Sketching Algorithms for Big Data Fall 2017

Lecture 17 — October 31, 2017

Prof. Piotr Indyk *Scribe: Cenk Baykal*

1 Overview

In the last lecture, we introduced the Sparse Fourier Transform (SFT) problem and discussed approaches under the assumption that the Fourier Transform of the input signal was 1-sparse, i.e., the input signal had a single non-zero Fourier coefficient. In particular, we covered the constant-time Two Point Sampling algorithm for the case of noiseless signals, and extended it to handle the noisy case, resulting in an algorithm based on a randomized binary search with a running time of $\mathcal{O}(\log n \log\log n)$.

In this lecture, we generalize to the k-sparse case for $k > 1$ in the absence of noise. We will cover an approach based on *isolating* large coefficients via judicious down sampling, with the goal of reducing the general k-sparse problem into multiple, easy-to-solve 1-sparse subproblems.

2 Isolation via Aliasing/Down-sampling

We will achieve isolation of coefficients using sub-sampling in the time domain, which as it turns out, corresponds to aliasing in the frequency domain. For ease of presentation, we will cover a simple-to-implement aliasing-based method that succeeds with constant probability and has an average case runtime complexity of $\mathcal{O}(k \log n)$.

The intuition behind our algorithm is that down sampling collapses the spectrum to something much shorter, effectively reducing the size of the problem, n. We will argue that under the assumption that the original spectrum has randomly distributed (as opposed to adversarially-chosen) entries, collisions do not occur too frequently, enabling us to disambiguate the collapsed spectrum.

The following theorem formalizes the fact that down sampling in the time domain corresponds to aliasing in the Fourier spectrum.

Theorem 1 (Aliasing Theorem): Given a signal $\mathbf{a} = (a_0, \dots, a_{n-1})$ of length n and a parameter $L \in \mathbb{N}_+$ that divides n, consider the vector $a' = (a_0, a_L, a_{2L}, \dots,)$ with n/L entries. Then,

$$\hat{a}'_u = \sum_{l=0}^{L-1} \hat{a}_{u+\frac{n}{L}\cdot l}.$$

Proof. See [4]. □

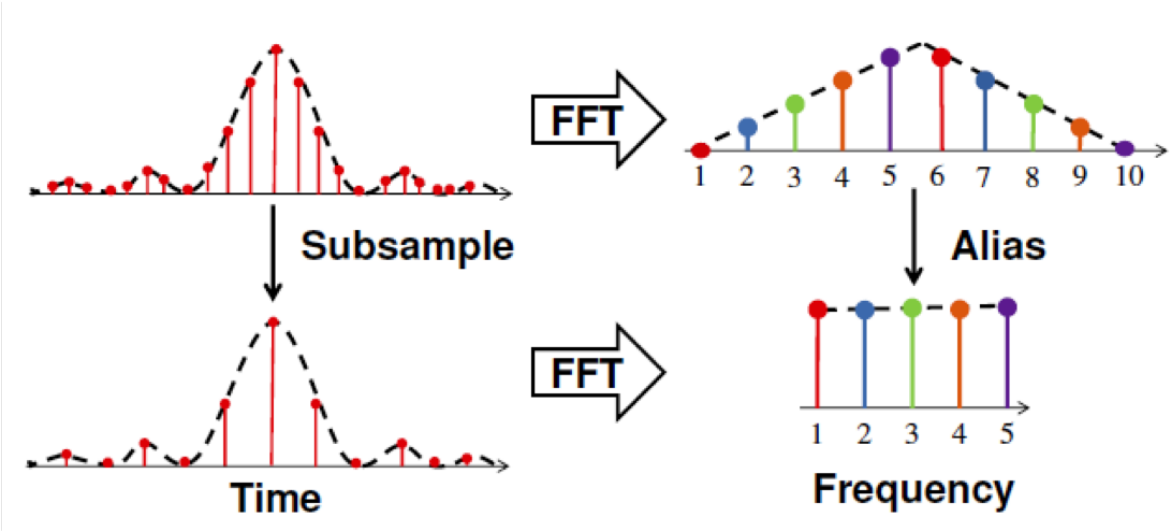

Figure 1: Visualization of the Aliasing Theorem. Sub-sampling in the time domain corresponds to aliasing in the frequency domain.

Algorithm 1 Aliasing FFT.

function AliasingFFT(a)
  $L \leftarrow n/k^2$
  Sample two corresponding sequences of signals, each of length k^2
  $\mathbf{a}' \leftarrow \{a_0, a_L, a_{2L}, \ldots\}$
  $\mathbf{a}'' \leftarrow \{a_1, a_{L+1}, a_{2L+1}, \ldots\}$
  Run FFT on $\mathbf{a}'$ and $\mathbf{a}''$ to obtain $\hat{\mathbf{a}}'$ and $\hat{\mathbf{a}}''$
  Recover $\hat{\mathbf{a}}$ by applying Two Point Sampling for each non-zero $\hat{\mathbf{a}}'_u$
  return $\hat{\mathbf{a}}$

3 AliasingFFT: SFT in $\mathcal{O}(k^2 \log k)$ Time

The basic idea behind our algorithm is to take the full spectrum consisting of n entries and collapse it to Ck^2 for some constant $C \geq 1$. For simplicity, we will henceforth assume $C = 1$, although other constants larger than 1 can be used to boost the success probability to higher constants. The following theorem establishes that given a signal $\mathbf{a}$ of length n, Alg. 1 can recover the Fourier coefficients, $\hat{\mathbf{a}}$ with probability at least $1/2$ in $\mathcal{O}(k^2 \log k)$ time.

Theorem 2 (Isolation by Aliasing): Suppose that $\hat{\mathbf{a}}$ is generated by summing up k Fourier coefficients in positions selected independently and uniformly at random from $\{0 \ldots, n-1\}$, i.e.,

$$\hat{\mathbf{a}} = \sum_{t=1}^{k} b_t \, e_{j(t)}, \tag{1}$$

where $b_1, \ldots, b_t$ are the corresponding magnitudes, $j(1), \ldots, j(t)$ are the corresponding uniformly drawn random indices, and e_i refers to ith the standard basis vector. Then, given the signal $\mathbf{a}$, we can recover $\hat{\mathbf{a}}$ in $\mathcal{O}(k^2 \log k)$ time with probability at least $1/2$.

Proof. First, we remark that by the Aliasing Theorem and the time-shift theorem:

$$\mathbf{a}'_u = \sum_{l=0}^{L-1} \hat{a}_{u+\frac{n}{L}\cdot l}$$
$$\mathbf{a}''_u = \sum_{l=0}^{L-1} \hat{a}_{u+\frac{n}{L}\cdot l} \omega^{u+l\cdot\frac{n}{L}}.$$

Thus, recovery of the coefficients can indeed be performed on Line 7 by applying the Two Sampling Algorithm from the previous lecture for each non-zero in $\hat{\mathbf{a}}'_u$.

To prove our claim, observe that it suffices to prove that with probability at least $1/2$, each non-zero coordinate in $\hat{\mathbf{a}}'$ only receives a contribution from a single Fourier component, i.e., there are no collisions in any of the bins. The failure probability is bounded by the probability that there exists an index such that the sum is composed of 2 or more terms, thus we have,

$$\mathbb{P}(\text{Failure}) \leq \binom{k}{2} \mathbb{P}(\text{Collision}) = \frac{k(k-1)}{2} \times \frac{L}{n} = \frac{k(k-1)}{2} \times \frac{1}{k^2} < 1/2,$$

where we used the union bound to argue over all $\binom{k}{2}$ possible pairs that can possibly collide. Thus, Alg. 1 succeeds with probability at least $1/2$.

Moreover, the runtime is dominated by two calls to FFT with inputs of length k^2, thus Alg. 1 runs in $\mathcal{O}(k^2 \log k)$ time. □

4 AliasingFFT+: Reducing $\mathcal{O}(k^2 \log k)$ to $\mathcal{O}(k \log n)$

In this section, we present an algorithm that improves upon Alg. 1 and runs in $\mathcal{O}(k \log n)$ time. The algorithm we will present was discovered independently in 2013 by Ghazi et al. [1], Pawar et al. [2], and Hsieh et al. [3].

Henceforth, we impose the following assumptions on the problem:

1. The locations, i.e., indices, of the non-zero Fourier coefficient entries are random and independent . That is, we need the locations of the Fourier peaks in $\hat{\mathbf{a}}$ to be randomly generated.
2. $n = p \cdot q$ where the integers p and q are co-prime, with $p < q$.
3. $k < q$ and $k, p, q = \theta(\sqrt{n})$.
4. $k < \varepsilon p$ for a fixed constant $\varepsilon \in (0, 1)$. Note: in the context of the grid representation described below and shown in Fig. 3, this assumption implies that there is less than one non-zero entry per column / row in expectation.

We remark that Assumption 1 was also required for the analysis of the previous AliasingFFT algorithm. Moreover, the algorithm that we will discuss can be extended to run in time $\mathcal{O}(k \log n)$ in the case of a slightly relaxed version of Assumption 3 where $k = o(\sqrt{n})$.

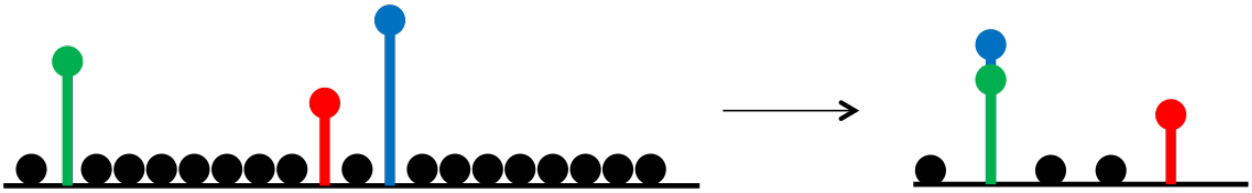

Figure 2: Visual depiction of isolation by aliasing, where a collision occurs in one entry. Our analysis shows via a Birthday paradox-like argument that for a properly spaced out (i.e., choice of parameter L), subsampling procedure, collisions do not occur with constant probability.

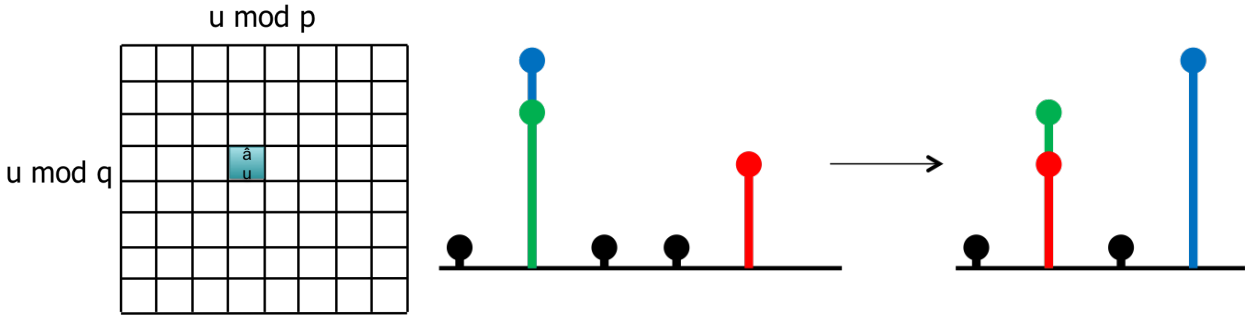

Figure 3: Left) Pictorial representation of binning using two co-prime aliasing filters. Right) Illustration of the frequency subtraction process.

4.1 Algorithm

The main idea behind our algorithm is a simple two step iterative process: (i) identify isolated frequencies in one filter and (ii) subtract them from the other, and repeat finitely many times. Another way to think about this process is to think of the frequency coefficients $\hat{a}_u$ as a two-dimension grid, i.e., map each frequency u to the unique matrix entry (since p and q are co-prime) given by $(u \mod q, u \mod p)$ and note that downsampling by p or q collapses columns or rows in the frequency domain by summing along the columns / rows (see Fig. 3). Note that this is a valid representation since $(u \mod q, u \mod p)$ is a bijection between $\{0, 1, \ldots, n-1\}$ and $(\{0, 1, \ldots, p-1\} \times \{0, 1, \ldots, q-1\}$ for p, q co-prime.

The intuition behind our algorithm is that by using a peeling process in the form of collapsing rows and columns, we use the row and column aliasing operations to bin the frequencies in each row and column. Since the frequency grid is only sparsely populated (in expectation, see Assumption 4), some of the rows and columns are likely to have exactly one coefficient, which can be recovered using the Two Point Sampling method that we saw in the last lecture and also leveraged in Alg. 1. After recovering the coefficient, we can remove its contribution (see in Fig. 3) and proceed with the remaining signal in a self-reducing manner, until $\mathcal{O}(\log n)$ iterations have been executed.

Algorithm 2 Aliasing FFT+.

1: **function** AliasingFFT+(a, p, q)
2: $T \leftarrow \{0, 1, \ldots, 2s\}$ for $s = \mathcal{O}(1)$
3: $a^{t,L} \leftarrow a_{t+L}, a_{t+2L}, \ldots,$ for $t \in T$, $L \in \{p, q\}$
4: $\hat{a}^{t,L} \leftarrow \text{FFT}(a^{t,L})$ ▷ Run FFT to obtain $\hat{a}^{t,L}_u = \sum_{l=0}^{L-1} \hat{a}_{u+\frac{n}{L}\cdot l}\omega^{\left(u+\frac{n}{L}\cdot l\right)t}$
5: $\hat{c}^{t,L} \leftarrow \hat{a}^{t,L}$
6: $C \leftarrow \emptyset$ ▷ Keep track of recovered coefficients
7: **for** iterations $i = 1, \ldots, \mathcal{O}(\log n)$ **do**
8: $\hat{c}^{t,L} \leftarrow \text{SubtractCoeffs}(\hat{a}^{t,L}, C, T)$
9: $C \leftarrow C \cup \text{Recover}(\hat{c}^{t,L}, p, T)$
10: $\hat{c}^{t,L} \leftarrow \text{SubtractCoeffs}(\hat{a}^{t,L}, C, T)$
11: $C \leftarrow C \cup \text{Recover}(\hat{c}^{t,L}, q, T)$
return C

12: **function** SubtractCoeffs$(\hat{a}^{t,L}, C, T)$
13: $\hat{c}^{t,L} \leftarrow \hat{a}^{t,L}$
14: **for** $(v, i) \in C, t \in T, L \in \{p, q\}$ **do**
15: $u \leftarrow i \mod n/L$
16: $\hat{c}^{t,L}_u \leftarrow \hat{a}^{t,L}_u - v\omega^{t\,i}$
return $\hat{c}^{t,L}$

17: **function** Recover$(\hat{c}^{t,L}, L, T)$
18: $C' \leftarrow \emptyset$
19: **for** $u = 0, \ldots, n/L - 1$ **do**
20: Apply two-point sampling on $\hat{c}^{0,L}_u$ and $\hat{c}^{1,L}_u$ to obtain the coefficient, index pair (v, i)
21: **if** $\hat{c}^{t,L}_u - v\omega^{ti} = 0$ for all $t \in T$ **then** ▷ Isolation Test
22: $C' \leftarrow C' \cup \{(v, i)\}$
return C'

4.2 Analysis

We subdivide the analysis of Alg. 2 into analyzing three components: running time, convergence, and correctness.

Running time Computation of $\hat{a}^{t,L}$, for $L \in \{p, q\}$ (Line 4) takes $\mathcal{O}((p+q)\log n)$ time. The SubtractCoeffs (x2) sub-procedure takes $\mathcal{O}(k)$ time per invocation (Lines 8 and 10) and Recover$(\hat{c}^{t,L}, p, T)$ and Recover$(\hat{c}^{t,L}, q, T)$ (Lines 9 and 11) take time $\mathcal{O}(p)$ and $\mathcal{O}(q)$ respectively.

Since the algorithm runs for at most $\mathcal{O}(\log n)$ iterations and since $p, q, k = \theta(\sqrt{n})$ by Assumption 3, the overall time complexity of Alg. 2 is $\mathcal{O}(k \log n)$.

Convergence Note that the process gets suck if either of the following occurs:

1. There is an alternating row/column cycle with two entries per each row/column, or
2. There is a chain of length $> \mathcal{O}(\log n)$.

To address the probability of the first event occurring, note that the probability of a cycle of length $2l$ occurring is at most

$$p^l q^l \times \left(\frac{\varepsilon}{q}\right)^{2l} < \varepsilon^{2l},$$

since there are at most $p^l q^l$ cycles of length $2l$ and each entry is non-zero with probability ε/q. Union bounding over all possible cycle lengths, we get that the summation over all l converges to a small constant for sufficiently small ε.

To bound the probability of the second event occurring, note that there are pq possible starting positions for the path, and each entry is non-zero with probability ε/q. Thereafter, the probability of each entry being non-zero remains ε/q, but there are now at most q choices in the current row/column (since we assumed $p < q$ by Assumption 2). Thus, the probability that a chain of length $l' > c \log n$ exists is bounded above by

$$pq \left(\frac{\varepsilon}{q}\right) \left(q \times \frac{\varepsilon}{q}\right)^{l'} = o(1),$$

for large enough c and $l' > c \log n$.

Putting it all together, we conclude that Alg. 2 does not get stuck with constant probability.

Correctness As the final component of our analysis, we will establish that our algorithm is correct with constant probability. To do so, we formalize the (probabilistic) correctness of the Isolation Test (Line 21 of Alg. 2) by the following Lemma.

Lemma 3 (Probabilistic Correctness of Isolation Test): Let $s = 2$. If for all $t \in T = \{0, 1, \ldots, 2s\}$, we have

$$\hat{a}_u^{t,L} = \sum_{l=0}^{L-1} \hat{a}_{u+\frac{n}{L}\cdot l} \omega^{\left(u+\frac{n}{L}\cdot l\right)t} = 0,$$

then

$$\mathbb{P}\left(\hat{a}_{u+\frac{n}{L}\cdot l} = 0 \ \forall l \in \{0, 1, \ldots, L-1\}\right) > 1 - \frac{C}{n}.$$

Proof. Let m be the number of non-zeros in $\hat{a} = \hat{a}_u, \hat{a}_{u+\frac{n}{L}}, \hat{a}_{u+\frac{n}{2L}}, \ldots$. We subdivide the proof into three cases:

Case 1 ($m \leq s$): Then, the solution to the linear system

$$\sum_v \hat{a}_v \omega^{vt} = 0 \quad \forall t \in T,$$

must be $\hat{a} = 0$, since the Vandermonde matrix $V = [\omega^{vt}]$ has full rank.

Case 2 ($s < m < \mathcal{O}(\log n)$): Choose $\hat{a}$ by selecting $(m-s)$-sparse $\hat{a}'$ first and complement it with s-sparse $\hat{a}''$, so that $\hat{a} = \hat{a}' + \hat{a}''$. Let V_T denote the Vandermonde matrix given by the partial Fourier matrix that is restricted to the rows in T. We consider the event $V_T(\hat{a}' + \hat{a}'') = 0$ and thus

we have

$$\begin{aligned}\mathbb{P}(V_T(\hat{a}' + \hat{a}'') = 0|\hat{a}') &= \mathbb{P}(V_T\hat{a}'' = -V_T\hat{a}'|\hat{a}') \\ &\leq \frac{1}{\binom{L-(m-s)}{s}} \\ &< \frac{C}{n},\end{aligned}$$

where we used the fact that there is at most one s-sparse $\hat{a}''$ such that $V_T\hat{a}'' = -V_T\hat{a}'$ and $L = \theta(\sqrt{n})$ by Assumption 3 since L corresponds to either p or q.

Case 3 ($m > \mathcal{O}(\log n)$): Since the probability of having a non-zero coefficient is ε/L, the probability of having more than $\mathcal{O}(\log n)$ non-zero coefficients is at most $1/n$.

□

Putting it all together, we have that our algorithm runs in $\mathcal{O}(k \log n)$ time and succeeds (i.e., is correct and does not get stuck) with constant probability.

References

[1] Badih Ghazi, Haitham Hassanieh, Piotr Indyk, Dina Katabi, Eric Price, Lixin Shi. Sample Optimal Average-Case Sparse Fourier Transform in Two Dimensions. *Allerton*, 2013.

[2] Sameer Pawar, Kannan Ramchandran. Computing a k-sparse n-length Discrete Fourier Transform using at most 4k samples and O(k log k) complexity. *ISIT*, 2013.

[3] S.-H. Hsieh, C.-S. Lu, S.-C. Pei. Sparse fast fourier transform by downsampling. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

[4] `http://www.dsprelated.com/dspbooks/mdft/Downsampling_Theorem_Aliasing_Theorem.html`