EEECS 205003 Session 29

Fast Fourier transform

Fast Fourier transform (FFT) revolutionalize signal processing

Basic idea

Speed up multiplication by F & F^{-1}, where F is the Fourier matrix

Q: How fast ?

For $n \times n$ F, $F\mathbf{c}$ uses n^2 multiplications FFT needs only $\frac{1}{2}n \log n$

Discrete Fourier transform (DFT)

A Fourier series is a way of writing a periodic function or signal as a combination of functions of different frequency

$f(x) = a_0 + a_1 cosx + b_1 sinx + a_2 cos2x + b_2 sin2x + \cdots$

when working with finite data sets, DFT is key to this decomposition:

$$y_l = \sum_{k=0}^{n-1} c_k e^{i\frac{2\pi}{n}kl}$$

In matrix form

$\mathbf{y} = F_n \mathbf{c}$

where $F_n = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{n-1} \\ 1 & w^2 & w^4 & \ddots & w^{2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & w^{n-1} & w^{2(n-1)} & \cdots & w^{(n-1)^2} \end{bmatrix}$ (Fourier matrix)

& $w = e^{\frac{i2\pi}{n}}$ or $w^n = 1$

Note 1: In EE & CS, rows & columns of a matrix often starts with 0 (not 1) and ends at $n-1$ (not n), we follow this convention here

Note 2: $F_n = {F_n}^T$ so F_n is symmetric (Not Hermitian !)

Note 3: $(F_n)_{jk} = w^{jk}$

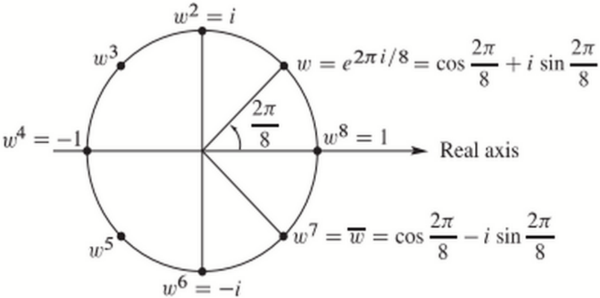
where $w = e^{\frac{i2\pi}{n}}$ and $w^n = 1$

$\Rightarrow$ All entries of F_n are on the unit circle in the complex plane

We can write

$w = \cos\left(\frac{2\pi}{n}\right) + i\sin\left(\frac{2\pi}{n}\right)$

(But harder to compute)



Note 4: columns of F_n are orthogonal

Fourier matrix: n=4

$w^4 = 1 \Rightarrow w = e^{\frac{i2\pi}{4}} = i$

$$F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & i^2 & i^3 \\ 1 & i^2 & i^4 & i^6 \\ 1 & i^3 & i^6 & i^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$$

(easy to check that columns of F_4 are orthogonal)

F_4 is not yet unitary $\because$ length of column=2

$\Rightarrow (\frac{1}{2}F_4)^H(\frac{1}{2}F_4) = I$ or $F_4^H F_4 = 4I$

$\Rightarrow F_4^{-1} = \frac{1}{4}F_4^H = \frac{1}{4}\overline{F_4}$ $(F_4^T = F_4)$

Once we know F, we get F^{-1} so when FFT gives a quick way to multiply by F, it does the same for F^{-1} ($F_n^{-1} = \frac{1}{n}\overline{F_n}$ in general)

4-point Fourier series

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = F_4\mathbf{c} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & w & w^2 & w^3 \\ 1 & w^2 & w^4 & w^6 \\ 1 & w^3 & w^6 & w^9 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Input: four complex DFT coefficient c_0, c_1, c_2, c_3

Output: four function values y_0, y_1, y_2, y_3

An example:

with DFT coefficient $(1, 0, 0, 0)$

$$\mathbf{y} = F_4\mathbf{c} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\mathbf{c} = F_4^{-1}\mathbf{y} = \frac{1}{4}\overline{F_4}\mathbf{y} = \frac{1}{4}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -i & -1 & i\\ 1 & -1 & 1 & -1\\ 1 & i & -1 & -i\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$$

Fast Fourier transform (one step)

Motivation: Normally $\mathbf{y} = F_n\mathbf{c}$ takes n^2 separate multiplications

We want to speed up the process

Observation 1:

If a matrix has many zeros, many multiplications can be skipped

But Fourier matrix has NO zeros !

Observation 2:

F_n has the special pattern of w^{jk} for its entries

Q: Can we use this to speed up computation ?

Yes! F_n can be factored in a way that produces many zeros

This is FFT !

Key idea

Connect F_n with $F_{\frac{n}{2}}$

Assume that n is a power of 2

There is a nice relationship between F_n & $F_{\frac{n}{2}}$: (based on $w_{2n}^2 = w_n$)

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

where D is a diagonal matrix with entries $(1, w, \cdots, w^{\frac{n}{2}-1})$

P is a $n \times n$ permutation matrix that puts the even columns ahead of odd columns

Ex: n=4

$$F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}, \begin{bmatrix} F_2 & \\ & F_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & & \\ 1 & i^2 & & \\ & & 1 & 1 \\ & & 1 & i^2 \end{bmatrix}$$

$$F_4 = \begin{bmatrix} 1 & & 1 & \\ & 1 & & i \\ 1 & & -1 & \\ & 1 & & -i \end{bmatrix} \begin{bmatrix} 1 & 1 & & \\ 1 & i^2 & & \\ & & 1 & 1 \\ & & 1 & i^2 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & & 1 & \\ & 1 & & \\ & & & 1 \end{bmatrix}$$

(sparse) (half zeros) (sparse)

Note:

$$\begin{bmatrix} 1 & & & \\ & & 1 & \\ & 1 & & \\ & & & 1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_2 \\ c_1 \\ c_3 \end{bmatrix}$$

then apply F_2 and F_2 on the evens & odds

complexity reduction:

multiplied by two size $\frac{n}{2}$ Fourier matrix requires $2(\frac{n}{2})^2 = \frac{1}{2}n^2$

multiplications+multiplication of two sparse matrix P & $\begin{bmatrix} I & D \\ I & -D \end{bmatrix}$

requires order n operations

total complexity $\cong \frac{1}{2}n^2$ operations

The full FFT by recursion

$F_n \to F_{\frac{n}{2}} \to F_{\frac{n}{4}} \to F_{\frac{n}{8}} \to \cdots$

Ex: n=1024

$$F_{1024} = \begin{bmatrix} I_{512} & D_{512} \\ I_{512} & -D_{512} \end{bmatrix} \begin{bmatrix} F_{512} & \\ & F_{512} \end{bmatrix} \begin{bmatrix} even \\ odd \\ perm \end{bmatrix}$$

$$\begin{bmatrix} F_{512} & \\ & F_{512} \end{bmatrix} = \begin{bmatrix} I & D & & \\ I & -D & & \\ & & I & D \\ & & I & -D \end{bmatrix} \begin{bmatrix} F & & & \\ & F & & \\ & & F & \\ & & & F \end{bmatrix}$$

$$\begin{bmatrix} \text{pick } 0, 4, 8, \cdots \\ \text{pick } 2, 6, 10, \cdots \\ \text{pick } 1, 5, 9, \cdots \\ \text{pick } 3, 7, 11, \cdots \end{bmatrix}$$

where $F = F_{256}$, $D = D_{256}$

Complexity:

$n^2 \rightarrow \frac{1}{2} n \log n$

Reason: Let $l = \log n \Rightarrow n = 2^l$

there are a total of l levels

$(\underbrace{F_n \rightarrow F_{\frac{n}{2}} \rightarrow F_{\frac{n}{4}} \rightarrow \cdots \rightarrow F_1}_{total \ \ level = l})$

For each level

$$F_n = \begin{bmatrix} I & D \\ I & -D \end{bmatrix} \begin{bmatrix} F_{\frac{n}{2}} & \\ & F_{\frac{n}{2}} \end{bmatrix} P$$

$\frac{n}{2}$ multiplications from the diagonal D's

so a total of $\frac{n}{2} \log n$ operations

A typical case $n = 1024$, $(1024)^2 \to \frac{1}{2}(1024)(10)$

This is 200 times faster !

(This is possible because F_n's are special matrices with orthogonal columns !)