MIT 6.893: Algorithms and Signal Processing Fall 2014

Lecture 3 — September 15, 2014

Prof. Piotr Indyk *Scribe: Ludwig Schmidt*

1 Introduction

In this lecture, we build on the ideas from the previous two lectures and give a sparse Fourier transform running in $O(k \log n)$ time under certain assumptions including an average case analysis. Last lecture, we saw an algorithm that computes the Fourier transform of an n-dimensional signal with an exactly k-sparse spectrum. The algorithm had a running time of $O(k^2 \log n)$ and the correctness of the algorithm was proven for the *average case*. The algorithm reduced the problem of computing a k-sparse spectrum to several instances of a 1-sparse spectrum, which we could then find with the methods from Lecture 1. A crucial tool was *binning* the frequencies via *aliasing* in the frequency domain, which we could achieve efficiently by sampling in the time domain and computing a smaller Fourier transform (c.f. the Aliasing Theorem in Lecture 2). Since the algorithm from last lecture used only a single frequency binning / aliasing filter, it required a large number of frequency bins to ensure that every nonzero frequency is isolated in a frequency bin. This limited the number of frequency bins to $\Omega(k^2)$ (birthday paradox), leading to the $O(k^2 \log n)$ running time.

In this lecture, we improve the time complexity to $O(k \log n)$ by using *two* frequency binnings / aliasing filters. We choose the aliasing filters so that no pair of frequencies can collide in both filters at the same time. The algorithm relies on the following assumptions:

- The signal dimension n is the product of two coprime integers p and q (we assume $p < q$).
- The frequency sparsity k satisfies $k < q$ (as a result, $k = O(\sqrt{n})$).
- The locations of the nonzero frequencies are random and independent.

In the following, we further assume that $k, p, q = \Theta(\sqrt{n})$ and only mention that the algorithm can be extended to achieve a $O(k \log n)$ running time also in the $k = o(\sqrt{n})$ regime.

In spite of these limitations, today's algorithm has found multiple applications in practice because of its simplicity. The algorithm was discovered independently multiple times, at least by [1], [2] and [3]. Our description here most closely follows [1].

2 Signal model

Before we describe our algorithm, we introduce a useful pictorial representations and clarify our input model.

2.1 Pictorial representation

To get an intuition for our algorithm, it is helpful to think of the frequency coefficients $\widehat{a}_u$ as a two-dimensional grid (Figure 1). This is a valid representation because $(u \mod q, u \mod p)$ is a bijection between $\{0, 1, \ldots, n-1\}$ and $\{0, 1, \ldots, p-1\} \times \{0, 1, \ldots, q-1\}$ for p, q coprime. Using this mapping, all coefficients with the same value mod q are in the same row and all coefficients with the same value mod p are in the same column.

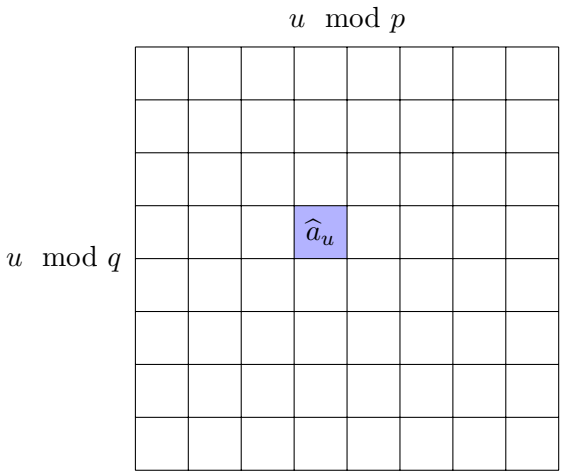


Figure 1: Two-dimensional signal representation.

In this represenation, the aliasing operations "collapse" the rows / columns by summing along the columns / rows. Note that the representation fails when p and q are not coprime because some coefficients then map to the same grid cell and hence collide in *both* aliasing filters.

2.2 Input model

We assume that our input is generated in two steps:

1. Fix a coefficient $\widehat{a}_u$ for every $u \in \{0, 1, \ldots, n-1\}$.
2. For every coefficient, toss a biased, independent coin: with probability $\frac{k}{n}$ keep the coefficient, otherwise set it to zero.

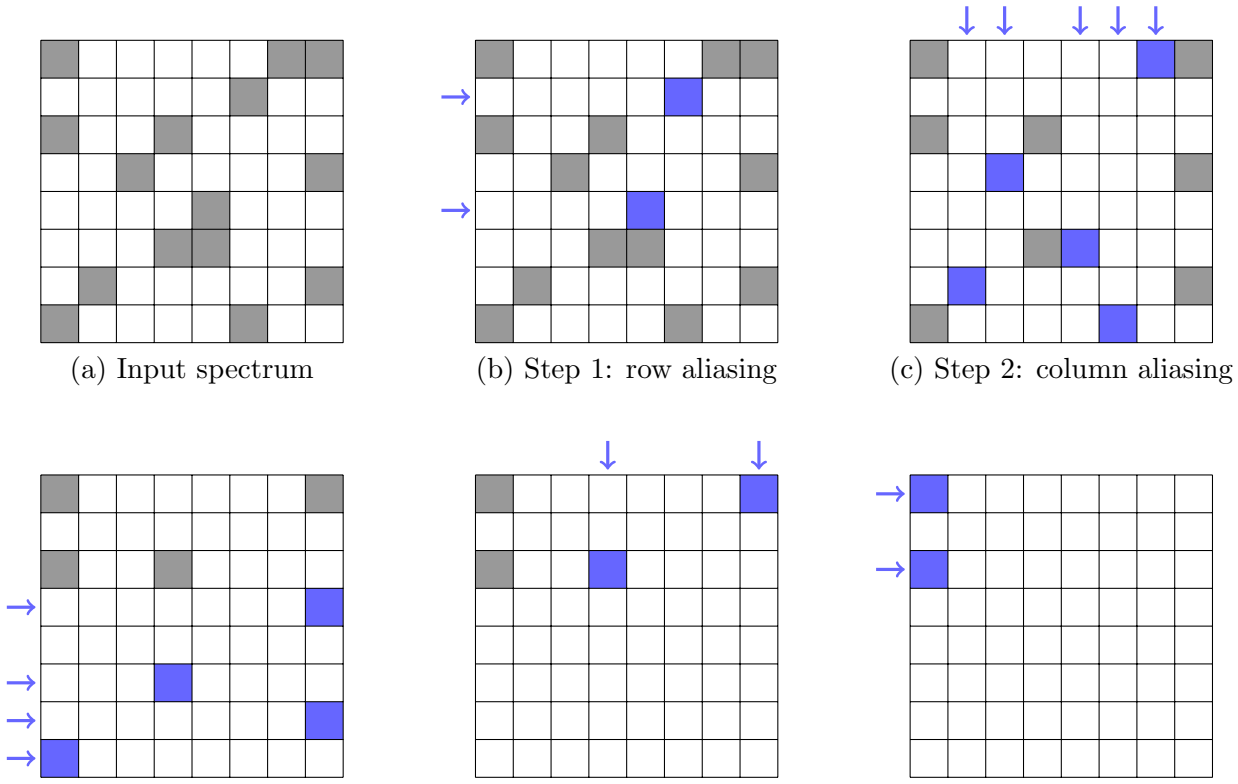
Furthermore, we assume that $k < \varepsilon p$ for a fixed small ε. So there is less than one nonzero entry per column / row *in expectation*.

It is important to note that step 1 can be adversarial, but our analysis crucially uses the fact that the decisions in step 2 are independent.

3 Algorithm

At a high-level, our algorithm can be seen as a peeling process. We use the row- and column-aliasing operations to bin the frequencies in each row / column. Since the frequency grid is only sparsely populated, some of the rows or columns are likely to have exactly one coefficient, which we can then recover using the two-point sampling technique from Lecture 1. After recovering the coefficient, we remove its contribution from the signal and proceed with the remaining signal, which might now contain new rows or columns with exactly one nonzero frequency coefficient. Using our average-case assumptions, this process converges after a logarithmic number of peeling iterations. See Figure 2 for an illustration of the peeling process.

An important ingredient in the algorithm outlined above is an efficient procedure for determining whether a given row / column contains exactly one non-zero coefficient. We call this part the "isolation test". We can reduce the isolation test to checking whether a row / column is empty by subtracting our signal estimate for this row / column (recovered with two-point sampling). Under



(a) Input spectrum (b) Step 1: row aliasing (c) Step 2: column aliasing

(d) Step 3: row aliasing (e) Step 4: column aliasing (f) Step 5: row aliasing

Figure 2: An illustration of the "peeling" recovery process on an 8×8 signal with 15 nonzero frequencies (gray squares). In each step, the algorithm recovers all 1-sparse columns and rows. The recovered frequencies are depicted with blue squares. After step 5, the algorithm has converged.

our average-case assumptions, we can show that a constant number of test entries in each row / column is sufficient for the isolation test (see Section 4). We now formally state our algorithm (see Algorithm 1).

Algorithm 1 Sparse Fourier transform with two aliasing filters.

1: **function** AliasingFFT(a, p, q)
2: Let $T = \{0, 1, \ldots, 2s\}$ for $s = O(1)$ ▷ Test set
3: Let $a^{t,L} = a_t$, a_{t+L}, …, a_{t+n-L} for $t \in T$ and $L \in \{p, q\}$
4: Compute $\widehat{a}^{t,L}$ ▷ Use FFT
5: $C \leftarrow \{\}$ ▷ Set of recovered coefficients
6: **for** $i \leftarrow 1, \ldots, O(\log n)$ **do**
7: $\widehat{c}^{t,L} \leftarrow$ SubtractCoeffs$(\widehat{a}^{t,L}, C, T)$
8: $C^p_{\text{new}} \leftarrow$ Recover$(\widehat{c}^{t,L}, p, T)$
9: $C^q_{\text{new}} \leftarrow$ Recover$(\widehat{c}^{t,L}, q, T)$
10: $C \leftarrow C \cup C^p_{\text{new}} \cup C^q_{\text{new}}$
11: **return** C

12: **function** Recover$(\widehat{c}^{t,L}, L, T)$
13: $C' \leftarrow \{\}$
14: **for** $u \leftarrow 0, \ldots, L-1$ **do**
15: Apply two-point sampling on $\widehat{c}^{0,L}_u$ and $\widehat{c}^{1,L}_u$ to recover
16: a coefficient v and a frequency i.
17: **if** $\widehat{c}^{t,L_u} - v\omega^{ti} = 0$ for all $t \in T$ **then** ▷ Isolation test
18: $C' \leftarrow C' \cup \{(v, i)\}$
19: **return** C'

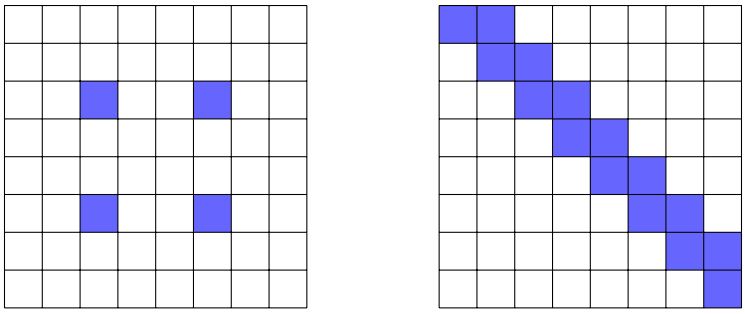
20: **function** SubtractCoeffs$(\widehat{a}^{t,L}, C, T)$
21: $\widehat{c}^{t,L} \leftarrow \widehat{a}^{t,L}$
22: **for** $(v, i) \in C$ **do**
23: $u \leftarrow i \mod L$
24: **for** $t \in T$ **do**
25: **for** $L \in \{p, q\}$ **do**
26: $\widehat{c}^{t,L}_u \leftarrow \widehat{c}^{t,L}_u - v\omega^{ti}$
27: **return** $\widehat{c}^{t,L}$

4 Analysis

Finally, we study the performance of our algorithm. For the convergence and running time analysis, we assume that all isolation tests are correct. In Section 4.3 we then show that the isolation tests succeed with large enough probablity in our average-case setting.

4.1 Running time

Since the test set T has a constant size, we can compute $\widehat{a}^{t,L}$ in line 4 with a standard FFT in total time $O((p + q) \log n)$. Furthermore, the subroutines in the for-loop from lines 6 to 10 have the following time complexities:



(a) Cycle (b) Long path

Figure 3: Cases in which the algorithm fails to converge.

- SubtractCoeffs($\widehat{a}^{t,L}, C, T$): $O(k)$
- Recover($\widehat{c}^{t,L}, p, T$): $O(p)$
- Recover($\widehat{c}^{t,L}, q, T$): $O(q)$

Since the loop is repeated $O(\log n)$ times and we have $k, p, q = \Theta(\sqrt{n})$ by assumption, we get an overall time complexity of $O(k \log n)$

4.2 Convergence

The algorithm fails to converge if one of the following two cases occurs:

- The sparsity pattern contains a "cycle" of nonzeros in which every nonzero frequency has at least one other nonzero in its row and column.
- The sparsity pattern contains a long path of nonzeros.

Figure 3 illustrates these two failure modes. The two cases above are the only cases in which the algorithm fails to converge because we can construct a cycle or a long path from every execution of the algorithm that does not terminate after $O(\log n)$ iterations of the for-loop in lines 6 to 10.

We now bound the probability of the two failure modes. Recall that a given frequency coefficient is nonzero with probability $\frac{k}{n} \leq \frac{\varepsilon}{q}$, independently of all other signal coefficients.

- Cycles: there are at most $p^l q^l$ cycles of length $2l$: pq choices for the first element, then alternatingly at most p or q choices for the following elements in the same row / column, and the last element of the cycle is fixed by the previous choices. Moreover, every element of the cycle is nonzero with probablity $\frac{\varepsilon}{q}$. So the total probability of a cycle of length $2l$ consisting of only nonzeros is

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

  where we used $p < q$. The sum over all l then converges to a small constant if ε is sufficiently small.

- Long paths: we bound the probability of a path with length at least l'. There are $n = pq$ starting positions for the path and each position is nonzero with probablity $\frac{\varepsilon}{q}$. For all following

elements, there are at most q choices in the current row / column and each element is nonzero with probability $\frac{\varepsilon}{q}$. Therefore the total probability of the paths with length at least l' is

$$pq\,\frac{\varepsilon}{q}\varepsilon^{l'}\ ,$$

which is subconstant for $l' > c\log n$ and a sufficiently large c.

So the algorithm converges with constant probability.

4.3 Correctness

We now show that the isolation test is correct with sufficiently high probability, so that the overall algorithm succeeds with constant probability. This then completes the analysis of the algorithm. For simplicity, we study the "emptyness test" here, which checks whether a given bin is empty. Formally, we have the following statement.

Lemma 1. *Let $s = 2$. If for all $t \in T$ we have*

$$\sum_{l=0}^{L-1} \widehat{a}_{u+\frac{n}{L}l}\,\omega^{(u+\frac{n}{L}l)t} \;=\; 0$$

then

$$\mathbb{P}[\widehat{a}_{u+\frac{n}{L}l} = 0 \text{ for all } l \in \{0,1,\ldots,L-1\}] \;>\; 1 - \frac{C}{n}\ .$$

Proof. Let m be the number of nonzeros in

$$\widehat{a} = \widehat{a}_u,\, \widehat{a}_{u+\frac{n}{L}},\, \ldots,\, \widehat{a}_{u+n-L}\ .$$

We now consider three cases:

- $m \le s$
  Then the solution of the linear system

  $$\sum_{v=0}^{L-1} \widehat{a}_v\,\omega^{vt} \;=\; 0 \qquad \text{for all } t \in T$$

  must be $\widehat{a} = 0$ because the Vandermonde matrix given by ω^{vt} has rank $|T|$.

- $s < m < O(\log n)$
  Choose $\widehat{a}$ by selecting an $(m-s)$-sparse $\widehat{a}'$ first and complementing it with an s-sparse $\widehat{a}''$, so that $\widehat{a} = \widehat{a}' + \widehat{a}''$. Let V_T be the Vandermonde matrix given by the partial Fourier matrix and restricted to the rows in T. In order to bound $\mathbb{P}[V_T(\widehat{a}' + \widehat{a}'')]$, it suffices to bound the following expression (using the law of total probability):

  $$\begin{aligned}\mathbb{P}_{\widehat{a}''}[V_T(\widehat{a}' + \widehat{a}'') = 0 \,|\, \widehat{a}'] \;&=\; \mathbb{P}_{\widehat{a}''}[V_T\widehat{a}'' = -V_T\widehat{a}' \,|\, \widehat{a}'] \\ &\le\; \frac{1}{\binom{L-(m-s)}{s}} \\ &<\; \frac{C}{n}\ .\end{aligned}$$

  The second line follows from the fact that there is at most one s-sparse $\widehat{a}''$ such that $V_T\widehat{a}'' = -V_T\widehat{a}'$ (here we use $|T| = 2s$ as defined in the algorithm). In the third line we use $L = \Theta(\sqrt{n})$ becaues L is either p or q.

- $m > O(\log n)$
  Since the probability of a given frequency coefficient being nonzero is at most $\frac{\varepsilon}{L}$, the total probability of this event is at most $\frac{1}{n}$.

□

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 k samples and $O(k \log k)$ complexity. *ISIT*, 2013.

[3] S.-H. Hsieh, C.-S. Lu, and S.-C. Pei. Sparse fast fourier transform by downsampling. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 56375641.