Sampling Algorithms: Lower Bounds and Applications *

Ziv Bar-Yossef†
Computer Science Division
U.C. Berkeley
Berkeley, CA 94720
zivi@cs.berkeley.edu

Ravi Kumar
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
ravi@almaden.ibm.com

D. Sivakumar
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
siva@almaden.ibm.com

August 1, 2001

Abstract

We develop a framework to study probabilistic sampling algorithms that approximate general functions of the form $f : A^n \to B$, where A and B are arbitrary sets. Our goal is to obtain lower bounds on the query complexity of functions, namely the number of input variables x_i that any sampling algorithm needs to query to approximate $f(x_1, \ldots, x_n)$.

We define two quantitative properties of functions — the *block sensitivity* and the *minimum Hellinger distance* — that give us techniques to prove lower bounds on the query complexity. These techniques are quite general, easy to use, yet powerful enough to yield tight results. Our applications include the mean and higher statistical moments, the median and other selection functions, and the frequency moments, where we obtain lower bounds that are close to the corresponding upper bounds.

We also point out some connections between sampling and streaming algorithms and lossy compression schemes.

*An extended abstract of this paper appeared in the Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing (STOC), pages 266-275, 2001.

†Part of this work was done while the author was visiting IBM Almaden Research Center. Supported by NSF Grant CCR-9820897.

1 Introduction

The need for computing with massive data sets has sparked off much interest in novel computational paradigms. These include, but are not restricted to, algorithms that probe only small (random) portions of the data; algorithms that work by making a few passes over the data [HRR99]; algorithms that operate on a stream of data with limited space and stringent constraints on time per data item [HRR99, AMS99, FKSV99]. Algorithms of this nature are typically intended to compute sufficiently good approximate solutions to the problem at hand. Several algorithmic and complexity questions arise in the context of these computational paradigms.

In this paper, we focus on one such paradigm: algorithms that work by randomly sampling a few entries of a large input. We develop the first systematic complexity theory of probabilistic sampling algorithms that approximate general functions of the form $f : A^n \to B$, where A and B are arbitrary sets. A highlight of our work is the level of generality we are able to achieve toward our goal: we consider functions where the domain and range are not necessarily metric spaces, which makes the issue of "approximate" computation rather subtle; even when the domain and range have metric properties, there are many standard notions of approximation, e.g., relative vs. additive, that need to be encompassed. Also, we include algorithms that may perform non-oblivious (adaptive) sampling of input entries.

Main contributions.

From the standpoint of modeling, we present a single unified view of what it means to approximate a general function on a possibly non-metric domain and range. As our computational model, we show how the standard decision tree model may be suitably adapted. Our main results are two techniques for obtaining lower bounds on the query complexity for approximating broad classes of functions. A primary advantage of these techniques is that the lower bounds are stated in terms of certain quantitative properties of functions, which make them quite general and very easy to use. We also present (general and specific) upper bounds that establish the tightness of our lower bounds. Finally, we point out connections between sampling algorithms, streaming algorithms, and lossy data compression.

Lower bounds.

We present two techniques for obtaining lower bounds on the query complexity of randomized sampling algorithms that approximately compute functions. Our first lower bound technique is based on an adaptation of the notion of *block sensitivity* [Nis91]. The advantage of this technique is that it applies to *any* function, even though the lower bounds may be somewhat weak. Moreover, it applies to the *expected* query complexity, not just the worst case query complexity. Our second lower bound technique is based on the notion of *Hellinger distance* [LL90] between probability distributions, and yields much stronger bounds for symmetric functions. These results are described in Sections 3 and 4.

Using the method based on the Hellinger distance, we obtain (in many cases, optimal) lower bounds on the query complexity for several problems. These include approximate computations of the median, minimum, maximum, and other selection functions, the mean and higher statistical

moments, and frequency moments F_k for $k \neq 1$. For the case of the mean, our lower bound matches the lower bound of Canetti, Even, and Goldreich [CEG95], and (modulo the machinery) is substantially simpler. This lower bound has another powerful consequence: it implies the main technical result of Radhakrishnan and Ta-Shma [RTS00], which they use to obtain lower bounds on extractor, disperser, and superconcentrator parameters.

Our lower and upper bounds for the frequency moments have some interesting implications. (Recall that given $X \in [m]^n$, for $k \geq 0$, the k-th frequency moment $F_k(X)$ is defined as $\sum_{i=1}^{m}(f_i(X))^k$, where $f_i(X)$ is the number of times i appears in the sequence X.) While any oblivious sampling algorithm that uses s samples can be simulated by a streaming algorithm that uses space (roughly) s, we show that the converse is not true for F_2: the streaming algorithm of Alon, Matias, and Szegedy [AMS99] uses $O(\log m)$ space for approximating F_2, while we prove an $\Omega(\sqrt{m})$ lower bound for sampling algorithms. (This style of "separation" of the two models is also demonstrated in [FKSV00].) We then show that for $k \geq 2$, the k-th frequency moment can be approximated using $O(m^{1-\frac{1}{k}})$ samples, which immediately implies the space upper bound of [AMS99] for all $k > 2$. Finally, we provide a simple proof for the sampling lower bound of Charikar et al. [CCMN00] for F_0. These results are described in Section 5.

We also investigate the question of how tight our lower bound methodologies are. We obtain a general theorem, which shows that at least in some special cases (symmetric functions on bounded domains), the lower and upper bounds are polynomially related. In addition, we point out some limitations of our techniques. These results are described in Section 6.

Connection to Lossy compression.

The question of how to compress data so that certain interesting functions of the data may be computed directly from the compressed data is of fundamental importance, both in theory and practice. The question becomes much more interesting (and harder) if we allow the compression to be *lossy*, that is, if we trade some qualitative degradation of the data for large compression factors. We present a formal model to study this question, and show that if a function has an efficient sampling algorithm or a streaming algorithm [HRR99, AMS99, FKSV99], then it admits lossy compression (in a technically precise sense). These results are described in Section 7.

In Section 8 we discuss related prior work and in particular the relationship of our model to the areas of Boolean decision tree complexity [Bd99], PAC and statistical learning Theory [Val84, KV94, Vap98], statistical decision theory [Ber85], statistical estimation theory [Van68], and statistical sequential analysis [Sie85]. Section 9 concludes with some open problems.

2 Preliminaries

In this section we introduce a notion of approximation for functions $f : A^n \to B$, where A and B are arbitrary sets. We then generalize Boolean decision trees to decision trees that approximately compute such functions. These decision trees will be our model for sampling algorithms.

2.1 Approximation Notions

An approximation for a function $f : A^n \to B$ is a function that maps inputs of f to subsets of its range B. This function assigns to every input x a set of values in B that are considered a good approximation for $f(x)$. It has two natural requirements, as specified in the following definition:

Definition 2.1 (Approximation) *An* approximation *for a function $f : A^n \to B$ is a family of functions $\{C_{f,\epsilon} : A^n \to 2^B\}_{\epsilon \geq 0}$, parameterized by an error parameter ϵ, which satisfies the following two conditions:*

(1) $\forall x \in A^n,\ C_{f,0}(x) = \{f(x)\}$

(2) $\forall x \in A^n\ \forall \epsilon > \epsilon' \geq 0,\ C_{f,\epsilon}(x) \supseteq C_{f,\epsilon'}(x)$

One might wonder why we define $C_{f,\epsilon}$ as a function of the inputs rather than as a function of the output values, i.e., why wouldn't $C_{f,\epsilon}$ determine for each value in the range what other values in the range are considered a good approximation for it? The rank and property testing approximations described below show that in some cases the approximation is indeed a function of the inputs: there can be two different inputs with the same output value but with different approximation sets.

Throughout this paper, for simplicity of notation, we implicitly assume that every function f is associated with a single approximation $C_{f,\epsilon}$ (the "natural" approximation for f). A few examples of popular approximation notions interpreted in light of the above definition:

(1) *Additive approximation:* (B, d_B) is a metric space.

$C_{f,\epsilon}(x) = \{y \in B \mid d_B(y, f(x)) \leq \epsilon\}$.

(2) *Relative approximation:* $B = \mathbf{R}$.

$C_{f,\epsilon}(x) = \{y \in B \mid (1-\epsilon)f(x) \leq y \leq (1+\epsilon)f(x)\}$.

(3) *Ratio approximation:* $B = \mathbf{R}$.

$C_{f,\epsilon}(x) = \{y \in B \mid \epsilon f(x) \leq y \leq (1/\epsilon)f(x)\}$.

(4) *Rank approximation:* An approximation notion for selection functions, like the median. The input for Select_q, the selection function of order q ($0 \leq q \leq 1$), is a set of n real numbers $a_1, \ldots, a_n$. If we order these numbers from smallest to largest: $a_{i_0} \leq \cdots \leq a_{i_{n-1}}$, then $\text{Select}_q(a_1, \ldots, a_n) = a_{i_{\lfloor q(n-1) \rfloor}}$. For example, the median is $\text{Select}_{\frac{1}{2}}$, minimum is Select_0, and maximum is Select_1. The approximation is $C_{\text{Select}_q,\epsilon}(a_1, \ldots, a_n) = \{a_{i_j} \mid j \in [\lfloor q'(n-1) \rfloor, \lfloor q''(n-1) \rfloor]], q' = \max(q-\epsilon, 0), q'' = \min(q+\epsilon, 1)\}$.

(5) *Property testing approximation:* An approximation notion for functions with a Boolean range. We assume some distance measure D on the input domain. The approximation is: $C_{f,\epsilon}(x) = \{1\}$, if $D(x,y) \leq \epsilon/2$ for some input y with $f(y) = 1$; $C_{f,\epsilon}(x) = \{0\}$, if $D(x,y) \geq \epsilon$ for all inputs y with $f(y) = 1$; and $C_{f,\epsilon}(x) = \{0,1\}$ otherwise.

A special kind of approximation is one that satisfies the "sunflower property," defined below. It is easy to check that approximations (2)–(5) above satisfy the sunflower property. Approximation

(1) satisfies it, if $B = \mathbf{R}$.

Definition 2.2 (Sunflower property) *An approximation $C_{f,\epsilon}$ is said to satisfy the* sunflower property*, if for every $D \subseteq A^n$, the following condition holds:*

$$(\forall x, x' \in D)[C_{f,\epsilon}(x) \cap C_{f,\epsilon}(x') \neq \emptyset] \Rightarrow \bigcap_{x \in D} C_{f,\epsilon}(x) \neq \emptyset.$$

A notion that will play an important role in our discussion is the following:

Definition 2.3 (Disjoint inputs) *Two inputs $x, y \in A^n$ are said to be ϵ-disjoint with respect to approximation $C_{f,\epsilon}$, if $C_{f,\epsilon}(x) \cap C_{f,\epsilon}(y) = \emptyset$.*

2.2 Decision Trees for General Functions

A *randomized decision tree* for a function $f : A^n \to B$ is a rooted labeled tree, not necessarily binary. Similar to Boolean decision trees, internal nodes denote either queries of input locations or random coin tosses, and leaves denote outputs. Given an input x, we use the queries and the random coin tosses to determine a path from the root to one of the leaves. This leaf specifies the output on x.

Formally, for every node v we denote by $\deg(v)$ the number of children it has and by $S(v)$ the number of query nodes on the path from the root to v. If v is a query node, it is labeled by an input variable $i \in [n]$ and a function $N_v : A^{S(v)} \to [\deg(v)]$. If the path reaches v, we apply N_v on the values queried so far to determine which child of v should be the next node on the path. If v is a random coin node, it is labeled by the special character \$, instructing the path to pick one of its children uniformly at random. If v is a leaf, it is labeled by a function $O_v : A^{S(v)} \to B$, which specifies the output of the tree, based on the values queried along the path. Note that any input x may be associated with several possible paths leading from the root to a leaf, depending on the random choices made in the random coin nodes. These random choices induce a distribution over the paths corresponding to x.

We discuss two notions of complexity for decision trees: the *expected query complexity* of a tree T on input x, denoted $S^e(T, x)$, is the expected number of query nodes on paths corresponding to x. The *worst-case query complexity* of T on x, denoted $S^w(T, x)$, is the maximum number of query nodes on paths corresponding to x. Here, the expectation and the maximum are over the distribution of paths. The expected and worst-case query complexity of T, $S^e(T)$ and $S^w(T)$, are the maximum of $S^e(T, x)$ and $S^w(T, x)$, respectively, over all inputs $x \in A^n$.

Notice that this model can simulate, with the same efficiency, various models of decision trees, including Boolean, comparison, and algebraic decision trees.

Yao's Theorem [Yao77] gives an equivalent characterization of a randomized decision tree as a distribution μ over deterministic decision trees. The expected query complexity of the tree on input x is the expected length (over μ) of the paths corresponding to x in these trees (and similarly for the worst-case query complexity).

Let $\epsilon \geq 0$ be an error parameter, $0 \leq \delta \leq 1$ a confidence parameter, and $f : A^n \to B$ a function with approximation $C_{f,\epsilon}$. A decision tree is said to (ϵ, δ)-*approximate* f, if for every input $x \in A^n$ the

probability of paths corresponding to x that output a value $y \in C_{f,\epsilon}(x)$ is at least $1-\delta$. The (ϵ,δ) *expected query complexity* of f is: $S^e_{\epsilon,\delta}(f) \stackrel{\text{def}}{=} \min\{S^e(T) \mid T \ (\epsilon,\delta)\text{-approximates } f\}$. We similarly define the worst-case query complexity of functions.

3 A General Lower Bound via Block Sensitivity

In this section we generalize the notion of block sensitivity (defined by Nisan [Nis91] for Boolean functions) and obtain a lower bound on the expected query complexity in terms of block sensitivity.

A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is said to be sensitive to a subset (a "block") $I \subseteq [n]$ on input x, if f flips its value when flipping all the bits in I. The block sensitivity of f on x, $bs(f,x)$, is the maximum number t of pairwise disjoint subsets $I_1,\ldots,I_t \subseteq [n]$, to which f is sensitive on x. The block sensitivity of f, $bs(f)$, is the maximum, over all inputs x, of $bs(f,x)$.

For general domains and ranges, the basic intuition we carry over from the Boolean case is that a function is sensitive to a block of variables if a change to the corresponding input elements results in a significant change to the function value. In the following definition we denote by $x^{(I \leftarrow Q)}$ the input obtained from x by changing the elements in $I \subseteq [n]$ to the values in $Q \in A^{|I|}$:

Definition 3.1 (Block sensitivity) *f is* ϵ-sensitive to a subset of variables $I \subseteq [n]$ on input x, *if there exists $Q \in A^{|I|}$ such that x and $x^{(I \leftarrow Q)}$ are ϵ-disjoint. $bs_\epsilon(f,x)$, the* ϵ-block sensitivity of f on x, *is the maximum number t of pairwise disjoint subsets $I_1,\ldots,I_t \subseteq [n]$, such that f is ϵ-sensitive to each of them on x. $bs_\epsilon(f)$, the* ϵ-block sensitivity of f, *is* $\max_{x \in A^n} bs_\epsilon(f,x)$.

Nisan [Nis91] proved that for Boolean functions f, $S^e_{0,\delta}(f) \geq (1-2\delta)bs(f)$. We generalize the proof of this theorem to our case.

Theorem 3.2 *For every $\epsilon \geq 0$, $0 \leq \delta \leq 1/2$, and $f : A^n \to B$, $S^e_{\epsilon,\delta}(f) \geq (1-2\delta)bs_\epsilon(f)$.*

Proof. Consider any decision tree T that (ϵ,δ)-approximates f. We will use the view of T as a distribution μ over deterministic decision trees $T_1,\ldots,T_m$. Note that deterministic decision trees have exactly one path from the root to a leaf corresponding to any input x. Thus a deterministic decision tree is either always "right" on x, i.e., outputs a value in $C_{f,\epsilon}(x)$, or always "wrong". Let us call a decision tree T_j "bad" for x, if it is wrong on x. Note that for every input x, $\Pr_{T_j \in \mu}(T_j \text{ is bad for } x) \leq \delta$.

Let x be the input that achieves the block sensitivity of f, and let $I_1,\ldots,I_t \subseteq [n]$ be the disjoint variable blocks to which f is ϵ-sensitive on x (note that $t = bs_\epsilon(f)$). Finally, let $Q_1,\ldots,Q_t$ be the assignments to these blocks, for which x and $x^{(I_k \leftarrow Q_k)}$ are ϵ-disjoint.

Consider a tree T_j in which the path corresponding to x does not query any of the variables in I_k for some $k \in [t]$. This means that the same path will correspond to $x^{(I_k \leftarrow Q_k)}$. Moreover, since all the values queried in this path are the same at x and at $x^{(I_k \leftarrow Q_k)}$, the leaf will output the same value y for both. However, since $C_{f,\epsilon}(x) \cap C_{f,\epsilon}(x^{(I_k \leftarrow Q_k)}) = \emptyset$ either $y \notin C_{f,\epsilon}(x)$ or $y \notin C_{f,\epsilon}(x^{(I_k \leftarrow Q_k)})$.

So T_j is bad either for x or for $x^{(I_k \leftarrow Q_k)}$. Therefore,

$$\Pr_{T_j \in \mu}(T_j \text{ does not query } I_k \text{ on } x) \leq \Pr_{\mu}(T_j \text{ is bad for } x) + \Pr_{\mu}(T_j \text{ is bad for } x^{(I_k \leftarrow Q_k)}) \leq 2\delta$$

Let $S(T,x)$ be the random variable counting the number of queries T performs on x. Define for every $k \in [t]$, a random variable X_k which counts the number of these queries that belong to I_k. Since $I_1, \ldots, I_t$ are pairwise disjoint, $S(T,x) \geq \sum_{k=1}^{t} X_k$. What we proved above implies that $\Pr(X_k \geq 1) \geq 1 - 2\delta$, implying that also $E(X_k) \geq 1 - 2\delta$. Therefore, by linearity of expectation,

$$S^e_{\epsilon,\delta}(f) \geq E(S(T,x)) \geq E(\sum_{k=1}^{t} X_k) = \sum_{k=1}^{t} E(X_k) \geq t(1-2\delta) = (1-2\delta) bs_\epsilon(f)$$

□

Let us consider an application of this lower bound for additive approximation of the mean: $\mu_1 : [0,1]^n \to [0,1]$. The best upper bound is $S^w_{\epsilon,\delta}(\mu_1) = O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$. Using Theorem 3.2 we prove the following:

Proposition 3.3 $S^e_{\epsilon,\delta}(\mu_1) \geq \Omega(\frac{1}{\epsilon}(1-2\delta))$

Proof. Let $x = 1^n$, and consider any subset $I \subseteq [n]$ of size $2\epsilon n$. Define an assignment $Q = 0^{|I|}$. Since $\mu_1(x) = 1$ and $\mu_1(x^{(I \leftarrow Q)}) = 1 - 2\epsilon$, x and $x^{(I \leftarrow Q)}$ are ϵ-disjoint, implying μ_1 is ϵ-sensitive to I on x. This argument works for any subset I of size $2\epsilon n$; thus, the maximum number of pairwise disjoint subsets to which μ_1 is ϵ-sensitive on x is $n/|I| = 1/2\epsilon$. Applying Theorem 3.2 completes the proof. □

Proposition 3.3 shows that the block sensitivity lower bound does not yield tight bounds in general. In the case of the mean, the bound produced is not tight not in terms of the error ϵ, and not in terms of the confidence δ. In Section 5 we show how our second lower bound technique gives a tight lower bound for the mean.

Nisan [Nis91] proved that for Boolean functions, block sensitivity and (deterministic/randomized) decision tree complexity are polynomially related. A particular corollary of this is that for Boolean functions, deterministic and randomized decision tree complexities are polynomially related. It is natural to ask if block sensitivity also gives a (polynomial) characterization of randomized decision tree complexity for approximating general functions, but it turns out not to be the case. While the ϵ-approximate median problem has a sampling algorithm that makes $O(\epsilon^{-2})$ probes (independent of n), it is easy to show that any deterministic algorithm needs at least $\Omega(n(1-\epsilon))$ probes.

4 A Lower Bound for Symmetric Functions

The query complexity lower bound via block sensitivity is very general but is also weak. Its dependence on both the error ϵ and the confidence δ frequently turn out to be sub-optimal. In this section we obtain stronger lower bounds that work only for *symmetric functions* — functions that are invariant under permutations of the input elements:

Definition 4.1 (Symmetric functions) *A function $f : A^n \to B$ is* ϵ-symmetric *if for every input $x \in A^n$ and every permutation $\pi \in S_n$, $C_{f,\epsilon}(x) = C_{f,\epsilon}(\pi(x))$ (where $\pi(x)_i = x_{\pi^{-1}(i)}$).*

The basic idea for the lower bounds is to show that for symmetric functions we can reduce the computer science sampling setting, in which different queries are not necessarily independent and do not have the same query distribution, to the statistics setting, in which all queries are independent and identically distributed ("the i.i.d. case"). We show that with almost no loss of efficiency, any general sampling algorithm for a symmetric function can be simulated by one that makes queries that are independent and uniformly distributed.

We then observe that a sampling algorithm of query complexity k that approximates $f : A^n \to B$ and makes only independent uniform queries can be used to form a *statistical test* that uses k samples to distinguish between the distributions U_x and U_y for any pair of ϵ-disjoint inputs x and y. Here, U_x and U_y are the distributions on A obtained by picking $i \in [n]$ uniformly at random and returning x_i and y_i respectively. Clearly, the closer U_x and U_y are, the harder it is to distinguish between them. Thus, by giving a lower bound on the number of samples required to distinguish any two such input distributions, we obtain a lower bound on the query complexity of the function.

We state the lower bounds in terms of new properties of functions. These properties use the Hellinger distance and the variation distance between distributions:

Definition 4.2 (Distance measures) *Let P and Q be two distributions on the same probability space Ω. The* variation distance *d and the* Hellinger distance *h between P and Q are defined as follows:*

$$d(P,Q) \stackrel{\text{def}}{=} \tfrac{1}{2}\textstyle\sum_{\omega\in\Omega} |P(\omega) - Q(\omega)| = \max_{D\subseteq\Omega} |P(D) - Q(D)|$$

$$h(P,Q) \stackrel{\text{def}}{=} (1 - \textstyle\sum_{\omega\in\Omega} \sqrt{P(\omega)Q(\omega)})^{\frac{1}{2}} = (\tfrac{1}{2}\textstyle\sum_{\omega\in\Omega}(\sqrt{P(\omega)} - \sqrt{Q(\omega)})^2)^{\frac{1}{2}}$$

We use these distance measures to define the following properties:

Definition 4.3 (Minimum Hellinger/Variation distance) *For a function $f : A^n \to B$, the* ϵ-minimum Hellinger distance of f *is $h_\epsilon(f) \stackrel{\text{def}}{=} \min\{h(U_x, U_y) \mid x, y \in A^n$ are ϵ-disjoint$\}$ and the* ϵ-minimum variation distance of f *is $d_\epsilon(f) \stackrel{\text{def}}{=} \min\{d(U_x, U_y) \mid x, y \in A^n$ are ϵ-disjoint$\}$.*

Our main theorem gives two lower bounds on the worst-case query complexity of symmetric function in terms of their minimum Hellinger and variation distances:

Theorem 4.4 (Main theorem) *For every $\epsilon \geq 0$ and for every ϵ-symmetric function $f : A^n \to B$, (1) if $\frac{1}{2n} < \delta < \frac{1}{2}$, $S^w_{\epsilon,\delta}(f) \leq \sqrt{2\delta n}$, and $h_\epsilon(f) \leq \frac{1}{2}$, then*

$$S^w_{\epsilon,\delta}(f) \geq \frac{1}{4h^2_\epsilon(f)} \ln \frac{1}{8\delta}$$

(2) if $\frac{2}{n} < \delta < \frac{1}{2}$, $S^w_{\epsilon,\delta}(f) \leq n/2 - \sqrt{n/2\delta}$, *and* $d_\epsilon(f) \leq \frac{1}{2}$, *then*

$$S^w_{\epsilon,\delta}(f) \geq \frac{1}{8d_\epsilon(f)} \ln \frac{1}{8\delta}$$

Our strategy for the proof of Theorem 4.4 is the following: We first consider (Section 4.1) a variant of the decision tree model, in which all queries are made according to a fixed query distribution. We then prove (Theorem 4.8) that, almost without any loss in efficiency, any symmetric function can be computed by such decision trees that probe the input at uniformly chosen positions. The consequence of this theorem is that if a decision tree makes k queries to an input x, the distribution it "sees" is U_x^k, the k-fold product of U_x.

Lemma 4.9 shows that a decision tree that (ϵ, δ)-approximates f using k (worst-case) queries can be used to distinguish between the distributions U_x^k and U_y^k for any ϵ-disjoint x, y, with advantage 2δ. This immediately implies that $d(U_x^k, U_y^k) \geq 1 - 2\delta$. Lemma 4.9 essentially reduces the problem of statistical hypothesis testing to the problem of (ϵ, δ)-approximating $f(x)$. In hypothesis testing, one is given samples from a distribution D which is known to be either P or Q, and is required to decide whether $D = P$ or $D = Q$. In our case the two distributions are U_x and U_y, and the hypothesis tester is the decision tree: if the tree outputs a value in $C_{f,\epsilon}(x)$, we decide "U_x", and otherwise we decide "U_y". Since x and y are ϵ-disjoint, and since the tree outputs an ϵ-approximation with probability at least $1 - \delta$, this gives a statistical test that succeeds with probability at least $1 - 2\delta$.

We then derive a lower bound (Lemma 4.11) on the number of samples needed to distinguish two distributions (with a certain advantage) in terms of their Hellinger distance. The tightness of this bound follows from a useful multiplicativity property of the Hellinger distance.

Some technical complications make the above argument work only for functions with query complexity which is at most $O(\sqrt{n})$. For functions with larger query complexity we can prove only a weaker lower bound in terms of the minimum variation distance.

In Section 4.6 we show a general reduction from worst-case query complexity to expected query complexity, which implies that for any function f, $S^w_{\epsilon,\frac{\delta}{3}}(f) \leq S^e_{\epsilon,\delta}(f) \cdot O(\log \frac{1}{\delta})$. This reduction implies that Theorem 4.4 holds also for the expected query complexity except for the dependence on the confidence δ:

Theorem 4.5 *For every* $\epsilon \geq 0$ *and for every* ϵ*-symmetric function* $f : A^n \to B$,

(1) if $\frac{1}{2n} < \delta < \frac{1}{2}$, $S^w_{\epsilon,\delta}(f) \leq \sqrt{2\delta n}$, *and* $h_\epsilon(f) \leq \frac{1}{2}$, *then*

$$S^e_{\epsilon,\frac{\delta}{3}}(f) \geq \frac{1}{8h^2_\epsilon(f)}$$

(2) if $\frac{2}{n} < \delta < \frac{1}{2}$, $S^w_{\epsilon,\delta}(f) \leq n/2 - \sqrt{n/2\delta}$, *and* $d_\epsilon(f) \leq \frac{1}{2}$, *then*

$$S^e_{\epsilon,\frac{\delta}{3}}(f) \geq \frac{1}{16d_\epsilon(f)}$$

For simplicity of notation, we will assume throughout this section that the domain A is finite. Everything we prove in the sequel can be generalized to arbitrary domains by replacing summation by integration.

4.1 Decision Trees with Fixed Query Distribution

We next consider a variation to the decision tree model presented in Section 2.2. Instead of assigning to each query node v a single query index $i_v \in [n]$, we assign it a *query strategy* $Q_v : ([n] \times A)^{S(v)-1} \to \mathcal{D}$, where $\mathcal{D}$ is a finite set of distributions on $[n]$. When the path from the tree's root reaches v, we pick a *query distribution* $D \in \mathcal{D}$ by applying Q_v to the indices queried so far on the path and to the outcomes of these queries. We then choose $i \in [n]$ according to D, query x_i, and continue as before. The child selection function N_v and the output function O_v are modified to take as input not only the outcomes of the queries made along the path leading to v, but also the indices queried (i.e., $N_v : ([n] \times A)^{S(v)} \to [\deg(v)]$ and $O_v : ([n] \times A)^{S(v)} \to B$). This seemingly stronger model is in fact equivalent to the one of Section 2.2 as long as $\mathcal{D}$ consists only of distributions with rational probabilities, since one can simulate these query distributions by random coin nodes.

An *oblivious* decision tree is one in which: (1) all nodes v with $S(v) = j$ have the same query strategy Q_j, and (2) these query strategies depend only on the indices queried so far and not on the outcomes of these queries (that is, $Q_j : [n]^{j-1} \to \mathcal{D}$). Oblivious decision trees choose their query distributions independently of the input at hand and of the outcomes of the random coin tosses; note, however, that they are not totally non-adaptive: the *number* of queries performed may still depend on the given input or the random coins. One could simulate the computation of an oblivious decision tree by first picking the k indices $i_1, \ldots, i_k \in [n]$ to be queried (where k is the maximum number of queries T may perform), and then running the tree with these queries (i.e., the j-th query will be i_j).

An oblivious decision tree T thus has a *fixed query distribution*, which we denote by $D^k(T)$. $D^k(T)$ is a distribution on $[n]^k$ obtained by the following iterative process: choose $i_1 \in [n]$ according to $Q_1()$; given $i_1, \ldots, i_{j-1}$, choose $i_j \in [n]$ according to $Q_j(i_1, \ldots, i_{j-1})$; return $(i_1, \ldots, i_k)$. We call a decision tree with fixed query distribution D^k a *D^k-decision tree*. Two examples of fixed query distributions we will use are the following:

(1) U^k – *Uniform queries with replacement*: each index is picked independently and uniformly at random from $[n]$. In this case, $Q_j(i_1, \ldots, i_{j-1})$ is the uniform distribution on $[n]$ for all j and for all $i_1, \ldots, i_{j-1}$.

(2) W^k – *Uniform queries without replacement*: each index is picked uniformly at random from the indices not chosen so far. Here, $Q_j(i_1, \ldots, i_{j-1})$ is the uniform distribution on $[n] \setminus \{i_1, \ldots, i_{j-1}\}$.

The two-phase view of a D^k-decision tree computation on some input x (i.e., first picking the (at most) k indices $i_1, \ldots, i_k$ to be queried, and then running the tree with these queries) implies that during the second phase of its execution, the tree has sequential access to the list $((i_1, x_{i_1}), \ldots, (i_k, x_{i_k}))$. This list can be viewed as one sample from the *joint query-outcome distribution* $\tilde{D}^k_x$ – a distribution on $([n] \times A)^k$ defined by picking $(i_1, \ldots, i_k) \in [n]^k$ according to D, and returning $((i_1, x_{i_1}), \ldots, (i_k, x_{i_k}))$.

A special class of oblivious decision trees are ones that are *index-oblivious*; that is, the child selection function N_v and the output function O_v of all nodes v depend only on the outcomes of queries made along the path leading to v and not on the indices queried (i.e., $N_v : A^{S(v)} \to [\deg(v)]$ and $O_v : A^{S(v)} \to B$). For such trees we can view the list T is given sequential access to as a sample from the *outcome distribution* D_x^k – a distribution on A^k obtained by picking $(i_1, \ldots, i_k) \in [n]^k$ according to D, and returning $(x_{i_1}, \ldots, x_{i_k})$.

4.2 Uniform Decision Trees for Symmetric Functions

The aim of this section is to show (Theorem 4.8) that for symmetric functions the restricted class of U^k-decision tree (i.e., ones that query only uniformly at random with replacement) is almost as good as general decision trees. We initially prove (Lemma 4.6) that any decision tree for a symmetric function can be simulated by an index-oblivious W^k-decision tree with the same query complexity; we then present (Lemma 4.7) two simulations of index-oblivious W^k-decision trees by U^k-decision trees: an index-oblivious simulation that works only for $k \leq O(\sqrt{n})$, and an index-aware simulation that works for $k \leq n/2 - o(n)$.

The results in this section are stated in terms of the worst-case query complexity, but they also hold for the expected query complexity.

Lemma 4.6 *Let $f : A^n \to B$ be an ϵ-symmetric function with $S^w_{\epsilon,\delta}(f) = k$. Then, there exists an index-oblivious W^k-decision tree of worst-case query complexity k that (ϵ, δ)-approximates f.*

Proof. Let T be a decision tree of worst-case query complexity k that (ϵ, δ)-approximates f. We use the standard decision tree view of T, like in Section 2.2 (i.e., each query node is associated with a fixed query index, and not with a query distribution; the child selection functions and output functions depend only the outcomes of queries and not on the locations queried). Without loss of generality, we assume that T never queries the same index more than once. We will show a construction of an index-oblivious W^k-decision tree T' that simulates T without loss of efficiency.

For simplicity of exposition, we assume that T is k-regular; that is, it always performs exactly k queries. This assumption is acceptable when dealing with the worst-case query complexity. Our arguments, however, can be modified to hold also when T is not k-regular, which is needed when dealing with the expected query complexity.

Given an input $x \in A^n$, T' picks k indices $i_1, \ldots, i_k \in [n]$ uniformly at random without replacement, and runs T with $i_1, \ldots, i_k$ as its queries. That is, when reaching the j-th query node v of T, T' ignores the query index labelling v, queries i_j instead, and applies the child selection function N_v on $x_{i_1}, \ldots, x_{i_j}$.

Clearly, T' is a W^k-decision tree of worst-case query complexity k. T' is index-oblivious, because it uses the same child selection and output functions as T (which do not take the query indices as input parameters, since we use the the standard decision tree view of T). We are left to prove that T' indeed (ϵ, δ)-approximates f.

For the analysis, we use Yao's theorem to view T as a distribution μ over deterministic decision

trees $T_1, \ldots, T_m$. Note that the output of each such T_j is fully determined by the input x, and thus T_j can be viewed as a function $T_j : A^n \to B$, mapping an input x to an output in B.

Using the distributional view of T, we can view the simulation of T' as follows. We denote by T_j^I the decision tree obtained from T_j by replacing all its query indices to $I \stackrel{\text{def}}{=} (i_1, \ldots, i_k)$. That is, all the nodes of T_j^I at depth d have i_{d+1} as a query label. T' runs in three phases: first it picks a decision tree T_j according to μ; then, it picks k indices $I = (i_1, \ldots, i_k)$ uniformly at random without replacement; finally, it runs T_j^I on the input x. Note that T_j^I, like T_j, is a deterministic decision tree, and thus can be viewed as a function $T_j^I : A^n \to B$. Using this view, we can characterize the error probability of T' on x as:

$$\Pr_{T_j \in \mu, I \in W^k}(T_j^I(x) \notin C_{f,\epsilon}(x))$$

From now on we fix x as some input. The main point is that the execution of T_j^I on x is identical to the execution of T_j on some permutation $\pi(x)$. Formally, let $v_1, \ldots, v_k, u$ be the nodes along the path of T_j^I determined by x; $v_1, \ldots, v_k$ are the query nodes of this path and u is the output node. Let $j_1, \ldots, j_k$ be the query indices labelling $v_1, \ldots, v_k$ in the tree T_j. Let S_I be the set of permutations satisfying $\pi(j_1) = i_1, \ldots, \pi(j_k) = i_k$. It is easy to see that for any $\pi \in S_I$, T_j, when running on $\pi(x)$, produces exactly the same path $v_1, \ldots, v_k, u$ as T_j^I when running on x. Moreover, all the query outcomes are identical in both cases, implying that $T_j^I(x) = T_j(\pi(x))$. This yields the following characterization of the error probability of T':

$$\Pr_{T_j \in \mu, I \in W^k}(T_j^I(x) \notin C_{f,\epsilon}(x)) = \Pr_{T_j \in \mu, I \in W^k, \pi \in S_I}(T_j(\pi(x)) \notin C_{f,\epsilon}(x))$$

We use the above observation to characterize the execution of T_j^I on x with a random I as running T_j on a random permutation $\pi(x)$. We note that the collections of permutations $\{S_I\}_{I \in W^k}$ is a partitioning of the symmetric group S_n into disjoint subsets; that is, (1) for all $I \neq I'$, $S_I \cap S_{I'} = \emptyset$ and (2) $\bigcup_{I \in W^k} S_I = S_n$.

To prove (1), let $I \neq I'$, and let i_d, i'_d be the first two indices different in I and I'. Since I and I' agree in their first $d-1$ indices, then the first d nodes on the paths in T_j^I and $T_j^{I'}$ determined by x are identical. Let j_d be the index labelling the d-th node on these paths. By definition, for all $\pi \in S_I$, $\pi(j_d) = i_d$, and for all $\pi \in S_{I'}$, $\pi(j_d) = i'_d$. Therefore, there is no $\pi \in S_I \cap S_{I'}$.

To prove (2), fix some permutation $\pi \in S_n$. We define an index set I inductively as follows: let j_1 be the index labelling the root of T_j; set $i_1 = \pi(j_1)$. Assume that $i_1, \ldots, i_{d-1}$ were defined; note that for every two I, I' starting with $i_1, \ldots, i_{d-1}$, the first d nodes on the paths of T_j^I and $T_j^{I'}$ determined by x are the same. Thus, let j_d be the index labelling the d-th node on these paths; we set $i_d = \pi(j_d)$. Note that $i_1, \ldots, i_k$ are distinct, because $j_1, \ldots, j_k$ are distinct and π is a permutation. Therefore, $I \stackrel{\text{def}}{=} (i_1, \ldots, i_k)$ belongs to W^k and $\pi \in S_I$.

It follows from the above that choosing a random $I \in W^k$ and then a random $\pi \in S_I$ is equivalent to choosing a random $\pi \in S_n$. Therefore,

$$\Pr_{T_j \in \mu, I \in W^k, \pi \in S_I}(T_j(\pi(x)) \notin C_{f,\epsilon}(x)) = \Pr_{T_j \in \mu, \pi \in S_n}(T_j(\pi(x)) \notin C_{f,\epsilon}(x)) = E_{\pi \in S_n}(\Pr_{T_j \in \mu}(T_j(\pi(x)) \notin C_{f,\epsilon}(x)))$$

Since f is ϵ-symmetric, then for all $\pi \in S_n$, $C_{f,\epsilon}(x) = C_{f,\epsilon}(\pi(x))$. Therefore,

$$E_{\pi \in S_n}(\Pr_{T_j \in \mu}(T_j(\pi(x)) \notin C_{f,\epsilon}(x))) = E_{\pi \in S_n}(\Pr_{T_j \in \mu}(T_j(\pi(x)) \notin C_{f,\epsilon}(\pi(x)))) \leq \delta$$

where the last inequality follows from the fact the T is erroneous with probability at most δ for any given input. □

The following lemma presents two simulations of index-oblivious W^k-decision trees by U^k-decision trees with almost the same query complexity. The first one works only for small enough k's ($k \leq O(\sqrt{n})$), but can be carried out index-obliviously; the second one works for almost any k ($k \leq n/2 - o(n)$), but is index-aware.

Lemma 4.7 *Let T be an index-oblivious W^k-decision tree of worst-case query complexity k that (ϵ, δ)-approximates a function $f : A^n \to B$. Then,*

(1) if $\frac{1}{2n} < \delta < \frac{1}{2}$ and $k \leq \sqrt{2\delta n}$, there exists an index-oblivious U^k-decision tree T' of worst-case query complexity k that $(\epsilon, 2\delta)$-approximates f.

(2) if $\frac{2}{n} < \delta < \frac{1}{2}$ and $k \leq \frac{n}{2} - \sqrt{\frac{n}{2\delta}}$, there exists an index-aware U^{2k}-decision tree T'' of worst-case query complexity $2k$ that $(\epsilon, 2\delta)$-approximates f.

Proof. Without loss of generality, we assume T always performs exactly k queries. Since T is a W^k-decision tree, we can view its computation as first picking k indices $i_1, \ldots, i_k \in [n]$ uniformly at random without replacement, and then running the tree with $i_1, \ldots, i_k$ as the query indices. The randomness in T can be split into two: (1) the choice of the query indices and (2) the random coin tosses performed later. We denote the random variable that corresponds to the index choice by I, and the random variable that corresponds to the random coin tosses by R.

Proof of (1):

T' picks $i_1, \ldots, i_k$ uniformly at random *with* replacement, and then simulates the second phase of T with $i_1, \ldots, i_k$ as its queries. Clearly, T' is index-oblivious (because T is), and its query complexity is k. We will show that if $k \leq \sqrt{2\delta n}$, T' errs with probability at most 2δ on any input x.

Let E denote the event that $i_1, \ldots, i_k$ are distinct, and let $\overline{E}$ denote its complement. Note that conditioning on E, the distribution of $i_1, \ldots, i_k$ is uniform without replacement. Thus, for any input x,

$$\begin{aligned}
&\Pr_{I \in U^k, R}(T'(x) \notin C_{f,\epsilon}(x)) = \\
&= \Pr_{I \in U^k, R}(T'(x) \notin C_{f,\epsilon}(x) \mid E) \cdot \Pr_{I \in U^k}(E) + \Pr_{I \in U^k, R}(T'(x) \notin C_{f,\epsilon}(x) \mid \overline{E}) \cdot \Pr_{I \in U^k}(\overline{E}) \\
&\leq \Pr_{I \in W^k, R}(T(x) \notin C_{f,\epsilon}(x)) + \Pr_{I \in U^k}(\overline{E}) \\
&\leq \delta + \Pr_{I \in U^k}(\overline{E})
\end{aligned}$$

We next show that $\Pr(\overline{E}) \leq \delta$, completing the proof of part (1). Define $\binom{k}{2}$ indicator random variables X_{jl}, such that $X_{jl} = 1$ iff $i_j = i_l$. $i_1, \ldots, i_k$ are not all distinct if and only if $\sum_{j,l} X_{jl} \geq 1$. The expectation of $\sum_{j,l} X_{jl}$ is $\binom{k}{2}\frac{1}{n} \leq \frac{k^2}{2n}$. Therefore, using Markov's inequality, $\Pr(\overline{E}) \leq \frac{k^2}{2n} \leq \delta$.

Proof of (2):

T'' picks $2k$ indices $i_1, \ldots, i_{2k}$ uniformly at random with replacement, selects the first k that are all distinct, and runs the second phase of T with these k indices as its queries. If the $2k$ indices contain less than k distinct ones, T'' outputs an arbitrary answer and halts. Note that T'' is *not* index-oblivious, since after picking the indices to be queried, it chooses from them the distinct ones. We next show that if $k \leq n/2 - \sqrt{n/2\delta}$, then T'' $(\epsilon, 2\delta)$-approximates f.

Again, we define E to be the event that at least k of the $2k$ indices chosen are distinct and $\overline{E}$ to be its complement. Note that conditioning on E, the distribution of the first k distinct indices is uniform without replacement. Thus, an identical argument to the one shown in part (1) proves that for any input x,

$$\Pr_{I \in U^{2k}, R}(T''(x) \notin C_{f,\epsilon}(x)) \leq \delta + \Pr_{I \in U^{2k}}(\overline{E})$$

In order to prove that $\Pr_{I \in U^{2k}}(\overline{E}) \leq \delta$, we define, as before $\binom{2k}{2}$ indicator random variables X_{jl}, such that $X_{jl} = 1$ iff $i_j = i_l$. If the list $i_1, \ldots, i_{2k}$ contains at most k distinct values, then $\sum_{j,l} X_{jl} \geq k$ (because the way to create the least number of collisions is to have two occurrences for each of the k values). On the other hand, the expectation of $\sum_{j,l} X_{jl}$ is $\binom{2k}{2}\frac{1}{n} \leq \frac{2k^2}{n}$. Using the pairwise independence of X_{jl} and Chebyshev's inequality we obtain:

$$\begin{aligned}
\Pr(\overline{E}) &\leq \Pr(\sum_{j,l} X_{jl} \geq k) \leq \Pr(\left|\sum_{j,l} X_{jl} - E(\sum_{j,l} X_{jl})\right| \geq k - \frac{2k^2}{n}) \leq \\
&\leq \frac{\mathrm{Var}(\sum_{j,l} X_{jl})}{k^2(1-\frac{2k}{n})^2} = \frac{\binom{2k}{2}\frac{1}{n}\frac{n-1}{n}}{k^2(1-\frac{2k}{n})^2} \leq \frac{\frac{2k^2}{n}}{k^2(1-\frac{2k}{n})^2} = \frac{2}{n(1-\frac{2k}{n})^2}
\end{aligned}$$

The last expression is at most δ for $k \leq n/2 - \sqrt{n/2\delta}$. □

Combining Lemma 4.6 and Lemma 4.7, we obtain the main result of this section:

Theorem 4.8 *Let $f : A^n \to B$ be an ϵ-symmetric function with $S^w_{\epsilon,\delta}(f) = k$. Then:*

(1) if $\frac{1}{2n} < \delta < \frac{1}{2}$ and $k \leq \sqrt{2\delta n}$, there exists an index-oblivious U^k-decision tree of worst-case query complexity k that $(\epsilon, 2\delta)$-approximates f.

(2) if $\frac{2}{n} < \delta < \frac{1}{2}$ and $k \leq \frac{n}{2} - \sqrt{\frac{n}{2\delta}}$, there exists an index-aware U^{2k}-decision tree of worst-case query complexity $2k$ that $(\epsilon, 2\delta)$-approximates f.

4.3 Approximation Implies Distinguishability

The main lemma of this section proves that any decision tree of worst-case query complexity k with fixed query distribution D^k that approximates a function f can be used to distinguish, using k samples, the joint query-outcome distributions $\tilde{D}^k_x$ and $\tilde{D}^k_y$ of any two ϵ-disjoint inputs $x, y \in A^n$. Moreover, if the tree happens to be index-oblivious, it can be used also to distinguish between the outcome distributions D^k_x and D^k_y.

Lemma 4.9 *Let T be a D^k-decision tree of worst-case query complexity k that (ϵ, δ)-approximates a function $f : A^n \to B$. Then for any two ϵ-disjoint inputs $x, y \in A^n$, $d(\tilde{D}^k_x, \tilde{D}^k_y) \geq 1 - 2\delta$. Furthermore, if T is index-oblivious, also $d(D^k_x, D^k_y) \geq 1 - 2\delta$.*

Proof. As in [CEG95], the basic idea underlying the proof will be to show that the distribution of values read off the input by the random queries along the tree's paths is very different when x is the input from when y is the input.

Without loss of generality, we assume that T is *regular*, that is, it always makes exactly k queries. We use the two-phase view of T: first, it picks $I = (i_1, \ldots, i_k) \in [n]^k$ according to D^k and then it runs the computation with $i_1, \ldots, i_k$ as its query indices. Note that the output of T is fully determined by the query indices $i_1, \ldots, i_k$, by the outcomes of these queries $x_{i_1}, \ldots, x_{i_k}$, and by the outcomes of its random coin tosses, which we denote by R. Thus, if we denote by $J_x = ((i_1, x_{i_1}), \ldots, (i_k, x_{i_k}))$ the random variable chosen according to the joint query-outcome distribution $\tilde{D}^k_x$, then T can be viewed as function that maps (J_x, R) to a value in the range B.

When T is index-oblivious, then its output is determined just by the outcomes of the query indices $x_{i_1}, \ldots, x_{i_k}$ and by the coin tosses, without an explicit dependence on the indices $i_1, \ldots, i_k$. Thus, in this case T maps (O_x, R) into B, where $O_x = (x_{i_1}, \ldots, x_{i_k})$ is chosen according to the outcome distribution D^k_x.

In the following we show that when T is index-aware the distributions of J_x and J_y are far from each other for ϵ-disjoint x and y, and similarly for O_x and O_y when T is index-oblivious. Since the argument for both cases is the same, we generically view T as mapping a pair (S_x, R) to B, and we will argue that the distance between the distributions of S_x and S_y is large. We denote the distribution of S_x by P_x, and its domain by S (in the first case $P_x = \tilde{D}^k_x$ and $S = ([n] \times A)^k$, and in the second case $P_x = D^k_x$ and $S = A^k$).

Let $q_z \stackrel{\text{def}}{=} \Pr_{S_z, R}(T(S_z, R) \notin C_{f,\epsilon}(z))$ for an input z. By definition, $q_z \leq \delta$ for every input z. We write q_z as $q_z = \sum_{s \in S} P_z(s) Q_z(s)$, where $Q_z(s) \stackrel{\text{def}}{=} \Pr_R(T(s, R) \notin C_{f,\epsilon}(z))$.

Note that since $C_{f,\epsilon}(x) \cap C_{f,\epsilon}(y) = \emptyset$, we have $Q_y(s) \geq \Pr_R(T(s, R) \in C_{f,\epsilon}(x)) = 1 - Q_x(s)$.

We can rewrite the sum $q_x + q_y$ as follows:

$$\begin{aligned}
q_x + q_y &= \sum_{s \in S} P_x(s) Q_x(s) + P_y(s) Q_y(s) \\
&\geq \sum_{s \in S} P_x(s) Q_x(s) + P_y(s)(1 - Q_x(s)) \\
&\geq \sum_{s \in S} \min(P_x(s), P_y(s)) \\
&= \frac{1}{2} \sum_{s \in S} P_x(s) + P_y(s) - |P_x(s) - P_y(s)| \\
&= 1 - d(P_x, P_y)
\end{aligned}$$

Since $q_x + q_y \leq 2\delta$, then $d(P_x, P_y) \geq 1 - 2\delta$. □

4.4 Sample Complexity for Distinguishing Distributions

The main aim of this section is to obtain a lower bound on the number of samples required in order to distinguish between two distributions. We will use two measures of distance between distributions: the variation distance and the Hellinger distance.

Variation distance is a measure of distinguishability between distributions. In order to get a lower bound on the number of samples required to distinguish two distributions P and Q with some advantage α, we need to find a lower bound on the minimal integer k for which $d(P^k, Q^k) \geq \alpha$. Here, P^k and Q^k are distributions on Ω^k obtained by picking k independent random samples from P and Q respectively. However, variation distance does not behave well under product distributions, and in particular there is no simple formula that describes $d(P^k, Q^k)$ in terms of $d(P, Q)$. We therefore use the Hellinger distance, which has a nice multiplicativity property and bounds the variation distance from both sides:

Proposition 4.10

(1) $1 - h^2(P^k, Q^k) = (1 - h^2(P, Q))^k$

(2) $h^2(P, Q) \leq d(P, Q) \leq h(P, Q)\sqrt{2 - h^2(P, Q)}$.

The proof appears, e.g., in [LL90]. We repeat it here for completeness:

Proof. Proof of (1):

$$\begin{aligned} 1 - h^2(P^k, Q^k) &= \sum_{\omega_1,\ldots,\omega_k \in \Omega} \sqrt{P^k(\omega_1, \ldots, \omega_k) Q^k(\omega_1, \ldots, \omega_k)} \\ &= \sum_{\omega_1,\ldots,\omega_k \in \Omega} \sqrt{P(\omega_1) \cdots P(\omega_k) \cdot Q(\omega_1) \cdots Q(\omega_k)} \\ &= (\sum_{\omega \in \Omega} \sqrt{P(\omega) Q(\omega)})^k \\ &= (1 - h^2(P, Q))^k \end{aligned}$$

Proof of (2):

$$\begin{aligned} h^2(P, Q) &= \frac{1}{2} \sum_{\omega \in \Omega} (\sqrt{P(\omega)} - \sqrt{Q(\omega)})^2 \leq \frac{1}{2} \sum_{\omega \in \Omega} \left| \sqrt{P(\omega)} - \sqrt{Q(\omega)} \right| (\sqrt{P(\omega)} + \sqrt{Q(\omega)}) \\ &= \frac{1}{2} \sum_{\omega \in \Omega} |P(\omega) - Q(\omega)| = d(P, Q) \end{aligned}$$

Using the Cauchy-Schwartz inequality:

$$\begin{aligned} d(P, Q) &= \frac{1}{2} \sum_{\omega \in \Omega} |P(\omega) - Q(\omega)| = \frac{1}{2} \sum_{\omega \in \Omega} \left| \sqrt{P(\omega)} - \sqrt{Q(\omega)} \right| (\sqrt{P(\omega)} + \sqrt{Q(\omega)}) \\ &\leq \sqrt{\frac{1}{2} \sum_{\omega \in \Omega} (\sqrt{P(\omega)} - \sqrt{Q(\omega)})^2 \cdot \frac{1}{2} \sum_{\omega \in \Omega} (\sqrt{P(\omega)} + \sqrt{Q(\omega)})^2} \end{aligned}$$

$$= \sqrt{h^2(P,Q)} \cdot \sqrt{\frac{1}{2}(\sum_{\omega\in\Omega} P(\omega) + \sum_{\omega\in\Omega} Q(\omega) + 2\sum_{\omega\in\Omega}\sqrt{P(\omega)Q(\omega)})}$$
$$= h(P,Q)\cdot\sqrt{2-h^2(P,Q)}$$

□

The following lemma provides the desired lower bound in terms of the distributions' Hellinger distance. It was pointed out to us by David Zuckerman [personal communication, September 2000].

Lemma 4.11 *Let P and Q be two distributions on Ω with $h^2(P,Q) \le 1/2$, let $0 < \alpha < 1$, and let k be an integer such that $d(P^k, Q^k) \ge \alpha$. Then: $k \ge \frac{1}{4h^2(P,Q)} \ln \frac{1}{1-\alpha^2}$*

Proof. By Proposition 4.10 (part (2)), we have

$$h(P^k, Q^k)\sqrt{2 - h^2(P^k, Q^k)} \ge d(P^k, Q^k) \ge \alpha,$$

which implies that $h^4(P^k, Q^k) - 2h^2(P^k, Q^k) + \alpha^2 \le 0$. The solution to this quadratic inequality gives $h^2(P^k, Q^k) \ge 1 - \sqrt{1-\alpha^2}$.

By Proposition 4.10 (part (1)), we have $1 - (1 - h^2(P,Q))^k \ge 1 - \sqrt{1-\alpha^2}$. Using the inequality $1 - x \ge e^{-2x}$ for $0 \le x \le 1/2$, we have:

$$k \ge \frac{\ln \frac{1}{\sqrt{1-\alpha^2}}}{\ln \frac{1}{1-h^2(P,Q)}} \ge \frac{1}{4h^2(P,Q)} \ln \frac{1}{1-\alpha^2}$$

□

4.5 The Lower Bound

We are now ready to prove our main theorem:

Proof. [of Theorem 4.4] Let $k = S^w_{\epsilon,\delta}(f)$. We prove each of the two parts separately:

Proof of part (1):

By part (1) of Theorem 4.8, if $k \le \sqrt{2\delta n}$, there exists an index-oblivious U^k-decision tree T that $(\epsilon, 2\delta)$-approximates f. Let x and y be the ϵ-disjoint inputs that achieve the function's minimum Hellinger distance: $h(U_x, U_y) = h_\epsilon(f)$. Lemma 4.9 implies that $d(U_x^k, U_y^k) \ge 1 - 4\delta$. Since U_x^k and U_y^k are the k-fold products of U_x and U_y, and since $h^2(U_x, U_y) \le 1/2$, we can apply the distinguishability lower bound (Lemma 4.11) to complete the proof of this part.

Proof of part (2):

By part (2) of Theorem 4.8, if $k \le \frac{n}{2} - \sqrt{\frac{n}{2\delta}}$, then there exists an index-aware U^{2k}-decision tree T that $(\epsilon, 2\delta)$-approximates f. Let x and y be the ϵ-disjoint inputs that achieve the function's minimum variation distance: $d(U_x, U_y) = d_\epsilon(f)$. Lemma 4.9 implies that $d(\tilde{U}_x^{2k}, \tilde{U}_y^{2k}) \ge 1 - 4\delta$.

We denote by $\tilde{U}_x$ and $\tilde{U}_y$ the distributions on $[n] \times A$ obtained by picking $i \in [n]$ uniformly at random and returning the pairs (i, x_i) and (i, y_i) respectively. Clearly, $\tilde{U}_x^{2k}$ and $\tilde{U}_y^{2k}$ are the $2k$-fold product distributions of $\tilde{U}_x$ and $\tilde{U}_y$. We next prove two simple facts about the distance between $\tilde{U}_x$ and $\tilde{U}_y$.

Claim 4.12 *For all inputs $x, y \in A^n$,*

$$h^2(\tilde{U}_x, \tilde{U}_y) = d(\tilde{U}_x, \tilde{U}_y) = Ham(x, y)$$

where $Ham(x, y)$ is the relative Hamming distance between x and y (i.e., $Ham(x, y) = \frac{1}{n}|\{i \in [n] \mid x_i \neq y_i\}|$).

Proof. By definition (see Definition 4.2), we have:

$$h^2(\tilde{U}_x, \tilde{U}_y) = \frac{1}{2} \sum_{i \in [n], a \in A} (\sqrt{\tilde{U}_x(i,a)} - \sqrt{\tilde{U}_y(i,a)})^2$$

$$d(\tilde{U}_x, \tilde{U}_y) = \frac{1}{2} \sum_{i \in [n], a \in A} |\tilde{U}_x(i,a) - \tilde{U}_y(i,a)|$$

Every term in each of the above sums corresponds to a pair $(i, a) \in [n] \times A$. In both sums, this term is $1/n$ if and only if exactly one of x_i, y_i equals a, and is 0 otherwise. Thus, each index $i \in [n]$ for which $x_i \neq y_i$ contributes $2/n$ to each of the two sums ($1/n$ in the term (i, x_i) and $1/n$ in the term (i, y_i)), while each index i for which $x_i = y_i$ contributes 0. It follows that:

$$h^2(\tilde{U}_x, \tilde{U}_y) = d(\tilde{U}_x, \tilde{U}_y) = \frac{1}{2} \cdot \frac{2}{n} \cdot |\{i \in [n] \mid x_i \neq y_i\}| = Ham(x, y)$$

□

Claim 4.13 *For all inputs $x, y \in A^n$,*

$$d(U_x, U_y) = \min\{d(\tilde{U}_{\pi(x)}, \tilde{U}_{\sigma(y)}) \mid \pi, \sigma \in S_n\}$$

Proof.

$\leq$:

Since U_x and U_y can obtained from $\tilde{U}_x$ and $\tilde{U}_y$ by projecting on their second coordinate, then $d(U_x, U_y) \leq d(\tilde{U}_x, \tilde{U}_y)$. Let $\pi^*, \sigma^* \in S_n$ be the permutations in which the minimum of $d(\tilde{U}_{\pi(x)}, \tilde{U}_{\sigma(y)})$ is obtained. Note that $d(U_x, U_y)$ is invariant under permutations: for all $\pi, \sigma \in S_n$, $d(U_x, U_y) = d(U_{\pi(x)}, U_{\sigma(y)})$. Thus,

$$d(U_x, U_y) = d(U_{\pi^*(x)}, U_{\sigma^*(y)}) \leq d(\tilde{U}_{\pi^*(x)}, \tilde{U}_{\sigma^*(y)}) = \min\{d(\tilde{U}_{\pi(x)}, \tilde{U}_{\sigma(y)}) \mid \pi, \sigma \in S_n\}$$

$\geq$:

We first describe a procedure to permute x and y. We then claim that the obtained permutations π^* and σ^* satisfy $d(U_x, U_y) \geq d(\tilde{U}_{\pi^*(x)}, \tilde{U}_{\sigma^*(y)})$.

The main idea is to move all the shared elements of x and y to their front, such that they will have a maximal common prefix. Formally, for each $a \in A$, denote by $f_x(a)$ and $f_y(a)$ the frequencies of a in x and y respectively (i.e., $f_x(a) = |\{i \in [n] \mid x_i = a\}|$ and similarly for $f_y(a)$). Let $m(a) = \min(f_x(a), f_y(a))$ and $M(a) = \max(f_x(a), f_y(a))$. Fix some arbitrary order on A, and permute x and y as follows: enumerate the elements in $a \in A$ that have at least one occurrence in either x or y from smallest to largest according to the fixed order; for each such a, move its first $m(a)$ occurrences in x and in y to the front of x and y respectively. Note that after this procedure ends, x and y have a common prefix of length $m \stackrel{\text{def}}{=} \sum_{a \in A} m(a)$; their two suffixes (of length $n - m$) are different in every position.

Denote by π^* and σ^* the permutations this procedure induces on x and y respectively. By Claim 4.12, $d(\tilde{U}_{\pi^*(x)}, \tilde{U}_{\sigma^*(y)})$ equals the relative Hamming distance between $\pi^*(x)$ and $\sigma^*(y)$. Therefore,

$$d(\tilde{U}_{\pi^*(x)}, \tilde{U}_{\sigma^*(y)}) = \frac{n-m}{n}$$

On the other hand, if we define $M \stackrel{\text{def}}{=} \sum_{a \in A} M(a)$, then we have:

$$d(U_{\pi^*(x)}, U_{\sigma^*(y)}) = d(U_x, U_y) = \frac{1}{2} \sum_{a \in A} |U_x(a) - U_y(a)| = \frac{1}{2} \sum_{a \in A} \frac{M(a) - m(a)}{n} = \frac{M-m}{2n}$$

Note that $M + m = 2n$ (since $M + m$ is the sum of frequencies of all $a \in A$ in x and y), implying that $(M - m)/2n = (n - m)/n$. Therefore,

$$d(U_x, U_y) = d(\tilde{U}_{\pi^*(x)}, \tilde{U}_{\sigma^*(y)}) \geq \min\{d(\tilde{U}_{\pi(x)}, \tilde{U}_{\sigma(y)}) \mid \pi, \sigma \in S_n\}$$

□

Going back to the proof of the theorem, we pick permutations $\pi, \sigma \in S_n$, such that $d(\tilde{U}_{\pi(x)}, \tilde{U}_{\sigma(y)}) = d(U_x, U_y)$. Note that $d(U_{\pi(x)}, U_{\sigma(y)}) = d(U_x, U_y) = d_\epsilon(f)$. Thus, without loss of generality, we can assume that x and y themselves were chosen such that $d(U_x, U_y) = d(\tilde{U}_x, \tilde{U}_y)$.

Since $\tilde{U}_x^{2k}$ and $\tilde{U}_y^{2k}$ are the $2k$-fold product distributions of $\tilde{U}_x$ and $\tilde{U}_y$, and since $h^2(\tilde{U}_x, \tilde{U}_y) = d(\tilde{U}_x, \tilde{U}_y) = d(U_x, U_y) \leq 1/2$, then we can apply the distinguishability lower bound, and obtain:

$$k \geq \frac{1}{8h^2(\tilde{U}_x, \tilde{U}_y)} \ln \frac{1}{8\delta} = \frac{1}{8d(U_x, U_y)} \ln \frac{1}{8\delta}$$

□

4.6 Expected Query Complexity vs. Worst-Case Query Complexity

In this section we show that the worst query complexity of any function is at most a factor of $O(\log \frac{1}{\delta})$ away from its expected query complexity:

Proposition 4.14 *For any function f, $S^w_{\epsilon,3\delta}(f) \leq 2S^e_{\epsilon,\delta}(f) \cdot \lceil \log \frac{1}{\delta} \rceil$.*

Proof. Let T be a decision tree of expected query complexity k that (ϵ, δ)-approximates f. We show that there exists a decision tree T that $(\epsilon, 3\delta)$-approximates f, and has worst-case query complexity of $2k\lceil \log \frac{1}{\delta} \rceil$.

Given an input x, T' runs at most $d \stackrel{\text{def}}{=} \lceil \log \frac{1}{\delta} \rceil$ independent simulations of T one after the other. Each simulation is carried out for at most $2k$ queries. If at any simulation T outputs a value before the $2k+1$-st query, T' outputs the same value and halts. Otherwise, it continues to the next simulation. If none of the d simulations outputs a value by the $2k$-th query, T' outputs an arbitrary value and halts.

It is clear that T' has a worst-case query complexity of at most $2kd = 2k\lceil \log \frac{1}{\delta} \rceil$. We are left to show that it $(\epsilon, 3\delta)$-approximates f. Let us denote by $S(T,x)$ the number of queries T performs on x. Clearly, $E(S(T,x)) \leq k$, and therefore, by Markov's inequality, $\Pr(S(T,x) > 2k) < 1/2$. Let us denote by F the event that $S(T,x) > 2k$ for all the d simulations of T. Since the simulations are independent, $\Pr(F) \leq (\frac{1}{2})^d \leq \delta$. Therefore,

$$\begin{aligned}
\Pr(T'(x) \notin C_{f,\epsilon}(x)) &= \Pr(T'(x) \notin C_{f,\epsilon}(x) \mid F) \Pr(F) + \Pr(T'(x) \notin C_{f,\epsilon}(x) \mid \overline{F}) \Pr(\overline{F}) \\
&\leq \Pr(F) + \Pr(T'(x) \notin C_{f,\epsilon}(x) \mid \overline{F}) \\
&\leq \delta + \frac{\Pr(T(x) \notin C_{f,\epsilon}(x) \wedge S(T,x) \leq 2k)}{\Pr(S(T,x) \leq 2k)} \\
&\leq \delta + 2\Pr(T(x) \notin C_{f,\epsilon}(x)) \leq 3\delta
\end{aligned}$$

□

Theorem 4.4 and Proposition 4.14 immediately imply now Theorem 4.5.

5 Applications

5.1 Statistical Moments

The k-th statistical moment, $\mu_k : [0,1]^n \to \mathbf{R}$, is defined as $\mu_k(x_1, \ldots, x_n) = \frac{1}{n}\sum_{i=1}^n x_i^k$. The first moment, for example, is simply the mean. An additive approximation of μ_k can be easily performed by taking a random sample of size $O(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$, and computing the sample's k-th moment. Chernoff-Hoeffding bound [Che52, Hoe63] implies that this yields an ϵ-additive approximation with probability $1-\delta$. Canetti et al. [CEG95] prove that the mean (and, implicitly, all the other moments too) require $\Omega(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$ samples. Using Theorem 4.4 we obtain a much simpler proof for this result:

Theorem 5.1 *For all* $\frac{1}{2n} < \delta < \frac{1}{2}$ *and* $\Omega(\sqrt{\frac{\ln(1/\delta)}{\sqrt{\delta}n}}) < \epsilon < \frac{1}{\sqrt{8}}$,

$$S^w_{\epsilon,\delta}(\mu_k) \geq \frac{1}{16\epsilon^2} \ln \frac{1}{8\delta}$$

Proof. Consider the two inputs (or equivalently, distributions) x and y. x is defined to be 0 with probability $1/2 - \epsilon$ and 1 otherwise; y is defined to be 0 with probability $1/2 + \epsilon$ and 1 otherwise. Since $\mu_k(x) = 1/2 + \epsilon$ and $\mu_k(y) = 1/2 - \epsilon$, x and y are ϵ-disjoint. We bound the Hellinger distance of x and y as follows: $h^2(U_x, U_y) = 1 - 2\sqrt{(1/2+\epsilon)(1/2-\epsilon)} = 1 - \sqrt{1-4\epsilon^2} \leq 1 - (1-4\epsilon^2) = 4\epsilon^2$. The lower bound follows from part (1) of Theorem 4.4. □

5.2 Frequency Moments

Given a sequence $x = x_1, \ldots, x_n$ where $x_i \in [m]$, the k-th frequency moment of x is defined to be $F_k(x) \stackrel{\text{def}}{=} \sum_{i=1}^m q_i^k$, where $q_i = |\{j \mid x_j = i\}|$, the number of occurrences of i in the sequence. We prove upper and lower bounds for a relative approximation of F_k that are almost matching.

Theorem 5.2 *Let* $r = \min\{m, n\}$. *Then, for any* $k \geq 2$, *and for all* $\frac{2}{n} < \delta < \frac{1}{2}$ *and* $\Omega(\sqrt{\frac{(2\ln(1/\delta))^k}{n}}) < \epsilon < 1$,

(1) $\Omega(r^{1-1/k} \frac{1}{\epsilon^{1/k}} \ln \frac{1}{\delta}) \leq S^w_{\epsilon,\delta}(F_k)$

(2) $S^w_{\epsilon,\delta}(F_k) \leq O(kr^{1-1/k} \frac{1}{\epsilon^{2/(k-1)}} \log \frac{1}{\delta})$

Proof. For simplicity, we assume $m \geq n$. For the lower bound, we will use part (2) of Theorem 4.4, since we do not have an upper bound of $O(\sqrt{n})$ on $S^w_{\epsilon,\delta}(F_k)$. Consider the following two inputs: the first input is $x = 1, \ldots, n$. The second input is constructed from x as follows. Divide the input of length n into $n^{1-1/k}/(3\epsilon)^{1/k}$ blocks of size $(3\epsilon n)^{1/k}$. y is x except in the first block where it is all 1's. Clearly, $F_k(x) = n$ while $F_k(y) = 3\epsilon n + (n - (3\epsilon n)^{1/k}) \approx n(1 + 3\epsilon)$, implying x and y are ϵ-disjoint. Since $d(U_x, U_y) \leq O(\epsilon^{1/k}/n^{1-1/k})$, the lower bound follows from Theorem 4.4.

Now, we show that this lower bound is almost tight. Note that for any input x, $U_x(i) = q_i/n$. Thus, computing $F_k(x)$ is equivalent to estimating the k-wise collision probability of U_x, since $F_k(x) = \sum_{i=1}^m q_i^k = n^k \sum_{i=1}^n U_x(i)^k$. So, the sampling algorithm picks ℓ samples $x_1, \ldots, x_\ell$ using U_x and counts the k-wise collisions C_k, our basic estimator. It is clear that $E(C_k) = \binom{\ell}{k} \|U_x\|_k^k$. The following lemma, whose proof generalizes that of [GR00] for F_2, helps us to bound the variance of this estimator.

Lemma 5.3 $\mathrm{Var}(C_k) \leq O(E(C_k)^{1+1/k})$.

Now, noting that $\|U_x\|_k^k \geq 1/r^{k-1}$, by Chebyshev inequality, $\Pr(|C_k - E(C_k)| \geq \epsilon E(C_k)) \leq O(1/(\epsilon^2 E(C_k)^{1-1/k})) \leq O(k^{k-1} r^{(k-1)^2/k}/(\epsilon^2 \ell^{k-1}))$. For this to be a constant, say, 3/4, we get $\ell \geq \Omega(kr^{1-1/k}/\epsilon^{2/(k-1)})$. Now, this experiment can be repeated $O(\log 1/\delta)$ times and the median of these various trials is correct with probability at least $1 - \delta$. □

We next give a simple new proof for (a slightly weaker version of) the sampling lower bound of Charikar et al. [CCMN00] for a ratio approximation of F_0:

Theorem 5.4 *For all* $\frac{2}{n} < \delta < \frac{1}{2}$ *and* $\sqrt{\frac{2}{n}} < \epsilon < \sqrt{\frac{1}{2\ln(1/\delta)}}$,

$$S^w_{\epsilon,\delta}(F_0) \geq \Omega(\epsilon^2 n \ln \frac{1}{\delta})$$

Proof. Let x be the input that is all 1's and let y be the input that has $n - \lfloor(1/\epsilon^2)\rfloor$ 1's, followed by $2, 3, \ldots, \lfloor(1/\epsilon^2)\rfloor + 1$. Note that $F_0(z)$ counts the number of distinct elements in z, and thus $F_0(x) = 1, F_0(y) \geq 1/\epsilon^2$, implying x and y are ϵ-disjoint. Now, $d(U_x, U_y) = \lfloor(1/\epsilon^2)\rfloor/n \leq 1/(n\epsilon^2)$. The lower bound follows from part (2) of Theorem 4.4. □

5.3 Selection Functions

Chernoff bound implies that the median of a random sample of size $O(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$ yields an ϵ-approximate median. The same result holds for Select_q for any constant $0 < q < 1$. However, for $q = 0$ (minimum) and for $q = 1$ (maximum), a sample of size $O(\frac{1}{\epsilon}\log\frac{1}{\delta})$ is sufficient. We next show that these bounds are tight:

Theorem 5.5 *For all* $\frac{1}{2n} < \delta < \frac{1}{2}$ *and* $\Omega(\sqrt{\frac{\ln(1/\delta)}{\sqrt{\delta}n}}) < \epsilon < \frac{1}{\sqrt{8}}$,

(1) For $\epsilon \leq q \leq 1 - \epsilon$, $S^w_{\epsilon,\delta}(\text{Select}_q) \geq \Omega(\frac{q(1-q)}{\epsilon^2}\log\frac{1}{\delta})$

(2) For $q < \epsilon, q > 1 - \epsilon$, $S^w_{\epsilon,\delta}(\text{Select}_q) \geq \Omega(\frac{1}{\epsilon}\log\frac{1}{\delta})$

Proof. Assume, initially, that $\epsilon \leq q \leq 1 - \epsilon$. We pick $x, y \in \{0,1\}^n$ as follows: x has $(q+\epsilon)n$ 0's and the rest are 1's, and y has $(q-\epsilon)n$ 0's and the rest are 1's. Clearly, $C_{\text{Select}_{q,\epsilon}}(x) = \{0\}$ and $C_{\text{Select}_{q,\epsilon}}(y) = \{1\}$, implying x and y are ϵ-disjoint. Since $h^2(U_x, U_y) \leq \epsilon^2/(q(1-q))$, the bound follows from part (1) of Theorem 4.4.

When $q < \epsilon$, we pick x as before and $y = 1^n$, and when $q > 1 - \epsilon$, we pick y as before and $x = 0^n$. The fact that $h^2(U_x, U_y) \leq 2\epsilon$ implies the lower bound. □

5.4 Extractor and Disperser Lower Bounds

Using the lower bounds for the mean and the maximum, we can immediately derive the lower bounds on the seed length of extractors and dispersers originally proved by Radhakrishnan and Ta-Shma [RTS00]. We first recall the definitions of dispersers and extractors:

Definition 5.6 (Disperser) *A function* $D : N \times T \to M$, *where* $|N| = 2^n, |T| = 2^t, |M| = 2^m$, *is a* (k,ϵ)-disperser, *if for all subsets* $K \subseteq N$ *of size at least* 2^k, $|D(K,T)| \geq (1-\epsilon)2^m$.

Definition 5.7 (Extractor) *A function $E : N \times T \to M$, where $|N| = 2^n, |T| = 2^t, |M| = 2^m$, is a* (k, ϵ)-extractor, *if for all distributions X on N with min-entropy at least k, $d(E(X, U_T), U_M) < \epsilon$, where U_T and U_M are the uniform distributions on T and M respectively.*

Zuckerman [Zuc97] showed that any extractor implies an (oblivious) sampling algorithm for the mean:

Lemma 5.8 (Zuckerman) *If there exists a (k, ϵ)-extractor $E : N \times T \to M$, then there exists a sampling algorithm T of worst-case query complexity 2^t that $(\epsilon, 1/2^{n-k-1})$-approximates the mean $\mu_1 : [0,1]^{|M|} \to [0,1]$.*

A similar argument can show that any disperser implies a sampling algorithm for the maximum:

Lemma 5.9 *If there exists a (k, ϵ)-disperser $D : N \times T \to M$, then there exists a sampling algorithm T of worst-case query complexity 2^t that $(\epsilon, 1/2^{n-k})$-approximates the maximum* $\max : \mathbf{R}^{|M|} \to \mathbf{R}$.

Using these connections and our lower bounds for the mean and the maximum, we can obtain a simple proof for extractor and disperser seed length lower bounds:

Theorem 5.10

(1) Let $D : N \times T \to M$ be a (k, ϵ)-disperser. Then $t \geq \log(n-k) + \log \frac{1}{\epsilon} - O(1)$.

(2) Let $E : N \times T \to M$ be a (k, ϵ)-extractor. Then $t \geq \log(n-k) + 2\log \frac{1}{\epsilon} - O(1)$.

Proof. Using Lemma 5.9, the existence of D implies the existence of a sampling algorithm T_D of worst-case query complexity 2^t that $(\epsilon, 1/2^{n-k})$-approximates $\max : \mathbf{R}^{2^m} \to \mathbf{R}$. Our sampling lower bound for the maximum (Theorem 5.5) implies that:

$$2^t \geq \Omega(\frac{1}{\epsilon}\log(2^{n-k})) = \Omega(\frac{1}{\epsilon}(n-k))$$

which derives part (1) of the theorem.

Similarly, Lemma 5.8 implies that the existence of E yields a sampling algorithm T_E that $(\epsilon, 1/2^{n-k-1})$-approximates $\mu_1 : [0,1]^{2^m} \to [0,1]$ with 2^t queries. Our lower bound for the mean (Theorem 5.1) implies that:

$$2^t \geq \Omega(\frac{1}{\epsilon^2}\log(2^{n-k-1})) = \Omega(\frac{1}{\epsilon^2}(n-k-1))$$

which derives part (2) of the theorem. □

6 Tightness of the Bounds

In this section, we discuss the tightness of our bounds. First, we show that for symmetric functions the lower bound achieved in Section 4 is always at least as strong as the lower bound via block sensitivity.

Theorem 6.1 *For all $\epsilon \geq 0$ and every ϵ-symmetric function $f : A^n \to B$, $\frac{1}{h_\epsilon^2(f)} \geq \frac{1}{d_\epsilon(f)} \geq bs_\epsilon(f)$.*

Proof. The first inequality follows from Proposition 4.10. For the second inequality let x be the input that achieves the block sensitivity of f, and let $I \subseteq [n]$ be the largest block to which f is ϵ-sensitive on x. Note that $bs_\epsilon(f) = bs_\epsilon(f, x) \leq n/|I|$.

Let $Q \in A^{|I|}$ be the assignment to I such that x and $y = x^{(I \leftarrow Q)}$ are ϵ-disjoint. Since x and y differ in $|I|$ positions, then $d(U_x, U_y) \leq |I|/n$. Therefore, by definition, $d_\epsilon(f) \leq |I|/n$. The theorem follows. □

We now show that the lower bound achieved in Section 4 is reasonably tight for functions over small finite domains (e.g., functions over a Boolean domain). The proof is due to Luca Trevisan [personal communication, March 2001].

Theorem 6.2 *If $f : A^n \to B$ is an ϵ-symmetric function with approximation $C_{f,\epsilon}$ that satisfies the sunflower property, then $S^w_{\epsilon,\delta}(f) \leq O(|A| \frac{1}{d_\epsilon^2(f)} \ln \frac{1}{\delta})$.*

Proof. Given input x, the algorithm for approximating f will try to construct a distribution Q_x which approximates U_x in variation distance to within an additive factor of $\beta = d_\epsilon(f)/18$.

The algorithm constructs Q_x by picking k independent samples $x_1, \ldots, x_k$ uniformly at random from x, and computing for all $a \in A$: $Q_x(a) = (1/k)|\{i \mid x_i = a\}|$. We choose $k = |A|/\beta^2$.

Define $e_a = |Q_x(a) - U_x(a)|$, the error made on a; we bound $E(e_a)$ as follows: $E(e_a)^2 \leq E(e_a^2) = E(|Q_x(a) - U_x(a)|^2) = \mathrm{Var}(Q_x(a))$. Define for all $i \in [k]$, $Z_i = 1$, if $x_i = a$ and $Z_i = 0$, otherwise. Then, $Q_x(a) = \frac{1}{k} \sum_{i \in [k]} Z_i$. We obtain $E(e_a)^2 \leq \mathrm{Var}(\frac{1}{k} \sum_{i \in [k]} Z_i) = \frac{1}{k^2} \sum_{i \in [k]} \mathrm{Var}(Z_i) = \frac{1}{k^2} \cdot k \cdot U_x(a)(1 - U_x(a)) \leq U_x(a)/k$. Now, if $U_x(a) \leq 1/|A|$, then $E(e_a) \leq \sqrt{(1/|A|)(\beta^2/|A|)} = \beta/|A|$. If $U_x(a) > 1/|A|$, then $E(e_a) \leq \sqrt{U_x(a)(\beta^2/|A|)} \leq \sqrt{U_x(a)^2 \beta^2} = \beta U_x(a)$. Thus, for all $a \in A$, $E(e_a) \leq \max\{\beta/|A|, \beta U_x(a)\}$.

Note that $d(Q_x, U_x) = \frac{1}{2} \sum_{a \in A} e_a$. Thus,

$$\begin{aligned} E(d(Q_x, U_x)) &= \frac{1}{2} \sum_{a \in A} E(e_a) \\ &\leq \frac{1}{2} \left(\sum_{a : U_x(a) \leq \frac{1}{|A|}} \frac{\beta}{|A|} + \sum_{a : U_x(a) > \frac{1}{|A|}} \beta U_x(a) \right) \\ &\leq \frac{1}{2} (|A| \cdot (\beta/|A|) + \beta \cdot 1) = \beta. \end{aligned}$$

Therefore, by Markov's inequality, $\Pr(d(Q_x, U_x) > 3\beta) < 1/3$.

The algorithm will generate ℓ independent approximations $Q_x^1, \ldots, Q_x^\ell$ for U_x, where $\ell = O(\ln(1/\delta))$. We call an approximation Q_x^i "good", if it is at distance of at most 3β from U_x. Chernoff bound implies that with probability at least $1 - \delta$, more than half of the approximations are good. Note that by the triangle inequality every two good approximations are at distance of at most 6β from

each other. The algorithm thus picks an approximation Q_x^j that is of distance at most 6β from more than half of the other approximations, if one exists. If indeed more than half of the approximations are good, then (1) Q_x^j exists (each good approximation can serve as Q_x^j) and (2) any Q_x^j we pick has at least one good approximation at distance of at most 6β; therefore, Q_x^j is at distance of at most 9β from U_x. We conclude that with probability $1-\delta$ our algorithms finds an approximation Q_x that satisfies $d(Q_x, U_x) \leq 9\beta = d_\epsilon(f)/2$.

The algorithm now looks for the set D of all inputs y for which $d(Q_x, U_y) < d_\epsilon(f)/2$. What we proved above shows that with probability $1-\delta$, $x \in D$. Moreover, by the triangle inequality for every two inputs $y, y' \in D$, $d(U_y, P_{y'}) < d_\epsilon(f)$. It follows from the definition of $d_\epsilon(f)$ that $C_{f,\epsilon}(y) \cap C_{f,\epsilon}(y') \neq \emptyset$. Using now the sunflower property of $C_{f,\epsilon}$, we obtain that there exists a value $b \in \bigcap_{y \in D} C_{f,\epsilon}(y) \subseteq C_{f,\epsilon}(x)$. The algorithm outputs b. □

The analysis of the technique applied in this theorem is tight in terms of its dependence on $|A|$, since there is a lower bound of $\Omega(|A|)$ on the number of samples required to approximate an unknown distribution in L_1 distance [BFR+00]. Therefore, in order to obtain a better upper bound one should apply a different technique.

For small domains this upper bound can be tight up to a constant factor. The tightness depends on the difference between $d_\epsilon(f)$ and $h_\epsilon(f)$. In the Boolean median, for example, $d_\epsilon(\text{median}) = h_\epsilon(\text{median}) = \epsilon$, which means that the bound is tight. In any case, since $d_\epsilon(f) \geq h_\epsilon^2(f)$, the difference between the upper and lower bounds is at most quadratic for small domains.

Finally, we show that block sensitivity is not a good lower bound in some cases.

Proposition 6.3 *There is a function $f : A^n \to B$ such that $S^w_{\epsilon,\delta}(f) = \Omega(\sqrt{n})$, while $bs_\epsilon(f) = O(1/\epsilon)$.*

Proof. Let f be the Boolean function on bounded-degree d graphs with n vertices such that $f(x) = 1$ iff x is a bipartite graph. x is represented by the graph's edge list (a list of dn integers in $[n]$). We consider the property testing approximation. If x is bipartite, to make it ϵ-far from being bipartite, we need to make at least ϵdn modifications to its edge list, implying $\Omega(\epsilon dn)$-sized blocks, which implies only $O(1/\epsilon)$ block sensitivity. On the other hand, Goldreich and Ron [GR97] prove that $S^w_{\epsilon,\delta}(f) = \Omega(\sqrt{n})$. □

7 Lossy Compression

In the context of computing with massive data sets, the following question is fairly natural and extremely important: what computations can be performed efficiently when data is available in compressed form? Specifically, it is of much interest to find algorithms that are able to compute interesting functions without completely decompressing the compressed data. Applications include pattern matching on compressed text files [ABF96], operations such as scene change detection and abrupt lighting change detection on video sequences [Cha95], and nearest neighbor computations.

This question becomes more interesting if, to gain larger compression factors, we are willing to accept some loss in the precision of the computation. The idea of *lossy compression* has been

around quite awhile, and is used in multimedia applications, where images and video sequences are routinely stored in compressed form. The guideline in these areas is that a lossy compression method is good if a typical human observer cannot tell the difference between an image and its version stored using the compression technique.

To our knowledge, there is no formal treatment of the notion of approximate computation from data stored using a lossy compression technique. The closest is the notion of computing from sketches, described in [FKSV99, BCFM98], and our definition below may be thought of as an extension of theirs.

Definition 7.1 (Lossy compression scheme) *For $\epsilon, \delta > 0$, a function $f : A^n \times F \to B$ is said to* admit a (c, ϵ, δ)-lossy compression scheme *if there exists a (probabilistic) compression function $\phi : A^n \to E$ and a (probabilistic) approximating function $\widetilde{f} : E \times F \to B$ such that:*

(1) $\log |E|/(n \log |A|) \leq c$;

(2) for all inputs $x \in A^n, y \in F$, $\widetilde{f}(\phi(x), y) \in C_{f,\epsilon}(x, y)$ with probability at least $1 - \delta$.

It is natural to wonder why f is taken to be a bivariate function in the definition above. If not, the trivial compression function $\phi = f$ and the trivial approximating function $\widetilde{f} =$ IDENTITY would be a perfectly good compression scheme. That is, if we wish to be able to compute $f(x)$ from the compression $\phi(x)$ of x, simply precompute $f(x)$ and let $\phi(x) = f(x)$. The point, in all real applications, is to be able to compress x without the knowledge of which y will be the argument for the computation of $f(x, y)$ (possibly in the future). For example, with image compression, the role of y is played by (reasonable) human observers, and the goal is to compress an image so that for any observer, the image and its compressed version look similar. In nearest neighbor computation, the role of y is played by a future "query" input for which we need to find the nearest database point from a set of database points stored in compressed form. The situation is similar for pattern matching.

We now point out that under certain conditions, a sampling algorithm for f implies a lossy compression scheme for f. The proof of the following proposition is straightforward.

Proposition 7.2 *If $f : A^n \times F \to B$ has an (ϵ, δ) sampling algorithm that, to compute $f(x, y)$, makes oblivious, uniformly random queries to entries of x, then f admits a (c, ϵ, δ)-lossy compression scheme, where c depends on the maximum space needed to store the internal state of the sampling algorithm.*

7.1 Streaming Algorithms

Consider a function $f : A^n \to B$. In the basic streaming model, a stream is an ordered sequence of pairs (i, x_i) where $i \in [n], x_i \in A$ that is presented on a unidirectional read-once tape. Each $i \in [n]$ occurs exactly once in the stream. The requirement of a streaming algorithm is that it should output the correct (possibly approximate) value of f irrespective of the ordering of the input sequence.

Definition 7.3 *An* (s,t)-streaming algorithm *for a function* $f : A^n \to B$ *is a streaming algorithm for* f *that, for every* $\epsilon, \delta \geq 0$ *uses work space* $s = s(n, |A|, \epsilon, \delta)$*, time* $t = t(n, |A|, \epsilon, \delta)$ *per data item, and for any input* x *with probability at least* $1 - \delta$*, outputs a* $y \in C_{f,\epsilon}(x)$*.*

Our next observation is that every sampling algorithm (a randomized decision tree) yields a streaming algorithm in a natural way.

Proposition 7.4 *A non-adaptive sampling algorithm for* f *that works for all* $\epsilon, \delta \geq 0$ *implies a streaming algorithm for* f *that uses space at most* $S^w_{\epsilon,\delta}(f) \cdot O(\log |A| + \log n)$*.*

Proof. Construct a streaming algorithm in the following manner. Without loss of generality, let the non-adaptive sampling algorithm toss coins to pick $S^w_{\epsilon,\delta}(f)$ indices to sample. Let this index set be I. Now, on an input stream consisting of tuples (i, x_i), if $i \in I$, the streaming algorithm stores x_i. After it has read all the samples, it runs the rest of the sampling algorithm. Clearly, the space used by the streaming algorithm is $S^w_{\epsilon,\delta}(f) \cdot O(\log |A| + \log n)$. □

Finally, we note that a streaming algorithm for a function yields a lossy compression scheme for the function. In particular, if the function is a metric, this yields a nearest neighbor scheme as well.

Proposition 7.5 *A streaming algorithm for* $f : A^n \times F \to B$ *that uses space* s *implies, for every* $\epsilon, \delta \geq 0$ *a* (c, ϵ, δ)*-lossy compression scheme for* f*, where* $c = O(s/(n \log |A|))$*.*

Proof. Given a streaming algorithm T for f that runs in space s, the compression function $\phi(x)$ is given by the contents of the work tape of T (of size $\leq s$) when T is run on x, pretending that it arrives as a stream. The approximating function $\tilde{f}$, given $\phi(x), y$, resumes the streaming algorithm T initializing the work tape to $\phi(x)$, and supplying y in a stream (in arbitrary order), and outputs the output of T. □

A partial converse to the above was proved by making additional assumptions about the encoding function of the lossy compression [FKSV99]. We omit all the parameters in the following statement.

Proposition 7.6 *[FKSV99] If* f *has a lossy compression scheme whose compression function has a streaming algorithm, then* f *has a streaming algorithm.*

7.2 Nearest Neighbor Schemes

We point out one interesting application of the connection to lossy compression. We define a metric nearest neighbor scheme and show that a lossy compression scheme for the metric yields an efficient nearest neighbor scheme.

Definition 7.7 (Nearest neighbor scheme) *For a metric space* A^d *with metric* f*, an* (s,t)-nearest neighbor scheme *is an algorithm that, for any* $\epsilon, \delta \geq 0$*, preprocesses* N *points* $X = \{x_1, \ldots, x_N\} \subseteq A^d$ *into a storage of size* $s = s(N, d, \epsilon, \delta)$ *such that for any query* $y \in A^d$*, with probability at least* $1 - \delta$*, it outputs an* $x \in X$ *in time* $t = t(N, d, \epsilon, \delta)$ *such that for all* $x' \in X$*,* $f(x, y) \leq (1 + \epsilon) f(x', y)$*.*

Proposition 7.8 *If $f : A^d \times A^d \to \mathbf{R}$ is a metric and has a (c, ϵ, δ)-lossy compression scheme with compression function ϕ and approximating function $\widetilde{f}$, then there is an $(Ncd \log |A|, Nt(\epsilon, \delta/N))$-nearest neighbor scheme for A^d with f as the metric, where $t(\epsilon, \delta)$ denotes the time for one application of $\widetilde{f}$ to succeed with error ϵ and confidence $1 - \delta$.*

Proof. We can take each $x_i \in X$ and compress under C to get a signature z_i of size $cd \log |A|$ and store it. Upon a query $y \in A^d$, for each stored signature z_i, we use the approximating function $\widetilde{f}$ to find the x_i such that $\widetilde{f}(z_i, y)$ is the minimum. The proposition follows from the properties of lossy compression. □

The parameters of the approximate nearest neighbor algorithm from Proposition 7.8 are quite poor in terms of N, since the algorithm simply applies $\widetilde{f}$ to each database item. If $\widetilde{f}$ is also a distance function in a smaller dimensional space, then one may recursively apply any of the known approximate nearest neighbor computation techniques to improve the speed. Unfortunately, $\widetilde{f}$ often involves the median operator, which is not a distance function. In an earlier version of this paper, we had claimed that this problem could be circumvented, but this claim appears to be erroneous. Independently, Indyk [Ind00] has shown how to handle some of the complications that arise due to the median; he applies the streaming algorithms of [AMS99, FKSV99] for distance functions to obtain embeddings of metric spaces into metric spaces.

8 Related Work

Most of the computer science literature on the query complexity of sampling algorithms concentrated on the variant of the model in which the functions are *Boolean* (i.e., $A = B = \{0, 1\}$) and the computation is required to be *exact*. This variant is captured by the famous Boolean decision tree model (see [Bd99] for a recent survey about this area).

The query complexity of deterministic, randomized, and non-deterministic computations of Boolean functions is well understood, and is known to be related to various function properties, like sensitivity and block sensitivity, certificate complexity, and degree of representing and approximating polynomials. The situation for the query complexity of approximation of general functions seems to be very different. For example, it is known that the query complexity of (even randomized) computation of any non-constant Boolean symmetric function is $\Omega(n)$ [Pat92, vR97]. On the other hand, the query complexity of approximating some non-Boolean symmetric functions (like the mean and the median) is only constant.

We are aware of only a few lower bounds in the theoretical computer science literature on the query complexity of non-Boolean function approximations; all of these are tailored to specific functions. Canetti et al. [CEG95] show a lower bound for additive approximation of the mean; Dagum et al. [DKLR95] (and also, implicitly, Schulman and Vazirani [SV99]) give lower bounds for relative approximation of the mean on any input x. Charikar et al. [CCMN00] prove a lower bound for ratio approximation of the frequency moment of order 0. Nayak and Wu [NW99] give a lower bound on the *quantum* query complexity of the median and some other statistics.

Sampling algorithms can be viewed as a special case of the general framework studied in statistical

decision theory [Ber85]. Classical decision theory takes a point of view, which is somewhat orthogonal to our goals, by seeking ways to get the best approximation from a given set of sample points. In particular, in their setting the algorithm has no control over the samples, their number, or their distribution.

Several sampling lower bounds from sub-areas of statistical decision theory are related to our work. The Cramér-Rao inequality (see, e.g., [Van68], Chapter 2.4) is a classical result in statistical estimation theory, which provides a lower bound on the variance of any unbiased estimator for an unknown parameter (that is, an estimator whose expectation is the true value of the parameter) in terms of the *Fisher information* of the sample distribution. Statistical learning theory [Val84, KV94, Vap98] includes lower bounds on the number of samples needed to learn an unknown function in terms of the VC dimension of the hypothesis space. The optimality of the sequential probability ratio test is a fundamental result in statistical sequential analysis (see, e.g., [Sie85], Chapter 1), which gives a lower bound on the expected number of samples needed to distinguish between two distributions.

All of the above bounds hold for algorithms whose queries are independent and identically distributed. Thus, the bounds do not apply, at least directly, to our setting, where the sampling algorithm may apply different sample distributions for each query, and may have queries that depend on each other. In fact, as we mentioned already before, our work on symmetric functions can be viewed as a reduction from the computer science setting to the statistical i.i.d. setting. We can then apply known tools from statistics, like the Hellinger distance, to obtain the lower bounds.

9 Discussion and Open Problems

This paper concentrates on a worst-case analysis of the (expected and worst-case) query complexity. Specifically, we defined the query complexity of a decision tree as its query complexity on the "worst" input. However, sometimes the query complexity varies significantly for different inputs and a worst-case analysis gives too pessimistic a view. For example, the query complexity of a relative approximation of the mean is $\Omega(n)$, since in such an approximation the inputs 0^n and $0^{n-1}1$ are ϵ-disjoint, while having Hellinger distance of $O(1/\sqrt{n})$. However, the mean of an input $x \in \{0,1\}^n$ with a constant number of 1's can be approximated with only $O(\epsilon^{-2}\log\delta^{-1})$ queries. This motivates the definition of the query complexity of a function $f : A^n \to B$ on a *specific* input $x \in A^n$, as $S^e_{\epsilon,\delta}(f,x) = \min\{S^e_{\epsilon,\delta}(T,x) \mid T \ (\epsilon,\delta)\text{-approximates } f\}$ (and similarly for $S^w_{\epsilon,\delta}(f,x)$). Our two lower bounds (Theorem 3.2 and Theorem 4.4) can be stated in terms of the query complexity of f on x: $S^e_{\epsilon,\delta}(f,x) \geq (1-2\delta)bs_\epsilon(f,x)$ and $S^w_{\epsilon,\delta}(f,x) \geq \Omega(\frac{1}{h^2_\epsilon(f,x)}\log\frac{1}{\delta})$, where $h^2_\epsilon(f,x) = \min\{h^2(U_x,U_y) \mid y \text{ is } \epsilon\text{-disjoint from } x\}$. Using this formulation, one can prove, for example, that a relative approximation of the mean on input $x \in \{0,1\}^n$ requires $\Omega(\frac{n}{\epsilon^2|x|}\log\frac{1}{\delta})$ queries, where $|x|$ denotes the Hamming weight of x.

As mentioned above, Boolean query complexity is known to be related to several function properties, one of which is block sensitivity. In this paper we explored only the generalization of block sensitivity to non-Boolean functions. Similar generalizations may be possible also for some of the other properties. However, it is not clear yet whether such generalizations would yield lower or upper bounds for the query complexity of general function approximation.

Our lower bound for the expected query complexity of symmetric functions (Theorem 4.5) has no dependence on the confidence δ. An interesting open problem is to improve this bound to include the missing $O(\log(1/\delta))$ factor.

The lower bound for symmetric function (Theorem 4.4) is based on the fact that an algorithm that approximates the function has to be able to distinguish between any pair of ϵ-disjoint inputs. However, in some cases there are many inputs that are all mutually disjoint from each other, and the algorithm manages to distinguish between all of them simultaneously. It seems that the *number* of mutually disjoint inputs should be a parameter in the lower bound (i.e., the greater this number is the higher the lower bound should be). One case where this comes into play is the estimation of the *empirical distribution* U_x of a given string x: $f : A^n \to [0,1]^{|A|}$, $f(x) = U_x$, $C_{f,\epsilon}(x) = \{D \mid d(D, U_x) < \epsilon\}$. For this function, Batu et al. [BFR+00] show a $\Omega(|A|)$ lower bound (with no dependence on ϵ or δ). Our results imply only an $\Omega(\epsilon^{-1}\log(1/\delta))$ lower bound (with no dependence on $|A|$). It seems that our bounded misses the $|A|$ factor, because it ignores the fact that this function has exponentially many mutually disjoint inputs.

Finally, an important open problem is to find tight lower bounds for the query complexity of non-symmetric functions. Currently, we have only the block sensitivity bound, which seems to be too weak.

Acknowledgments

We thank Ron Fagin, Michael Jordan, Christos Papadimitriou, Moni Naor, Andrew Ng, Luca Trevisan, and David Zuckerman for useful discussions.

References

[ABF96] A. Amir, G. Benson, and M. Farach. Let sleeping files lie: Pattern matching in Z-compressed files. *Journal of Computer and System Sciences*, 52(2):299–307, 1996.

[AMS99] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.

[BCFM98] A. Broder, M. Charikar, A. Frieze, and M. Mitzenmacher. Min-wise independent permutations. In *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 327–336, 1998.

[Bd99] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: A survey, 1999. Available at `http://www.cwi.nl/~rdewolf`.

[Ber85] J. O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag, 1985.

[BFR+00] T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White. Testing that distributions are close. In *Proceedings of the 41st IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 259–269, 2000.

[CCMN00] M. Charikar, S. Chaudhuri, R. Motwani, and V. Narasayya. Towards estimation error guarantees for distinct values. In *Proceedings of the 19th Annual ACM Symposium on Principles of Database Systems (PODS)*, pages 268–279, 2000.

[CEG95] R. Canetti, G. Even, and O. Goldreich. Lower bounds for sampling algorithms for estimating the average. *Information Processing Letters*, 53:17–25, 1995.

[Cha95] S.-F. Chang. Compressed-domain techniques for image/video indexing and manipulation. In *Invited article in IEEE International Conference on Image Processing, Special Session on Digital Image/Video Libraries and Video-on-demand*, 1995.

[Che52] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *American Mathematical Society*, 23:493–507, 1952.

[DKLR95] P. Dagum, R. Karp, M. Luby, and S. Ross. An optimal algorithm for monte carlo estimation. In *Proceedings of the 36th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 142–149, 1995.

[FKSV99] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate L^1-difference algorithm for massive data streams. In *Proceedings of the 40th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 501–511, 1999.

[FKSV00] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. Testing and spot-checking of data streams. In *Proceedings of the 11th IEEE Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 165–174, 2000.

[GR97] O. Goldreich and D. Ron. Property testing in bounded degree graphs. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 406–415, 1997.

[GR00] O. Goldreich and D. Ron. On testing expansion in bounded-degree graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 2000. TR00-020.

[Hoe63] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[HRR99] M. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. In *DIMACS series in Discrete Mathematics and Theoretical Computer Science*, volume 50, pages 107–118, 1999.

[Ind00] P. Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *Proceedings of the 41st IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 189–197, 2000.

[KV94] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory.* The MIT Press, 1994.

[LL90] L. Le Cam and G. Lo Yang. *Asymptotics in Statistics - Some Basic Concepts*, pages 24–30. Springer-Verlag, 1990.

[Nis91] N. Nisan. CREW PRAMs and Decision Trees. *SIAM Journal on Computing*, 20(6):999–1007, 1991.

[NW99] A. Nayak and F. Wu. The quantum query complexity of approximating the median and related statistics. In *Proceedings of the 31st Annual ACM Symposium on the Theory of Computing (STOC)*, pages 384–393, 1999.

[Pat92] R. Paturi. On the degree of polynomials that approximate symmetric boolean functions. In *Proceedings of the 24th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 468–474, 1992.

[RTS00] J. Radhakrishnan and A. Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM J. on Discrete Math*, 13(1):2–24, 2000.

[Sie85] D. Siegmund. *Sequential Analysis - Tests and Confidence Intervals.* Springer-Verlag, 1985.

[SV99] L. Schulman and V. V. Vazirani. Majorizing estimators and the approximation of #P-complete problems. In *Proceedings of the 31st Annual ACM Symposium on the Theory of Computing (STOC)*, pages 288–294, 1999.

[Val84] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[Van68] H. L. Van Trees. *Detection, Estimation, and Modulation Theory.* Jon Wiley & Sons, Inc., 1968.

[Vap98] V. N. Vapnik. *Statistical Learning Theory.* John Wiley & Sons, Inc., 1998.

[vR97] J. von zur Gathen and J. R. Roche. Polynomials with two values. *Combinatorica*, 17(3):345–362, 1997.

[Yao77] A.C-C. Yao. Probabilistic computations: toward a unified measure of complexity. In *Proceedings of the 18th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977.

[Zuc97] D. Zuckerman. Randomness-optimal oblivious sampling. *Random Structures and Algorithms*, 11(4):345–367, 1997.