6.893: Algorithms and Signal Processing

Piotr Indyk
What this class is about

• Algorithms
  – Binary search
  – Dynamic programming
  – Hashing
  – Randomization
  – Approximation algorithms
  – Streaming/sketching
  – Clustering
  – ...

• Signal processing
  – Fourier transform
  – Wavelets
  – Sparse approximations
  – Signal models
  – Compressive sensing
  – Vector quantization
  – ...

  – ....
Specific topics

- Sparse Fourier Transform
- Non-uniform Fourier Transform
- Prony/Matrix Pencil Method
- Other sparse approximation methods (wavelets, etc)
- (Model-based) compressive sensing
- Distance/vector quantization
- ....

Will post an approximate calendar and list of papers on the course website
What this class is not about

• Not an introduction to signal processing
• Not focused on applications (although will cover some)
• Not a TQE class
Course logistics

- Prerequisites: familiarity with algorithms (at 6.046 level) and basic signal processing (at 6.003 level)

- Deliverables:
  - Scribe notes (both listeners and registered students)
    - Will pass around the signup sheet on Wednesday
  - Project
    - Proposals due mid October
    - Short project presentations in December
Sparsity

- Signal $a = a_0 \ldots a_{n-1}$
- Sparse approximation: approximate $a$ by $a'$ parametrized by $k$ coefficients, $k \ll n$
- Applications:
  - Compression
  - Denoising
  - ....
Sparse approximations – transform coding

• Transform coding:
  – Compute $T_a$ where $T$ is an $n \times n$ orthonormal matrix
  – Compute $H_k(T_a)$ by keeping only the largest* $k$ coefficients of $T_a$
  – $T^{-1}(H_k(T_a))$ is a $k$-sparse approximation to a w.r.t. $T$

*In magnitude
Transform coding: examples

Fourier Transform + thresholding + Inverse FT

Wavelet Transform + thresholding + Inverse WT

Sparsity $k=0.1 \ n$
Other sparse approximations

- Histograms, a.k.a., piece-wise constant approximations
- Piece-wise polynomials
- Trigonometric polynomials
- ....

This course: how to compute all of this efficiently
Fourier Transform

• Input (time domain): \(a=a_0 \ldots a_{n-1}\)
  – Today: \(n=2^l\)

• Output (frequency domain): \(\hat{a}=\hat{a}_0 \ldots \hat{a}_{n-1}\), where
  \[
  \hat{a}_u = \frac{1}{n} \sum_j a_j e^{-2\pi i/n u_j}
  \]

• Notation: \(\omega = \omega_n = e^{2\pi i/n}\) is the n-root of unity

• Then
  \[
  \hat{a}_u = \frac{1}{n} \sum_j a_j \omega^{-uj}
  \]

• Matrix notation: \(\hat{a} = F \ a\)
  - \(F_{uj}=1/n \ \omega^{-uj}\)
  - \(F^{-1}_{ju} = \omega^{uj}\)
Algorithms

• Fast Fourier Transform: computes \( \hat{a} \) from a (and vice versa) in \( O(n \log n) \) time

• \( H_k(\hat{a}) \) can be then computed in \( O(n) \) time via selection algorithms

• Sparse Fourier Transform:
  – Directly computes \( k \) largest coefficients of \( \hat{a} \) (approximately - formal definition in a moment)
  – Running time: \( O(k \log^2 n) \) or faster
  – Sub-linear time

• Today: \( k=1 \)
Sparse Fourier Transform (k=1)

• Warmup: \( \hat{a} \) is exactly 1-sparse, i.e., \( \hat{a}_{u'} = 0 \) for all \( u' \neq u \), for some \( u \)
• I.e., the signal is “pure” – only one frequency is present
• Need to find \( u \) and \( \hat{a}_u \)
Two-point sampling

- We have
  \[ a_j = \hat{a}_u \, \omega^{uj} \]
- Sample \( a_0 \), \( a_1 \)
- Observe:
  - \( a_0 = \hat{a}_u \)
  - \( a_1 = \hat{a}_u \, \omega^u \Rightarrow a_1/a_0 = \omega^u \)
  - Can read \( u \) from the angle of \( \omega^u \)
- Constant time algorithm!
- Unfortunately, it relies heavily on signal being pure
What if $\hat{a}$ is not 1-sparse?

• Ideally, would like to find 1-sparse $\hat{a}^*$ that contains the largest entry of $\hat{a}$

• We will need to allow approximation: compute a 1-sparse $\hat{a}'$ such that

$$||\hat{a}-\hat{a}'||_2 \leq C \ ||\hat{a}-\hat{a}^*||_2$$

where $C$ is the approximation factor

– L2/L2 guarantee

• Note that the guarantee is meaningful only for signals where $C||\hat{a}-\hat{a}^*||_2 < ||\hat{a}||_2$

• …or, equivalently, $\Sigma_{u \neq u'} \hat{a}_u^2 < \epsilon \hat{a}_u^2$

• We will assume this holds for small enough $\epsilon$
Finding the “heavy hitter” in â

- Will describe the algorithm assuming the exactly 1-sparse case first, and deal with epsilons later
- Suppose $a_j = \hat{a}_u \omega^{uj}$
- We will find $u$ bit by bit (binary search)
Bit 0: compute $u \mod 2$

- Suppose $u=2v+b$, we want $b$
- Compute:
  - $a_{0+r} = \hat{a}_u \omega^{ur}$
  - $a_{n/2+r} = \hat{a}_u \omega^{u n/2} \omega^{ur} = \hat{a}_u \omega^{2v n/2 + b n/2} \omega^{ur} = \hat{a}_u (-1)^b \omega^{ur}$
- Test: $b = 0$ iff $|a_0 - a_{n/2}| < |a_0 + a_{n/2}|$

Actual test:

$$b=0 \text{ iff } |a_r - a_{n/2+r}| < |a_r + a_{n/2+r}|,$$
where $r$ is chosen uniformly at random from $0\ldots n-1$
Bit 1

• Can pretend that \( b = u \mod 2 = 0 \). Why?
  – Claim: if \( a'_j = a_j \omega^{bj} \) then \( â'_u = â'_{u-b} \)
    (“Time shift” theorem. Follows directly from the definition of FT)
  – If \( b=1 \) then we replace \( a \) by \( a' \)

• Since \( u = 2v \), we have
  \[
  a_j = â_u \omega^{uj} = â_{2v} \omega^{2vj} = â'_v \omega^{n/2 \cdot vj}
  \]
  for a frequency vector \( â'_v = â_{2v} \), \( v = 0 \ldots n/2-1 \)

• So, \( a_0 \ldots a_{n/2-1} \) are time samples of \( â'_0 \ldots â'_{n/2-1} \), and \( â'_v \) is the heavy hitter in \( â' \)

• It suffices to compute bit 0 of \( v \) by sampling \( a_0 \ldots a_{n/2-1} \)
• I.e., \( v = 0 \mod 2 \) iff \( |a_r - a_{n/4+r}| < |a_r + a_{n/4+r}| \)
Bit reading recap

- Bit $b_0$: test if
  $$|a_r - a_{n/2+r}| < |a_r + a_{n/2+r}|$$

- Bit $b_1$: test if
  $$|a_r - a_{n/4+r} \omega^{(n/4+r)} b_0| < |a_r + a_{n/4+r} \omega^{(n/4+r)} b_0|$$

- Bit $b_2$: test if
  $$|a_r - a_{n/8+r} \omega^{(n/8+r)} (b_0 + 2b_1)| < |a_r - a_{n/8+r} \omega^{(n/8+r)} (b_0 + 2b_1)|$$

- ...

- Altogether, we use $O(\log n)$ samples to identify $u$

- $O(\log n)$ time algorithm in the 1-sparse case
Dealing with epsilons

- We now have
  \[ a_j = \hat{a}_u \omega_{uj} + \sum_{u' \neq u} \hat{a}_{u'} \omega_{u'j} \]
  \[ = \hat{a}_u \omega_{uj} + \mu_j \]

- Observe that \( \sum \mu_j^2 = n \sum_{u' \neq u} \hat{a}_{u'}^2 \)

- Therefore, if we choose \( r \) at random from 0…n-1, we have
  \[ E_r[\mu_r^2] = \sum_{u' \neq u} \hat{a}_{u'}^2 \]

- Since \( \sum_{u' \neq u} \hat{a}_{u'}^2 < \epsilon \hat{a}_u^2 \), it follows that
  \[ E[\mu_r^2] < \epsilon \hat{a}_u^2 \]
Dealing with epsilons, ctd

• Claim: For all values of $\mu_r$, $\mu_{n/2+r}$ satisfying
  
  \[ 2 (|\mu_r|+|\mu_{n/2+r}|) < |\hat{\alpha}_u| \]

  the outcome of the comparison
  
  \[ |a_r-a_{n/2+r}| < |a_r+a_{n/2+r}| \]

  is the same.

• From Markov inequality we have
  
  \[ \Pr_r [ |\mu_r| > |\hat{\alpha}_u|/4 ] = \Pr_r [ |\mu_r|^2 > |\hat{\alpha}_u|^2/16 ] \leq 16 \epsilon \]

• Corollary: If $16\epsilon < 1/6$, then each bit test is correct with probability $>2/3$
Finding the heavy hitter: algorithm/analysis

• For each bit $b_i$, make $O(\log \log n)$ independent tests, and use majority vote

• From Chernoff bound, the probability that $b_i$ is estimated incorrectly is $< 1/(4 \log n)$

• The probability that the coordinate is estimated incorrectly is at most

$$\frac{\log n}{4 \log n} = \frac{1}{4}$$

• Total complexity: $O(\log n \times \log \log n)$
Estimating the value of the heavy hitter

- Recall that
  \[ a_j = \hat{a}_u \omega^{uj} + \sum_{u' \neq u} \hat{a}_{u'} \omega^{u'j} \]
  \[ = \hat{a}_u \omega^{uj} + \mu_j \]
  where \( E[\mu_r^2] = \sum_{u' \neq u} \hat{a}_{u'}^2 = ||\hat{a} - \hat{a}^*||_2 \)

- By Markov inequality
  \[ \Pr_r [ |\mu_r| > 2 ||\hat{a} - \hat{a}^*||_2 ] \leq \frac{1}{4} \]

- In this case, the estimate \( \hat{a}'_u = a_j \omega^{-uj} \) satisfies
  \[ |\hat{a}'_u - \hat{a}_u| \leq 2 ||\hat{a} - \hat{a}^*||_2 \]

- If we set \( \hat{a}'_u = 0 \) for all \( u' \neq u \) then
  \[ ||\hat{a} - \hat{a}'||_2 \leq |\hat{a}'_u - \hat{a}_u| + ||\hat{a} - \hat{a}^*||_2 \leq 3||\hat{a} - \hat{a}^*||_2 \]
  so \( \hat{a}' \) satisfies the L2/L2 guarantee from slide 14