The Fast Fourier Transform

Andreas Klappenecker

Fourier Transform in Matrix Form

For a polynomial $A(x) = a_0 + a_1x + \ldots + a_{n-1}x^{n-1}$ of degree n-1, a conversion from coefficient representation to p.v. representation at n distinct points 1, ω, ..., ω^{n-1} can be done as follows:

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)^2} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

Fast Fourier Transform

Evaluate a degree n-1 polynomial $A(x) = a_0 + \ldots + a_{n-1}\, x^{n-1}$ at its n^{th} roots of unity: ω^0, ω^1 1, …, ω^{n-1}.

Divide. Divide the polynomial into even and odd powers.

$$A(x) = A_{even}(x^2) + x\, A_{odd}(x^2).$$

Conquer. Evaluate $A_{even}(x)$ and $A_{odd}(x)$ at the $\frac{1}{2}n^{th}$ roots of unity: $\nu^0, \nu^1, \ldots, \nu^{n/2-1}$.

Combine.

$$A(\omega^k) = A_{even}(\nu^k) + \omega^k A_{odd}(\nu^k), \quad 0 \le k < n/2$$

$$A(\omega^{k+n/2}) = A_{even}(\nu^k) - \omega^k A_{odd}(\nu^k), \quad 0 \le k < n/2$$

Fourier Transform in Matrix Form

Split into even and odd powers

$$\begin{pmatrix} A(1) \\ A(\omega) \\ \vdots \\ A(\omega^{n-1}) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)^2} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

Realize that 1, ω, …, $\omega^{n/2-1}$, -1, $-\omega$, …, $-\omega^{n/2-1}$ (upper/lower part)

Fast Fourier Transform

Evaluate a degree n-1 polynomial $A(x) = a_0 + \ldots + a_{n-1} x^{n-1}$ at its n^{th} roots of unity: ω^0, ω^1 1, …, ω^{n-1}.

Divide. Divide the polynomial into even and odd powers.

$$A(x) = A_{even}(x^2) + x\, A_{odd}(x^2).$$

Conquer. Evaluate $A_{even}(x)$ and $A_{odd}(x)$ at the $\frac{1}{2}n^{th}$ roots of unity: $\nu^0, \nu^1, \ldots, \nu^{n/2-1}$.

Combine.

$$A(\omega^k) = A_{even}(\nu^k) + \omega^k A_{odd}(\nu^k), \quad 0 \le k < n/2$$

$$A(\omega^{k+n/2}) = A_{even}(\nu^k) - \omega^k A_{odd}(\nu^k), \quad 0 \le k < n/2$$

Fast Fourier Transform

$$F_n = \begin{pmatrix} I & D \\ I & -D \end{pmatrix} \begin{pmatrix} F_{n/2} & 0 \\ 0 & F_{n/2} \end{pmatrix} P$$

$$D = \mathrm{diag}(1, \nu, \nu^2, \cdots, \nu^{n/2-1})$$

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ & & & \vdots & & & \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ & & & \vdots & & & \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}$$

FFT Algorithm

```
FFT(n, a_0,a_1,…,a_{n-1}) {
    if (n == 1) return a_0

    (e_0,e_1,…,e_{n/2-1}) ← FFT(n/2, a_0,a_2,a_4,…,a_{n-2})
    (d_0,d_1,…,d_{n/2-1}) ← FFT(n/2, a_1,a_3,a_5,…,a_{n-1})

    for k = 0 to n/2 - 1 {
        ω^k ← e^{2πik/n}
        y_k     ← e_k + ω^k d_k
        y_{k+n/2} ← e_k - ω^k d_k
    }

    return (y_0,y_1,…,y_{n-1})
}
```

Summary

- The FFT evaluates a polynomial of degree $n-1$ at n-th roots of unity in $O(n \log n)$ steps.
- Inverse of FFT just as fast.
- Can multiply two polynomials of degree $n-1$ in $O(n \log n)$ time using FFT of length $2n$.
- Find more details in CLRS or Kleinberg/Tardos.