Linear transform

compute

$$y = Tx$$

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

Example: DFT

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

$$y_k = \sum_{\ell=0}^{n-1} e^{-2\hat{\imath}jk\ell/n} \cdot x_\ell, \quad \text{for } 0 \le k < n, \qquad \boxed{\hat{\imath} = \sqrt{-1}}$$

2. form: set $\omega_n = e^{-2\hat{\imath}j/n}$ (primitive, nth root of 1)

$$y_k = \sum_{\ell=0}^{n-1} \omega_n^{k\ell} x_\ell, \quad \text{for } 0 \le k < n$$

3. form: $x = (x_0, \dots, x_{n-1})^T$, $y = (y_0, \dots, y_{n-1})^T$, compute

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

For example:

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

Transform algorithms

you want to compute $y = Tx$, T is $n \times n$.
an algorithm is given by a factorization

$$T = T_1 T_2 \cdots T_m, \qquad \text{all } T_i \text{ are } n \times n$$

namely you can compute y as:

$$t_1 = T_m x$$
$$t_2 = T_{m-1} x \qquad (m \text{ steps} = m \text{ MVMs})$$
$$\vdots$$
$$y = T_1 \cdot t_{m-1}$$

this reduces the operations count only if

- the T_i are sparse
- m is not too large

Note: generic MVM (i.e., matrix is not known beforehand) has complexity $\mathcal{O}(n^2)$

Example: fast Fourier transform (FFT) for $n=4$

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

compare cost:
- by definition: 12 adds, 4 mults by j (all complex)
- using FFT (4 steps): 8 adds, 1 mult by j

the sparse matrices are structured:

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

(explained next)

Structured matrices

- $DFT_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ (butterfly matrix)
- $I_n = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix}$
- $\mathrm{diag}(a_0, \dots, a_{n-1}) = \begin{pmatrix} a_0 & & \\ & \ddots & \\ & & a_{n-1} \end{pmatrix}$
- $A \oplus B = \begin{pmatrix} A & \\ & B \end{pmatrix}$
- $A \otimes B = [a_{k,\ell} \cdot B]_{0 \le k,\ell < n}$ if $A = [a_{k,\ell}]_{0 \le k,\ell < n}$

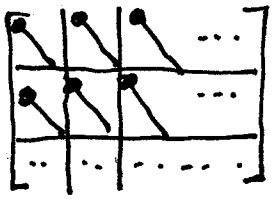
  A $n \times n$, B $m \times m$ $\Rightarrow$ $A \otimes B$ $nm \times nm$

Example:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \otimes DFT_2 = \left[\begin{array}{cc:cc} 1 & 1 & 2 & 2 \\ 1 & -1 & 2 & -2 \\ \hdashline 3 & 3 & 4 & 4 \\ 3 & -3 & 4 & -4 \end{array}\right]$$

coarse structure ↑ fine structure ↑

$I_n \otimes A = \begin{bmatrix} A & & \\ & A & \\ & & \ddots \\ & & & A \end{bmatrix}$ contains n As

$A \otimes I_n =$ 

- each block is an $a_{i,e} \cdot I_n$
- all circles together constitute one A
- contains n As