0368.416701 Sublinear Time Algorithms	04 Jan, 2016
Lecture 12	
Lecturer: Ronitt Rubinfeld	*Scribe: Maor Rahamim, Nimrod Busany, Chen Liderman*

1 Estimating the Sum of Powers of Degree One Fourier Coefficients

1.1 Reminders

On the previous lecture, we have showed the following properties for Boolean functions $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$

Definition 1 $\chi_s(x) = \Pi_{i \in S} x_i$

Definition 2 $\hat{f}(s) = \langle f, \chi_s \rangle = \frac{1}{2^n} \sum_x f(x) \chi_s(x)$

Theorem 3 $f(x) = \sum_s \hat{f}(s) \chi_s(x)$

Theorem 4 *Parseval / Plancherel:*
$\langle f, g \rangle = \sum_{S,T} \hat{f}(S) \hat{g}(T) \langle \chi(S), \chi(T) \rangle$
$\langle f, f \rangle = \sum \hat{f}(S)^2 = 1$, *if f is Boolean (since* $\langle \chi_S, \chi_S \rangle = 1$*)*

1.2 Estimating the Sum of Powers of Degree One Fourier Coefficients

In this section we are interested in estimating the sum of powers of degree one Fourier coefficients. An example for a use of this sum is testing if a function is a dictator function. For such a test, this sum can indicate if a function is determined by a single bit. Furthermore, the approach we present here for estimating the sum of degree one powers, can be used to iterative estimate sums of any degree.

Let us denote by $f(i)$, $\hat{f}(s)$ for $|S| = 1$; We would like to estimate: $\sum_{i=1}^{n} \hat{f}(i)^{P \in \mathbb{N}}$
Let us propose the following algorithm, given a parameter $\eta \in \{0, 1\}$:

1. pick $X^{(1)}, X^{(2)}, ..., X^{(P-1)}$, randomly (uniformly) from $\{\pm 1\}^n$
2. pick noise vector μ^n s.t., each entry:
   $+1$, w.p. $\frac{1}{2} + \frac{\eta}{2}$
   -1, w.p. $\frac{1}{2} - \frac{\eta}{2}$
3. $y \leftarrow f(X^{(1)}) f(X^{(2)}) ... f(X^{(P-1)}) f(X^{(1)} \odot X^{(2)} ... \odot X^{(P-1)} \odot \mu)$
   $^*\odot$ is coordinate-wise multiplication
4. output y

Claim 5 $\mathbb{E}[y] = \sum_{S \subseteq [n]} \eta^{|S|} \hat{f}(s)^P$

Proof
$\mathbb{E}[y] = \mathbb{E}[f(X^{(1)}) f(X^{(2)}) ... f(X^{(P-1)}) f(X^{(1)} \odot X^{(2)} ... \odot X^{(P-1)} \odot \mu)] =_{using\ Thm.\ 3}$
$\mathbb{E}[(\sum_{S_1} \hat{f}(S_1) \chi_{S_1}(X^{(1)})) ... (\sum_{S_{P-1}} \hat{f}(S_{P-1}) \chi_{S_{P-1}}(X^{(P-1)}))(\sum_{S_P} \hat{f}(S_P) \chi_{S_P}(X^{(1)} \odot X^{(2)} ... \odot X^{(P-1)} \odot \mu))]$
$= \sum_{S_1, S_2, ..., S_P} \hat{f}(S_1) \hat{f}(S_2) ... \hat{f}(S_P) \mathbb{E}[\chi_{S_1 \Delta S_P}(X^{(1)}) \chi_{S_2 \Delta S_P}(X^{(2)}) ... \chi_{S_{P-1} \Delta S_P}(X^{(P-1)}) \chi_{S_P}(\mu)]$
$=_* \sum_{S_1, S_2, ..., S_P} \hat{f}(S_1) \hat{f}(S_2) ... \hat{f}(S_P) \mathbb{E}[\chi_{S_1 \Delta S_P}(X^{(1)})] \mathbb{E}[\chi_{S_2 \Delta S_P}(X^{(2)})] ... \mathbb{E}[\chi_{S_{P-1} \Delta S_P}(X^{(P-1)})] \mathbb{E}[\chi_{S_P}(\mu)]$
$\rightarrow_{**} \mathbb{E}[y] = \sum_S \hat{f}(S)^P \mathbb{E}[\chi_{S_P}(\mu)]$

* using the independence between the different vectors
** if $S_1 = S_2 = ... = S_P \to S_i \Delta S_P = \emptyset \to \mathbb{E}\left[\chi_\emptyset\right] = 1$, else some $S_i \neq S_P \to \mathbb{E}\left[S_i \Delta S_P\right] = 0$

Therefore, we ar left to compute $\mathbb{E}\left[\chi_{S_P}(\mu)\right]$:

$\mathbb{E}\left[\chi_{S_P}(\mu)\right] = \Pi_{i \in S_P} \mathbb{E}\left[\mu_i\right] =_{***} \eta^{|S_P|}$

*** $\mathbb{E}\left[\mu_i\right] = 1(\frac{1}{2} + \frac{\eta}{2}) - 1(\frac{1}{2} - \frac{\eta}{2}) = \eta$

$\Rightarrow \mathbb{E}\left[y\right] = \sum_{S \subseteq \left[n\right]} \eta^{|S|} \hat{f}(s)^P$

■

Let us note that the noise factor allows us to eliminate high-order Fourier coefficients as $\eta^{|S|}$ decays as $|S|$ increases.

1.3 Plan for estimating $\sum_i \hat{f}(i)^P$

Let us show how we can use our estimate of y to estimate $\sum_i \hat{f}(i)^P$; Based on our last observation it is clear that the sum of powers is effected the most by 0/1 degree Fourier coefficients. Therefore, we will try to approximate these terms and show that we can neglect high order terms. Let us consider the following algorithm:

1. Estimate $\mathbb{E}\left[f(X^{(1)})f(X^{(2)})...f(X^{(P)})\right] = \sum_{|S|=0} \eta^0 \hat{f}(S)^P = \hat{f}(\emptyset)^P$ to additive $\pm\frac{\eta^2}{2}$

   (by randomly sampling vectors from $\{\pm 1\}^n$, computing their f() values and calculating the avg.)

2. Estimate $\mathbb{E}\left[f(X^{(1)})f(X^{(2)})...f(X^{(P-1)})f(X^{(1)} \odot X^{(2)} ... \odot X^{(P-1)} \odot \mu)\right] =$
   $=_{(=\mathbb{E}\left[y\right])} \sum_{S \subseteq \left[n\right]} \eta^{|S|} \hat{f}(s)^P$ to additive $\pm\frac{\eta^2}{2}$

   (using the algorithm we have seen earlier)

Let us denote by $\gamma = \sum_{|S|>0} \eta^{|S|} \hat{f}(S)^P$, then we can obtain an additive $\pm\eta^2$ approximation of it by subtracting (2)-(1)

Claim 6 $\frac{\gamma}{\eta}$ *is a "good" estimate of* $\sum_{|S|=1} \hat{f}(S)^P$

Proof

$\sum_{|S|=1} \hat{f}(S)^P = \frac{\sum_{|S|=1} \eta \hat{f}(S)^P}{\eta} = \frac{\sum_{|S|>0} \eta^{|S|} \hat{f}(S)^P}{\eta}_{*} - \frac{\sum_{|S|>1} \eta^{|S|} \hat{f}(S)^P}{\eta}_{**}$

* $= \frac{\gamma}{\eta}$

** will need to show that this argument is small and can be ignored

$\sum_{|S|>1} \eta \hat{f}(S)^P \leq \eta^2 \sum_{|S| \geq 2} |\hat{f}(S)^P| \leq_{***} \eta^2 \sqrt{\sum_{|S| \geq 2} \hat{f}(S)^2}_{****} \sqrt{\sum_{|S| \geq 2} (\hat{f}(S)^{P-1})^2}_{****} \leq \eta^2$

*** cauchy-swartz inequality

**** ≤ 1, based on the Theorem for Boolean Parseval (see reminder)

Hence we get that:

$\sum_{|S|=1} \hat{f}(S)^P = \frac{\sum_{|S|>0} \eta^{|S|} \hat{f}(S)^P}{\eta} - \frac{\sum_{|S|>1} \eta^{|S|} \hat{f}(S)^P}{\eta} \leq \frac{\gamma}{\eta} - \frac{\eta^2}{\eta} = \frac{\gamma}{\eta} - \eta$

This concludes the proof.

■

To sum up, we showed that the algorithm provides an additive estimate of $\pm\eta$ to the sum of powers of degree one Fourier coefficients using $\frac{p}{\eta}$ queries.

2 Interactive Proofs

Assume we have some user U who wants to compute a function f on an input x. Furthermore assume the user is computationally bounded and cannot compute $f(x)$ on his own, but can outsource the computation to an "untrusted computation expert". The "expert" will return $f(x)$ and a "proof" that $f(x)$ is "good".

2.1 Website Hits

U owns a website, and a company C claims that at least k clicks were made through their website to enter U's website.

Goal: If the number of valid clicks is greater than k return "PASS" with high probability, and if the number of valid clicks is less than $(1-\epsilon)k$ return "FAIL" with high probability.

* Assume we have a way of verifying that a click is legal.

Protocol:

1. Check that there are at most $k\epsilon/2$ fake entries. ($O(1/\epsilon)$).
2. Check that there are at most $k\epsilon/2$ duplicates.

Consider the following proof: C will build a table T with all possible clicks description, $t_1, t_2, ..., t_n$ and send both tables T and X, where X is the array $x_1, ...x_k$. C will also send forward and back pointers both from T to X and from X to T in the following way:

1. For each cell $t_i \in T$, t_i is some click's possible description. If we have such a click in table X, then t_i will hold a pointer to that cell in X. Thus, we have a pointer from t_i to x_j if x_j's click description is exactly t_i . If we don't have a click in X with t_i's description, then t_i will hold a $NULL$ pointer.
2. For each cell in $x_j \in X$, we will hold a back-pointer to the appropriate cell t_i such that t_i is the description of click x_j.

Now U will verify the proof with the following procedure:

1. Repeat $O(1/\epsilon)$ times:
   (a) pick $j \in [k]$ at random
   (b) $l \leftarrow X[j]$
   (c) if $T(l) = x_j$ continue, otherwise return $FAIL$
2. Return $PASS$

Behavior: If there are any duplicates in X, then T won't "know" on which x_i to point. Thus, we can easily see that if there are $\geq k\epsilon/2$ duplicates in X, then in each round we fail with probability $\geq \epsilon/2$, so in $O(1/\epsilon)$ rounds, we will catch a false proof with constant probability.

2.2 Bin-Packing problem

Input:

1. A positive integer B
2. A set of n positive elements $x_1, x_2, ..., x_n$ where each $x_i \in [B]$
3. k bins of size B

Prover: We want to know whether we can fit all these elements into the k bins, i.e, each element is insetrted into some bin and the total size of the elements in each bin is less than B.

* This is also a well-known NP-complete problem.

Goal: If all the elements fit we want that the algorithm will returns "$PASS$" with probability 1, and if at most $(1-\epsilon)n$ of the elements fit, returns "$FAIL$" with high probability.

Consider the following proof: We will have k arrays (one for each bin) $A_1, ..., A_k$ and each array will be of size B. If x_i is of weight w and appears in bin A_j , then A_j will contain w consecutive instances of x_i. In addition, we will have an extra array X of size n such that $X[j]$ indicates the number of bins i in which x_j is packed and the offset m in A_i at which x_j starts to appear w consecutive times.

To verify this proof, the verifier will use the following procedure:

1. Repeat $O(1/\epsilon)$ times:
   (a) pick an element x_i of size w , $i \in n$ at random.
   (b) query $X[i]$ to get the number of bin j, and the offset m, for x_i.
   (c) verify that x_i appears w consecutive times in A_j starting at index m, if it is not - return $FAIL$, else-continue.
2. Return $PASS$