Overview of Fourier sampling over the Boolean cube

Eric Blais
University of Waterloo

October 18, 2014

Sublinear-time Fourier transform algorithms

The two big questions for this talk:

1. When the spectrum of $f : \{0,1\}^n \to \mathbb{R}$ is sparse, can we compute its Fourier transform in sublinear time?

2. Can we test if $f : \{0,1\}^n \to \mathbb{R}$ has a sparse spectrum in sublinear time?

(Spoiler: Yes, and yes.)

Boolean functions

$$f : \{0, 1\}^n \rightarrow \{0, 1\}$$

Boolean functions

$$f : \{0,1\}^n \to \mathbb{R}$$

Fourier transform of Boolean functions

Definition (Parity functions)

For any $S \subseteq [n]$, the function $\chi_S : \{0,1\}^n \to \{-1,1\}$ is defined by

$$\chi_S(x) = (-1)^{\sum_{i \in S} x_i}.$$

- Notation: $[n] := \{1, 2, \ldots, n\}$.
- Parity functions are also known as *linear functions* or *characters*.
- The parity functions form an orthonormal basis of functions mapping $\{0,1\}^n$ to $\mathbb{R}$ under the inner product

$$\langle f, g \rangle = \mathop{\mathrm{E}}_{x}[f(x)g(x)].$$

Fourier transform of Boolean functions

Definition (Fourier coefficients)

The Fourier coefficient of $f : \{0,1\}^n \to \mathbb{R}$ corresponding to $S \subseteq [n]$ is

$$\hat{f}(S) = \mathop{\mathrm{E}}_{x}\left[f(x)\chi_S(x)\right].$$

Theorem (Fourier inversion formula)

Every function $f : \{0,1\}^n \to \mathbb{R}$ can be represented as

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S)\chi_S(x).$$

Fourier transform of Boolean functions

Theorem (Plancherel's identity)

For every two functions $f, g : \{0,1\}^n \to \mathbb{R}$,

$$\mathop{\mathrm{E}}_{x}\left[f(x)g(x)\right] = \sum_{S \subseteq [n]} \hat{f}(S)\hat{g}(S).$$

Corollary (Parseval's identity)

For every function $f : \{0,1\}^n \to \mathbb{R}$,

$$\mathop{\mathrm{E}}_{x}[f(x)^2] = \sum_{S \subseteq [n]} \hat{f}(S)^2.$$

Part I: Sparse Fourier transforms for Boolean functions

The question

Algorithm model:

- Can query the value of $f(x)$ at any input $x \in \{0,1\}^n$.
- Randomized algorithm, can fail with probability $\leq \delta$.

Input assumptions:

- Consider only **bounded** functions $f : \{0,1\}^n \to [0,1]$.
- f is s-*sparse*: it has at most s non-zero Fourier coefficients.

Question: How efficiently can we estimate the Fourier coefficients of f up to additive error $\pm\epsilon$?

Query-efficient Fourier transform

Theorem

Let $\mathcal{F}$ be a family of subsets of $[n]$ that is given to the algorithm and contains all the non-zero Fourier coefficients of $f : \{0,1\}^n \to [0,1]$. Then the algorithm can approximate its Fourier transform with $q = O(\frac{1}{\epsilon^2} \log \frac{|\mathcal{F}|}{\delta})$ queries.

Query-efficient Fourier transform

Theorem

Let $\mathcal{F}$ be a family of subsets of $[n]$ that is given to the algorithm and contains all the non-zero Fourier coefficients of $f : \{0,1\}^n \to [0,1]$. Then the algorithm can approximate its Fourier transform with $q = O(\frac{1}{\epsilon^2} \log \frac{|\mathcal{F}|}{\delta})$ queries.

Proof.

Hoeffding/Chernoff bound + the following simple algorithm:

- Draw q elements $x^{(1)}, \ldots, x^{(q)} \in \{0,1\}^n$ independently and uniformly at random.
- Estimate $\tilde{f}(S) = \frac{1}{q} \sum_{i=1}^{q} f(x^{(i)}) \chi_S(x^{(i)})$ for each $S \in \mathcal{F}$.

□

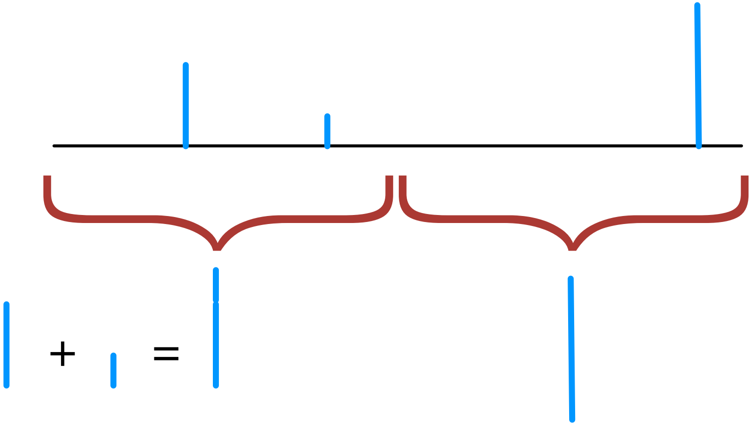
Query-efficient Fourier transform

Theorem

Let $\mathcal{F}$ be a family of subsets of $[n]$ that is given to the algorithm and contains all the non-zero Fourier coefficients of $f : \{0,1\}^n \to [0,1]$. Then the algorithm can approximate its Fourier transform with $q = O(\frac{1}{\epsilon^2} \log \frac{|\mathcal{F}|}{\delta})$ queries.

- So we can estimate *all* the Fourier coefficients of $f : \{0,1\}^n \to [0,1]$ with $q = O(\frac{n}{\epsilon^2} + \frac{1}{\epsilon^2} \log \frac{1}{\delta})$ queries. (!)
- ... but the running time of the simple algorithm is $\Omega(|\mathcal{F}|) = \Omega(2^n)$ in this case.
- Taking $\mathcal{F} = \{S \subseteq [n] : |S| \leq k\}$ yields the *Low-Degree Algorithm*. [Linial, Mansour, Nisan '93]

Better running time through binary search?



The key lemma

Lemma

Fix $1 \leq k \leq n$ *and let* $H = \{y \in \{0,1\}^n : y_{k+1} = \cdots = y_n = 0\}$. *For any* $T \subseteq [k]$,

$$\sum_{S \subseteq [n] : S \cap [k] = T} \hat{f}(S)^2 = \mathop{\mathrm{E}}_{x \in \{0,1\}^n, y \in H} \left[f(x) f(x \oplus y) \chi_T(y) \right].$$

Proof idea: Let $P_T f : \{0,1\}^n \to \mathbb{R}$ be the projection of the function obtained by defining

$$\widehat{P_T f}(S) = \begin{cases} \hat{f}(S) & \text{if } S \cap [k] = T \\ 0 & \text{otherwise.} \end{cases}$$

The key insight is that $P_T f(x) = \mathrm{E}_{y \in H}[f(x+y)\chi_T(y)]$. The proof follows from Parseval's identity and elementary rearranging.

The key lemma

Lemma

Fix $1 \le k \le n$ *and let* $H = \{y \in \{0,1\}^n : y_{k+1} = \cdots = y_n = 0\}$*.*
For any $T \subseteq [k]$*,*

$$\sum_{S \subseteq [n] : S \cap [k] = T} \hat{f}(S)^2 = \mathop{\mathrm{E}}_{x \in \{0,1\}^n, y \in H} \left[f(x) f(x \oplus y) \chi_T(y) \right].$$

Consequence:

- We can estimate $\sum_{S \subseteq [n] : S \cap [k] = T} \hat{f}(S)^2$ to accuracy $\pm\epsilon$ with $q = O(\frac{1}{\epsilon^2}, \log \frac{1}{\delta})$ queries.

Time-efficient Fourier transform

Theorem (Goldreich-Levin / Kushilevitz-Mansour)

Let $f : \{0,1\}^n \to [0,1]$ have an s-sparse spectrum. There is an algorithm that approximates the Fourier transform of f in time $O(\frac{ns}{\epsilon^4} \log \frac{1}{\delta})$.

Proof.

1. Initialize $k = 0$ and $\mathcal{L}_0 = \{\emptyset\}$.
2. While $k \leq n$,
3. For each $T \in \mathcal{L}_k$,
4. Add T to $\mathcal{L}_{k+1}$ if $\sum_{S:S\cap[k+1]=T} \hat{f}(S)^2 \geq \frac{\epsilon^2}{2}$.
5. Add $T \cup \{k+1\}$ to $\mathcal{L}_{k+1}$ if $\sum_{S:S\cap[k+1]=T\cup\{k+1\}} \hat{f}(S)^2 \geq \frac{\epsilon^2}{2}$.
6. Return the estimates for each Fourier coefficient in $\mathcal{L}_n$.

Applications

The Low-Degree Algorithm can be used to learn

- DNFs
- Decision lists
- k-juntas (Time: $O(n^k)$.)
- AC^0 circuits (Time: $O(n^{\mathrm{polylog}(n)})$.)

The GL/KM Algorithm can be used to learn

- Parity decision trees.
- k-juntas in time $O(2^k + \mathrm{poly}(n))$.

Remarks

1. Both the Low-Degree Algorithm and GL/KM Algorithm work in the more general setting where f is only promised to be *close* to s-sparse.

2. Despite implementing a binary search approach, the GL/KM Algorithm is *non-adaptive*; all the queries can be selected in advance.

3. Can the GL/KM algorithm be extended/modified to work in settings where there are strong restrictions on the set of allowed queries?

Part II: Testing Fourier sparsity

The question

- The function $f : \{0,1\}^n \to \mathbb{R}$ is *s-sparse* if it has at most s non-zero Fourier coefficients.
- The function $f : \{0,1\}^n \to \mathbb{R}$ is *ϵ-far from s-sparse* if for every function $g : \{0,1\}^n \to \mathbb{R}$ that is s-sparse,

$$\mathrm{dist}(f,g) := \mathop{\mathrm{E}}_{x}[(f(x) - g(x))^2] \geq \epsilon.$$

Question: How efficiently can we test whether $f : \{0,1\}^n \to [0,1]$ is s-sparse or ϵ-far from s-sparse?

Testing with the Fourier transform

Theorem

There is an algorithm for testing whether $f : \{0,1\}^n \to [0,1]$ is s-sparse or ϵ-far from s-sparse with $O(\frac{ns^3}{\epsilon^2})$ queries.

Algorithm:

- Run the Goldreich-Levin algorithm to compute the Fourier coefficients of f with accuracy $\pm\sqrt{\frac{\epsilon}{4s}}$.
- If the function returned by the GL algorithm is not s-sparse, reject.
- Otherwise, estimate $\mathrm{E}_x[(f(x) - g(x))^2]$ with $O(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$ queries and accept iff this distance is less than $\frac{\epsilon}{2}$.

A more efficient test

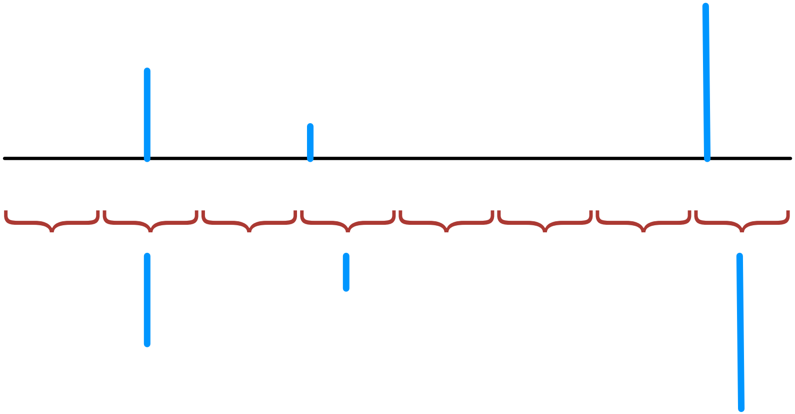
Theorem (Gopalan, O'Donnell, Servedio, Shpilka, Wimmer)

There is an algorithm for testing whether $f : \{0,1\}^n \to [0,1]$ *is* s*-sparse or* ϵ*-far from* s*-sparse with* $\mathrm{poly}(s, \frac{1}{\epsilon}, \log \frac{1}{\delta})$ *queries.*

Remarks

- Note that the query complexity is *independent* of n.
- In particular, this implies that the algorithm cannot estimate the Fourier coefficients of f—or even identify which ones are the non-zero coefficients.

Idea for the GOSSW test



Proof components for the GOSSW Theorem

Definition (Random hashing)

Draw $x^{(1)}, \ldots, x^{(t)} \in \{0,1\}^n$ uniformly and independently at random. For each string $z \in \{0,1\}^t$, define

$$\mathcal{B}(z) = \{S \subseteq [n] : \chi_S(x^{(i)}) = z_i \text{ for every } i \in [t]\}.$$

Lemma (Key ingredient 1)

For every $z \in \{0,1\}^t$ and distinct sets $S, T \subseteq [n]$,

1. $\Pr[S \in \mathcal{B}(z)] = 2^{-t}$.
2. $\Pr[S \in \mathcal{B}(z) \mid T \in \mathcal{B}(z)] = 2^{-t}$.
3. *Fix a family $\mathcal{F}$ of k subsets of $[n]$. If $t \geq 2\log k + \log \frac{1}{\delta}$, then the sets in $\mathcal{F}$ all land in different buckets except with probability at most δ.*

Proof components for the GOSSW Theorem

Lemma (Key ingredient 2)

For any $z \in \{0,1\}^t$,

$$\sum_{S \in \mathcal{B}(z)} \hat{f}(S)^2 = \mathop{\mathrm{E}}_{x \in \{0,1\}^n, y \in H} \left[f(x) f(x \oplus y) \chi_T(y) \right],$$

where $H = \left\{ y \in \{0,1\}^n : (-1)^{y \cdot x^{(i)}} = 1 \text{ for every } i \in [t] \right\}$ *and* $x^{(1)}, \ldots, x^{(t)} \in \{0,1\}^n$ *are the elements drawn in the random hashing process.*

Proof sketch of the GOSSW Theorem

Theorem (Gopalan, O'Donnell, Servedio, Shpilka, Wimmer)

There is an algorithm for testing whether $f : \{0,1\}^n \to [0,1]$ *is* s*-sparse or* ϵ*-far from* s*-sparse with* $\mathrm{poly}(s, \frac{1}{\epsilon}, \log \frac{1}{\delta})$ *queries.*

Algorithm:

- Randomly hash the subsets of $[n]$ into buckets $\mathcal{B}(z)$, $z \in \{0,1\}^t$ for $t = O(\log s)$.
- Estimate $\sum_{S \in \mathcal{B}(z)} \hat{f}(S)^2$ for each $z \in \{0,1\}^t$.
- Accept iff at most s buckets have total weight at least $\frac{\epsilon}{100 \cdot 2^t}$.

Proof sketch of the GOSSW Theorem

Algorithm:

- Randomly hash the subsets of $[n]$ into buckets $\mathcal{B}(z)$, $z \in \{0,1\}^t$ for $t = O(\log s)$.
- Estimate $\sum_{S \in \mathcal{B}(z)} \hat{f}(S)^2$ for each $z \in \{0,1\}^t$.
- Accept iff at most s buckets have total weight at least $\frac{\epsilon}{100 \cdot 2^t}$.

Analysis:

- If f is s-sparse, the algorithm accepts whenever the estimates are accurate.
- If f is far from s-sparse because it has $s+1 > s$ large Fourier coefficients, then with high probability they are separated by the random hash and the test rejects.
- Otherwise, if f is far from s-sparse because it has many small coefficients, then with high probability many of the buckets have weight at least $\frac{\epsilon}{100 \cdot 2^t}$.

Remarks

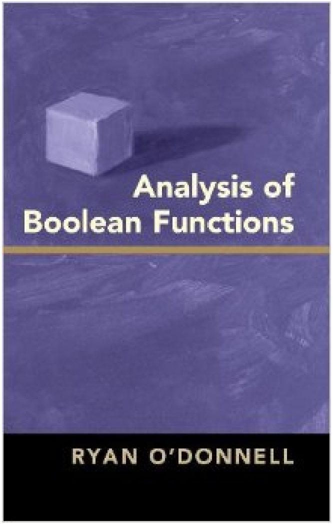
1. The query complexity of the GOSSW algorithm is roughly $O(s^{14})$. Can we do better?

   (Spoiler: yes!)

2. The function $f : \{0,1\}^n \to \mathbb{R}$ has *Fourier dimension* d if $f(x) = g(\chi_{S_1}(x), \chi_{S_2}(x), \ldots, \chi_{S_d}(x))$ for some function $g : \{0,1\}^d \to \mathbb{R}$ and subsets $S_1, \ldots, S_d \subseteq [n]$. Gopalan et al. showed that we can test d-dimensionality with $O(d2^{2d})$ queries.

3. The GOSSW algorithm requires membership query access. Can we still define an efficient Fourier sparsity tester in more restricted query models?

Further reading



Thank you!