Polynomial Division from a Fourier Perspective

Tara Braswell

April 2025

Introduction

- Polynomial Multiplication = Convolution of Coefficient Sequences = pointwise multiplication of Fourier Transforms of Coefficient Sequences
- We can (probably) reverse this to do Polynomial Long Division in specific, and deconvolution in general
- It almost works. Where it doesn't is interesting.
- Why we need minimization algorithms like Split Bregman for deconvolution

Conventions

- for generalities sake, $[\cdot]$ indicates a finite sequence/vector/list, and $[\cdot]_{+k}$ indicates a list padded with k Zeros at the end.
- let $[P(x)]$ be the coefficient sequence for polynomial P(x)
- let $\mathcal{F}[P(x)]$ be the discrete Fourier transform of P(x) coefficients ($\mathcal{F}$ specifically indicating the matrix of appropriate size)
- for the sake of computational clarity, all polynomials are monic (highest degree coefficient is 1)
- we will assume that a is the constant in a linear term $(x + a)$, corresponding to a root $-a$

First issue:

- For DFT on compact domain, convolution does not map to multiplication. *Circular Convolution* maps to multiplication.
- sequence length mismatch: we want to divide polynomials of different lengths, pointwise. this is a pointwise operation

Solution: Zero Padding

- Putting zeros at the end of our divisor sequence solves both these issues.
- Zeros prevent 'wraparound' terms in circular convolution from adding additional info
- because we expect our quotient and divisor to be interchangeable, our quotient should have zeros after its transformed back (think last box in synthetic division)
- zero padding our dividend is like multiplying a polynomial by $(x + 0)$: the degree of the polynomial increases

synthetic-division.jpg

Fourier Transform of (x+a)

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_N & \omega_N^2 & \dots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^{(2)(2)} & \dots & \omega_N^{(2)(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_N^{N-1} & \omega_N^{(N-1)(2)} & \dots & \omega_N^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} 1 \\ a \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 1+a \\ 1+(\omega_N)a \\ 1+(\omega_N)^2 a \\ \vdots \\ 1+(\omega_N)^{N-1} a \end{bmatrix}$$

- ω_N is the N^{th} root of unity
- Higher degree polynomials can be calculated through multiplication of their linear factors
- explicit calculation of $\mathcal{F}[x-a]_{+k}$ is $\mathcal{O}(n)$. Calculating k of these and multiplying them together is $\mathcal{O}(kn)$. FFT on them all premultiplied is $\mathcal{O}(n \log n)$

Examples

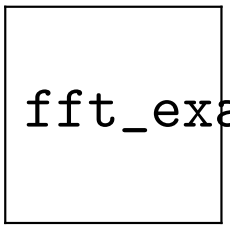
$DFT([x-a]_{+k})$:

- $P(x) = x$: $[P(x)]_{+k} = [1, 0, \ldots, 0]$, $\mathcal{F}[P(x)]_{+k} = [1, 1, \ldots, 1]$ (dual of delta function is a constant function)
- $P(x) = x + i$: $[P(x)]_{+1} = [1, i, 0]$
  $\mathcal{F}[P(x)]_{+1} = [1 + i, 1 + (\frac{\sqrt{3}}{2} - \frac{1}{2}i), 1 + (-\frac{\sqrt{3}}{2} - \frac{1}{2}i)]$
- $P(x) = x + i$: $[P(x)]_{+2} = [1, i, 0, 0]$, $\mathcal{F}[P(x)]_{+1} = [1 + i, 2, 1 - i, 0]$

Working through $\frac{x^2+1}{x+i}$

1. $\mathcal{F}[x^2+1] = [2, 1+(-\frac{1}{2} - \frac{\sqrt{3}}{2}i), 1+(-\frac{1}{2} + \frac{\sqrt{3}}{2}i)] = [2, \frac{1}{2} - \frac{\sqrt{3}}{2}i, \frac{1}{2} + \frac{\sqrt{3}}{2}i]$
2. $\frac{\mathcal{F}[x^2+1]}{\mathcal{F}[x+i]_{+1}} = \frac{[2, \frac{1}{2} - \frac{\sqrt{3}}{2}i, \frac{1}{2} + \frac{\sqrt{3}}{2}i]}{[1+i, \frac{2+\sqrt{3}}{2} - \frac{1}{2}i, \frac{2-\sqrt{3}}{2} - \frac{1}{2}i]} =$
   $[1 - i, 1 - \frac{\sqrt{3}}{2} + \frac{1}{2}i, 1 - \frac{\sqrt{3}}{2} + \frac{1}{2}i] = \mathcal{F}[x - i]_{+1}$

Computer World



Caveats

- division by (x+0) does not change anything. That's okay, and expected. Fourier tricks don't account for sequence length.
- As demonstrated prior, transform of divisor may have zeros. if $1 + a = 0$, first term will always be zero.

Pathologies at Roots of Unity

- whenever our divisor has a root at a root of unity, (ie, $a = -\omega_N^k$), we risk having a zero. (Conjecture I can't prove: we always will for N even)
- this can be changed by padding additional zeros, changing the roots of unity.
- This will not work if $a = -\omega_N^0 = -1$!

roots_1.png

Analytical justification

Why does dividing by x-1 cause so many problems?

- for polynomials P that are divisible by Q(x)=(x-1), both $(\mathcal{F}[P(x)])_1$ and $(\mathcal{F}[Q(x)]_{+1})_1$ equal zero. This is not an infinite value, this is some $\frac{0}{0}$ that cannot be defined without additional information. We may be able to formalize this with a distributional approach but it will not appear computationally (ie, principal values).
- Given scope of this problem, we know $\frac{0}{0}$ should be $1 + a_{n-2} + \cdots + a_0$. We can find this traditional division methods in this project, but we the Fourier domain is missing this information.
- this info may be recoverable through backwards evaluation of other Fourier coefficients, but this task increases in complexity as N does, and large enough Ns may leave this undetermined

Analytical justification pt 2

- We are essentially doing a contour integral around the unit circle an infinite number of times $\int_{\mathbb{R}} f(t)(e^{-2i\pi\xi t}dt) = \int_{\mathbb{R}} f(t)(e^{-2i\pi\xi t}dt) = \int_{\mathbb{R}} f(t)\frac{1}{-2i\pi\xi}de^{-2i\pi\xi t} = \oint_{t\in\mathbb{R}} \frac{1}{-2i\pi\xi} f(\frac{\ln\omega}{-2i\pi\xi})d\omega$
- poles off the unit circle will be collected as residues, poles on the unit circle break this integral
- DFT is defined as a function over the unit circle ($\mathbb{T}$), and (x-1) has a zero has on the unit circle.
- Hardy Spaces: polynomials are only integrable in L^p if $p < 1$. these norms are much easier to study on the unit circle/Hardy spaces, in which functions are only invertible on the unit circle if they have no zeroes on the unit circle.[**harmonic**]

Alternatives?

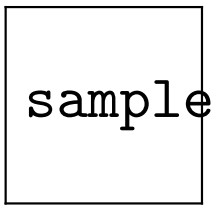
- Synthetic Division: actually faster ($\mathcal{O}(n)$), but only works for linear factors. Violates the spirit of this project.
- constraints: we know $([R]_{+1})_1 = 1, ([R]_{+1})_n = 0$. Works nicely if $R(x)$ is linear, but underdetermined for more.
- Split Bregman algorithm/general minimization techniques: most useful in application, but computationally intensive. also against the spirit of this project.
- Using polynomials/functions themselves: not $L^p(\mathbb{R})$ for $p \geq 1$. promising though, esp. considering Hardy space issues.

All Functions are Periodic if You Don't Care

2 parallel concepts:

- Polynomials are not in $L^p(\mathbb{R}), p \geq 1$, but the are in $L^p_{loc}(\mathbb{R})$
- Our "polynomial" examined on $[a, b]$ can be assumed periodic if we are outside $[a, b]$

So just sample the thing!



How to Sample (Even though it doesn't work)

For dividing $P(x)$ by $Q(x)$, $\deg(P) \geq \deg(Q)$

1. pick "enough" values of x_i,
2. evaluate $P(x_i), Q(x_i)$ and create sequences$[P(x_1), \ldots P(x_{n+1})]$,$[Q(x_1), \ldots Q(x_{n+1})]$.
3. Fourier, pointwise divide, inverse Fourier for new values at each x_i. call this new sequence $R = [R(x_1), \ldots R(x_{n+1})]$.
4. interpolate/curve fit polynomial. Given inevitable error and an overdetermined system, this will be a least squares argument.

"Enough" values is a little unclear, as Nyquist-Shannon bounds don't work (no guarantee function is even in $C^1([\min(x_i), \max(x_i)]$. Resulting value will inevitably be inaccurate (using sines to approximate a polynomial after all) and suffer from some aliasing as a result of this. (I really wanted this to work out better, but the idea came from a misreading of some CS lecture notes online)[**polynomial˙fft**]

A Quick Divergence on Split Bregman

- Split-Bregman:
  $[R(x)]^* = \arg\min ||\partial[R(x)]||_1 + \frac{\mu}{2}|||[Q(x)] * [R(x)] - [P(x)]||_2^2$
- (Split) Bregman algorithm fundamentally based on convex optimization/L^1 regularization for sparsity promotion.
- We dont actually need sparsity, we just need the last coefficient (or last several) to equal zero. Instead, we can optimize$[R(x)]^* = \arg\min |||[Q(x)] * [R(x)] - [P(x)]||_2^2$ subject to constraints. In theory, these constraints might not be necessary.
- Machine learning shines here-randomize values and backpropagate. I can't keep track of all the exact derivatives of these twisted values, but Machine learning
- Remember: all of this was just for division by (x-1)!

Conclusion

- Deconvolution is significantly harder than convolution, because the appearance of zeros in the Fourier domain are hard to avoid for some kernels.
- Zero padding our sequences solves a lot of problems and adds very little additional issues
- All our most beautiful algorithmic dreams are eventually replaced by least-squares arguments

bibliography