1 Function approximation methods

In this set of notes, we will discuss methods to compute approximations of $f(x)$ based on data (discrete) or the function itself (continuous). We will no longer ask that our approximant $p(x)$ be an interpolant; we will instead directly require it to be "close" to $f(x)$ in some sense (best fit to the data or smallest norm for the error).

1.1 Discrete approximation

Let's discuss the discrete case first: when discussing interpolation, we assumed we had n data points $(x_j, y_j)_{j=1}^n$ for x_j distinct nodes in $[a, b]$. The key assumption behind this is that $y_j = f(x_j)$. That is, we assumed our observations to be accurate function evaluations.

In practice, this is often not the case. Instead, $y_j = f(x_j) + \varepsilon_j$; that is, our observations are, as far as the relationship between x and y goes, noisy. This could be due to:

- **Modeling error:** we may have missed dependencies of y on other variables.
- **Measurement / computation error:** the process through which we got y_j can have uncertainties / errors associated with it.

Usually, ε_j can be assumed to be sampled from a probability distribution. You can carry out a statistical analysis of the function approximants we discuss here based on these "noise models"; that is beyond the scope of our course, but what is discussed below can give you a place to start.

General idea: Let V_m be a vector space of functions (e.g. polynomials, splines, trigonometric poly, rational, exponentials) useful to approximate the underlying function we are working with. Let $\{\phi_k\}_{k=0}^m$ be a basis for this space. We can write our approximant as:

$$p(x) = \sum_{k=0}^{m} a_k \phi_k(x) \tag{1.1}$$

Because our observations are noisy, we should **NOT** make m greater than or equal to n, as at this point *we will be fitting noise*. In other words: we want our observations $n+1$ to be much greater than the number of parameters in our method $m+1$. That way, we can hope to approximate trends in our data.

For this reason, we have to drop the ask that $p(x_j) = y_j$. Instead, we want to find $p(x) \in V_m$ such that, given the vector $\mathbf{y}$ of observations and $\mathbf{p}$ of model evaluations, $\mathbf{p}$ is the minimizer of:

$$\min ||\mathbf{p} - \mathbf{y}|| \tag{1.2}$$

For some vector norm $||\cdot||$.

Discrete Least Squares

For instance, if we make this the euclidean norm (and square it, since the minimizer doesn't change), this becomes a **discrete least squares** problem:

$$\min ||\mathbf{p} - \mathbf{y}||_2^2 = \min \sum_{j=0}^{n} \left(\sum_{k=0}^{m} a_k \phi_k(x_j) - y_j \right)^2 \tag{1.3}$$

As complicated as this looks, as a function of our unknowns $q(a_0, a_1, \ldots, a_k)$, we are asking for the minimum of a convex quadratic function. This unique minimizer is the unique solution to $\nabla q(\mathbf{a}) = 0$, which gives us a set of linear equations.

In class, we did an example for a linear polynomial fit, computing $\frac{dq}{da_1}$ and $\frac{dq}{da_2}$ and setting them equal to zero. This led to the linear system of the form:

$$\begin{bmatrix} \sum_{j=0}^{n} 1 & \sum_{j=0}^{n} x_j \\ \sum_{j=0}^{n} x_j & \sum_{j=0}^{n} x_j^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} \sum_{j=0}^{n} y_j \\ \sum_{j=0}^{n} x_j y_j \end{bmatrix} \tag{1.4}$$

We can derive a general result for Discrete Least Squares by re-writing our objective function in terms of matrix-vector arithmetic. Given the $n+1 \times m+1$ matrix $\mathbf{M}$ with entries $m_{j,k} = \phi_k(x_j)$, the DLS problem becomes:

$$\min ||\mathbf{Ma} - \mathbf{y}||^2 = \min \mathbf{a}^T\mathbf{M}^T\mathbf{Ma} - 2\mathbf{y}^T\mathbf{M}^T\mathbf{a} + \mathbf{y}^T\mathbf{y} \tag{1.5}$$

If $\mathbf{M}$ is of full rank, the matrix $\mathbf{M}^T\mathbf{M}$ is SPD, and this quadratic is strictly convex. Its minimizer happens when the gradient is equal to zero, which we know is when:

$$\mathbf{M}^T\mathbf{Ma} = \mathbf{M}^T\mathbf{y} \tag{1.6}$$

(we have dropped a factor of 2 from the gradient). This set of $m+1$ equations is known as the **normal equations**, and the system matrix $\mathbf{G} = \mathbf{M}^T\mathbf{M}$ is known as the Grammian matrix, with entries $G_{i,j} = M(:,i)^T M(:,j) =< \phi_i(\mathbf{x}), \phi_j(\mathbf{x}) >$. The right-hand-side is of the form $b_i = M(:,i)^T\mathbf{y} =< \phi_i(\mathbf{x}), \mathbf{y} >$.

We could go ahead and solve this linear system to find the unique discrete least squares solution. However, in a number of relevant cases like polynomial interpolation given equispaced data, this matrix is *horribly conditioned*. Intuitively speaking (and this can be made more precise using a Singular Value Decomposition of $\mathbf{M}$), we can show that the condition number $\kappa(\mathbf{G})$ behaves like the square of the condition number of the Vandermonde matrix!

Bonus: efficient and stable linear algebra based algorithms for DLS

Here we will briefly introduce two algorithms which allow us to solve the least squares problem without the issues that come from the bad conditioning of $\mathbf{G}$. This is included in the notes for completion and for the curious reader.

Both of these algorithms have in common that they use a matrix factorization of $\mathbf{M}$; one that involves writing it in terms of orthonormal bases for its range (and in one case, also its domain). Using orthogonality allows us to simplify and solve this problem in a stable and efficient manner.

- **Singular Value Decomposition based algorithm, and the pseudoinverse:** Let $\mathbf{M} = \mathbf{U\Sigma V^T}$, where $\mathbf{U, V}$ are orthogonal matrices (their columns form an orthonormal basis) and Σ is a diagonal matrix with non-negative diagonal entries $\Sigma(i,i) = \sigma_i \geq 0$, where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. This is known as a *Singular Value Decomposition*. It can be obtained from eigendecompositions of $\mathbf{M^TM}$ and $\mathbf{MM^T}$, and there are efficient algorithms to compute it in Python, Matlab, etc.

  If we plug-in the SVD of $\mathbf{M}$ into the system of normal equations, we get the following:

$$\mathbf{M^T M a} = \mathbf{V \Sigma^T \Sigma V^T a} = \mathbf{V \Sigma^T U^T y} \tag{1.7}$$

$$\mathbf{\Sigma^T \Sigma (V^T a)} = \mathbf{\Sigma^T (U^T y)} \tag{1.8}$$

$$\mathbf{V^T a} = \left(\mathbf{\Sigma^T \Sigma}\right)^{-1} \mathbf{\Sigma^T (U^T y)} \tag{1.9}$$

$$\mathbf{a} = \mathbf{V \Sigma^\dagger U^T y} \tag{1.10}$$

where $\Sigma^\dagger$ is an $m+1 \times n+1$ diagonal matrix with diagonal entries $1/\sigma_j$ if $\sigma_j > 0$, and zero otherwise.

This SVD-based algorithm also tells us how to find a least squares minimizer *if matrix* $\mathbf{M}$ *is not full rank*, as can be the case in a number of applications. If this is the case, the DLS problem is not uniquely solvable. However, using the SVD and properties of unitary matrices, we can look at the objective function directly:

$$\min ||\mathbf{U \Sigma V^T a} - \mathbf{y}||_2^2 = \min ||\mathbf{\Sigma V^T a} - \mathbf{U^T y}||_2^2 \tag{1.11}$$

$$= \min ||\Sigma \boldsymbol{\beta} - \mathbf{U}^T \mathbf{y}||_2^2 \tag{1.12}$$

$$= \min \sum_{j=1}^{r} (\sigma_j \beta_j - u_j^T \mathbf{y})^2 + \sum_{j=r+1}^{m} (u_j^T \mathbf{y})^2 \tag{1.13}$$

where $\boldsymbol{\beta} = \mathbf{V^T a}$. This suggests that the coefficient vector $\boldsymbol{\beta} = \mathbf{\Sigma^\dagger U^T y}$, or $\mathbf{a} = \mathbf{V \Sigma^\dagger U^T y}$ is *the least squares minimizer with the smallest norm* (since we made all the free betas equal to 0).

- **QR decomposition (Gram-Schmidt) based algorithm:** Another approach to solving the discrete LS problem is to use orthogonality to find the LS solution, both in the full rank and deficient rank cases. This corresponds to methods based on the QR decomposition of $\mathbf{M}$. Given r to be the rank of $\mathbf{M}$, we can compute

$$\mathbf{M} = \mathbf{Q} \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}$$

with $\mathbf{Q}$ size $n+1 \times r$ with orthonormal columns and $\mathbf{R}$ square, triangular matrix, then the LS solution is given by $\mathbf{a} = \mathbf{R^{-1} Q^T y}$, where the system for $\mathbf{R}$ can be solved efficiently using a triangular solve.

Generalizations of DLS

We can slightly generalize the discrete least squares problem in two ways relevant to applications.

Weighted LS: The first one is to minimize a weighted sum of least-square errors. That is, given SPD matrix $\mathbf{W}$, we have the inner product $< \mathbf{x}, \mathbf{y} >_\mathbf{W} = x^T \mathbf{W y}$ and associated norm $||\mathbf{x}||_\mathbf{W}$. We can then ask to minimize:

$$\min ||\mathbf{M a} - \mathbf{y}||_\mathbf{W}^2 \tag{1.14}$$

Most often, this matrix is diagonal $\mathbf{W} = \text{diag}(\mathbf{w})$. This then becomes,

$$\min ||\mathbf{Ma} - \mathbf{y}||_{\mathbf{W}}^2 = \min \sum_{j=0}^{n} w_j \left(p(x_j) - y_j\right)^2 \tag{1.15}$$

Regardless, we can carry out *the same exact analysis* of the quadratic in our vector of unknown coefficients, and find the system:

$$\mathbf{M^T W M a} = \mathbf{M^T W y} \tag{1.16}$$

Ridge Regression / Shrinkage / Tikhonov regularization: The idea here is to add a penalty term to limit the size of the coefficient vector; this often has a smoothing effect. That is, we want to minimize:

$$\min ||\mathbf{Ma} - \mathbf{y}||_2^2 + \lambda ||\mathbf{a}||^2 \tag{1.17}$$

for non-negative parameter λ. Carrying out the same analysis yields the system:

$$(\mathbf{M^T M} + \lambda \mathbf{I})\mathbf{a} = \mathbf{M^T y} \tag{1.18}$$

you can see that the larger λ is, it improves upon the condition number by shifting all eigenvalues of the Grammian away from zero.

Of course, there are more general penalties one can add; a popular one is to replace the euclidean norm of $\mathbf{a}$ with the 1-norm (LASSO), or with the norm of $\Gamma\mathbf{a}$ representing norms of model function derivatives (e.g. smooth spline fit).

1.2 Continuous function approximation

Unlike the discrete case, we now want to find a function $p \in V_m$ that is, in terms of a given norm or distance function, the best approximant in this finite dimensional space for $f(x)$. We will assume, at the very least, that $f \in C([a, b])$.

In what sense might I try to approximate $f(x)$? We can use the q norms for functions:

- **Minimax:** we want p such that it minimizes $||f - p||_\infty = \sup\{|f(x) - p(x)|\}$
- L^2 **norm (Continuous LS):** we wish to minimize $||f - p||_2^2 = \int_a^b (f(x) - p(x))^2 dx$.
- **weighted** L^2 **norm (Continuous LS):** we wish to minimize $||f - p||_w^2 = \int_a^b (f(x) - p(x))^2 w(x) dx$.
- L^q **norm, for** $q \geq 1$**:** we wish to minimize $||f - p||_q = \int_a^b (f(x) - p(x))^q dx$.

For polynomial approximation, we have the following theorems at our disposal:

Theorem 1.1 (Existence of a minimizer) *Let $f \in C([a, b])$. Given $n \geq 0$, there exists $p \in \mathcal{P}_n$ such that it minimizes $||p - f||_q$.*

For the sup norm, we have the Weierstrass theorem, that tells us that given ε, we can find a polynomial approximant that is uniformly that close to f. This, in turn, also tells us that the error for the minimizer for degree $\leq n$ goes to 0 as $n \to \infty$!

Theorem 1.2 (Weierstrass) *Let $f \in C([a,b])$ and $\varepsilon > 0$. Then, there exists p_ε polynomial of degree $n(\varepsilon)$ such that*

$$||f - p||_\infty < \varepsilon \tag{1.19}$$

This implies that, if $\rho_n[f] = \min_{p \in \mathcal{P}_n} ||f-p||_\infty$, then

$$\lim_{n\to\infty} \rho_n[f] = 0 \tag{1.20}$$

1.2.1 Continuous Least Squares problem:

Once again, we want to find $p \in V_m$ such that it is the minimizer of a continuous L^2 or weighted L^2 norm of the error. For this purpose, we again write down $p(x) = \sum_{k=0}^m a_k \phi_k(x)$ for a basis with terms ϕ_k. That is:

$$\min \int_a^b \left(\sum_{k=0}^m a_k \phi_k(x) - f(x) \right)^2 w(x) dx = \min < \sum_{k=0}^m a_k \phi_k(x) - f(x), \sum_{k=0}^m a_k \phi_k(x) - f(x) >_w^2$$
$$= \mathbf{aGa} - 2\mathbf{b}^T\mathbf{a} + ||f||_w^2$$

with $\mathbf{G}(i,j) =< \phi_i, \phi_j >_w$ and $\mathbf{b}(i) =< \phi_i, f >_w$. This has exactly the same structure as the discrete case, except now our inner products are integrals of functions instead of discrete sums!

Same as in the discrete case, the Gram matrix can be horribly conditioned; for instance, if we are doing polynomial approximation and use the monomial basis. Instead, we can choose our basis for V_m so that it is **orthogonal**: this makes $\mathbf{G}$ a diagonal matrix with entries $||\phi_k||_w^2$. It also implies, then that the coefficients can be found via the formula:

$$a_k = \frac{< \phi_k, f >_w}{< \phi_k, \phi_k >_w} \tag{1.21}$$

These coefficients should look familiar to us: given an orthogonal basis, these correspond to the *scalar orthogonal projection* of $f(x)$ onto the space V_m. The actual projection is given by:

$$p(x) = \sum_{k=0}^m \frac{< \phi_k, f >_w}{< \phi_k, \phi_k >_w} \phi_k(x) \tag{1.22}$$

Orthogonal bases are, by many accounts, the best bases, and this carries out to infinite dimensional spaces like $C([a,b])$ and $L^2([a,b])$. We have, for example, the following results:

- **Bessel's inequality:** $||p||_w^2 = \sum_{k=0}^m a_k^2 \leq ||f||_w^2$
- **Parseval identity:** $||f||_w^2 = \sum_{k=0}^\infty a_k^2$

Building orthogonal bases:

Same as for bases of vectors in $\mathbb{R}^n$, we can use the Gram-Schmidt process to transform a basis $\{\psi_0, \psi_1, \ldots, \psi_m\}$ of V_m into an orthogonal basis $\{\phi_0, \phi_1, \ldots, \phi_m\}$. In particular, if we want to find orthogonal bases of the space of polynomials of degree n, we can start with the monomial (canonical) basis $\{1, x, x^2, \ldots, x^n\}$. We will find a basis of orthogonal polynomials such that:

- Φ_k is a polynomial of degree k.

- $\text{span}(\{\Phi_0, \Phi_1, \ldots, \Phi_k\}) = \mathcal{P}_k$, that is, the first $k+1$ terms span $\mathcal{P}_k$.
- $< \Phi_k, \Phi_i >_w = 0$ for $i < k$.
- $< \Phi_k, q >_w = 0$ FOR ALL $q \in \mathcal{P}_{k-1}$.

Example 1: Legendre polynomials Given the interval $[-1, 1]$ and weight function $w(x) = 1$ (which is the simplest one), we wish to find an orthogonal basis. For this purpose, we follow the Gram-Schmidt process:

$$P_0(x) = 1 \tag{1.23}$$

$$P_1(x) = x - \frac{< x, 1 >}{< 1, 1 >} 1 = x - 0 = x \tag{1.24}$$

$$P_2(x) = x^2 - \frac{< x^2, 1 >}{< 1, 1 >} 1 - \frac{< x^2, x >}{< x, x >} x = x^2 - \frac{2/3}{2} 1 = x^2 - \frac{1}{3} \tag{1.25}$$

$$\tag{1.26}$$

and so on. This gives us the basis of Legendre polynomials. Note that sometimes these polynomials might be normalized: popular ones are making them an orthonormal basis, or scaling such that $P_k(1) = 1$.

3-term recurrence

It is a well-known fact for Legendre as well as for general bases of orthogonal polynomials that they in fact satisfy a 3-term recursion; this is a formula similar to that of the Fibonacci sequence. That is: if we know P_n and P_{n-1}, we can obtain P_{n+1}. This is one of many extremely useful formulas one can derive for these families of polynomial bases.

In general, we can show that if we take $\Phi_{-1} = 0, \Phi_0 = 1$, then we can derive a recursion formula of the form:

$$\Phi_{n+1}(x) = (x - \beta_n)\Phi_n(x) - \gamma_n \Phi_{n-1}(x) \tag{1.27}$$

where

$$\beta_n = \frac{< x\Phi_n, \Phi_n >_w}{< \Phi_n, \Phi_n >_w} \quad \gamma_n = \frac{< \Phi_n, \Phi_n >_w}{< \Phi_{n-1}, \Phi_{n-1} >_w} \tag{1.28}$$

the proof of this statement, which we did in detail in class, is usually done by induction; the case $n = 1$ simply yields the first term of Gram-Schmidt. For the general induction step, we must use the assumed orthogonality properties of Φ_k for $k \leq n$ to show that Φ_{n+1} is, as defined, orthogonal to all basis terms for $k < n - 1$. Then, we can show that the coefficients β_n, γ_n are exactly what is needed so that Φ_{n+1} is orthogonal to Φ_n and Φ_{n-1}.

Example 2: Chebyshev polynomials (of the 1st kind)

A very important example of an orthogonal basis, which is tremendously useful in applications like spectral and pseudospectral methods, is the Chebyshev basis. This basis corresponds to the interval $[-1, 1]$ and weight $w(x) = \frac{1}{\sqrt{1-x^2}}$.

Instead of deriving these using G-S, we gave a formula for them in terms of trigonometric identities, and from it we showed orthogonality AND the recursion relation directly. This is a general feature of Chebyshev polynomial families, of which there are 4 kinds.

We define:

$$T_n(x) = \cos n\theta \quad \text{for} \quad x = \cos\theta$$

We then showed that $T_0 = 1$, $T_1 = x$ and from identities for angle sum or product of cosines, we can show the recurrence relation:

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \tag{1.29}$$

Orthogonality can be explicitly proven, again using the change of variables $x = \cos\theta$:

$$< T_n, T_m >= \int_{-1}^{1} T_n(x)T_m(x)\frac{dx}{\sqrt{1-x^2}} = \int_0^{\pi} \cos(n\theta)\cos(m\theta)d\theta \tag{1.30}$$

this gives us 0 if $n \neq m$, π if $n = m = 0$ and $\pi/2$ otherwise. That gives us the formula for the polynomial approximant to the weighted continuous L^2 problem in $[-1, 1]$:

$$a_0 = \frac{1}{\pi}\int_0^{\pi} f(\cos\theta)d\theta \tag{1.31}$$

$$a_k = \frac{2}{\pi}\int_0^{\pi} f(\cos\theta)\cos n\theta d\theta \quad k \in \mathbb{N} \tag{1.32}$$

$$p(x) = \sum_{k=0}^{m} a_k T_k(x) \tag{1.33}$$

computing these coefficients can then be sped up using a Fast Cosine Transform (FCT).

Remark on Chebyshev approximation: The Chebyshev L^2 approximant is known to have very desirable properties; in particular, one can show that it is in fact "near minimax" (the sup norm of the error is very well controlled, and close to the optimum in some sense).

Chebyshev approximation of functions is also extremely useful to solve a number of problems of interest, e.g. PDEs. This has led to software packages that do scientific computing with functions based on Chebyshev polynomial representations; namely Nick Trefethen's **Chebfun**. There is a wealth of cool properties and numerical methods based on L^2 and Chebyshev polynomial approximants.

Corollary of Chebyshev: Why are Chebyshev nodes so good?

One can show that the Chebyshev nodes we used in polynomial interpolation (to avoid the Runge phenomenon) are, in fact, the zeroes of the $n+1$ Chebyshev polynomial. As a result, one can show that the $\Psi(x)$ that shows up in the interpolation error estimate is none other than $\frac{1}{2^n}T_{n+1}(x)$. But $T_{n+1}(x)$ is between -1 and 1 (it is a cosine function). Hence, $|\Psi(x)| \leq \frac{1}{2^n}$! This explains why Chebyshev nodes give us *the best nodes for interpolation.*

1.3 Continuous approximation with non-polynomials

We can use the formulation above to treat L^2 and weighted L^2 approximants using spaces of trigonometric functions, splines, etc. All the structure that is needed for discrete or continuous LS is the one provided by the Hilbert space (inner-product space). Each of these examples is of great interest in applications, and in particular, one can show the L^2 approximant for periodic smooth functions in $[\pi, \pi]$ is none other than the truncated Fourier series (at this point, this should not be shocking).

1.3.1 Minimax approximation:

The Minimax approximant is the polynomial such that the sup norm of the error is controlled. Even if we limit ourselves to polynomials of degree $\leq n$, computing the minimax is substantially harder than computing the L^2 approximation (inner product spaces are always the nicest).

To compute the minimax approximation (if we are not happy with taking Chebyshev), the main result we have to use is the Chebyshev Equioscillation Theorem, that characterizes what the minimax approximant satisfies (yes, Chebyshev was very active in this area):

Theorem 1.3 (Chebyshev Equioscillation Theorem) *Let $f \in C([a,b])$, $n \geq 0$. Then, there exists* ***a unique*** *$q \in \mathcal{P}_n$ such that $\rho_n[f] = ||f - q||_\infty$ (recall $\rho_n[f]$ is the minimum sup norm in this space).*

Further, this polynomial q ***is the only polynomial in*** *$\mathcal{P}_n$ for which there are at least $n+2$ distinct points $a \leq x_0 < x_1 < \cdots < x_{n+1} leq b$ such that*

$$f(x_j) - q(x_j) = \sigma(-1)^j \rho_n[f] \tag{1.34}$$

with $\sigma = \pm 1$.

This suggests the following: assuming f is at least continuously differentiable, then the $n+2$ extrema will occur either at endpoints or in interior points, which will be then critical points for the error $E(x) = f(x) - q(x)$. We can then write down the equations:

$$E(x_j) = (-1)^j \rho \tag{1.35}$$
$$E'(x_j) = 0 \tag{1.36}$$

in the unknowns x_j, ρ and the coefficients a_k that we use to represent $q(x)$ in some polynomial basis. This is a non-linear system of equations; we may, for instance, try to solve it using methods such as Newton.

The **Remez exchange method** gives us another idea based on this set of equations: given a guess for our nodes x_j, the resulting system of equations is then *linear* in ρ and a_k. We solve this linear system, and then find the $n+2$ points where the error $E(x)$ is the greatest. These, then, become our new guesses for x_j. This method is iterative, and can be shown to converge quadratically regardless of initial choice for the distinct $n+2$ nodes.

Extensions: the extension of Remez and other minimax algorithms to rational functions is a research question of great current relevance. Look rational Remez to learn more.