Efficiently Learning Fourier Sparse Set Functions

ETH zürich EPFL

Andisheh Amrollahi*, Amir Zandieh*, Michael Kapralov, Andreas Krause

Learning & Adaptive Systems

*: the first two authors contributed equally

Overview

Given: set function $F: 2^V \to \mathbb{R}$ .Can make (noisy) value queries

Goal: "learn" F with as few queries as possible

Assumptions: 1) sparsity: at most k non-zero Fourier coefficients
2) low frequency: degree of Fourier polynomials at most d

Motivation

Can represent a set function F where t is the indicator variable of the subsets

$$x_t : \{0,1\}^n \to \mathbb{R} \quad x_t = \sum_{f \in \{0,1\}^n} \hat{x}_f (-1)^{\langle f,t \rangle}$$

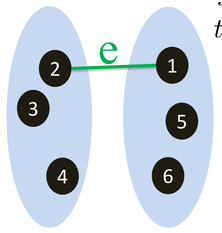
Fourier basis functions

1. Graph cuts

$f_e =$ 1 1 0 0 0 0

$t_A =$ 0 1 1 1 0 0

Frequency degree d = 2



$$x_t = \frac{1}{2}(1 - (-1)^{\langle f_e, t_A \rangle})$$

One frequency for each edge

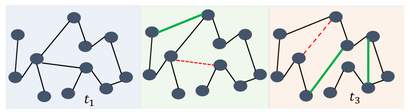
$k = |E| + 1$

$A = \{2,3,4\} \quad V \setminus A = \{1,5,6\}$

$F(A) = 3$

Size of the cut between A and V \ A

Graph sketching



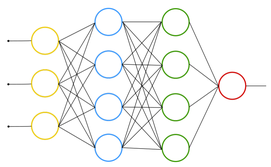
2. Black box optimization

$F(x_1 x_2 x_3 x_4 x_5) =$

$F_1(x_1) + F_2(x_3 x_4) + F_3(x_2 x_5)$

$d = 2$

$F(x_1 x_2 x_3 x_4 x_5)$ = Validation error using hyperparameters **x**



[5]

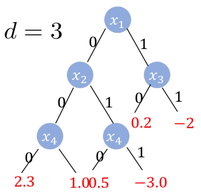
$x_1 = \begin{cases} 0 \text{ if optimizer is ADAM} \\ 1 \text{ if optimizer is SGD} \end{cases}$

$x_2 = \begin{cases} 0 \text{ if filtersize} = 3x3 \\ 1 \text{ if filtersize} = 5x5 \end{cases}$

...

3. Decision Trees

$d = 3$



Intuitively a decision tree of depth d captures order d interactions

Techniques

$$h(\tilde{f}) = \sigma^\top \tilde{f} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tilde{f}$$

1. Hashing

$x_t : \{0,1\}^n \to \mathbb{R} \quad \hat{x}_f : \{0,1\}^n \to \mathbb{R}$

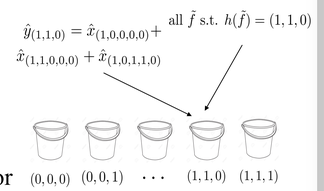
$\sigma \in \{0,1\}^{n \times b}$ each element is chosen independent uniformly at random

Hash function $h(f) \triangleq \sigma^\top f$ ⟵ Pairwise independent

$B = 2^b$ # of Buckets ⟵ $B \approx O(k)$

Avoid randomness assumptions on support

$y_t = x_{\sigma t} \implies \hat{y}_f = \sum_{\tilde{f}: h(\tilde{f}) = f} \hat{x}_f$



2. Frequency detection

Let $a \in \{0,1\}^n$ be a shift (measurement) vector

$y_t = x_{\sigma t + a} \implies \hat{y}_f = \sum_{\tilde{f}: h(\tilde{f}) = f} \hat{x}_f (-1)^{\langle a, f \rangle}$

Fix σ and repeat for carefully chosen values of a such that every frequency f is recoverable from linear measurements $\langle a_i, f \rangle$

Solutions:

1- Set $a_i = e_i, \quad i \in [n]$ #measurements = $O(n)$

2- If frequency is one-sparse do binary search

$f =$ 0 0 0 1 0 0 0 0 #measurements= $O(\log n)$

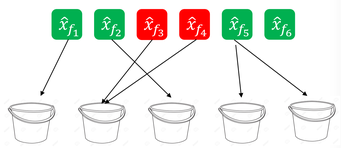
3- If frequency is d-sparse, hash and do binary search

#measurements= $O(d \log n)$

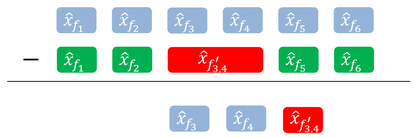
Compressive sensing over finite fields for better sampling complexities

3. Peeling

We allow for collisions which result in erroneous frequency recovery for colliding frequencies



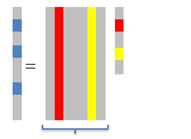
As long as there are not too many collisions sparsity increases after subtraction



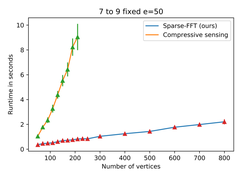
Results

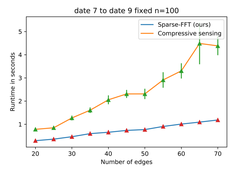
	Compressive sensing	Sparse FFT	Ours
Runtime	$\tilde{O}(kn^d)$ [2][3]	$\tilde{O}(kn^{(2)})$ [1][4]	$\tilde{O}(kn)$
Sampling complexity	[2][3] $O(k(\log k)^2 \log(n^d))$	[1][4] $O(kn)$	$O(k \log(n^d))$
Assumptions	None	Randomness of support	None
Robustness	Worst case noise	Gaussian noise + binary valued support	Worst case noise

- Obtain best of both worlds algorithm where runtime scales linear in d (sparse FFT) while maintaining optimal sampling complexities (compressive sensing)
- Remove assumptions on randomness of support
- Handle worst case noise



Experiments





- Track evolution of a graph through time by looking at the values of a few cuts
- Runtimes scale linear in number of nodes and edges

Table 1: Sampling and computational complexity

No. of vertices	CS method Runtime	CS method Samples	Our method Runtime	Our method Samples
70	1.14	767	0.85	6428
90	1.88	812	0.92	6490
110	3.00	850	0.82	6491
130	4.31	880	1.01	7549
150	5.34	905	1.16	7942
170	6.13	927	1.22	7942
190	7.36	947	1.18	7271
210	8.24	965	1.28	7271
230	*	*	1.38	7942
250	*	*	1.38	7271
300	*	*	1.66	8051
400	*	*	2.06	8794
500	*	*	2.42	8794
600	*	*	3.10	9646
700	*	*	3.35	9646
800	*	*	3.60	9646

References

[1]: R. Scheibler et al. , "A Fast Hadamard Transform for Signals With Sublinear Sparsity in the Transform Domain," in *IEEE Transactions on Information Theory*, 2015.

[2]: S. Peter, and A. Krause. "Learning Fourier sparse set functions." *Artificial Intelligence and Statistics,* 2012.

[3]: I. Haviv, and O. Regev. "The restricted isometry property of subsampled Fourier matrices". *Geometric Aspects of Functional Analysis, 2017*

[4]:X. Li, J. K. Bradley, S. Pawar and K. Ramchandran, "The SPRIGHT algorithm for robust sparse Hadamard Transforms," *IEEE International Symposium on Information Theory,* 2014.

[5]:*Hazan, Elad, Adam Klivans, and Yang Yuan. "Hyperparameter Optimization: A Spectral Approach." ICLR (2018).*