Coste Computacional DFT

Sea la DFT de una secuencia x[n] de longitud N:

$$X[k]=\sum_{n=0}^{N-1} x[n]\cdot e^{-i\frac{2\pi}{N}nk}$$

$$W_{kn}=e^{-i\frac{2\pi}{N}nk} \qquad \begin{pmatrix} X[0] \\ X[1] \\ \dots \\ X[k] \\ \\ \end{pmatrix} = \begin{pmatrix} W_{00} & W_{01} & W_{02} & \\ W_{10} & W_{11} & & \\ W_{20} & W_{21} & & \\ & & & W_{kn} \\ & & & \end{pmatrix} \cdot \begin{pmatrix} x[0] \\ x[1] \\ \dots \\ x[n] \\ \\ \end{pmatrix}$$

DFT puede escribirse como la aplicación de una matriz W_{nk} al vector con la secuencia x[n] ~ del orden de N^2 operaciones

Algoritmo FFT (Fast Fourier Transform)

Sea la DFT de una secuencia x[n] de longitud N:

$$\mathrm{X}[\mathrm{k}]=\sum_{\mathrm{n}=0}^{\mathrm{N}-1} \mathrm{x}[\mathrm{n}]\cdot \mathrm{e}^{-\mathrm{i}\frac{2\pi}{\mathrm{N}}\mathrm{nk}}$$

Supongamos que N es par:

$$\mathrm{X}[\mathrm{k}]=\sum_{\mathrm{n}=0}^{\mathrm{N}-1} \mathrm{x}[\mathrm{n}]\cdot \mathrm{e}^{-\mathrm{i}\frac{2\pi}{\mathrm{N}}\mathrm{nk}} = \sum_{\mathrm{n\_pares}} + \sum_{\mathrm{n\_impares}}$$

Los n pares en [0, N-1] = 2·m con m=0, 1, …, N/2-1

Los n impares en [0, N-1] = 2m+1 con m=0,1,…, N/2-1

Algoritmo FFT (Fast Fourier Transform)

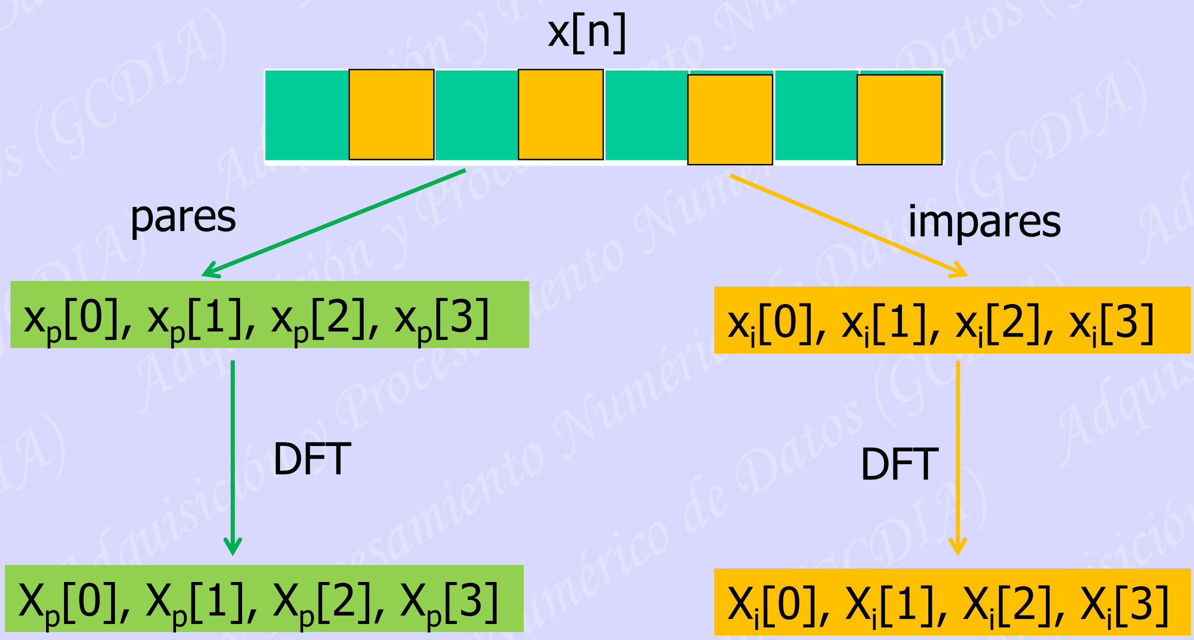
$$X[k] = \sum_{n\_pares} + \sum_{n\_impares}$$

$$X[k] = \sum_{m=0}^{N/2-1} x[2m] \cdot e^{-i\frac{2\pi}{N}(2m)k} + \sum_{m=0}^{N/2-1} x[2m+1] \cdot e^{-i\frac{2\pi}{N}(2m+1)k}$$

$$X[k] = \boxed{\sum_{m=0}^{N/2-1} x_p[m] \cdot e^{-i\frac{2\pi}{N/2}mk}} + e^{-i\frac{2\pi}{N}k} \cdot \boxed{\sum_{m=0}^{N/2-1} x_i[m] \cdot e^{-i\frac{2\pi}{N/2}mk}}$$

DFT ($x_p[m]$ de tamaño N/2) DFT ($x_p[m]$ de tamaño N/2)

Ejemplo para secuencia de longitud N=8



Ejemplo para secuencia de longitud N=8

$X_p[0], X_p[1], X_p[2], X_p[3]$

$X_i[0], X_i[1], X_i[2], X_i[3]$

$$X[0] = X_p[0] + e^{-i\frac{2\pi}{N}0} X_i[0] \qquad X[0] = X_p[0] + e^{-i\frac{2\pi}{N}0} X_i[0]$$

$$X[1] = X_p[1] + e^{-i\frac{2\pi}{N}1} X_i[1] \qquad X[1] = X_p[1] + e^{-i\frac{2\pi}{N}1} X_i[1]$$

$$\ldots \qquad \ldots$$

$$X[4] = X_p[4] + e^{-i\frac{2\pi}{N}4} X_i[4] \qquad X[4] = X_p[0] + e^{-i\frac{2\pi}{N}4} X_i[0]$$

$$X[5] = X_p[5] + e^{-i\frac{2\pi}{N}5} X_i[5] \qquad X[5] = X_p[1] + e^{-i\frac{2\pi}{N}5} X_i[1]$$

$$\ldots \qquad \ldots$$

Ejemplo para secuencia de longitud N=8

$$\begin{aligned}
X[0] &= X_p[0] + e^{-i\frac{2\pi}{N}0} X_i[0] \\
X[1] &= X_p[1] + e^{-i\frac{2\pi}{N}1} X_i[1] \\
X[2] &= X_p[2] + e^{-i\frac{2\pi}{N}2} X_i[2] \\
X[3] &= X_p[3] + e^{-i\frac{2\pi}{N}3} X_i[3] \\
X[4] &= X_p[0] + e^{-i\frac{2\pi}{N}4} X_i[0] \\
X[5] &= X_p[1] + e^{-i\frac{2\pi}{N}5} X_i[1] \\
X[6] &= X_p[2] + e^{-i\frac{2\pi}{N}6} X_i[2] \\
X[7] &= X_p[3] + e^{-i\frac{2\pi}{N}7} X_i[3]
\end{aligned}$$

Tamaño N

$$\begin{pmatrix} X[k] \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} X_p[k] \end{pmatrix} \\ \begin{pmatrix} X_p[k] \end{pmatrix} \end{pmatrix} + \begin{pmatrix} e^{-i\frac{2\pi}{N}k} \end{pmatrix} \cdot \begin{pmatrix} \begin{pmatrix} X_i[k] \end{pmatrix} \\ \begin{pmatrix} X_i[k] \end{pmatrix} \end{pmatrix}$$

Tamaño N/2

ADQUISIÓN Y PROCESAMIENTO NUMÉRICO de DATOS / PROCESAMIENTO DIGITAL de la SEÑAL ANTONIO TABERNERO GALÁN

Algoritmo FFT (Fast Fourier Transform)

Hacer una DFT de tamaño N $X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-i\frac{2\pi}{N}nk}$

es equivalente a hacer 2 DFT de tamaño N/2

$$X[k] = \boxed{\sum_{m=0}^{N/2-1} x_p[m] \cdot e^{-i\frac{2\pi}{N/2}mk}} + e^{-i\frac{2\pi}{N}k} \cdot \boxed{\sum_{m=0}^{N/2-1} x_i[m] \cdot e^{-i\frac{2\pi}{N/2}mk}}$$

+ N operaciones extra (suma y multiplicación por $e^{-i\frac{2\pi}{N}k}$)

Algoritmo FFT (Fast Fourier Transform)

En operaciones: DFT(N) = 2·DFT(N/2) + N operaciones

Si N es múltiplo de 4, N/2 es par y puedo repetir el proceso de separar cada DFT en muestra parres/impares, etc:

DFT(N/2) = 2·DFT(N/4) + N/2 operaciones

Y por lo tanto:

DFT(N) = 2·DFT(N/2) + N
= 2·(2·DFT(N/4) + N/2) + N
= 4·DFT(N/4) + N + N

Algoritmo FFT (Fast Fourier Transform)

Si N es potencia de 2, $N=2^P$, puedo seguir dividiendo:

DFT(N) = N/2·DFT(2) + N + N + … N

P-1 pasos

Una DFT de tamaño 2 son 2 operaciones: $X[0]=x[0]+x[1]$

$X[1]=x[0]-x[1]$

DFT(N) = N/2·2 + N + N + … + N operaciones

= P· N = $N \cdot \log_2(N)$

Posiblemente necesite una constante para reflejar el hecho de las operaciones con complejos, pero lo importante es que el coste computacional es proporcional a $N \cdot \log_2(N)$, no N^2

Convolución a través de la FFT

Convolución de 2 secuencias de longitud N: z[n]=x[n]*y[n]

z[n]=x[n]*y[n] → coste computacional del orden de N^2

x[n] → $N \cdot \log_2(N)$ → X[k]

y[n] → $N \cdot \log_2(N)$ → Y[k] tiene un coste de

Z[k]=X[k]·Y[k] (N multiplicaciones)

z[n] ← $N \cdot \log_2(N)$ ← Z[k]

En total $3 \cdot N \cdot \log_2(N) + N = 3 \cdot N \cdot (\log_2(N)+1)$ operaciones

Para $N=8192=2^{13}$, el ratio es $N/ 3 \cdot (\log_2(N)+1) = 8192/40$, del orden de 200 veces más rápido.

Otras aplicaciones

				1	2	3	4
			x	5	6	7	8
				8	16	24	32
			7	14	21	28	
		6	12	18	24		
+	5	10	15	20			
	5	16	34	60	61	52	32
	7	0	0	6	6	5	2

Acarreando las decenas al siguiente número

Otras aplicaciones

				1	2	3	4
			x	5	6	7	8
				8	16	24	32
			7	14	21	28	
		6	12	18	24		
+	5	10	15	20			
	5	16	34	60	61	52	32

Convolución de las secuencias

x=[1 2 3 4], y=[5 6 7 8]

conv(x,y) =

5 16 34 60 61 52 32

Nuestro algoritmo para multiplicar 2 número “largos” es equivalente a hacer una convolución.

Puede hacerse mucho más rápido a través de la FFT (si el nº de cifras de los números involucrados es grande)