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 $\hat{a}_u$ contains at most $k$ non-zero coefficients with “polynomial precision” (For example, one simple way is to assume that $\hat{a}_u$ 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 $\hat{a}$ 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 $\hat{a}$. 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 $\sigma$ is uniformly chosen from odd numbers in $[n]$, and $\beta$ is uniformly chosen from $[n]$. Line 6 to Line 13 do the binning, i.e., it puts the coefficients of $\hat{a} - \hat{e}$ into $B$ bins, where $\hat{e}$ is the signal already recovered and stored in list. We do the binning for $\hat{a}$ on Line 6-7, and subtract the binning for $\hat{e}$ on Line 8-13. Binning is the bottleneck of our algorithm, so we should carefully design it.

**Discussion.** In order to compute $\hat{c}_{jn/B}, j = 0, \ldots, B - 1$ (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 $\hat{a} \ast \hat{g}$. 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 $b_t = \sum_t (a \times g)_{i+tb}$, 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, supp($\hat{a} - \hat{e}$) 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.

1: function Partial-Recovery($B, \gamma, k, \text{list}$)  
2:    Choose random $\sigma, \beta$  
3:    Define $a'_j = a_{\sigma j} \omega^{-j \beta}$ \hspace{1cm} $\triangleright$ so that $\hat{a}'_u = \hat{a}_{1/\sigma(u+\beta)}$  
4:    Define $a''_j = a'_{j+1}$ \hspace{1cm} $\triangleright$ so that $\hat{a}''_u = \hat{a}'_u \omega^u$  
5:    Let $g$ to be the $(\varepsilon, \varepsilon', \delta, O(k \log n/\gamma)$ flat window function constructed in Lecture 4  
6:    Compute $\hat{c}'_{jn/B}, j = 0, \ldots, B-1$, where $\hat{c}' = \hat{a}' * \hat{g}$ \hspace{1cm} $\triangleright$ see discussion  
7:    Compute $\hat{c}''_{jn/B}, j = 0, \ldots, B-1$, where $\hat{c}'' = \hat{a}'' * \hat{g}$ \hspace{1cm} $\triangleright$ similar to previous line  
8:    for each (val,pos) in list do  
9:       let $u = \sigma \cdot \text{pos} - \beta$  
10:      let $j$ be the closest bin to $u$  
11:     off = $u - jn/B$  
12:    $\hat{c}'_{jn/B} = \hat{c}'_{jn/B} - \text{val} \cdot \hat{g}_{\text{off}}$  
13:    $\hat{c}''_{jn/B} = \hat{c}''_{jn/B} - \text{val} \cdot \omega^u \cdot \hat{g}_{\text{off}}$  
14:    for $j \leftarrow 0, \ldots, B-1$ do  
15:       if $|\hat{c}'_{jn/B}| > 1/2$ then  
16:          $\text{val} = \hat{c}'_{jn/B}$  
17:          $\text{pos} = 1/\sigma(\text{round}(\text{phase}(\hat{c}'_{jn/B}/\hat{c}'_{jn/B}))) + \beta)$  
18:          Add (val,pos) to outputList  
19:    return outputList

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

1: function MAIN  
2:    list = $\emptyset$  
3:    for $t \leftarrow 1, \ldots, \log k/3$ do  
4:       $k_t = k/8^{t-1}$, $B_t = Ck/4^t$, $\gamma_t = 1/(C2^t)$ (this defines $\varepsilon_t$ and $\varepsilon'_t$)  
5:       list = list + Partial-Recovery($B_t, \gamma_t, k_t, \text{list}$)  
6:    return list
3 Analysis

We start the analysis with two claims for Algorithm 1.

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

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

Claim 2. For each $u$ in supp($\hat{a}$), 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 $\hat{e}_t$ be the contents of list at the end of stage $t$ in Algorithm 2. Define a “good” event $E_t$ to be

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

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,

$$\Pr[\not E_t | E_{t-1}] < \Pr[\text{fraction of failures } > 1/(3 \times 8)] < O(1)/C \cdot 2^{-t}$$

and

$$\Pr[\not E_1 \lor \not E_2 \lor \ldots \lor \not E_{\log_2 k/3}] < O(1)/C \cdot (1/2 + 1/4 + \ldots) = O(1)/C.$$
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