MIT 6.893: Algorithms and Signal Processing Fall 2014

Lecture 5 — September 22, 2014

Prof. Piotr Indyk *Scribe: Zhengyu Wang*

1 Introduction

In this lecture, we give an $O(k \log n)$-time sparse Fourier transform algorithm (for the exactly k-sparse case) that works even for worst case inputs, based on the *pseudo-random spectrum permutation* technique (to permute coefficients randomly so that it is as if the average case) and *flat window functions* (to bin coefficients in an efficient and desirable way) that we discussed in the previous lecture. We follow the idea of Section 5 in Lecture 4, and give a complete proof of Theorem 7 in Lecture 4.

The result mentioned in the lecture first appears in [1].

2 $O(k \log n)$-time Algorithm

We assume that n is a power of 2, and $\widehat{a}_u$ contains at most k non-zero coefficients with "polynomial precision" (For example, one simple way is to assume that $\widehat{a}_u$ in $\{-n^{O(1)}, \ldots, n^{O(1)}\}$). And the proposed algorithm outputs at most k potential coefficients, and each actual non-zero coefficient of $\widehat{a}$ is output correctly with probability $1 - p$ for some constant $p > 0$.

We first show a basic block of the $O(k \log n)$-time algorithm in Algorithm 1. Basically, the function of the algorithm is to recover most coefficients of $\widehat{a}$. For the parameters, we have the relations that $B = 1/(2\varepsilon)$, $\varepsilon' = (1 - \gamma)\varepsilon$ and $\delta = 1/n^{O(1)}$. B is the number of bins, and $\varepsilon, \varepsilon', \delta$ altogether specify the parameters of the flat window function used as the filter function on input signal a.

Now we give some explanations for Algorithm 1. Line 2 randomly picks a linear permutation where σ is uniformly chosen from odd numbers in $[n]$, and β is uniformly chosen from $[n]$. Line 6 to Line 13 do the binning, i.e., it puts the coefficients of $\widehat{a} - \widehat{e}$ into B bins, where $\widehat{e}$ is the signal already recovered and stored in list. We do the binning for $\widehat{a}$ on Line 6-7, and subtract the binning for $\widehat{e}$ on Line 8-13. Binning is the bottleneck of our algorithm, so we should carefully design it.

Discussion. In order to compute $\widehat{c'}_{jn/B}, j = 0, \ldots, B - 1$ (on Line 6 in Algorithm 1), we have two options.

- The direct option is to compute the DFT of $(a \times g)_{-w/2}, \ldots, (a \times g)_{w/2}$ where $w = O(k \log(n)/\gamma)$. However, it runs in $O((k \log(n)/\gamma) \cdot \log n)$ time, and it computes too many samples of $\widehat{a} * \widehat{g}$. So we will not adopt this option.

- The second option is to alias $a \times g$ into B bins first. That is, first to compute $b_i = \sum_t (a \times g)_{i+tB}$, and then to compute the DFT of $b_0, \ldots, b_{B-1}$. It runs in $O(k \log(n)/\gamma) + O(B \log B) = O(k \log(n)/\gamma)$ time. We will compute in this way.

Line 14-18 do the updates. We will prove later, that after the updates, $\mathrm{supp}(\widehat{a} - \widehat{e})$ shrinks by a constant fraction with good probability.

Now we give the $O(k \log n)$-time algorithm in Algorithm 2. It basically repeats Algorithm 1 with carefully chosen parameters.

Algorithm 1 Basic Block for $O(k \log n)$-time Sparse Fourier Transformation.

function PARTIAL-RECOVERY(B, γ, k, list)
  Choose random σ, β
  Define $a'_j = a_{\sigma j}\omega^{-j\beta}$ ▷ so that $\widehat{a}'_u = \widehat{a}_{1/\sigma(u+\beta)}$
  Define $a''_j = a'_{j+1}$ ▷ so that $\widehat{a''}_u = \widehat{a'}_u\omega^u$
  Let g to be the $(\varepsilon, \varepsilon', \delta, O(k \log n/\gamma)$ flat window function constructed in Lecture 4
  Compute $\widehat{c}'_{jn/B}, j = 0, \cdots, B-1$, where $\widehat{c'} = \widehat{a'} * \widehat{g}$ ▷ see discussion
  Compute $\widehat{c}''_{jn/B}, j = 0, \cdots, B-1$, where $\widehat{c''} = \widehat{a''} * \widehat{g}$ ▷ similar to previous line
  for each (val,pos) in list **do**
    $u = \sigma \cdot \text{pos} - \beta$
    let j be the closest bin to u
    off $= u - jn/B$
    $\widehat{c'}_{jn/B} = \widehat{c'}_{jn/B} - \text{val} \cdot \widehat{g}_{\text{off}}$
    $\widehat{c''}_{jn/B} = \widehat{c''}_{jn/B} - \text{val} \cdot \omega^u \cdot \widehat{g}_{\text{off}}$
  for $j \leftarrow 0, \ldots, B-1$ **do**
    if $|\widehat{c'}_{jn/B}| > 1/2$ **then**
      $\text{val} = \widehat{c'}_{jn/B}$
      $\text{pos} = 1/\sigma(\text{round}(\text{phase}(\widehat{c''}_{jn/B}/\widehat{c'}_{jn/B})) + \beta)$
      Add (val, pos) to outputList
  return outputList

Algorithm 2 $O(k \log n)$-time Sparse Fourier Transformaion.

function MAIN
  list= $\emptyset$
  for $t \leftarrow 1, \ldots, \log k/3$ **do**
    $k_t = k/8^{t-1}$, $B_t = Ck/4^t, \gamma_t = 1/(C2^t)$ (this defines ε_t and ε'_t)
    list = list + Partial-Recovery(B_t, γ_t, k_t, list)
  return list

3 Analysis

We start the analysis with two claims for Algorithm 1.

Claim 1. *At most k entries in $\widehat{c}'_0, \ldots, \widehat{c}'_{B-1}$ have magnitudes $> 1/2$, and therefore are reported.*

This is true because each spectrum to be recovered contributes to exactly one entry in $\widehat{c}'_0, \ldots, \widehat{c}'_{B-1}$, whose values are negligible otherwise.

Claim 2. *For each u in supp$(\widehat{a})$, the probability u is not correctly reported is at most $O(k\varepsilon + \gamma)$.*

Proof. u will be not correctly reported either because it collides with other coefficients or because subsampling hits its "slope part" instead of the "flat peak".

- The probability of being mapped within $O(\varepsilon n)$ of some other coefficient is $O(k\varepsilon)$. This is because there are at most k other coefficients, and the probability to collide is $O(\varepsilon)$ (Lemma 2 in Lecture 4).
- Probability that the coefficient is not sampled properly is at most $O(\gamma)$ since the position of the center of the "flat peak" is chosen uniformly at random.

□

Now we prove correctness and run time guarantee.

3.1 Correctness

Let $\widehat{e}_t$ be the contents of list at the end of stage t in Algorithm 2. Define a "good" event E_t to be

$$r_t = ||\widehat{a}_t - \widehat{e}_t||_0 < k/8^t.$$

By Claim 2, conditioned on E_{t-1}, for any u, the probability of failure to recover u is at most the sums of

$$r_{t-1}\varepsilon_t = O(k/8^{t-1} \cdot 1/(Ck/4^t)) = O(1)/C \cdot 1/2^t$$

and

$$\gamma_t = 1/(C2^{t-1}).$$

Therefore,

$$\mathbb{P}[\bar{E}_t | E_{t-1}] < \mathbb{P}[\text{fraction of failures} > 1/(3 \times 8)] < O(1)/C \cdot 2^{-t}$$

and

$$\mathbb{P}[\bar{E}_1 \vee \bar{E}_2 \vee \ldots \vee \bar{E}_{\log k/3}] < O(1)/C \cdot (1/2 + 1/4 + \ldots) = O(1)/C.$$

3.2 Run Time Guarantee

The time for binning is

$$\sum_{t=1}^{\log k/3} O(k_t \log n/\gamma_t) = O(k \cdot \log n) + O(k/4 \cdot \log n) + \ldots = O(k \log n).$$

And the time for list operations is also bounded by

$$O(\log n) \cdot k.$$

References

[1] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Near-optimal algorithm for sparse Fourier transform. *STOC*, 2012.