1 Introduction

In this lecture, we will focus on applications and future directions of the Sparse Fourier Transform we have learned in class.

2 Applications

So far we have been learning sparse Fourier transforms with an aim of minimizing time and the number of samples we need. Hence, some of the main applications of sparse Fourier transforms come as no surprise:

- Optimizing time
  - GPS
  - Hardware (special sFFT chips)
- Optimizing samples (and time)
  - Spectrum sensing

2.1 GPS Locking

In GPS locking, a device, such as a cell phone, is receiving a signal from satellites. In order for the device to figure out where it is, it needs to judge the distance to the various satellites it is listening to, from which it can triangulate its position. We can assume the device has a precise clock, so all it needs to do is figure out how long it takes for each signal to reach it and it can extrapolate the distance. The device also has the codewords of all the satellites it communicates with.

A natural solution to this is to take the received signal, shift it by some amount, and pointwise multiply it with the code – when this gives a high value, the shift is equal to the delay in transmission. But this is exactly what convolution does, so we can use a Fourier transform to approximate the delay more efficiently. We take an FFT of our signal, pointwise multiply it with an FFT of the code, then use an IFFT and read off the position of the peak. Notice that the IFFT can actually be a sparse Fourier transform, as the output signal has one peak. Note that the FFT of the received signal is not necessarily sparse, since the code is pseudorandom. However, since the sparse IFFT is not going to look at all its samples, we don’t actually need to compute all the samples in the FFT, so we can save some work there. Overall, we can get a ~2.2x improvement for the general problem (which also includes the recovery of the Doppler shift of the signal) and 5x for tracking (which assumes that the Doppler shift is known). See [1] for more details.
2.2 sFFT Chip

A chip with sFFT in hardware that can get good speedup over software implementations of sFFT. Uses aliasing with a ”rugged” version of two-point sampling to handle $n = 3^6 2^{10}$ and sparsity up to 750. Has roughly 88 times speed up over intel i7 when doing sparse FFT. [2]

2.3 Low Power Spectrum Sensing

With more mobile devices coming online by the day, we are running out of spectrum to allocate. However, many bands are actually not that occupied – even if the band is allocated, there is a lot of whitespace. But how can you efficiently figure out where the white space is, when you have to monitor large ranges of spectrum on the order of GHz? High-speed ADC’s (analog-to-digital converters) to sample at high speed are extremely expensive and power-hungry, so commercial devices rely on sequential scanning with more affordable ADC’s, which only give them tens of MHz at a time.

Noting that the usage of the frequency band is sparse, we can try to use a sparse FFT to figure out where the spectrum is being used. Then we can use a few low speed ADC’s with coprime sampling rates to estimate large frequencies with our sparse-FFT with aliasing. In practice, this method is fairly successful: we can sense frequency usage in a 1 GHz band using 3 ADC’s sampling at tens of MS/s. Details in [3] (see also [4]).
3 Future Directions

3.1 Summary of Results

Below (Figure 3) is a table of the current results in sparse Fourier transforms. Seeing results in a table like this, a natural question is whether we can do better, and, if so, how much better? While time lower bounds are difficult, there is hope for sample lower bounds. For average-case inputs, we have algorithms that match sample complexity lower bounds. However, for worst case inputs there is a gap between the best algorithms and the sample-complexity lower bounds. For example, a sample lower bound for the robust flat window scheme is \(O(k \log \frac{\theta}{\delta})\). Note that currently our robust flat window scheme has an extra logarithmic factor in the sample complexity, so a current open question is to close this gap.
3.2 Question 1: Lower Sample Complexity through Better “Windows?”

Consider our “window” approach. The flat windows we use are sines multiplied by Gaussians. It is necessary that the support of this in the frequency domain be small for accurate recovery – otherwise, when we convolve the frequency representation with the frequency representation of our signal, we want the $k$ large coefficients to not get summed up with each other. Hence, we know we want the support of our flat window in the frequency domain to be at most $n/k$ so that we have a good chance of no two Fourier coefficients falling within the same window. In order to optimize sample complexity, we also want our support in the time domain to be small. Recall that we compute the product in the time domain to convolve in the frequency domain – computing this product requires us to take samples of our signal at every point for which the window is non-zero in the time domain.

The fact that we’d like to optimize sample complexity while preserving correctness leads naturally to the following question: are there any functions with small time domain support and small frequency domain support? Specifically, our flat windows had $|\{g_i > \frac{1}{n^2}\}| \times |\{|\hat{g}_i| > \frac{1}{n^2}\}| = O(n \log n)$. Are there any functions $f$ that have $|\{|f_i| > \frac{1}{n^2}\}| \times |\{|\hat{f}_i| > \frac{1}{n^2}\}| = o(n \log n)$?

The answer is yes – the spike train (which transforms into a spike train) satisfies $|\{|f_i| > 0\}| \times |\{|\hat{f}_i| > 0\}| = n$, which is actually the lowest this product can go by the time-frequency uncertainty principle. The problem is, if two coefficients end up going to the same bin, it means that the period of the spike train is divisible by the aliasing value. This will not change by doing any affine permutation, so we cannot spread out the coefficients with an affine permutation. Hence, we want to find a function other than the spike train with a small support. Suppose $n$ is prime (this eliminates the possibility of a spike train). Is there a non-trivial function $f$ such that $|\{|f_i| > \frac{1}{n^2}\}| \times |\{|\hat{f}_i| > \frac{1}{n^2}\}| = o(n \log n)$?

3.3 Question 2: Uniform Bounds

Suppose we would like a deterministic algorithm. It is known that there is a set of $O(k \log^4 n)$ samples that works for all signals, although the recovery time is $n \text{polylog} n$ (this follows from the
general theory of compressive sensing which we will cover in late October). The fastest deterministic sublinear time algorithm has $k^2 \text{polylog}_n$ complexity [5]. The rough idea is that if we could alias with $k\text{polylog}_n$ primes, most of them would not cause collisions upon aliasing, since two distinct coefficients can be the same modulo a prime for at most $O(\log n)$ primes. Unfortunately, there are not $k\text{polylog}_n$ primes smaller than $n$ – we make up for this by sampling a continuous-time, interpolated version of the signal with $k\text{polylog}_n$ prime rates to simulate aliasing by primes. Question: Can we get $k^{2-a}\text{polylog}_n$ for some $a > 0$?

3.4 Question 3: Block-Sparsity

Suppose that the $k$ large coefficients occupy at most $k/B$ blocks. Does this help? In November, we’ll learn about model-based compressive sensing, which helps reduce $k \log n$ to $k$ for $B > \log n$ in the robust case, but requires more general samples. There is a difficulty that arises: in this model-based approach we’re trying to take advantage of additional structure to the Fourier coefficients. However, the pseudo-random permutations we’ve used destroy this structure. Aliasing, on the other hand, will preserve the block structure...

References


