Linear transforms

compute

$$y = Tx$$

where x is the input vector, y the output vector and T the transform matrix.

Example: DFT

1. form (standard in signal processing): given $x_0, \dots, x_{n-1}$

compute $y_k = \sum_{\ell=0}^{n-1} e^{-2\pi i k\ell/n} x_\ell$, for $k = 0 \dots n-1$ $\quad \boxed{i = \sqrt{-1}}$

$$= \sum_{\ell=0}^{n-1} \omega_n^{k\ell} x_\ell, \text{ for } k = 0 \dots n-1, \quad \omega_n = e^{-\frac{2\pi i}{n}}$$

primitive nth root of 1

2. form (we will use): $x = (x_0, \dots, x_{n-1})^T$ is given

compute $y = DFT_n x$, $\quad DFT_n = [\omega_n^{k\ell}]_{0 \le k,\ell < n}$

$[y = (y_0, \dots, y_{n-1})^T$ is the output$]$

Examples:

$$DFT_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad DFT_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$$

How many complex ops for DFT_4?

Transform algorithms

an algorithm for $y = Tx$ is given by a factorization

$$T = T_1 T_2 \cdots T_m$$

Namely, instead of $y = Tx$ you can compute

$$\left.\begin{aligned} t_1 &= T_m x \\ t_2 &= T_{m-1} t_1 \\ &\dots \\ y &= T_1 \cdot t_{m-1} \end{aligned}\right\} \text{ m steps}$$

This reduces the op count only if

- the T_i are sparse
- m is not too large

Note: For generic T, $y = Tx$ is $\mathcal{O}(n^2)$

Example: Cooley-Tukey FFT for $n = 4$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & i \end{bmatrix} \begin{bmatrix} 1 & 1 & & \\ 1 & -1 & & \\ & & 1 & 1 \\ & & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Compare (complex) cost:

by definition: 12 adds, 4 mults by i

using FFT: 8 adds, 1 mult by i

The sparse matrices are structured:

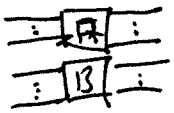
$$DFT_4 = (DFT_2 \otimes I_2)\, \mathrm{diag}(1,1,1,i)(I_2 \otimes DFT_2)\, L^4_2$$

(explained next)

→ back to slides

Structured matrices

- $DFT_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
- $I_n = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix}$
- $\mathrm{diag}(a_0, \ldots, a_{n-1}) = \begin{pmatrix} a_0 & & \\ & \ddots & \\ & & a_{n-1} \end{pmatrix}$
- $A \oplus B = \begin{pmatrix} A & \\ & B \end{pmatrix}$



- $A \otimes B = [a_{k,\ell} \cdot B]_{k,\ell}$ where $A = [a_{k,\ell}]_{k,\ell}$

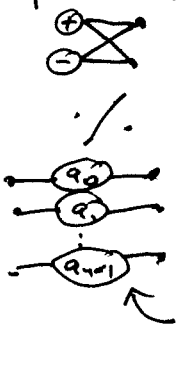
most important:

$I_n \otimes A = \begin{pmatrix} A & & \\ & \ddots & \\ & & A \end{pmatrix}$ contains n A's

$A \otimes I_n = \begin{pmatrix} a_{0,0} I_n & \cdots & \\ \vdots & \ddots & \\ & & \end{pmatrix}$ ($a_{k,\ell} \cdot I_n$) contains n A's

e.g., all bullets together constitute one A

data flow (right to left)



- L^n_k: stride permutation matrix

more later