail? because there are zero degrees of freedom? bad choice of points? bad function?

- Hermite polynomial interpolation
  - Data: (x_i, y_i, y_i') , $i = 1, .., n$.
  - Objective: Find a polynomial of degree $2n - 1$, $p(x)$, which agrees with the data, i.e.,

$$\begin{aligned} y_i &= p(x_i),\ \ i = 1, .., n \\ y_i' &= p'(x_i),\ \ i = 1, .., n \end{aligned}$$

  - Result: If the x_i are distinct, there is a unique interpolating polynomial
- Least squares approximation
  - Data: A function, $f(x)$.
  - Objective: Find a function $g(x)$ from a class G that best approximates $f(x)$, i.e.,

$$g = \arg\min_{g \in G} \|f - g\|^2$$

Orthogonal polynomials

- General orthogonal polynomials
  - Space: polynomials over domain D
  - Weighting function: $w(x) > 0$
  - Inner product: $\langle f, g \rangle = \int_D f(x)g(x)w(x)dx$
  - Definition: $\{\phi_i\}$ is a family of orthogonal polynomials w.r.t $w(x)$ iff

$$\left\langle \phi_i, \phi_j \right\rangle = 0, \; i \neq j$$

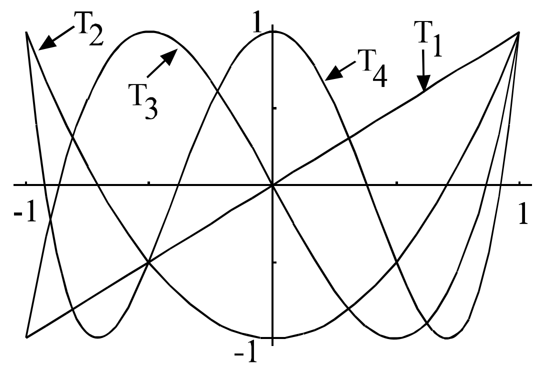
  - We can compute orthogonal polynomials using recurrence formulas

$$\begin{aligned}\phi_0(x) &= 1 \\ \phi_1(x) &= x \\ \phi_{k+1}(x) &= (a_{k+1}x + b_k)\,\phi_k(x) + c_{k+1}\phi_{k-1}(x)\end{aligned}$$

- Chebyshev polynomials
  - $[a,b] = [-1,1]$ and $w(x) = \left(1-x^2\right)^{-1/2}$
  - $T_n(x) = \cos(n \cos^{-1} x)$
  - Recursive definition

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_{n+1}(x) &= 2x\, T_n(x) - T_{n-1}(x), \end{aligned}$$

  - Graphs



- General intervals
  - Few problems have the specific intervals and weights used in definitions
  - One must adapt the polynomials to fit the domain through linear COV:
    * Define the linear change of variables that maps the compact interval $[a, b]$ to $[-1, 1]$

$$y = -1 + 2\frac{x-a}{b-a}$$

    * The polynomials $\phi_i^*(x) \equiv \phi_i\left(-1 + 2\frac{x-a}{b-a}\right)$ are orthogonal over $x \in [a, b]$ with respect to the weight $w^*(x) \equiv \left(-1 + 2\frac{x-a}{b-a}\right)$ iff the $\phi_i(y)$ are orthogonal over $y \in [-1, 1]$ w.r.t. $w(y)$

Regression

- Data: $(x_i, y_i), i = 1, .., n.$
- Objective: Find a function $f(x;\beta)$ with $\beta \in R^m$, $m \leq n$, with $y_i \doteq f(x_i), i = 1, .., n.$
- Least Squares regression:

$$\min_{\beta \in R^m} \sum \left(y_i - f\left(x_i; \beta\right)\right)^2$$

Algorithm 6.4: Chebyshev Approximation Algorithm in $\mathbb{R}^1$

- Objective: Given $f(x)$ defined on $[a, b]$, find its Chebyshev polynomial approximation $p(x)$
- Step 1: Compute the $m \geq n + 1$ Chebyshev interpolation nodes on $[-1, 1]$:

$$z_k = -\cos\left(\frac{2k-1}{2m}\ \pi\right)\ ,\ \ k = 1, \cdots, m.$$

- Step 2: Adjust nodes to $[a, b]$ interval:

$$x_k = (z_k + 1)\left(\frac{b-a}{2}\right) + a, k = 1, ..., m.$$

- Step 3: Evaluate f at approximation nodes:

$$w_k = f(x_k)\ ,\ \ k = 1, \cdots, m.$$

- Step 4: Compute Chebyshev coefficients, $a_i, i = 0, \cdots, n$:

$$a_i = \frac{\sum_{k=1}^{m} w_k T_i(z_k)}{\sum_{k=1}^{m} T_i(z_k)^2}$$

to arrive at approximation of $f(x, y)$ on $[a, b]$:

$$p(x) = \sum_{i=0}^{n} a_i T_i\left(2\frac{x-a}{b-a} - 1\right)$$

Minmax Approximation

- Data: $(x_i, y_i), i = 1, .., n.$
- Objective: L^∞ fit

$$\min_{\beta \in R^m} \max_i \|y_i - f(x_i; \beta)\|$$

- Problem: Difficult to compute

- Chebyshev minmax property

Theorem 1 *Suppose $f : [-1,1] \to R$ is C^k for some $k \geq 1$, and let I_n be the degree n polynomial interpolation of f based at the zeroes of $T_{n+1}(x)$. Then*

$$\| f - I_n \|_\infty \leq \left(\frac{2}{\pi} \log(n+1) + 1 \right)$$
$$\times \frac{(n-k)!}{n!} \left(\frac{\pi}{2}\right)^k \left(\frac{b-a}{2}\right)^k \| f^{(k)} \|_\infty$$

- Chebyshev interpolation:
  - converges in L^∞; essentially achieves minmax approximation
  - works even for C^2 and C^3 functions
  - easy to compute
  - does *not* necessarily approximate f' well

Splines

Definition 2 *A function $s(x)$ on $[a, b]$ is a spline of order n iff*

1. *s is C^{n-2} on $[a, b]$, and*
2. *there is a grid of points (called nodes) $a = x_0 < x_1 < \cdots < x_m = b$ such that $s(x)$ is a polynomial of degree $n - 1$ on each subinterval $[x_i, x_{i+1}]$, $i = 0, \ldots, m - 1$.*

Note: an order 2 spline is the piecewise linear interpolant.

- Cubic Splines
  - Lagrange data set: $\{(x_i,\ y_i) \mid i = 0,\ \cdots\ ,\ n\}$.
  - Nodes: The x_i are the nodes of the spline
  - Functional form: $s(x) = a_i + b_i\,x + c_i\,x^2 + d_i\,x^3$ on $[x_{i-1},\ x_i]$
  - Unknowns: $4n$ unknown coefficients, $a_i, b_i, c_i, d_i, i = 1, \cdots n$.

- Conditions:
  - $2n$ interpolation and continuity conditions:

$$\begin{aligned} y_i =& a_i + b_i x_i + c_i x_i^2 + d_i x_i^3, \\ & i = 1, ., n \\ y_i =& a_{i+1} + b_{i+1} x_i + c_{i+1} x_i^2 + d_{i+1} x_i^3, \\ & i = 0, ., n-1 \end{aligned}$$

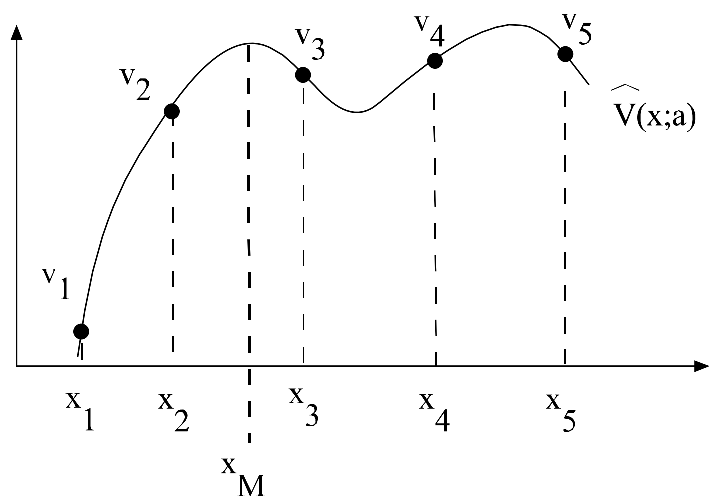
  - $2n-2$ conditions from C^2 at the interior: for $i = 1, \cdots n-1$,

$$\begin{aligned} b_i + 2c_i x_i + 3d_i x_i^2 =& b_{i+1} + 2c_{i+1}\, x_i + 3d_{i+1} x_i^2 \\ 2c_i + 6d_i x_i =& 2c_{i+1} + 6d_{i+1} x_i \end{aligned}$$

  - Equations (1–4) are $4n-2$ linear equations in $4n$ unknown parameters, a, b, c, and d.
  - construct 2 side conditions:
    * *natural spline:* $s''(x_0) = 0 = s''(x_n)$; it minimizes total curvature, $\int_{x_0}^{x_n} s''(x)^2\, dx$, among solutions to (1-4).
    * *Hermite spline*: $s'(x_0) = y_0'$ and $s'(x_n) = y_n'$ (assumes extra data)
    * *Secant Hermite spline:* $s'(x_0) = (s(x_1) - s(x_0))/(x_1 - x_0)$ and $s'(x_n) = (s(x_n) - s(x_{n-1}))/(x_n - x_{n-1})$.
    * *not-a-knot*: choose $j = i_1, i_2$, such that $i_1 + 1 < i_2$, and set $d_j = d_{j+1}$, $j = i_1, i_2$.
  - Solve system by special (sparse) methods; see spline fit packages

Shape Issues

- Approximation methods and shape
  - Concave (monotone) data may lead to nonconcave (nonmonotone) approximations.
  - Example



  - Shape problems destabilize value function iteration

- Shape-preserving orthogonal polynomial approximation
  - Let Least squares Chebyshev approximation preserving increasing concave shape with Lagrange data (x_i, v_i)

$$\begin{aligned}\min_{c_j} \quad & \sum_{i=1}^{m}\left(\sum_{j=0}^{n} c_j\phi_j(x_i) - v_i\right)^2 \\ \text{s.t.} \quad & \sum_{j=1}^{n} c_j\phi_j'(x_i) > 0, \\ & \sum_{j=1}^{n} c_j\phi_j''(x_i) < 0, \qquad i = 1, \ldots, m.\end{aligned}$$

  - Least squares Chebyshev approximation preserving increasing concave shape with Hermite data (x_i, v_i, v_i')

$$\begin{aligned}\min_{c_j} \quad & \sum_{i=1}^{m}\left(\sum_{j=0}^{n} c_j\phi_j(x_i) - v_i\right)^2 + \lambda\sum_{i=1}^{m}\left(\sum_{j=0}^{n} c_j\phi_j'(x_i) - v_i'\right)^2 \\ \text{s.t.} \quad & \sum_{j=1}^{n} c_j\phi_j'(x_i) > 0, \qquad i = 1, \ldots, m, \\ & \sum_{j=1}^{n} c_j\phi_j''(x_i) < 0, \qquad i = 1, \ldots, m.\end{aligned}$$

    where λ is some parameter.

- L1 Shape-preserving approximation
  - L1 increasing concave approximation

$$
\begin{aligned}
\min_{c_j} \quad & \sum_{i=1}^{m} \left| \sum_{j=1}^{n} c_j \phi_j(x_i) - v_i \right| \\
\text{s.t.} \quad & \sum_{j=1}^{n} c_j \phi_j'(z_k) \geq 0, \qquad k = 1, \ldots, K \\
& \sum_{j=1}^{n} c_j \phi_j''(z_k) \leq 0, \qquad k = 1, \ldots, K
\end{aligned}
$$

  - NOTE: We impose shape on a set of points, z_k, possibly different, and generally larger, from the approximation points, x_i.

– This looks like a nondifferentiable problem, but it is not when we rewrite it as

$$
\begin{aligned}
\min_{c_j,\lambda_i} \quad & \sum_{i=1}^{m} \lambda_i \\
\text{s.t.} \quad & \sum_{j=1}^{n} c_j \phi_j' \left(z_k\right) \geq 0, \qquad k = 1, \ldots, K \\
& \sum_{j=1}^{n} c_j \phi_j'' \left(z_k\right) \leq 0, \qquad k = 1, \ldots, K \\
-\lambda_i \leq & \sum_{j=1}^{n} c_j \phi_j \left(x_i\right) - v_i \leq \lambda_i, \qquad i = 1, \ldots, m \\
0 \leq & \lambda_i, \qquad i = 1, \ldots, m
\end{aligned}
$$

- Use possibly different points for shape constraints; generally you want more shape checking points than data points.
- Mathematical justification: semi-infinite programming
- Many other procedures exist for one-dimensional problems, but few procedures exist for two-dimensional problems

Multidimensional approximation methods

- Lagrange Interpolation
  - Data: $D \equiv \{(x_i, z_i)\}_{i=1}^N \subset R^{n+m}$, where $x_i \in R^n$ and $z_i \in R^m$
  - Objective: find $f : R^n \to R^m$ such that $z_i = f(x_i)$.
  - Need to choose nodes carefully.
  - Task: Find combinations of interpolation nodes and spanning functions to produce a nonsingular (well-conditioned) interpolation matrix.

Tensor products

- General Approach:
  - If A and B are sets of functions over $x \in R^n$, $y \in R^m$, their tensor product is

$$A \otimes B = \{\varphi(x)\psi(y) \mid \varphi \in A,\ \psi \in B\}.$$

  - Given a basis for functions of x_i, $\Phi^i = \{\varphi^i_k(x_i)\}_{k=0}^{\infty}$, the *n-fold tensor product* basis for functions of $(x_1, x_2, \ldots, x_n)$ is

$$\Phi = \left\{ \prod_{i=1}^{n} \varphi^i_{k_i}(x_i) \mid k_i = 0, 1, \cdots,\ i = 1, \ldots, n \right\}$$

- Orthogonal polynomials and Least-square approximation
  - Suppose Φ^i are orthogonal with respect to $w_i(x_i)$ over $[a_i, b_i]$
  - Least squares approximation of $f(x_1, \cdots, x_n)$ in Φ is

$$\sum_{\varphi \in \Phi} \frac{\langle \varphi, f \rangle}{\langle \varphi, \varphi \rangle} \varphi,$$

    where the product weighting function

$$W(x_1, x_2, \cdots, x_n) = \prod_{i=1}^{n} w_i(x_i)$$

    defines $\langle \cdot, \cdot \rangle$ over $D = \prod_i [a_i, b_i]$ in

$$\langle f(x), g(x) \rangle = \int_D f(x) g(x) W(x) dx.$$

Algorithm 6.4: Chebyshev Approximation Algorithm in $\mathbb{R}^2$

- Objective: Given $f(x, y)$ defined on $[a, b] \times [c, d]$, find its Chebyshev polynomial approximation $p(x, y)$
- Step 1: Compute the $m \geq n + 1$ Chebyshev interpolation nodes on $[-1, 1]$:

$$z_k = -\cos\left(\frac{2k-1}{2m}\ \pi\right)\ ,\ k = 1, \cdots, m.$$

- Step 2: Adjust nodes to $[a, b]$ and $[c, d]$ intervals:

$$x_k = (z_k + 1)\left(\frac{b-a}{2}\right) + a, k = 1, ..., m.$$

$$y_k = (z_k + 1)\left(\frac{d-c}{2}\right) + c, k = 1, ..., m.$$

- Step 3: Evaluate f at approximation nodes:

$$w_{k,\ell} = f(x_k, y_\ell)\ ,\ k = 1, \cdots, m.\ ,\ \ell = 1, \cdots, m.$$

- Step 4: Compute Chebyshev coefficients, $a_{ij}, i, j = 0, \cdots, n$:

$$a_{ij} = \frac{\sum_{k=1}^{m} \sum_{\ell=1}^{m} w_{k,\ell} T_i(z_k) T_j(z_\ell)}{\left(\sum_{k=1}^{m} T_i(z_k)^2\right)\left(\sum_{\ell=1}^{m} T_j(z_\ell)^2\right)}$$

  to arrive at approximation of $f(x, y)$ on $[a, b] \times [c, d]$:

$$p(x, y) = \sum_{i=0}^{n} \sum_{j=0}^{n} a_{ij} T_i\left(2\frac{x-a}{b-a} - 1\right) T_j\left(2\frac{y-c}{d-c} - 1\right)$$

Complete polynomials

- Taylor's theorem for $\mathbb{R}^n$ produces the approximation

$$f(x) \doteq f(x^0) + \sum_{i=1}^n \frac{\partial f}{\partial x_i}(x^0)\ (x_i - x_i^0)$$

$$+\frac{1}{2}\sum_{i_1=1}^n \sum_{i_2=1}^n \frac{\partial^2 f}{\partial x_{i_1}\partial x_{i_k}}(x_0)(x_{i_1} - x_{i_1}^0)(x_{i_k} - x_{i_k}^0) + ...$$

  - For $k = 1$, Taylor's theorem for n dimensions used the linear functions $\mathcal{P}_1^n \equiv \{1, x_1, x_2, \cdots, x_n\}$
  - For $k = 2$, Taylor's theorem uses $\mathcal{P}_2^n \equiv \mathcal{P}_1^n \cup \{x_1^2, \cdots, x_n^2, x_1x_2, x_1x_3, \cdots, x_{n-1}x_n\}$.

- In general, the kth degree expansion uses the *complete set of polynomials of total degree k in n variables.*

$$\mathcal{P}_k^n \equiv \{x_1^{i_1} \cdots x_n^{i_n} \mid \sum_{\ell=1}^n i_\ell \le k,\ 0 \le i_1, \cdots, i_n\}$$

- Complete orthogonal basis includes only terms with total degree k or less.

- Sizes of alternative bases

degree k	$\mathcal{P}_k^n$	Tensor Prod.
2	$1 + n + n(n+1)/2$	3^n
3	$1 + n + \frac{n(n+1)}{2} + n^2 + \frac{n(n-1)(n-2)}{6}$	4^n

  - Complete polynomial bases contains fewer elements than tensor products.
  - Asymptotically, complete polynomial bases are as good as tensor products.
  - For smooth n-dimensional functions, complete polynomials are more efficient approximations

- Construction
  - Compute tensor product approximation, as in Algorithm 6.4
  - Drop terms not in complete polynomial basis (or, just compute coefficients for polynomials in complete basis).
  - Complete polynomial version is faster to compute since it involves fewer terms
  - Almost as accurate as tensor product; in general, degree $k+1$ complete is better then degree k tensor product but uses far fewer terms.

Shape Issues

- Much harder in higher dimensions
- No general method
- The L2 and L1 methods generalize to higher dimensions.
  - The constraints will be restrictions on directional derivatives in many directions
  - There will be many constraints
  - But, these will be linear constraints
  - L1 reduces to linear programming; we can now solve huge LP problems, so don't worry.