LECTURE NOTES

ECO 613/614

FALL 2007

KAREN A. KOPECKY

Function Approximation

1 Interpolation

Interpolation is a form of function approximation in which the approximating function (*interpolant*) and the underlying function must agree at a finite number of points. In some cases additional restrictions may be imposed on the interpolant. For example its first derivative evaluated at a finite number of points may have to agree with that of the underlying function. Other examples include additional constraints imposed on the interpolant such as monotonicity, convexity, or smoothness requirements.

A choice must be made about which family of functions the interpolant is a member of. Some families of functions commonly used for interpolation include:

- Polynomials (Polynomial Interpolation)
- Trigonometric functions (Fourier Approximation)
- Rational functions (Pade Approximation)

Suppose we are interested in approximating some real-valued function f. The interpolant, $\tilde{f}$, is chosen to be a *linear combination* of some set of basis functions, $\phi_1(x), \ldots, \phi_n(x)$. The basis functions are linearly independent and *span* the family of functions chosen for the interpolation (any function in the family can be written as a linear combination of basis functions). Thus we have,

$$\tilde{f}(x) \equiv \sum_{j=1}^{n} w_j \phi_j(x). \tag{1}$$

Notice that we have now reduced the problem of characterizing an infinite-dimensional object, f to the problem of determining the n weights, w_j, $j = 1 \ldots n$. Since we have n unknowns we need at least n conditions to determine the coefficients. The simplest and

most common conditions imposed are that the interpolant *interpolate* or match the value of the original function at n selected *interpolation nodes*, x_i, $i = 1, \ldots, n$, or,

$$\tilde{f}(x) \equiv \sum_{j=1}^{n} w_j \phi_j(x_i) = f(x_i), \quad i = 1, \ldots, n. \tag{2}$$

Notice that this is a system of *linear* equations in the weights, w_j's. Hence we can rewrite it using matrix notation. Let ϕ be the $n \times n$ *interpolation matrix* and w the $n \times 1$ vector of unknown weights. In addition, let $y_i = f(x_i)$. Then an equivalent representation of the linear system is

$$\phi w = y.$$

Here $\phi_{ij} = \phi_j(x_i)$ or the jth basis function evaluated at the ith interpolation node. Note that $\tilde{f}$ is well-defined if the interpolation matrix, ϕ, is nonsingular. It is also worthwhile to note that interpolation is, in fact, a special case of a general function approximation method, namely, regression. To see this suppose that we have more evaluation nodes then basis functions. While, in general, it will not be possible to satisfy that the interpolant and true function agree at all the nodes we can construct an approximating function by minimizing the sum of squared errors. Where the error is given by

$$e_i = f(x_i) - \sum_{j=1}^{n} w_j \phi_j(x_i)$$

This implies that

$$w = (\phi' \phi)^{-1} \phi' y$$

Notice that this is equivalent to $\phi^{-1} y$ when the number of nodes and basis functions are the same and ϕ is invertible.

One can derive interpolation schemes based on matching a different set of conditions than the common setup derived above. For example, an interpolation scheme can be devised that requires the interpolant and the true function to agree at n_1 points and their derivatives to agree at n_2 where $n_1 + n_2 = n$, the degree of interpolation. This result would be achieved by solving the following linear system:

$$\sum_{j=1}^{n} c_j \phi_j(x_i) = f(x_i) \quad i = 1, \ldots, n_1,$$

$$\sum_{j=1}^{n} c_j \phi_j'(x_i) = f'(x_i) \quad i = 1, \ldots, n_2$$

All the matters is that we have the same number of conditions as basis functions and that the interpolating matrix is nonsingular.

Remark that there is some arbitrariness involved in interpolation since there are arbitrarily many functions that pass through a finite number of points. Interpolation methods are often classified as either *spectral* or *finite element* methods depending whether the basis functions are nonzero over the entire domain of the true function (except possibly at a finite number of points) or nonzero only on a subinterval of the domain. The most common spectral method is *polynomial interpolation* which uses polynomials as basis functions and the most common finite element method is *spline interpolation* which uses functions that are low-order polynomials on subintervals of the domain and zero otherwise as basis functions.

Decisions about what type of interpolation to do and what basis functions to use will depend on (1) what you wish to do with the function (evaluate, integrate, differentiate, etc.) and (2) what properties you want the interpolant to have (usually certain properties of the underlying function).

The minimal conditions that an interpolation scheme should satisfy is that (1) the interpolant should, at least theoretically, be able to approximate the true function arbitrarily well by increasing the number of basis functions and nodes. (2) the linear system should be sufficiently well-conditioned and easy to solve, and (3) the interpolant should be sufficiently easy to work with in the context it is needed. i.e., easy to evaluate and otherwise manipulate (differentiate, integrate, etc.).

1.1 Polynomial Interpolation

The motivation for using polynomials to approximate functions comes from the following theorem:

Theorem 1 (Weierstrass)**.** *If $C[a, b]$ is the set of all continuous function on $[a, b]$ then for all $f \in C[a, b]$ and $\varepsilon > 0$ there exists a polynomial p for which*

$$\sup_{a \le x \le b} |f(x) - p(x)| \le \varepsilon.$$

In addition, if $f \in C^k[a,b]$ then there exists a sequence of polynomials, p_n, where the degree of p_n is n, such that

$$\lim_{n\to\infty} \max_{x\in[a,b]} |f^{(l)}(x) - p_n^{(l)}| = 0$$

for $l \leq k$.

In other words, there exists a polynomial that approximates any continuous function over a compact domain arbitrarily well. (I.e., the approximation error as measured by the sup norm is arbitrarily small.) While valuable conceptually, the Weierstrass Theorem is useless in practice since it doesn't tell us how to find a good approximating polynomial nor the degree of interpolation needed to reach a desired level of accuracy.

1.1.1 Monomial Basis with Evenly-spaced Nodes–A Big Mistake

A naive individual may proceed as follows. First choose as their basis functions the monomials, i.e, $1, x, x^2, x^3, \ldots$, and second choose as their interpolating nodes the collection of n evenly-spaced points over the interval of interest. This approach is problematic for 2 reasons: (1) the monomials is a poor choice for basis functions and (2) an evenly-spaced grid is a poor choice for the interpolating nodes. Let's first focus on the first problem and then talk about the nodes.

The first problem has to do with the choice of basis functions. This choice implies that

$$\phi_j(x) = x^{j-1}, \quad j = 1, \ldots, n$$

and are interpolating polynomial has the form

$$p_{n-1}(x) = w_1 + w_2 x + \cdots + w_n x^{n-1}.$$

If y_i represents the value of our underlying function at location $x_i, \quad i = 1, \ldots, n$ then the weights are determined by solving the $n \times n$ linear system,

$$\begin{bmatrix} 1 & x_1 & \cdots & x_1^{n-1} \\ 1 & x_2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}. \tag{3}$$

The matrix associated with this linear system is called a *Vandermonde matrix*. It can be shown that it's determinant is always non-zero. Thus theoretically, the system is well-defined and a unique solution exists. Unfortunately, in practice, the Vandermonde matrix is known for being often ill-conditioned, especially for high-degree polynomials. This is because as the degree increases the functions become progressively less distinguishable making the columns of the Vandermonde matrix nearly linearly dependent. One possible way to handle this problem is to attempt to rescale the system so as to improve the condition of the matrix. We discussed rescaling before. For this particular system, rescaling such that the matrix elements all lie between $[-1, 1]$ improves the condition somewhat. This can be accomplished by setting

$$\phi_j(x) = \left(\frac{x-c}{d}\right)^{j-1},$$

where $c = (x_1 + x_n)/2$ and $d = (x_n - x_1)/2$.

A better approach is to avoid the Vandermonde matrix all together. Both the conditioning of the linear system and the work required to solve it can be improved upon by changing to a different basis. Remark that we are not changing the interpolating polynomial (remember that it is unique) only the way we represent it.

1.1.2 Better Choices for Basis Functions

Lagrange Basis Functions Alternatively, one can use the Lagrange polynomials as a basis. The jth polynomial is given by

$$l_j(x) = \frac{\prod_{k=1,k\neq j}^{n}(x - x_k)}{\prod_{k=1,k\neq j}^{n}(x_j - x_k)}. \tag{4}$$

Thus

$$l_j(x_i) = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{if} \quad i \neq j \end{cases}$$

Notice that the matrix of the linear system is the identity. Thus the interpolant is

$$p_{n-1}(x) = y_1 l_1(x) + y_2 l_2(x) + \cdots + y_n l_n(x).$$

Pros:

- Extremely easy to determine the interpolating polynomial.

Cons:

- Lagrangian form of the polynomial more expensive to evaluate than monomial form.
- Also more difficult to integrate, differentiate, etc.

Orthogonal Polynomials Often a set of orthogonal polynomials is used as a basis. This basis does a good job of "sampling" the interval. Thus the problem is unlikely to be poorly conditioned. In order to define orthogonal polynomials we first define the notion of a weighting function and inner product:

Definition 2 (Weighting functions)**.** *A weighting function,* $w(\cdot)$ *on* $[a, b]$ *is a positive function almost everywhere, such that* $\int_a^b w(u)du < \infty$.

Definition 3 (Inner Product)**.** *Consider two functions,* f *and* g *defined at least on* $[a, b]$*. The inner product with respect to the weighting function,* $w(\cdot)$ *is given by* $\langle f, g\rangle = \int_a^b f(u)g(u)w(u)du$.

Definition 4 (Orthogonal Polynomials)**.** *The family of polynomials* $\{\psi_k(\cdot)\}_k$ *is orthogonal on* $[a, b]$ *with respect to the weighting function* $w(\cdot)$ *if and only if:* $\langle \psi_i, \psi_j\rangle = 0$ *for* $i \neq j$*. Polynomials are orthonormal if they are orthogonal and* $\langle \psi_k, \psi_k\rangle = 1$ *for all* k.

Chebyshev Polynomials A widely used class of orthogonal polynomials are Chebyshev's polynomial. They are defined by

$$\begin{aligned} T_k : [-1, 1] &\rightarrow [-1, 1] \\ T_k(u) &= \cos(k \cos^{-1}(u)), \quad k = 0, 1, \ldots. \end{aligned}$$

Chebyshev's polynomial also satisfy the recurrence relation

$$T_{k+1}(u) = 2uT_k(u) - T_{k-1}(u),$$

where $T_0(u) = 1$ and $T_1(u) = u$. Note that it is easier to evaluate orthogonal polynomials using the recursive form rather than the closed form. In fact their recursive structure is part of their attractiveness since it makes them so easy to evaluate and generate. Chebyshev's polynomial are orthogonal with respect to $w(u) = (1 - u^2)^{-1/2}$. Thus for two Chebyshev polynomials, $T_i(x)$ and $T_j(x)$, $i \neq j$,

$$\int_{-1}^{1} T_i(x)T_j(x)w(x)dx = 0.$$

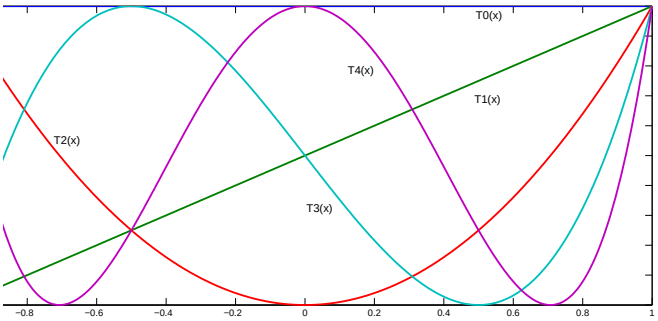

Figure 1: Chebyshev Polynomials $T_k(x),\ k = 0, 1, \ldots, 4$

In addition, in the case $i = j = 0$,

$$\int_{-1}^{1} T_0(x)T_0(x)w(x)dx = \pi.$$

While for $i = j \neq 0$ we have

$$\int_{-1}^{1} T_i(x)T_i(x)w(x)dx = \frac{\pi}{2}.$$

Orthogonal polynomials are especially useful when doing least squares approximation and for generating gaussian quadrature rules as we'll see later on. In addition, Chebyshev polynomials in particular are a popular choice for approximating the solutions to functional equations, such as those that arise from our economic models. This is because the weights (coefficients) associated withe the approximation are extremely easy to obtain as we will see in a moment. Other sets of orthogonal polynomials include Legendre, Laguerre, and Hermite.

1.1.3 Optimal Grid Points

Why is an evenly-spaced grid bad? Suppose we choose our grid points to be equally spaced on the interval $[a, b]$. Then the higher the degree of the approximating polynomial the more "wiggles" it will have. As one moves closer to the endpoints of the interval these oscillations become more dramatic. Hence the interpolant is more likely to be a poor approximation of the true function close to the endpoints. Often these oscillations are not desirable since they do not characterize well the underlying function and in some cases cause the error to grow as we increase the degree sending the sup norm of the error to infinity as n goes to

infinity. This occurs because although Weistrass guarantees that there exists a sequence of polynomials of increasing degree that will eventually converge to the true function, it doesn't guarantee that we are on such a sequence nor that for any given polynomial of degree n it is the best one of its size. And in fact, with an evenly-spaced grid, it will not be. The following theorem, called the Equioscillation Theorem, tells us that a unique, best, degree n polynomial exists. Here, best is defined as the one whose absolute error, defined by the sup norm, is the smallest of all polynomials of degree less than or equal to n.

Theorem 5 (Equioscillation Theorem)**.** *If $C[a,b]$ is the set of all continuous function on $[a,b]$ and $f \in C[a,b]$, then there is a unique polynomial of degree n, $q_n^\star$, such that*

$$||f - q_n^\star||_\infty = \inf_{deg(q) \leq n} ||f - q||_\infty \equiv \rho_n(f)$$

In addition, the polynomial $q_n^\star$ is also the unique polynomial for which there are at least $n+2$ points $a \leq x_0 < x_1 < \cdots < x_{n+1} \leq b$ such that for $m = 1$ or $m = -1$,

$$f(x_j) - q_n^\star(x_j) = m(-1)^j \rho_n(f), \quad j = 0, \ldots, n+1. \tag{5}$$

Equation (5) is called the *equioscillation property.* Geometrically is says that the maximum error of a cubic approximation, for example, should be achieved at least five times and that the sign of the error should alternate between these points. This is a useful theorem because it tells us what our error should look like when we are trying to find the best approximation. If we plot the error and have a very different shape, we know that it is theoretically possible to do better. A general rule of thumb is the closer our error looks to this shape the better the overall approximation.

While we know that a polynomial that minimizes the sup norm of the error exists in theory, such a polynomial is difficult to find in practice. It can be shown, though, that if one uses as his $m = n + 1$ grid points the Chebyshev nodes or the roots of the mth Chebyshev polynomial then the approximate will be nearly the optimal one (i.e., the one that has the smallest possible error as measured by the sup norm).

The mth Chebyshev polynomial has m distinct roots on $[-1, 1]$, where the roots, $\{z_i\}_{i=1}^m$ have the following expression:

$$z_i = -\cos\left(\frac{(2i-1)\pi}{2m}\right).$$

The Cheybshev nodes are the abscissas of m points in the plane that are equally centered around the unit circle. Notice that there are more Cheybshev points near the endpoints of the interval where we see larger errors when using an evenly-spaced grid.

According to Rivlin's Theorem, the approximation error associated with the nth-degree Chebyshev-node polynomial interpolant, p_n, cannot be larger than $2\pi \log(n) + 2$ times the lowest error attainable with any other polynomial approximant of the same order, or

$$||f - p_n||_\infty \leq [2\pi \log(n) + 2]\rho_n(f)$$

For $n = 100$ this factor is approximately 30, which is small compared to errors that have magnitudes that are powers of 10 larger than the optimum. In practice, the accuracy obtained with Cheybchev-node polynomial interpolation is often much better than that indicated by Rivlin's bound, especially if the function being approximated is smooth.

Another theorem, Jackson's Theorem, gives us a useful result. It says that if f is continuously differentiable, then we can bound the approximation error caused by the nth degree Chebyshcev-node polynomial interpolant by

$$||f - p_n||_\infty \leq \frac{6}{n}||f'||_\infty(b - a)[\log(n)/\pi + 1].$$

This error bound can potentially be estimated in practice providing a way to determine the number of grid points in order to achieve a desired level of accuracy.

1.1.4 Chebyshev Interpolation with Optimal Grid Points

In addition to being mutually orthogonal, the Chebyshev polynomials have an attractive characteristic called *discrete orthogonality*. Thus if $\{z_i\}_{i=1}^m$ are the m roots of the mth order Chebyshev polynomial then for all $i, j < m$,

$$\sum_{k=1}^{m} T_i(x_k)T_j(x_k) = \begin{cases} 0, & i \neq j, \\ m/2, & i = j, \neq 0 \\ m, & i = j = 0. \end{cases}$$

This property allows us to derive simple expressions for the interpolation weights when the grid points used are the Cheybshev nodes. To see this suppose we wish to approximate a function $f(\cdot)$ defined on $[-1, 1]$ using the first $m = n+1$ Chebyshev polynomials, i.e., with an

order n polynomial. We take our grid points to be the m roots of the mth order Chebyshev polynomial. Then are problem is to find the j weights $\{w_j\}_{i=0}^{n}$ such that

$$\sum_{j=0}^{n} w_j T_j(z_k) = f(z_k), \quad k = 1, \ldots, m.$$

First pick some $i \in \{0, 1, \ldots, n\}$ and multiply both sides by $T_i(z_k)$ obtaining,

$$\sum_{j=0}^{n} w_j T_i(z_k) T_j(z_k) = T_i(z_k) f(z_k).$$

Now sum across $k = 1, 2, \ldots, m$ to get,

$$\sum_{k=1}^{m} \sum_{j=0}^{n} w_j T_i(z_k) T_j(z_k) = \sum_{k=1}^{m} T_i(z_k) f(z_k).$$

From the discrete orthogonality property, the terms on the left-hand-side where $i \neq j$ are equal to zero. Thus,

$$w_i \sum_{k=1}^{m} T_i(z_k) T_i(z_k) = \sum_{k=1}^{m} T_i(z_k) f(z_k),$$

and for $i = 0$, the discrete orthogonality property yields,

$$w_0 m = \sum_{k=1}^{m} T_0(z_k) f(z_k).$$

Hence,

$$w_0 = \frac{1}{m} \sum_{k=1}^{m} f(z_k).$$

For $i \in \{1, 2, \ldots, n\}$ we get,

$$w_i \frac{m}{2} = \sum_{k=1}^{m} T_i(z_k) f(z_k),$$

or

$$w_i = \frac{2}{m} \sum_{k=1}^{m} T_i(z_k) f(z_k),$$

Thus the weights are very easy to compute.

1.1.5 Interval Conversion

Also note that even though the Chebyshev polynomials are defined over the interval $[-1, 1]$ this interval can be easily generalized to $[a, b]$ by linearly transforming the data, i.e, finding

α and β such that $x = \alpha z + \beta$ and $\beta - \alpha = a$ and $\beta + \alpha = b$. Doing this we find that if $z \in [-1, 1]$ and $x \in [a, b]$ then

$$x = \frac{b-a}{2} z + \frac{a+b}{2},$$

and

$$z = 2\frac{x-a}{b-a} - 1.$$

In some situations we may want the approximating function and the underlying function to agree at the endpoints of the interval. The Chebyshev nodes are always internal thus we are extrapolating when computing the function at the endpoints. These extrapolations tend to be poorer approximations than the ones at locations closer to the center. If a good approximation at the endpoints is desirable we can modify the transformation from $[-1, 1]$ to $[a, b]$ by linearly transforming the data such that a no longer corresponds with -1 but with z_1 and b no longer corresponds with 1 but with z_m. We have m collocation points $\{z_i\}_{i=1}^m$ such that $-1 < z_1 < z_2 < \cdots < z_{m-1} < z_m < 1$. Note that,

$$z_1 = -cos\left(\frac{\pi}{2m}\right),$$

and

$$z_m = -cos\left(\frac{(2m-1)\pi}{2m}\right) = -z_1.$$

It is easy to see that the proper transformation is,

$$x = \frac{\frac{1}{cos\left(\frac{\pi}{2m}\right)} z + 1}{2}(b-a) + a = \frac{sec\left(\frac{\pi}{2m}\right) z + 1}{2}(b-a) + a$$

, and

$$z = \frac{1}{sec\left(\frac{\pi}{2m}\right)}\left(\frac{2(x-a)}{b-a} - 1\right).$$

The grid points $\{x_i\}_{i=1}^m$ constructed in this way satisfy $a = x_1 < x_2 < \cdots < x_{m-1} < x_m = b$ and are called *extended Chebyshev array*.

- **Miranda, Mario J. and Paul L. Fackler.** 2002. *Applied Computational Economics and Finance.* Cambridge, MA: MIT Press.

- **Nocedal, Jorge and Stephen J. wright.** 1999. *Numerical Optimization.* Springer-Verlag New York, Inc.

- **Press, William H.; Saul A. Teukolsky; William T. Vetterling; and Brian P. Flannery.** 1992. *Numerical Recipes in C.* New York, N.Y.: Press Syndicate of the University of Cambridge.