Rational approximation

MATH 6610 Lecture 28

November 20, 2020

Types of approximation

We considered two types of approximation:

- Fourier Series approximation (periodic functions)
- Polynomial approximation (mostly interpolation)

Both of these methods have certain (dis)advantages.

Types of approximation

We considered two types of approximation:

- Fourier Series approximation (periodic functions)
- Polynomial approximation (mostly interpolation)

Both of these methods have certain (dis)advantages.

The last type of approximation we'll consider is *rational* approximation.

General setup: univariate scalar-valued functions, but can be complex valued.

$f : \mathbb{C} \to \mathbb{C}.$

Rational functions

A function ~~R~~ r $: \mathbb{C} \to \mathbb{C}$ is a rational function if it is a ratio of polynomials:

$$r(z) := \frac{p(z)}{q(z)}, \qquad p, q \in P_n,$$

$\quad = \operatorname{span}\{1, z, \ldots z^n\}$

where P_n is the space of polynomials of degree n and less.

Terminology: r is a rational function of "type $(\deg p, \deg q)$".

We'll assume throughout that p and q have no common (non-constant) divisors.

The function r is a ("strictly") *proper rational function* if $\deg p < \deg q$.

Note that p and q are not unique without specifying a normalization.

$$r(z) = \frac{z^2}{z^2} = \frac{1}{1}$$

if $\deg p \geq \deg q \implies r(z) = w(z) + \dfrac{u(z)}{v(z)}$,

u, v, w are polynomials.

$$r(z) = \frac{p(z)}{q(z)} = \frac{3 \cdot p(z)}{3 \cdot q(z)}$$

For normalization, we'll assume $q(z) = 1 +$ (higher order terms)

Rational functions

A function $R : \mathbb{C} \to \mathbb{C}$ is a rational function if it is a ratio of polynomials:

$$r(z) := \frac{p(z)}{q(z)}, \qquad\qquad p, q \in P_n,$$

where P_n is the space of polynomials of degree n and less.

Terminology: r is a rational function of "type $(\deg p, \deg q)$".

We'll assume throughout that p and q have no common (non-constant) divisors.

The function r is a ("strictly") *proper rational function* if $\deg p < \deg q$.

Note that p and q are not unique without specifying a normalization.

Goal: given f, construct r such that $f \approx r$.

Why is this better (worse?) than polynomial approximation or Fourier Series?

Some functions are very efficiently represented by rational functions.

Padè approximation (very well-known)

One strategy for constructing rational functions is Padè approximation.

The main idea: choose $r = p/q$ such that

$$f(z) = \frac{p(z)}{q(z)} + \mathcal{O}(x^{n+m+1}), \qquad \deg p = m, \qquad \deg q = n.$$

I.e., match Taylor coefficients to as high an order as possible. (up to order m+n).

p has m+1 degrees of freedom

q has n degrees of freedom $(q(z) = 1 + \ldots)$

Padè approximation

One strategy for constructing rational functions is Padè approximation.

The main idea: choose $r = p/q$ such that

$$f(z) = \frac{p(z)}{q(z)} + \mathcal{O}(x^{n+m+1}), \qquad \deg p = m, \qquad \deg q = n.$$

I.e., match Taylor coefficients to as high an order as possible.

Specifically, suppose p and q have the form,

$$r(z) = \frac{p(z)}{q(z)} = \frac{\sum_{j=0}^{m} a_j x^j}{1 + \sum_{j=1}^{n} b_j x^j},$$

for some coefficients $a_0, \ldots, a_m$ and $b_1, \ldots, b_n$. (and let's define $b_0 = 1$)

How are Pade approximante formed? Taylor Series

$f(z) = \frac{p(z)}{q(z)}$ up to order $m+n$.

$f(z)$ up to order $m+n$ is $\sum_{j=0}^{m+n} c_j x^j$ for some (known)

coefficients c_j.

$$\sum_{j=0}^{m+n} c_j x^j \approx \frac{\sum_{j=0}^{m} a_j x^j}{\sum_{j=0}^{n} b_j x^j}$$ goal: compute a_j, b_j.

$$\left(\sum_{j=0}^{m+n} c_j x^j\right)\left(\sum_{j=0}^{n} b_j x^j\right) = \sum_{j=0}^{m} a_j x^j$$

$$\left.\begin{aligned}
x^0 &: \; b_0 c_0 = a_0 \\
x^1 &: \; b_0 c_1 + b_1 c_0 = a_1 \\
x^2 &: \; b_0 c_2 + b_1 c_1 + b_2 c_0 = a_2 \\
&\vdots \\
x^j &: \; \sum_{k=0}^{j} b_k c_{j-k} = a_j \quad (0 \le j \le m)
\end{aligned}\right\}$$ linear in b_j, a_j.

continue for higher orders $(j > m)$: RHS $= 0$.

$$x^j, \, j > m: \quad \sum_{k=0}^{j} b_k c_{j-k} = 0.$$

There are n unknown coefficients $\{b_k\}_{k=1}^{n}$,

So take n conditions:

$$\sum_{k=0}^{j} b_k c_{j-k} = 0 \quad j = m+1, m+2, \ldots, m+n.$$

Set of n linear equations for n unknowns.

$\rightarrow$ can solve via linear algebra.

Second step: solve $\sum_{k=0}^{j} b_k c_{j-k} = a_j$ for $j = 0, \ldots m$

for the coefficients $\{a_j\}_{j=0}^{m}$

Padè approximation

One strategy for constructing rational functions is Padè approximation.

The main idea: choose $r = p/q$ such that

$$f(z) = \frac{p(z)}{q(z)} + \mathcal{O}(x^{n+m+1}), \qquad \deg p = m, \qquad \deg q = n.$$

I.e., match Taylor coefficients to as high an order as possible.

Specifically, suppose p and q have the form,

$$r(z) = \frac{p(z)}{q(z)} = \frac{\sum_{j=0}^{m} a_j x^j}{1 + \sum_{j=1}^{n} b_j x^j},$$

for some coefficients $a_0, \ldots, a_m$ and $b_1, \ldots, b_n$. The computation can be accomplished in a two-step procedure:

- Compute $\{b_j\}_{j=1}^{n}$ with a linear system matching orders $m+1, \ldots, m+n$.
- Compute $\{a_j\}_{j=0}^{m}$ with a linear system matching orders $0, \ldots, m$.

Rational approximation practicalities

$$f(z) = \frac{p(z)}{q(z)} + \mathcal{O}(x^{n+m+1}), \qquad \deg p = m, \qquad \deg q = n.$$

In order to match coefficients, we need the Taylor expansion of f.

Rational approximation practicalities

$$f(z) = \frac{p(z)}{q(z)} + \mathcal{O}(x^{n+m+1}), \qquad \deg p = m, \qquad \deg q = n.$$

In order to match coefficients, we need the Taylor expansion of f.

This is not so practical, but it does reveal a very useful strategy: linearization.

Consider, e.g., interpolation: (alternative to Pade)

$$r(z_j) = \frac{p(z_j)}{q(z_j)} = f(z_j), \qquad j = 1, \ldots, m+n+1.$$

The difficulty in imposing these conditions: they depend <u>non</u>linearly on coefficients.

Rational approximation practicalities

$$f(z) = \frac{p(z)}{q(z)} + \mathcal{O}(x^{n+m+1}), \qquad\qquad \deg p = m, \qquad\qquad \deg q = n.$$

In order to match coefficients, we need the Taylor expansion of f.

This is not so practical, but it does reveal a very useful strategy: linearization.

Consider, e.g., interpolation:

$$r(z_j) = \frac{p(z_j)}{q(z_j)} = f(z_j), \qquad\qquad j = 1, \ldots, m+n+1.$$

The difficulty in imposing these conditions: they depend <u>non</u>linearly on coefficients.

Linearization: impose these conditions in a different way:

$$q(z_j) f(z_j) = p(z_j), \qquad\qquad j = 1, \ldots, m+n+1.$$

This results in a linear system for the a_j, b_j coefficients.

Linearizations

$$f(z) = \frac{p(z)}{q(z)} \qquad \longrightarrow \qquad q(z)f(z) = p(z).$$

For interpolation and Padè approximation, linearization does not change formulation.

For other conditions, e.g., least-squares, linearization is different.

However, linearization provides a concrete solution strategy.

Linearizations

$$f(z) = \frac{p(z)}{q(z)} \quad \longrightarrow \quad q(z)f(z) = p(z).$$

For interpolation and Padè approximation, linearization does not change formulation.

For other conditions, e.g., least-squares, linearization is different.

However, linearization provides a concrete solution strategy.

There is one problem that linearization doesn't solve: how to ensure a good approximation?

One answer: there is an algorithm that <u>empirically</u> gives good approximation results: AAA.

Barycentric form

Consider an alternative "barycentric" formulation for a rational function:

f_j, w_j are arbitrary scalars.

$$r(z) = \frac{\sum_{j=1}^{m} \frac{w_j f_j}{z - z_j}}{\sum_{j=1}^{m} \frac{w_j}{z - z_j}}. = \frac{n(z)}{d(z)}$$

, n, d not polynomials.

By eliminating denominators: this is a type $(m-1, m-1)$ rational function.
(It's actually also a polynomial if w_j are chosen correctly....)

Barycentric form

Consider an alternative "barycentric" formulation for a rational function:

$$r(z) = \frac{\sum_{j=1}^{m} \frac{w_j f_j}{z - z_j}}{\sum_{j=1}^{m} \frac{w_j}{z - z_j}}.$$

By eliminating denominators: this is a type $(m-1, m-1)$ rational function. (It's actually also a polynomial if w_j are chosen correctly....)

The coefficients f_j and w_j are freely chosen complex numbers.

There are some important properties of this approximation:

- If $w_j \neq 0$, then r does <u>not</u> have a pole at $z = z_j$.
- If $w_j \neq 0$, then $r(z_j) = f_j$.
- The above are true independent of how $w_j \neq 0$ are chosen.

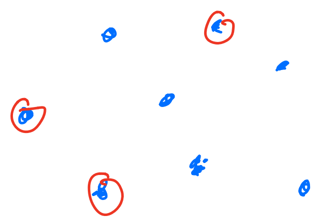
for $z \approx z_j$: $r(z) \approx \dfrac{\frac{w_j f_j}{z - z_j}}{\frac{w_j}{z - z_j}} = f_j$

AAA Algorithm basic idea

"Adaptive Antoulas-Anderson" algorithm.

Given nodes & function values $f(Z_j)=F_j$

$\mathbb{C}$



: given data (Z_j, F_j) $j=1 .. M$

: interpolation points (determines z_j, f_j in Barycentric form)

: linearized least-squares on these data points to determine w_j.

The AAA algorithm

m: # of interpolation points.

$$r(z) = \frac{\sum_{j=1}^{m} \frac{w_j f_j}{z - z_j}}{\sum_{j=1}^{m} \frac{w_j}{z - z_j}}.$$

Given data,

$$(Z_1, \ldots, Z_M), \quad (F_1, \ldots, F_M),$$

with $f(Z_j) = F_j$, and $M \gg m$.

The AAA algorithm

$$r(z) = \frac{\sum_{j=1}^{m} \frac{w_j f_j}{z - z_j}}{\sum_{j=1}^{m} \frac{w_j}{z - z_j}}.$$

Given data,

$$(Z_1, \ldots, Z_M), \quad (F_1, \ldots, F_M),$$

with $f(Z_j) = F_j$, and $M \gg m$.

AAA core ideas:

- "Intelligently" choose interpolation locations $\{z_1, \ldots, z_m\} \subset \{Z_1, \ldots, Z_M\}$ (Hence choose z_j, f_j appropriately)
- The $\{w_j\}_{j=1}^{m}$ can be chosen arbitrarily: choose them to minimize a least-squares residual.

The algorithm proceeds in an alternating fashion. Let $m = 0$.

1. Choose z_{m+1} (and hence f_{m+1}.).
2. Compute weights $\{w_j\}_{j=1}^{m+1}$ using least-squares.
3. $m \leftarrow m + 1$ and repeat steps.

AAA algorithm interpolation

How is z_{m+1} chosen? (Interpolation points)

- If $m = 0$, choose

$$j^* = \arg\max_j |F_j|, \qquad z_1 = Z_{j*}.$$

- If $m > \cancel{1}$ 0, choose

$$j^* = \arg\max_j |F_j - r(Z_j)|, \qquad z_{m+1} = Z_{j*}.$$

The approximation r above is the m-point barycentric rational approximation from the previous step.

AAA algorithm least squares

$$r(z) = \frac{n(z)}{d(z)} = \frac{\sum_{j=1}^{m} \frac{w_j f_j}{z - z_j}}{\sum_{j=1}^{m} \frac{w_j}{z - z_j}}.$$

How are the weights $\{w_j\}_{j=1}^m$ chosen?

First note that there is ambiguity in the normalization of the weights, so enforce

$$\|w\|_2 = 1, \qquad\qquad w = (w_1, \ldots, w_m)^T.$$

AAA algorithm least squares

linearization

$$r(z) = \frac{n(z)}{d(z)} = \frac{\sum_{j=1}^{m} \frac{w_j f_j}{z-z_j}}{\sum_{j=1}^{m} \frac{w_j}{z-z_j}}. \approx f(z) \Rightarrow f(z)d(z) \approx n(z)$$

How are the weights $\{w_j\}_{j=1}^{m}$ chosen?

First note that there is ambiguity in the normalization of the weights, so enforce

$$\|w\|_2 = 1, \qquad w = (w_1, \ldots, w_m)^T.$$

The weights are now chosen in the *linearized* least squares sense:

$$w^* = \underset{w \in \mathbb{C}^m}{\arg\min} \sum_{j \in S_m} \left| d(Z_j) F_j - n(Z_j) \right|^2,$$

where the index set S_m corresponds to the indices j such that Z_j is not an interpolation node:

$$S_m := \left\{ j \in \{1, \ldots, M\} \;\middle|\; Z_j \notin \{z_1, \ldots, z_m\} \right\}.$$

$$|d(z_j)\,F_j - n(z_j)|^2$$

$$= \left| \sum_{k=1}^{m} \frac{F_j w_k}{\cancel{z}\, z_j - z_k} - \sum_{k=1}^{m} \frac{f_k w_k}{\cancel{z}\, z_j - z_k} \right|^2$$

$$= \left| \sum_{k=1}^{m} \left(\frac{F_j - f_k}{z_j - z_k} \right) w_k \right|^2$$

Define L_m to be an $(M-m) \times m$ matrix.

Let $\{s_1, s_2 \ldots s_{M-m}\} = S_m$ (least-squares indices).

$$(L_m)_{j,k} = \frac{F_{s_j} - f_k}{z_{s_j} - z_k}$$

k: interpolation points

j: least-squares points.

$$= |(L_m w)_j|^2$$

Least-squares residual:

$$\sum_{j=1}^{M-m} \left| d(z_{s_j}) F_{s_j} - n(z_{s_j}) \right|^2$$

$$= \sum_{j=1}^{M-m} \left| (L_m w)_j \right|^2 = \| L_m w \|_2^2$$

$$L_m = \underbrace{\begin{pmatrix} \frac{F_{s_1} - f_1}{z_{s_1} - z_1} & \frac{F_{s_1} - f_2}{z_{s_1} - z_2} & \cdots & \\ \frac{F_{s_2} - f_1}{z_{s_2} - z_1} & \ddots & & \\ \vdots & & \ddots & \\ & & & \frac{F_{s_{M-m}} - f_m}{z_{s_{M-m}} - z_m} \end{pmatrix}}_{\text{col index: interpolation points}}$$

row indices are least-squares points.

The Loewner matrix

The AAA least-squares minimization problem is equivalent to,

$$\text{Compute } w \in \mathbb{C}^m \text{ such that } \|w\|_2 = 1 \text{ and } \|L_m w\|_2 \text{ is minimized}$$

where L_m is the *Loewner matrix*. With

$$S_m = \{s_1, \ldots, s_{M-m}\},$$

then

$$L_m \in \mathbb{C}^{(M-m)\times m}, \qquad (L_m)_{k,j} = \frac{F_{s_k} - f_j}{Z_{s_k} - z_j},$$

for $k = 1, \ldots, M - m$, and $j = 1, \ldots, m$.

The Loewner matrix

The AAA least-squares minimization problem is equivalent to,

$$\text{Compute } w \in \mathbb{C}^m \text{ such that } \|w\|_2 = 1 \text{ and } \|L_m w\|_2 \text{ is minimized}$$

$\shortparallel$ $\sigma_{min}(L_m)$

where L_m is the *Loewner matrix*. With

$$S_m = \{s_1, \ldots, s_{M-m}\},$$

then

$$L_m \in \mathbb{C}^{(M-m)\times m}, \qquad (L_m)_{k,j} = \frac{F_{s_k} - f_j}{Z_{s_k} - z_j},$$

for $k = 1, \ldots, M-m$, and $j = 1, \ldots, m$. I.e., w is a (unit-norm) minimal right-singular vector of L_m.

The AAA algorithm (Nonlinear approximation)

In summary, here are steps for the AAA algorithm:

Set $m = 0$, set $r(z) = 0$.
Initialize the Loewner matrix L_0 as an $M \times 0$ matrix.

1. Compute z_{m+1} as

$$j^* = \arg\max_j |F_j - r(Z_j)|, \qquad z_{m+1} = Z_{j*}$$

   and set $f_{m+1} = F_{j*}$.
2. Construct L_{m+1} by adding a column and removing a row.
   (Columns correspond to interpolation points, rows to the rest of the points.)
3. Compute $w \in \mathbb{C}^{m+1}$ as the minimal right-singular vector of L_{m+1}.
4. Construct r for $m+1$ using the new weights w and new point (z_{m+1}, f_{m+1}).
5. Terminate if $\max_j |F_j - r(Z_j)|$ is "small enough". (10^{-40} ...)
6. Otherwise, $m \leftarrow m + 1$ and repeat.