EE 203001 SES # 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, $F\underline{c}$ 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 fcn or signal as a comb. of fcns of diff. freq.

$$f(x) = a_0 + a_1 \cos x + b_1 \sin x + a_2 \cos 2x + b_2 \sin 2x + \cdots$$

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

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

In matrix form

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

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

(Fourier matrix)

& $w = e^{i 2\pi / n}$ or $w^n = 1$

Note 1: In EE & CS, rows & cols 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 $w = e^{i 2\pi / n}$ and $w^n = 1$

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

We can write

$$w = \cos(2\pi/n) + i\sin(2\pi/n)$$

(But harder to compute)

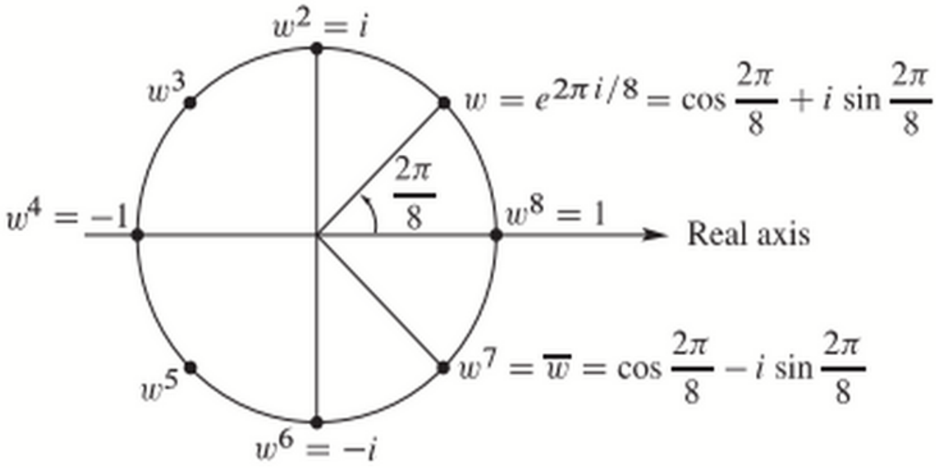

Figure 62: The eight solutions to $z^8 = 1$ are $1, w, w^2, \ldots, w^7$ with $w = (1+i)/\sqrt{2}$.

Note 4: col.s of F_n are orthogonal

<u>Fourier matrix: $n = 4$</u>

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

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

(easy to chk that col.s of F_4 are orthogonal)

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

$$\Rightarrow \left(\tfrac{1}{2}F_4\right)^H\left(\tfrac{1}{2}F_4\right) = I \text{ or } F_4^H F_4 = 4I$$

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

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

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

Input: four complex DFT coeff.

c_0, c_1, c_2, c_3

Output: four fcn values

y_0, y_1, y_2, y_3

An example:

with DFT coeff. $(1, 0, 0, 0)$

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

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

Fast Fourier transform (one step)

Motivation: Normally $\underline{y} = F_n \underline{c}$ 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 $F_{n/2}$

Assume that n is a power of 2

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

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

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

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

Ex: $n=4$

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

$$F_4 = \underbrace{\begin{bmatrix} 1 & & 1 & \\ & 1 & & i \\ 1 & & -1 & \\ & 1 & & -i \end{bmatrix}}_{\text{(sparse)}} \underbrace{\begin{bmatrix} 1 & 1 & & \\ 1 & i^2 & & \\ & & 1 & 1 \\ & & 1 & i^2 \end{bmatrix}}_{\text{(half zeros)}} \underbrace{\begin{bmatrix} 1 & & & \\ & & 1 & \\ & 1 & & \\ & & & 1 \end{bmatrix}}_{\text{(sparse)}}$$

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 $n/2$ Fourier matrix

requires $2(n/2)^2 = \frac{1}{2}n^2$ multiplications

+ muliplication of two sparse matrix

P & $\begin{bmatrix} I & D \\ I & -D \end{bmatrix}$ requires order n operations

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

The full FFT by recursion

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

Ex: $n = 1024$

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

$$\begin{bmatrix} F_{512} & \\ & F_{512} \end{bmatrix} = \begin{bmatrix} I & D & & \\ I & -D & & \\ & & I & D \\ & & I & -D \end{bmatrix} \begin{bmatrix} F & & & \\ & F & & \\ & & F & \\ & & & F \end{bmatrix} \cdot \begin{bmatrix} \text{pick } 0, 4, 8, \cdots \\ \text{pick } 2, 6, 10, \cdots \\ \text{pick } 1, 5, 9, \cdots \\ \text{pick } 3, 7, 11, \cdots \end{bmatrix}$$

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

Complexity:

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

Reason: Let $\ell = \log n \Rightarrow n = 2^{\ell}$

there are a total of ℓ levels

$$\left(\underbrace{F_n \to F_{n/2} \to F_{n/4} \to \cdots \to F_1}_{\text{total level } = \ell} \right)$$

For each level

$$F_n = \underbrace{\begin{bmatrix} I & D \\ I & -D \end{bmatrix} \begin{bmatrix} F_{n/2} & \\ & F_{n/2} \end{bmatrix}}_{\frac{n}{2} \text{ multiplications}} P$$

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

A typical case $n = 1024$,

$$(1024)^2 \longrightarrow \frac{1}{2}(1024)\cdot(10)$$

This is 200 times faster!

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