Convolution

Input: vectors $(a_0, a_1, \ldots, a_{n-1})$
$(b_0, b_1, \ldots, b_{n-1})$

Output: vector $c = a * b$, where

$$c_k = \sum_{i=0}^{k} a_i b_{k-i} = \sum_{\substack{0 \le i, j \le n-1 \\ i+j=k}} a_i b_j$$

$c_0 \quad c_1 \quad c_2$

$$\begin{array}{cccccc} a_0 b_0 & a_0 b_1 & a_0 b_2 & \cdots & & a_0 b_{n-1} \\ a_1 b_0 & a_1 b_1 & a_1 b_2 & \cdots & & a_1 b_{n-1} \\ a_2 b_0 & a_2 b_1 & a_2 b_2 & \cdots & & a_2 b_{n-1} \\ \vdots & & & & & \\ a_{n-1} b_0 & a_{n-1} b_0 & \cdots & & & a_{n-1} b_{n-1} \end{array}$$

c_{2n-2}

Why do we care

- polynomial mult
- signal processing

replace $(a_0, a_1, \ldots, a_{n-1})$ with

$$a_i' = \frac{1}{Z} \sum_{j=i-k}^{i+k} a_j \, e^{-(j-i)^2}$$

$$A(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_{n-1} x^{n-1}$$

$$B(x) = b_0 + b_1 x + b_2 x^2 + \dots + b_{n-1} x^{n-1}$$

$$C(x) = A(x) \cdot B(x) = c_0 + c_1 x + \dots + c_{2n-2} x^{2n-2}$$

where $c = (a_0, a_1, \dots, a_{n-1}) * (b_0, b_1, \dots, b_{n-1})$

We focus on computing $C(x)$ given $A(x)$ and $B(x)$

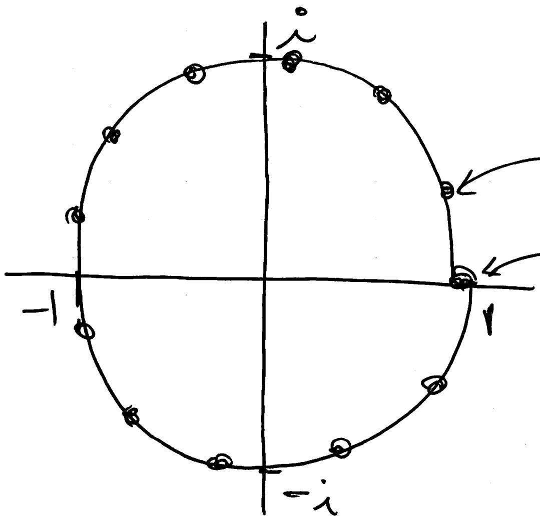
Goal: beat trivial $O(n^2)$ algo

Idea:

- pick $2n$ values $x_1, \dots, x_{2n}$
- evaluate $A(x_1) \dots A(x_{2n})$
  $B(x_1) \quad B(x_{2n})$
- compute $C(x_1) \quad C(x_{2n})$
- recover $C(x)$ from $\{C(x_i) : i = 1, \dots 2n\}$

Choosing the right values to sample

Let $w(j, 2n) = e^{\frac{j}{n}\pi i}$ be the 2nth roots of unit that is the 2n solutions to $z^{2n} = 1$



$e^{\frac{1}{n}\pi i}$

$e^{\frac{0}{n}\pi i} = 1$

<u>Obs</u>: $w^2(j, 2n) = w(j, n)$

~~proof~~ $\left[e^{\frac{j}{n}\pi i}\right]^2 = e^{\frac{2j}{n}\pi i} = e^{\frac{j}{n/2}\pi i}$

Evaluating all roots in one go

★ $A(x) = A_{even}(x^2) + x\, A_{odd}(x^2)$, where

$A_{even}(x) = a_0 + a_2 x + a_3 x^2 + \dots\; a_{n-2}\, x^{\frac{n}{2}-1}$

$A_{odd}(x) = a_1 + a_3 x + a_5 x^2 + \dots\; a_{n-1}\, x^{\frac{n}{2}-1}$

To compute $A(\omega(0,2n))$ $A(\omega(1,2n))$ ... $A(\omega(2n-1,2n))$

recursively compute

$A_{even}(\omega(0,n))$ $A_{even}(\omega(1,n))$ ... $A_{even}(\omega(n-1,n))$

$A_{odd}(\omega(0,n))$ $A_{odd}(\omega(1,n))$ ... $A_{odd}(\omega(n-1,n))$

Combine then with ★ to get desired output

$T(n) = 2T(n/2) + O(n) \implies T(n) = O(n \log n)$

How to reconstruct $C(x)$

$C(x) = c_0 + c_1 x + \ldots + c_{2n-2} x^{2n-2}$

$$D(x) = \sum_{s=0}^{2n-1} C(\omega(s, 2n))\, x^s$$

$D(x) = d_0 + d_1 x + \ldots + d_{2n-2} x^{2n-2}$

<u>Claim:</u> $\frac{1}{2n} D(\omega(2n-s, 2n)) = c_s$ $\Rightarrow$ compute C by evaluate D

$$D(\omega(j, 2n)) = \sum_{s=0}^{2n-1} C(\omega(s, 2n)) (\omega(j, 2n))^s$$

$$= \sum_{s=0}^{2n-1} \sum_{t=0}^{2n-1} c_t (\omega(s, 2n))^t (\omega(j, 2n))^s$$

$$= \sum_s \sum_t c_t\, e^{\frac{ts}{n}\pi i + \frac{s \cdot j}{n}\pi i}$$

$$= \sum_s \sum_t c_t\, \omega^s(t+j, 2n)$$

$$= \sum_{t=0}^{2n-1} c_t \left(\sum_{s=0}^{2n-1} \omega^s(t+j, 2n) \right)$$

$= 0$ for $\omega(t+j, 2n) \neq 1$ since $x^{2n} - 1 = 0$

$= 2n$ for $t = 2n - j$

Complete Algorithm

$$
\left.\begin{array}{llll}
\text{evaluate } A(x) & \text{at } x = \omega(j, 2n) & \text{for } j = 1 .. 2n \\
\text{evaluate } B(x) & \text{at } x = \omega(j, 2n) & \text{for } j = 1, .. 2n
\end{array}\right\} O(n \log n)
$$

$$
\text{compute } C(x) \quad \text{at } x = \omega(j, 2n) \text{ for } j = 1 .. 2n \; \} \; O(n)
$$

$$
\text{evaluate } D(x) \quad \text{at } x = \omega(j, 2n) \text{ for } j = 1, .. 2n \; \} \; O(n \log n)
$$

$$
\text{let } c_s = \frac{D(\omega(2n-s), 2n)}{2n} \quad \text{for } s = 0, \ldots 2n-1 \; \} \; O(n)
$$

$$
\Longrightarrow \quad O(n \log n) \text{ time}
$$

to compute convolution