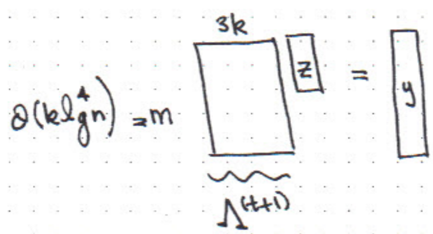
Lecture #3: Sublinear Sparse Fourier Algorithms

- All of the previous algos. we discussed were inefficient. Yes, they are sample efficient but not computationally; we take considerably fewer samples than the full signal but then do lots of FFTs! All to return only k items! Very inefficient (unless your application is really only constrained by sampling time/power space/...).
- let's look at architecture of CoSaMP to understand the bottlenecks and what we should change:

① $r = y - \Phi x^{(t)} = \Phi(x - x^{(t)})$ = sample from residual signal
or
sample from current approx$^{\underline{n}}$ and subtract from measurements.

Don't compute $\Phi x^{(t)}$ explicitly!

② $$w^{(t+1)} = \underset{\substack{z \in \mathbb{C}^n \\ \operatorname{supp} z \subset \Lambda^{(t+1)}}}{\operatorname{argmin}} \left\| y - \Phi z \right\|_2$$



solve this over-determined system. It's small!

OR — knowing $\Lambda^{(t+1)}$ = 3k-term supp set, estimate ~~find~~ the vals of the entries of z on that support set.

③ $\operatorname{supp}_{2k}(\Phi^* r)$

these are our two big bottlenecks

given samples (of residual), how to find/identify top 2k entries WITHOUT computing $\Phi^* r$

- This is actually revisionist history! SFT algos have been around for a while, just in slightly different form.

1991 Kushilevitz & Mansour (ML)
1989/1993 Goldreich & Levin (Crypto/codes)
] Hadamard transform (Fourier analysis on Boolean cube)

1992 Kushilevitz (ML)
2002 GGIMS (streaming/sublinear algs)
2003 Akavia, Goldwasser, Safra (crypto.)
2005 Gilbert, Muthukrishnan, Strauss
]
- Fourier analysis on $\mathbb{Z}_n$ for various flavors of n (n prime, $n = 2^\ell, \ldots$)
- Randomized algos with constant prob. of error (i.e., per signal)
- ℓ_2/ℓ_2 results

running time (+ # samples) $= O\left(k \lg^{O(1)}(n)\right)$

2010 – onwards LOTS OF WORK!
big flurry of algos, implementations, and hardware.
Det'ic constructions, too.

See 2015 SP Mag survey, with Iwen, Indyk, Schmidt.
most prominent one is HIKP2012b = Hassaneh, Indyk, Katabi, Price.

- Let's outline basic components & techniques (that don't suffer from the bottlenecks).

① To get intuition, let's assume spectrum of x (on $\mathbb{Z}_n$ ~~[illegible]~~) consists of a SINGLE non-zero freq.
  - two samples are sufficient to get position and value/coeff. of freq.
  - $\lg(n)$ samples suffice if tone + noise.

i.e., build a filterbank

② If we have [illegible] freq. in small [illegible] subsets of Fourier space

- if each bin has only ONE ~~freq~~ tone from the signal in it, then the filter bank succeeds in isolating the significant freqs, and, if we can sample from such a filtered signal, we have reduced our problem to case (1).

for close freqs., we permute the spectrum randomly and then apply filterbank.

⇒ sample complexity & run time is proportional to # bins.

---

(1) Single freq. recovery : suppose x = fixed freq $\omega \in \mathbb{Z}_n$ with coeff $\alpha_\omega \in \mathbb{C}$.

$$x_j = \alpha_\omega \frac{1}{\sqrt{n}} e^{2\pi i \omega j/n} \qquad j \in \mathbb{Z}_n$$

a. calculate ω by choosing $j, j+1 \in \mathbb{Z}_n$ and calculating

$$\text{phase}\left(\frac{x_{j+1}}{x_j} = \frac{\alpha_\omega \sqrt{n}}{\alpha_\omega \sqrt{n}} e^{2\pi i \omega (1)/n} = e^{2\pi i \omega/n} \right)$$

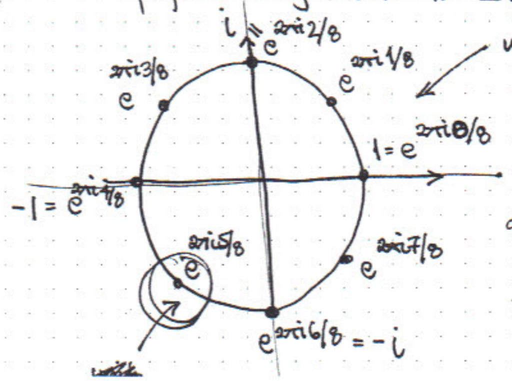
$$= \text{phase}\left(\cos\left(\frac{2\pi\omega}{n}\right) + i \sin\left(\frac{2\pi\omega}{n}\right) \right).$$

b. once you know ω, calculate α_ω by computing

$$x_j \sqrt{n}\, e^{-2\pi i \omega j/n} = \alpha_\omega \cdot 1.$$

---

this doesn't work very well if x is noisy. Have to perform binary search to ~~zoom~~ zoom in on ω. Let's suppose $n=8$.



divide into 2 halves and test if $e^{2\pi i \omega/8}$ closer to $-i$ or i
-1 or 1

(23)

(i) $|x_j| \cdot |i - e^{2\pi i \omega/8}| = |i \cdot x_j - x_{j+1}| \overset{?}{<} |i \cdot x_j + x_{j+1}| = |x_j| \cdot |i + e^{2\pi i \omega/8}|$

(ii) $|x_j| \cdot \underbrace{|1 - e^{2\pi i \omega/8}|} = |x_j - x_{j+1}| \overset{?}{<} |x_j + x_{j+1}| = |x_j| \underbrace{|1 + e^{2\pi i \omega/8}|}$

if $|-1 + e^{2\pi i \omega/8}| < |1 + e^{2\pi i \omega/8}|$, (i.e., $e^{2\pi i \omega/8}$ is closer to $+1$)
then we'll see

$$|x_j - x_{j+1}| < |x_j + x_{j+1}|$$

[and similarly for $\pm i$]

if $e^{2\pi i \omega/8}$ is closer to $+1$, then $\omega \in \{7,0,1\}$ and $\omega \notin \{3,4,5\}$

[similarly for $\pm i$ and $\{5,6,7\}$ and $\{1,2,3\}$.]

let's suppose we learn $\omega \in \{4,5,6\}$, then we can simplify our problem

define $x_j' = e^{-2\pi i \cdot 4 \cdot (2j/8)} x_{2j}$

$$= e^{-2\pi i \cdot 4 \cdot (2j/8)} \frac{\alpha_\omega}{\sqrt{n}} e^{2j 2\pi i \omega/8}$$

$$= \frac{\alpha_\omega}{\sqrt{n}} \cdot e^{2\pi i (\omega - 4) j/4}$$

for $j \in \mathbb{Z}_{n/2} = \mathbb{Z}_4$.

domain is half as big.

$\Rightarrow$ halved our problem.

- we shifted/rotated domain into first quadrant by multiplying by $e^{-2\pi i 4j/8}$
- then discarded odd entries.

and iterate....

[this is in the Gilbert-Strauss-Tropp 2008 survey, albeit expressed diff'ly.]

Let's do an example. Suppose $n = 2 \cdot 5 \cdot 7 = 70$ [a smooth # = product of lots of small primes] and suppose

$$x_j = \alpha_\omega \cdot \frac{1}{\sqrt{n}} \cdot e^{2\pi i j \omega / n}$$

[x is 1-tone with freq. $\omega \in \mathbb{Z}_n$ for $j \in \mathbb{Z}_n$]

let's form a short vector $a \in \mathbb{C}^2$ by sampling x at $j = 0, n/2$.

$$a_0 = x_0 = \alpha_\omega \cdot \frac{1}{\sqrt{n}}$$

$$a_1 = x_{n/2} = \alpha_\omega \cdot \frac{1}{\sqrt{n}} e^{2\pi i \omega \cdot n/2n} = \alpha_\omega \cdot \frac{1}{\sqrt{n}} (-1)^\omega$$

and compute the Fourier transform of a $\left(\hat{a} \in \mathbb{C}^2\right)$:

$$\hat{a}_0 = \alpha_\omega \cdot \frac{1}{\sqrt{n}} \left(\frac{1 + (-1)^\omega}{\sqrt{2}} \right)$$

$$\hat{a}_1 = \alpha_\omega \cdot \frac{1}{\sqrt{n}} \left(\frac{1 + (-1)^{\omega+1}}{\sqrt{2}} \right)$$

→ ω is an integer and it's either EVEN or ODD
⇒ only ONE of $\hat{a}_0$ & $\hat{a}_1$ will be non-zero
if $\hat{a}_0 \neq 0$, then $\omega \equiv 0 \bmod 2$
if $\hat{a}_1 \neq 0$, then $\omega \equiv 1 \bmod 2$

[Can learn ω mod 5 and 7 by sampling x at intervals $n/5$, $n/7$ ~~xxxxx~~ and then computing shorter FFTS. [aliased FFTs can do in parallel!] Then, from moduli, can reconstruct ω.]

<u>Thm</u> (CRT)

Any integer x is uniquely specified mod n by its remainders mod m rel. prime ints $p_1, p_2, \ldots, p_m$ as long as $\prod_{i=1}^{m} p_i \geq n$.

→ <u><Q></u>: How many samples? How fast?

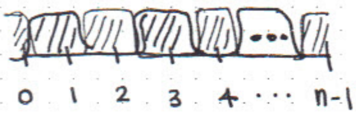
$$2 + 5 + 7 = 14$$

FFT FFT FFT and 3 short FFTs.

the algorithm is a <u>small</u> linear system size = # prime factors. ~ $\log(n)$

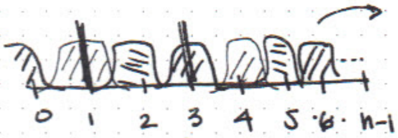
2. Filtering to Isolate Freqs.

It's important to view our CRT argument as a sampling from a filtered version of our signal in order to figure out how to handle signals with more than 1 freq.



→ the sampling at 0, n/2 "filters" the signal into 2 freq. "bins" — those that are EVEN & ODD.

Similarly for the other factors bins for different conj. classes mod 5, 7.

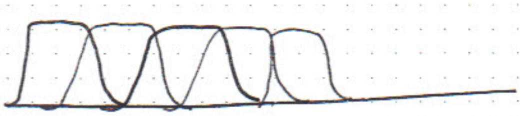


→ ex: 3 bins for 0, 1, 2 mod 3. 2 freqs in spectrum.

If ~~the~~ signal has 2 freqs., then the freqs. can't be equal mod 3 (CRT says they can't collide if you take enough primes)

More generally: need ~~k~~ k^2 bins to isolate the k freqs.

⇒ ~~the~~ small # of short FFTs give enough info. to get all k freqs.



→ LOTS of research on good near-perfect bandpass filters
e.g. sinc * Gaussian
[need a short # coeffs on time...]
diff. sublinear algos. have used diff. filters: Gaussians, Dolph-Chebysch., indicators, spike trains, ...

⟨Q⟩: How do we handle freqs. that are close together?
These bins are det'ic!

but sample eff'ly

Need:

① permute spectrum efficiently ⟶ use pairwise indep. random permutations

② sample on time side from the permuted spectrum ⟶ use 2 invariance props. of the FT.

in conjunct^n with

a) dilation

$$a_j = x_{cj} \longleftrightarrow \hat{a}_j = \hat{x}_{c^{-1}j} \quad \text{assuming } c^{-1} \text{ exists mod } n$$

b) translation

$$a_j = e^{2\pi i bj/n} \longleftrightarrow \hat{a}_j = \hat{x}_{j-b}$$

$\Rightarrow$ if we choose b, c indep., unif'ly at random $(c \in \mathbb{Z}_n^*)$, then

$$a_j = e^{2\pi i bj/n} x_{cj}$$

has a permuted spectrum $\hat{a}_j = \hat{x}_{c^{-1}j-b}$; i.e.,

if $\hat{x}_\omega = \alpha_\omega$, then $\hat{a}_{(\omega c+b) \bmod n} = \alpha_\omega$

we permute the spectrum

<u>You check:</u> ① if b, c indep., unif. random $(c \in \mathbb{Z}_n^*)$, then $\omega \mapsto \omega c + b$ is a random permutat^n

② it's actually a pairwise indep. rand. permutat^n

$$\left.\begin{aligned} c\omega_1 + b &= \xi_1 \\ c\omega_2 + b &= \xi_2 \end{aligned}\right\} \quad \text{what's the prob. of collision?}$$

Put all these pieces together in an iterative procedure:

while not done {

IDENTIFY freqs. with big Fcoeffs

ESTIMATE Fcoeffs of those identified freqs.

SUBTRACT (samples of) current approx.ⁿ from (samples of) orig. signal. (i.e, remeasure).

}

[GST 2008 has full pseudo code that is easy to implement!] (note caveats!)

[HIKP1,2 also have pseudocode]

There are some implementations:

- AAFFT (on sourceforge)
- SFFT v1,2,3 (on MIT webpage)
- ETH (on MIT webpage)

see IEEE SP Mag 2008, GST for a nicer layout + diagrams!

SFT(x, k)

Input: x has length $n = 2^{l}$ (need sampling access OR gen. samples FIRST and input those)
k # of desired freqs.

Output: $\Lambda = \{(\omega, \alpha_\omega)\}$ = list containing $O(k)$ (freq, coeff) pairs

$K \leftarrow 8k$
$\Lambda \leftarrow \emptyset$ $\Big] \longrightarrow$ Initialize.

for $j=1$ to 5 {

$\Omega \leftarrow$ Identification(x, Λ, K)

$c \leftarrow$ Estimation(x, Λ, Ω)

for each $\omega \in \Omega$ {

if $(\omega, \alpha_\omega) \in \Lambda$, then replace with $(\omega, \alpha_\omega + c_\omega)$ in Λ

else add new pair (ω, c_ω) to Λ

}

Retain K pairs in Λ with largest (abs. val.) coeff

}

Return Λ or prune to top k pairs.

Sample-Residual $(x, \Lambda, t, \sigma, K)$

for $k=1$ to K {

$u_k \leftarrow x(t + \sigma(k-1) \bmod n)$ $\longrightarrow$ sample arith. prog. from signal

$v_k \leftarrow \sum_{(\omega, \alpha_\omega) \in \Lambda} \left(\alpha_\omega e^{2\pi i \omega t/n} \right) e^{2\pi i \omega \sigma/n (k-1)}$ $\longrightarrow$ non-uniform FFT

}

Return $u - v \longrightarrow$ residual

Identification (x, Λ, K)

reps $\leftarrow$ 5
$\omega_k \leftarrow 0$ for $k = 1, 2, .., K$ → initialize

Draw $\sigma \sim \text{Unif}\{1, 3, 5, .., n-1\}$ → random odd dilatⁿ factor (needs to be odd so it's invert. mod 2^ℓ).

for $b = 0$ to $\lg(n/2)$ { → loop from LSB to MSB
  vote$_k \leftarrow 0$ for $k = 1, 2, .., K$

  for $j = 1$ to reps {
    Draw $t \sim \text{Unif}\{0, 1, .., n-1\}$ → random sample pt.
    $u \leftarrow$ sample-shattering$(x, \Lambda, t, \sigma, K)$
    $v \leftarrow$ Sample-shattering$(x, \Lambda, t + n/2^{b+1}, \sigma, K)$ → samples for correlated testing b^{th} bit

    for $k = 1$ to K {
      $E_0 \leftarrow u_k + e^{-\pi i \omega_k / 2^b} v_k$
      $E_1 \leftarrow u_k - e^{-\pi i \omega_k / 2^b} v_k$
      if $|E_1| \geq |E_0|$, then vote$_k \leftarrow$ vote$_k + 1$ → vote when bit = 1
    } → apply bit-test to demod. sig.

    for $k = 1$ to K {
      if vote$_k >$ reps/2 then $\omega_k \leftarrow \omega_k + 2^b$ → majority vote for bit value
    }
  }
}

Return unique$(\omega_1, \omega_2, .., \omega_K)$ → remove duplicate freqs.

Sample-shattering$(x, \Lambda, t, \sigma, K)$

$z \leftarrow$ Sample-residual$(x, \Lambda, t, \sigma, K)$

$z \leftarrow$ FFT(z)

return (z)

Estimation(x, Λ, Ω)

reps $\leftarrow 5$

for $j=1$ to reps {

Draw $\sigma \sim$ Unif. $\{1, 3, 5, \ldots, n-1\}$

$t \sim$ Unif. $\{0, 1, 2, \ldots, n-1\}$

$u \longleftarrow$ sample-residual $(x, \Lambda, t, \sigma, K)$

for $l = 1$ to $|\Omega|$ {

$c_l(j) = \sum_{k=1}^{K} u_k e^{2\pi i (\omega_l \sigma / n)(k-1)}$ → in parallel, non-unif FFT

$c_l(j) \leftarrow \left(\frac{n}{K}\right) e^{-2\pi i \omega_l t / n} \, c_l(j)$ → demodulate and scale the est.

}

}

$c_l \leftarrow$ Median$\{c_l(j) \mid j=1, \ldots$ reps$\}$ for $l = 1, 2, \ldots, |\Omega|$ → helps with robustness.

return $c_1, c_2, \ldots, c_{|\Omega|}$