Randomized Algorithms 2013A
Lecture 3 – Proof of Hoeffding’s bound and sketching algorithms*

Robert Krauthgamer

We first finish something from last class on streaming algorithms, showing a key application of point queries.

1 Heavy hitters via point queries

**Heavy hitters set:** \( HH_\phi^p(x) = \{i : |x_i| \geq \phi \|x\|_p \} \).

Observe that the number of \( HH_\phi^1 \) is bounded by \( 1/\phi \).

Hence, we may hope to compute it using small space. However, we cannot expect to solve it exactly, since this set is very sensitive to small changes in \( x_i \) and we cannot “remember” the exact value of each \( x_i \).

**Approximate HH problem:**

Parameters: \( \phi > \varepsilon > 0 \).

Goal: return a set \( S \subseteq [n] \) such that

\[
HH_\phi^p \subseteq S \subseteq HH_{\phi-\varepsilon}^p.
\]

**Reduction of HH to point query:**

Assume we have an algorithm for \( \ell_p \) point queries with parameter \( \alpha = \varepsilon/2 \) and error probability \( 1/3n \).

Execute this algorithm to compute for every \( i \in [n] \) an estimate \( \tilde{x}_i \) (this step takes time \( O(n \log n) \) or even more) and report the set \( S = \{i \in [n] : |\tilde{x}_i| \geq (\phi - \varepsilon/2)\|x\|_p \} \).

Remark: This assumes we know \( \|x\|_p \) exactly. We saw in previous class how to approximate \( \|x\|_2 \).

Storage: For \( p = 1 \), we saw in previous class how to answer such point queries via a Count-Min sketch using \( O(\varepsilon^{-2} \log n) \) machine words.

*These notes summarize the material covered in class, usually skipping proofs, details, examples and so forth, and possibly adding some remarks, or pointers. The exercises are for self-practice and need not be handed in. In the interest of brevity, most references and credits were omitted.
Analysis: With probability $\geq \frac{2}{3}$, all the $n$ estimates are correct within additive $\varepsilon/2$. In this case, $S$ contains all the $\phi$-HH, and is contained in the $(\phi - \varepsilon)$-HH.

## 2 Proof of Hoeffding’s bound

We will prove one variant of the deviations bounds stated in the first class. After proving this theorem, we will see a version of it that resolves the concern raised in the analysis of quicksort that the indicators are really independent.

**Theorem 1:** Let $X_1, \ldots, X_n \in [0, 1]$ be independent random variables, and let $\mu_1, \ldots, \mu_n$ be such that for all $i \in [n]$, $\mathbb{E}X_i \leq \mu_i$. Then

$$\forall t > 0, \quad \Pr[\sum_i X_i \geq \sum_i \mu_i + t] \leq e^{-t^2/2n}.$$  

Proof: The main idea called Chernoff’s method is to use Markov’s inequality on the moment generating function $e^{\lambda X}$, which requires to analyze, $\lambda \mapsto \mathbb{E}[e^{\lambda X}]$, for an “optimized” choice of $\lambda > 0$.

The proof seen in class requires the following lemma, whose proof uses basic calculus.

**Lemma 2:** Let $Y \in [a, b]$ be a random variable with $\mathbb{E}Y = 0$. Then

$$\forall \lambda > 0, \quad \mathbb{E}[e^{\lambda Y}] \leq e^{\lambda^2(b-a)^2/8}.$$  

We saw in class a somewhat simpler proof for the case $[a, b] = [-1, 1]$, which is the case we actually used for Hoeffding.

Exer: Use/adapt the proof to bound deviation to the other direction. (Hint: Looks at $1 - X_i$, which is equivalent to looking at $-Y$.)

**Theorem 3:** Let $X_1, \ldots, X_n \in [0, 1]$ be random variables such that for all $i$ and $X_1, \ldots, X_{i-1}$ we have $\mathbb{E}[X_i \mid X_1, \ldots, X_{i-1}] \leq \mu_i$. Then

$$\Pr[\sum_i X_i \geq \sum_i \mu_i + t] \leq e^{-t^2/2n}.$$  

Exer: Prove this theorem by adapt the previous proof.

Hint: The key step where we used independence is changed along the following lines:

$$\mathbb{E}[e^{\sum_i X_i}] = \mathbb{E}_{X_1, \ldots, X_{n-1}}[\mathbb{E}_{X_n}[e^{\sum_i X_i} \mid X_1, \ldots, X_{n-1}]]$$  

\[= \mathbb{E}_{X_1, \ldots, X_{n-1}}[e^{\sum_{i \leq n-1} X_i} \cdot \mathbb{E}_{X_n}[e^{X_n} \mid X_1, \ldots, X_{n-1}]]\]  

and now apply the lemma where $Y$ is the conditioned $X_n - \mathbb{E}[X_n \mid X_1, \ldots, X_{n-1}]$. 

2
3 Sketching Algorithms

What is Sketching: We have some input \( x \), which we want to “compress” into a sketch \( s(x) \) (much smaller), but want to be able to later compute some \( f(x) \) only from the sketch. Often, randomization helps.

Applications: Many in the context of massive data sets (internet, query logs).

Example we already saw: Sketching \( x \in \mathbb{R}^n \) so that later we could estimate any \( x_i \) (point queries).

We consider today the problem of estimating the \( l_p \) distance between two vectors \( x, y \) within factor \( 1 + \varepsilon \).

Problem definition: Estimating \( \ell_p \) distance:

Parameters: approximation \( \varepsilon > 0 \) and integer \( n \).

Algorithms: a randomized sketching function \( s = s_r \) (here \( r \) is the random coins) and an answer function \( a \), such that for all \( x, y \in [n]^n \),

\[
\Pr_r[a(s_r(x), s_r(y)) = (1 \pm \varepsilon)\|x - y\|_p] \geq 2/3.
\]

Note: \( a \) operates on the sketches; might use the randomness (\( a = a_r \)). We care mostly about the sketch size \( |s(x)| \), usually measured in bits. We care “less” about computation time.

Example: \( \ell_2 \) distance between two vectors:

Let \( s \) be the linear sketch \( L : [n]^n \rightarrow \mathbb{Z}^k \) for \( k = O(1/\varepsilon^2) \) that we saw in the previous class for estimating the \( \ell_2 \) norm. We want function \( a \) to apply algorithm \( B \) (from previous class) to \( x - y \). Is it possible?

Recall algorithm \( B \) basically computes the linear sketch \( L(x - y) \), and outputs the average squared-coordinate \( \frac{1}{k}\|L(x - y)\|_2^2 \). This is just \( \frac{1}{k}\|Lx - Ly\|_2^2 \) (since \( L \) is linear), hence function \( a \) can compute this estimate from its inputs \( Lx \) and \( Ly \).

The above achieves \( (1 \pm \varepsilon)\)-approximation for the \( \ell_2 \)-squared distance, and thus also \( (1 \pm \varepsilon) \)-approximation for \( \ell_2 \) distance.

Sketch size: \( |s(x)| \leq O(\varepsilon^{-2} \log n) \) bits.

Exer: Use the above to derive a solution for \( \ell_1 \) distance. (Hint: Convert to unary.)

Example application: closest/furthest pair:

Input: \( n \) vectors \( x^1, \ldots, x^n \in [n]^n \).

Goal: Find \( i \neq j \) that minimizes/maximizes \( \|x^i - x^j\|_2 \).

Exer: Show that an approximate solution within \( 1 \pm \varepsilon \) factor can be computed in time \( O(n^2 \varepsilon^{-2} \log n) \).

Theorem [Equality testing]: For every \( n \) and \( t \) there is a randomized sketching algorithm, meaning \( s(\cdot) \) and \( a(\cdot, \cdot) \), that uses \( t \) bits and such that for all \( x, y \in \{0, 1\}^n \) can determine whether \( x = y \) with probability \( 1 - 2^{-t} \).
Proof: Let \( h : \{0,1\}^n \rightarrow \{0,1\}^t \) be a random (hash) function determined by the common randomness. Let \( s(x) = h(x) \) and let \( a(s_1, s_2) \) be the indicator for \( s_1 = s_2 \). Clearly, if \( x = y \) then referee always outputs 1 (i.e., YES). If \( x \neq y \), then referee outputs 0 (i.e., NO) with probability \( 1 - 2^{-t} \).

Algorithm with fewer random bits (same sketch size): We start with the algorithm for \( t = 1 \). Choose a random \( r \in \{0,1\}^n \) using the common randomness. Define \( s(x) = \sum_{i=1}^{n} x_i r_i \pmod{2} \) which is the inner product \( \langle x, r \rangle \). For general \( t \), repeat the above \( t \) times (in parallel) and let \( s(x) \) be their concatenation. As before, \( a(s_1, s_2) \) be the indicator for \( s_1 = s_2 \).

The analysis was seen in class, using the principle of deferred decision.