New Lower Bounds for Statistical Query Learning

Ke Yang [*]

April 12, 2003

Abstract

We prove two lower bounds in the Statistical Query (SQ) learning model. The first lower bound is on weak-learning. We prove that for a concept class of SQ-dimension d, a running time of $\Omega(d/\log d)$ is needed. The SQ-dimension of a concept class is defined to be the maximum number of concepts that are "uniformly correlated", in that each pair of them have nearly the same correlation. This lower bound matches the upper bound in [BFJ+94], up to a logarithmic factor. We prove this lower bound against an "honest SQ-oracle", which gives a stronger result than the ones against the more frequently used "adversarial SQ-oracles". The second lower bound is more general. It gives a continuous trade-off between the "advantage" of an algorithm in learning the target function and the number of queries it needs to make, where the advantage of an algorithm is the probability it succeeds in predicting a label minus the probability it doesn't. Both lower bounds extend and/or strengthen previous results, and solve an open problem left in [Y01].

A preliminary version of this paper appeared in [Y02].

1 Introduction

1.1 The Statistical Query Model

The Statistical Query (SQ) model was first introduced by Kearns [K93]. Unlike the Probably Approximately Correct (PAC)-model [V84], a learning algorithm in the SQ model doesn't see explicit examples or their labels. Instead, the algorithm queries an "SQ-oracle" with boolean questions about random labeled examples. For each query, the oracle replies with an estimate of the probability that the answer is "YES" for a random example. An apparent restriction of the PAC model, the SQ model proved to be a very powerful and useful notion. Kearns showed how to efficiently simulate an SQ learning algorithm using a PAC algorithm, even in the presence of noise. He also proved that many PAC algorithms are indeed "SQ-typed", meaning that they can be converted to work in the SQ model. Particularly interesting is the case of noise-tolerant learning, where a random fraction of the examples are labeled wrong. In fact, most of the noise-tolerant PAC learning algorithms are "SQ-typed", due to the noise-tolerant nature inherent in the SQ model.

1.2 Lower Bounds on the SQ Model

Given that the SQ model is so useful, it is rather desirable to fully understand its strengths and limits. When the SQ model was introduced, people were interested in the question "which concept classes are SQ-learnable, and which are not?". In fact, in the seminal paper on the

[*]Computer Science Department, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213, USA. E-mail: `yangke@cs.cmu.edu`. Partially supported by the CMU SCS Alumni Fellowship and the NSF Aladdin center, Grant CCR-0122581.

SQ model, Kearns [K93] proved that PARITY functions are not SQ-learnable. Blum. et. al. [BFJ+94] defined the notion of "SQ-dimension" of a concept class, which is, roughly speaking, the number of "almost uncorrelated" concepts in this class. The SQ-dimension characterizes the weak SQ-learnability very well. In fact, Blum et. al. proved that, to weakly learn a concept class of SQ-dimension d, $O(d)$ queries suffice (though the algorithm might not be efficient), and $\Omega(d^{1/3})$ queries are necessary. However, the upper and lower bounds don't match, and the result only applies to weak learning.

Jackson [J00] strengthened the previous result by proving that for an "SQ-based" algorithm, $\Omega(2^n)$ examples are needed to learning the class of PARITY functions over n bits. An SQ-based algorithm is constructed by first designing an SQ learning algorithm, and then generically simulating each SQ query by sampling and averaging. This lower bound matches the corresponding upper bound. With this very strong result, Jackson was able to show a quadratic gap on learning PARITY functions with a high $(1/2 - 1/\mathsf{poly}(n))$ noise rate by SQ-learning and by PAC-learning. He demonstrated a PAC algorithm that learns noisy PARITY with running time $O(2^{n/2})$, while any SQ-based algorithm needs running time $\Omega(2^n)$.

Along another line of research, Yang [Y01] considered the problem of learning *uniformly correlated* concepts in the SQ model. That paper extended the notion of "SQ-dimension" to the case where each pair of concepts are "correlated in the same way". For correlated concept classes, it can be trivial to weakly learn a concept (in some cases, even without making any query), but it could take exponential time to do strong learning. Yang proved an $\Omega(d^{1/2}S)$ lower bound for learning such a concept class of SQ-dimension d with an "extra advantage" S. This result implies the SQ-unlearnability of the class of Linear Threshold Functions (LTFs) over finite fields. He also showed a PAC algorithm that beats the the best possible SQ algorithms in learning a special class of LTFs. An open problem from [Y01] was to characterize the SQ-learnability of general concept classes where the concepts can be *arbitrarily* correlated.

1.3 Variants of SQ-oracles

Different models of SQ-oracles have been proposed in the literature. Probably the most used one, which we call the "adversarial model", is the one introduced by Kearns in his original paper [K93]. In this model, the algorithm gives the oracle a query function g and an additive "error tolerance" ϵ. The oracle is allowed to reply with *any* real number that is ϵ-close to the expected value of g on a random labeled example. Notice this oracle is adversarial in natural, since it has the freedom to (adversarially) choose which real number to reply. This is a very nice model to prove *upper bounds*, since this gives a *worst-case* guarantee: an algorithm that works with such a adversarial oracle will surely work well with an SQ-oracle simulated by sampling. However, there are problems for proving *lower bounds* for this model. A typical strategy in the lower bound proofs [K93, BFJ+94] is to construct a "bad" oracle. This oracle doesn't commit to any particular target concept. Rather, on each query, the oracle replies with a value that is "consistent" with many "candidate concepts". The argument is that if not enough queries are made, the oracle will always be able to give answers that are consistent with many concepts. In this case, the learning algorithm won't be able to learn very well.

However, this model has a drawback: in practice, the SQ model isn't this adversarial. In practice, one normally *simulates* an SQ-oracle using a PAC learning algorithm. The simulation is done by repeated sampling and averaging the query function over these examples. There is nothing "adversarial" in the simulation and the simulated SQ-oracle behaves differently from the adversarial oracle used in the lower bound proofs.

Several efforts have been made to remedy this problem. Jackson [J00] introduced the notion of an "SQ-based PAC algorithm" to prove the $\Omega(2^n)$ lower bound for learning PARITY. Intuitively, an SQ-based PAC algorithm is constructed in the following way. One first designs a learning algorithm in the SQ-model. Then the SQ algorithm is "generically" simulated by a PAC algorithm in the most straightforward way. One important assumption about this generic simulation is that the algorithm cannot use any "internal knowledge" about the query function, nor any "clever tricks" to make the simulation more efficient. This assumption might seem a bit too strong in practice, since clever tricks are normally quite desirable in algorithm design. For example, Aslam and Decatur considered using *relative error* instead of additive error to efficiently simulate an SQ oracle [AS95]. Jackson's result didn't rule out the possibility that algorithms using such a simulation can learning PARITY functions more efficiently.

Yang [Y01] introduced the model of "honest SQ-oracles". In this model, the oracle receives a query g and a *sample count* M. Then the oracle independently picks M random labeled examples and returns the average of g on these examples. Intuitively, the honest SQ-oracle model can be regarded as the "SQ-based PAC" model without the "no-clever-trick" assumption. Any lower bound for algorithms using the honest SQ-oracle automatically translates to a lower bound for "SQ-based PAC" algorithms and a lower bound in the adversarial SQ-oracle model. An additional advantage of the "honest SQ-oracle" model is that it simplifies the presentation of the complexity of a learning algorithm. In the adversarial SQ-oracle model, two parameters go into the complexity of a learning algorithm: the error tolerance ϵ and the total number of queries. An efficient algorithm needs to make both $1/\epsilon$ and the total number of queries polynomial, and the trade-off between these two parameters is not always clear. However, in the honest SQ-oracle model, there is only one parameter, namely, the total sample count. This makes the presentation of the complexity of learning algorithms much simpler. The technique used in [Y01] was to keep track different "scenarios" when different concepts are chosen as targets, and to measure the "all-pair statistical distance" across the scenarios. This technique allows us to work in the new model, but it wasn't powerful enough to yield a tight lower bound.

1.4 Our Contributions

In this paper, we prove two lower bounds for SQ-learning algorithms. The first lower bound considers a concept class that contains a subset of uniformly correlated concepts, where each pair is correlated "in the same way". We define the *SQ-dimension* of a concept class to be the maximum number of such uniformly correlated concepts in this class. We also consider the *total sample count* of an algorithm, which is the sum of the sample counts of all the queries it makes to an honest SQ-oracle. We prove that a learning algorithm must have sample count of $\Omega(d/\log d)$ to learn a concept class of SQ-dimension d[1]. This lower bound almost matches the upper bound given in [BFJ+94] (up to a logarithmic factor), and is strong enough to imply the quadratic separation of SQ- and PAC- learning of the PARITY functions at a high noise rate. Furthermore, this lower bound applies to a wide range of concept classes and uses a different model for SQ-oracles. The proof uses several techniques, some of which could be of independent interest. As in [Y01], we still keep track of the "differences" across the scenarios with different concepts as targets. However, we use a different measure, namely the all-pair KL-divergence. This new measure can yield a very tight bound when the queries are "reasonably unbiased", i.e., when the probability that the query function returns "+1" on a random example is close to 1/2. But the KL-divergence measure becomes very bad when we have biased queries. In that case, we use an unbiased query to "simulate" a biased one. Combining these 2 techniques together allows us to prove a very tight lower bound.

The second lower bound is for *arbitrary* concept classes. We define a "correlation matrix", whose entries are the correlations of pairs of concepts. We prove that for an algorithm to have ξ advantage in learning by making q queries to an adversarial SQ-oracle with tolerance ξ, then the sum of the $(q+1)$ largest eigenvalues of the correlation matrix must be at least $s\cdot\xi^2$. Here s is the cardinality of the concept class. This result shows a continuous trade-off between the advantage an algorithm can have and the number of queries it needs to make. This trade-off can be used in designing learning algorithms, where certain requirements on the accuracy and confidence

[1]Despite its name, the total sample count is in fact a bound on the *time complexity*, rather than the *sample complexity* of the learning algorithm. The lower bound assumes that fresh samples are drawn for each new query. This might not be the most efficient way to use samples; in fact, in many situations, samples can be reused. However, the total sample count does give an bound on the times it takes to process all the samples.

are to be satisfied. This lower bound is a very general one: as we shall see in Section 4.4, this lower bound almost immediately implies some previous results [BFJ+94, Y01], and sometimes yields even stronger ones. The proof to this lower bound uses the Singular Value Decomposition (SVD). This connection could also be of independent interest.

1.5 Organization of the Rest of the Paper

We present the notations and definitions used in the paper in Section 2. We prove the two lower bounds in Section 3 and Section 4, respectively. We conclude the paper in Section 5.

2 Notations and Definitions

2.1 Mathematical Notations

All logarithms are natural log. All random distributions are over binary strings. We naturally identify a random variable with a probabilistic distribution. For a random variable X, we use $X(x)$ to denote the probability that $X = x$. For distributions X and Y, we use XY to denote their *direct product*:

$$XY(x, y) = X(x) \cdot Y(y).$$

This definition coincides with that of the *concatenation* of independent random variables X and Y.

2.2 Learning Concepts

We use Σ to denote a finite set of binary strings, and we are interested in learning *concepts* over Σ, i.e., functions that map elements in Σ to $\{-1, +1\}$. Elements in Σ are also called *examples.* A collection of concepts is often denoted by $\mathcal{F} = \{f_1, f_2, ..., f_S\}$, and is called a *concept class.* Each member in $\mathcal{F}$ is a *candidate concept*, and one of them, called a *target concept* is chosen for the learning algorithm to learn.

We fix an arbitrary order for the elements in Σ, and then identify a function over Σ with the vector represented by its truth table. Let D be a (fixed) probabilistic distribution over Σ. There exists an inner product of functions over Σ induced by D, defined as

$$\langle f, g\rangle_D = \sum_{x\in\Sigma} D(x)f(x)g(x)$$

When there is no danger of ambiguity, we omit the subscript D. The inner product of f and g is also called the *correlation* between f and g. In the case that both f and g concepts, their correlation $\langle f, g\rangle$ is the probability f and g agree on a random input minus the probability they disagree.

The *norm* of function f is defined as $||f|| = \sqrt{\langle f, f\rangle}$. Clearly all concepts have norm 1.

2.3 The Statistical Query Model

We present two different definitions of SQ-oracles.

Definition 1 (Adversarial SQ-Oracle) *A query to an* adversarial SQ-oracle *for concept f is a pair (g, τ), where $g : \Sigma\times\{-1, +1\} \to \{-1, +1\}$ is a boolean function, called the* query function, *and $\tau \in [0, 1]$ is real number, called the* tolerance. *The oracle replies with an (arbitrary) real number s such that*

$$|s - E_D[g(X, f(X))]| \le \tau$$

This definition, introduced by by Kearns [K93], is used by most literature in the SQ model.

Definition 2 (Honest SQ-Oracle) *A query to an* honest SQ-oracle *for concept f is a pair (g, M), where $g : \Sigma \times \{-1, +1\} \to \{-1, +1\}$ is a boolean function, called the* query function, *and M is a positive integer written in* unary[2]*, called the* sample count. *The oracle returns r, a random variable defined by $r = \frac{1}{M}\sum_{i=1}^{M} g(X_i, f(X_i))$ where each X_i is independently chosen according to D, and for different queries, different (and independent) examples are drawn. The* total sample count *of an algorithm is simply the sum of all the sample counts of all the queries it makes.*

This definition was introduced by Yang [Y01]. It is straightforward to show that one can use an honest SQ-oracle to simulate an adversarial oracle (with negligible error). Intuitively, the honest SQ-oracle describes how a PAC algorithm simulates an SQ algorithm more precisely than the adversarial SQ-oracle. In this sense, the honest SQ-oracle is more "realistic". Notice that this honest SQ-oracle model assumes that the oracle always draws fresh samples for each query. This might not be the most efficient way for sampling, since it is possible to collect a batch of labeled samples and then reuse them for each query, when the concept class has a low VC-dimension [FW95]. Therefore, a lower bound in the honest SQ-oracle model isn't a lower bound of the sample complexity of a learning algorithm but rather the time complexity.

3 The First Lower Bound: Uniform Concept Classes

We prove the first lower bound on learning a "uniform" concept class where every pair of concepts are "correlated in the same way", i.e., each pair has almost the same correlation. Our lower bound is against an honest SQ-oracle. With respect to such an oracle, we define the *total sample count* of an algorithm to be the sum of the sample counts of all the queries it makes. Since we require the sample counts to be written in unary, the total sample count of an algorithm is a lower bound on its running time. We shall prove a lower bound on the total sample count of the learning algorithms.

Before stating the theorem, we need a definition on the SQ-dimension of a concept class. We adopt the definition from [Y01].

Definition 3 (SQ-Dimension) *The* SQ-dimension *of a concept class $\mathcal{F}$ with respect to a distribution D and a correlation $\lambda \leq 1/2$, denoted by $\mathsf{SQ\text{-}DIM}(\mathcal{F}, D, \lambda)$, is the largest natural number d such that there exist d concepts $f_1, f_2, ..., f_d \in \mathcal{F}$ with the property that for all $i \neq j$,*

$$|\langle f_i, f_j\rangle - \lambda| \leq 1/d^3. \tag{1}$$

We often omit D when there is no danger of confusion.

Intuitively, the SQ-dimension of $\mathcal{F}$ is the size of the maximum subclass of $\mathcal{F}$ whose every pair has almost the same correlation λ.

Now we are ready to state our theorem.

Theorem 1 *Suppose $\mathcal{F}$ is a concept class of SQ-dimension $d > 200$ with respect to correlation λ. Let A be a learning algorithm that learns $\mathcal{F}$ with accuracy ϵ and confidence δ with respect to an honest SQ-oracle. If $24(\epsilon + \delta + 1/d) \leq 1 - \lambda - 1/d^3$, then the total sample count of A is at least $d/(300 \log d)$.*

Our proof strategy is similar to that in [Y01]. Roughly speaking, one considers d different "scenarios": in the j-th scenario, the concept f_j is the target. A quantity Δ, namely the "all-pair KL-divergence" is defined over these scenarios. Three lemmas are then proved:

1. The quantity Δ is initially zero, when no query is performed.
2. Each query increases Δ by a "small" amount.

[2]We require M to be written in unary to make sure that an algorithm wouldn't be able to make a query that requires too many (exponentially many) examples.

3. After all the queries are made, Δ must be “large”.

Then, from these three lemmas, one concludes that many queries are required.

Before actually proving this theorem, we present some results that are needed for the theorem.

3.1 Fourier Analysis and the SQ-dimension

The first result is from [Y01]

Lemma 1 (Fourier Analysis for the Query Function, [Y01]) *We follow the notations in Definition 3. Let $f_1, f_2, ..., f_d$ be the concepts satisfying $|\langle f_i, f_j\rangle - \lambda| \le 1/d^3$, and d be the SQ-dimension of $\mathcal{F}$. Let $g(x, y)$ be a query function. Then there exist real numbers $\beta_1, \beta_2, ..., \beta_d$, such that $\sum_{i=1}^{d} \beta_i^2 \le 1 + 100/d$ for $d > 100$ and*

$$E_D[g(X, f_i(X))] = C_g + \sqrt{1-\lambda} \cdot \beta_i \tag{2}$$

for every $i = 1, 2, ..., d$. where C_g is a constant that is independent of the target concept f_i. ■

We call C_g the *inherent bias* of g.

3.2 The KL-divergence

We present a definition of the KL-divergence, or Kullback-Leibler divergence [KL51].

Definition 4 (KL-divergence) *For 2 random variables P, Q with identical support, we define their* KL*-divergence to be* [3]

$$\mathsf{KL}(P||Q) = \sum_x P(x) \cdot \log\left(\frac{P(x)}{Q(x)}\right). \tag{3}$$

The KL-divergence is also known as the *relative entropy.* The reader is referred to [KL51, K59, B95, CT91] for a comprehensive treatise on the KL-divergence. We state several properties of the KL-divergence and postpone their proofs to Appendix A. These properties are highly unlikely to be original. We list them here only for the completeness of the paper and for the readers' convenience.

Lemma 2 (Identical Distribution) *If P and Q have identical distributions, then $KL(P||Q) = 0$.* ■

This is obvious from the definition.

Lemma 3 (Direct Product) *Let P^1, P^2 be distributions of identical support. Let Q_x^1 and Q_x^2 be distributions parametrized by x, such that for any x, Q_x^1 and Q_x^2 have identical support and $\mathsf{KL}(Q_x^1||Q_x^2) \le s$. We use $P^1 Q_{P^1}^1$ to denote the random variable forms by picking x according to P^1 following by picking y according P_x^1 and then output $x; y$. We have*

$$\mathsf{KL}(P^1 Q_{P^1}^1)||P^2 Q_{P^2}^2) \le \mathsf{KL}(P_1||P_2) + s. \tag{4}$$

Lemma 4 (Monotonicity) *Let P and Q be random variables of identical support and let ϕ be a deterministic function. Then*

$$\mathsf{KL}(\phi(P)||\phi(Q)) \le \mathsf{KL}(P||Q). \tag{5}$$

Intuitively, this lemma states that deterministic procedures cannot increase entropy.

[3] A slight difference for our definition here is that we are using the natural log, instead of log base-2. But the properties will not be affected by the change of the base.

Lemma 5 (Binomial Distribution) *Let P and Q be Bernoulli distributions with parameters p and q, i.e., $P(0) = 1 - p$, $P(1) = p$, $Q(0) = 1 - q$, and $Q(1) = q$. Let P_n and Q_n be binomial distributions with parameters (n, p) and (n, q), respectively, i.e., $P_n(k) = \binom{n}{k} p^k (1-p)^{n-k}$ and $Q_n(k) = \binom{n}{k} q^k (1-q)^{n-k}$. Then we have*

$$\mathsf{KL}(P_n || Q_n) = n \cdot \mathsf{KL}(P, Q). \tag{6}$$

Definition 5 (KL-divergence) *For n distributions $X_1, X_2, ..., X_n$ of identical support, we define their* all-pair KL-divergence *(*APKL*) to be*

$$\mathsf{APKL}(X_1, X_2, ..., X_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathsf{KL}(X_i || X_j). \tag{7}$$

3.3 Two Lemmas About the All-Pair KL-divergence

We model a learning algorithm A as a randomized Turing Machine. It makes a total of q queries, and at the end of these queries, a random input X is presented to A. Again, we assume that $f_1, f_2, ..., f_d$ are the d concepts satisfying that $|\langle f_i, f_j \rangle - \lambda| \leq 1/d^3$ for $i \neq j$. Suppose f_j is the target concept, then A must predict $f_j(X)$ with sufficient accuracy.

In the case that f_j is the target concept, we define the *state* of A after the k-th query to be the binary string S_k^j that describes the contents on A's tapes, the position of the heads, the current internal state of A. We define S_0^j to be the state of A before it starts. Notice each S_k^j is a random variable: the randomness comes from both the SQ-oracle and the random tape A uses. The j-th scenario, S^j is simply the concatenation of all the states: $S^j = (S_0^j, S_1^j, ..., S_q^j)$.

We define Δ_k to be the all-pair KL-divergence of $S_k^1, S_k^2, ..., S_k^d$:

$$\Delta_k = \mathsf{APKL}(S_k^1, S_k^2, ..., S_k^d).$$

Intuitively, Δ_k measures how "differently" A behaves with different target concepts. We shall focus on how Δ_k changes with k.

We first state and prove two lemmas about the all-pair KL-divergence. The first lemma says that the all-pair KL-divergence is initially zero, and the second one says after all the queries are made, the all-pair KL-divergence is "large".

Lemma 6 *The all-pair KL-divergence is initially zero, that is $\Delta_0 = 0$.*

Proof: This is obvious since A hasn't made any queries yet, and the state of A is independent of the target concept. Thus S_0^1, S_0^2, ..., S_0^d have identical distributions. ■

Lemma 7 *If the learning algorithm A has accuracy ϵ and confidence δ, satisfying $24(\epsilon + \delta) \leq 1 - \lambda - 1/d^3$, then after all the queries are made, we have*

$$\Delta_q \geq 0.2 d^2$$

for $d \geq 3$.

Proof: We prove a stronger statement, that after all queries are made, the KL-divergence between S_q^i and S_q^j is at least 0.23 for any $i \neq j$.

After all queries are made, the algorithm A is ready to take a random X distributed according to D and make a prediction on the label of X. However, we consider a different experiment: we give A a random X' chosen from another distribution D', which depends on i and j. We use A^i (resp. A^j) to denote the distribution of the output of A, in the case that f_i (resp. f_j) is the target concept. Notice A^i is a deterministic function of S_q^i and X', and A^j of S_q^j and X'. So by Lemma 4, we have

$$\mathsf{KL}(S_q^i || S_q^j) \geq \mathsf{KL}(A^i || A^j).$$

Now we describe the distribution D'. We partition Σ into Σ_0 and Σ_1, such that $f_i(x) = f_j(x)$ for all $x \in \Sigma_0$ and $f_i(x) \neq f_j(x)$ for all $x \in \Sigma_1$. Then we know that a randomly chosen $X \in \Sigma$ according to the distribution D has probability at least $\frac{1-\lambda-1/d^3}{2}$ to be in Σ_1, since the correlation between f_i and f_j is at most $\lambda + 1/d^3$.

We further partition Σ_1 into Σ_1^+ and Σ_1^- such that $f_i(x) = +1$ for $x \in \Sigma_1^+$ and $f_i(x) = -1$ for $x \in \Sigma_1^-$. WLOG, we assume that

$$\Pr{}_D[x \in \Sigma_1^+ | x \in \Sigma_1] \geq 1/2.$$

The distribution D' is simply the distribution D conditioned on $x \in \Sigma_1^+$.

Notice that for any element $x \in \Sigma_1^+$, we have $f_i(x) = +1$ and $f_j(x) = -1$. If A is perfect, i.e., if $\epsilon = \delta = 0$, then A^i should be the constant $+1$ and A^j should be the constant -1, and $KL(A^i||A^j) = \infty$. In the case that A is not perfect, we can still bound $KL(A^i||A^j)$ from below.

Consider the case that the concept $\hat{f}$ is chosen as a target concept. If a random $X \in \Sigma$ is drawn according to D, then with probability at least $\frac{1-\lambda-1/d^3}{4}$, it is in Σ_1^+. We say A is "lucky" if it has an accuracy of ϵ in predicting $\hat{f}(X)$ for a random $X \in \Sigma$. Then we know that A is lucky at least $1 - \delta$ of the time.

If A is lucky, then it only makes a mistake for an ϵ fraction of the inputs. So the probability A makes a mistake on a random $X' \in \Sigma_1^+$ is at most

$$\frac{\epsilon}{(1-\lambda-1/d^3)/4}$$

If A is unlucky, anything could happen. But this only happens with probability at most δ.

So the total probability A makes a mistake in predicting $\hat{f}(X')$ on a random input $X' \in \Sigma_1^+$ is at most

$$\frac{\epsilon}{(1-\lambda-1/d^3)/4} + \delta \leq \frac{4(\epsilon+\delta)}{1-\lambda-1/d^3} \leq 1/6$$

if $24(\epsilon+\delta) \leq 1-\lambda-1/d^3$. This is true for any target concept $\hat{f}$.

By union bound, we know that with probability at least $2/3$, A predicts both $f_i(X')$ and $f_j(X')$ correctly for a random $X' \in \Sigma_1^+$.

Therefore, if f_i is the target concept, then the probability A outputs a "+1" is at least $2/3$; if f_j is the target concept, then the probability A outputs a "+1" is at most $1/3$. Then we conclude that

$$KL(S_q^i||S_q^j) \geq KL(A^i||A^j) \geq \frac{2}{3}\log(\frac{2/3}{1/3}) + \frac{1}{3}\log(\frac{1/3}{2/3}) = \frac{\log 2}{3} \geq 0.23$$

for any pair $i \neq j$. So we have

$$\Delta_q \geq \sum_{i \neq j} KL(S_q^i||S_q^j) \geq 0.23d(d-1) \geq 0.2d^2$$

for $d \geq 3$. ■

3.4 Ideal Queries

We prove a third lemma about the all-pair KL-divergence, namely, that each query doesn't increase the all-pair KL-divergence much. However, technically we need to first consider a special type of queries. We call them the "ideal queries".

Definition 6 (Ideal Query Function) *We say a query function $g(x, y)$ is an* ideal query function*, if*

$$|\, E_D[g(X, f_i(X))] \,| \leq \frac{1}{2} \tag{8}$$

for $i = 1, 2, ..., d$.

Intuitively, if $g(x,y)$ is ideal, then it is not highly biased in any scenario. Ideal queries are nice since they don't increase the all-pair KL-divergence much, as shown in the following lemma.

Lemma 8 *Suppose the k-th query to an honest SQ-oracle by the algorithm is (g,M), where g is an ideal query. Then*

$$\Delta_k - \Delta_{k-1} \leq 3Md$$

for $d > 200$.

Proof: We define $P_i = g(X, f_i(X))$ to be a random variable, where X is a random element in Σ distributed according to D. We define $p_i = \mathsf{Pr}\,[P_i = +1]$. Then, according to Lemma 1, we have

$$p_i = \frac{1 + E_D[g(X, f_i(X))]}{2} = \frac{1 + C_g + \sqrt{1-\lambda}\cdot\beta_i}{2}. \tag{9}$$

We use Q_i to denote the distribution of the answer from the honest SQ-oracle when f_i is the target concept. Then obviously Q_i is the binomial distribution with parameter (M, p_i). By Lemma 5, we have

$$\mathsf{KL}(Q_i||Q_j) = M \cdot \mathsf{KL}(P_i||P_j). \tag{10}$$

We denote the all-pair KL-divergence of $Q_1, Q_2, ..., Q_d$ by ξ. Then we shall bound ξ from above.

By standard calculus, we know that for any positive real numbers x and y, there exists a real number z between x and y such that

$$\log(x) - \log(y) = \frac{x-y}{z}$$

since the derivative of $\log(x)$ is $1/x$. Now if both x and y are within $[1/4, 3/4]$, then so is z, and thus we will have

$$|\log(x) - \log(y)| = \left|\frac{x-y}{z}\right| \leq 4|x-y|.$$

Therefore, we have (notice that by the definition of ideal queries, we have $1/4 \leq p_i \leq 3/4$ for all i's):

$$\begin{aligned}
\mathsf{KL}(P_i||P_j) + \mathsf{KL}(P_j||P_i) &= p_i \log(\frac{p_i}{p_j}) + (1-p_i)\log(\frac{1-p_i}{1-p_j}) + p_j \log(\frac{p_j}{p_i}) + (1-p_j)\log(\frac{1-p_j}{1-p_i}) \\
&= (p_i - p_j)(\log p_i - \log p_j) + [(1-p_i) - (1-p_j)][\log(1-p_i) - \log(1-p_j)] \\
&\leq 4(p_i - p_j)^2 + 4[(1-p_i) - (1-p_j)]^2 \\
&= 8(p_i - p_j)^2.
\end{aligned}$$

Furthermore, by Lemma 5

$$\begin{aligned}
\xi &= \mathsf{APKL}(Q_1, Q_2, ..., Q_d) = M \cdot \sum_{i<j} (\mathsf{KL}(P_i||P_j) + \mathsf{KL}(P_j||P_i)) \\
&\leq 8M \cdot \sum_{i<j} (p_i - p_j)^2.
\end{aligned}$$

Substituting in Equation 9, we have

$$\begin{aligned}
\xi &\leq 8M \cdot \sum_{i<j} (p_i - p_j)^2 \\
&= 2M(1-\lambda) \sum_{i \neq j} (\beta_i - \beta_j)^2 \\
&= 2M(1-\lambda) \left[d \cdot \sum_i \beta_i^2 - \left(\sum_i \beta_i \right)^2 \right] \\
&\leq 2M(1-\lambda) \left[d \cdot \sum_i \beta_i^2 \right]
\end{aligned}$$

Since $\sum_i \beta_i^2 \leq 1 + 100/d$, we have

$$\xi \leq 2M(1-\lambda)(d+100) \leq 3Md \tag{11}$$

for $d > 200$.

Next, we show that

$$\Delta_k - \Delta_{k-1} \leq \xi. \tag{12}$$

The reason is that for every i, the state of A after the k-th query, S_k^i, is a deterministic function of S_{k-1}^i, Q_i, and the random bits used by A. However, the random bits used by A have the same distribution for all scenarios. Therefore we can invoke Lemma 3 and Lemma 4 and prove (12). In particular, for every pair (i,j) we can set $P^1 = S_{k-1}^i$, $P^2 = S_{k-1}^j$, and set Q_x^1 and Q_x^2 to be the replies of the SQ-oracle in S_k^i and S_k^j, where x refers to the state of A.[4] Then we know that S_k^i is a deterministic function of $P^1 Q_{P^1}^1$ and S_k^j of $P^2 Q_{P^2}^2$. Therefore, $\Delta_k - \Delta_{k-1}$ is upper bounded by all-pair KL-divergence of the Q_i's.

Now putting (11) and (12) together, we have

$$\Delta_k - \Delta_{k-1} \leq 3Md.$$

■

Now if we combine Lemma 6, Lemma 7, and Lemma 8, we can already prove an $\Sigma(d)$ lower bound on the running time of algorithms that only use ideal queries:

Lemma 9 *Suppose $\mathcal{F}$ is a concept class of SQ-dimension $d > 200$ with correlation λ. Let A be a learning algorithm that learns $\mathcal{F}$ with accuracy ϵ and confidence δ with respect to an honest SQ-oracle. If all the queries A makes are ideal queries and $24(\epsilon + \delta) \leq 1 - \lambda - 1/d^3$, then the total sample count of A is at least $d/15$.*

Proof: Suppose the total sample count of algorithm A is t. By Lemma 6, Lemma 8, and the direct product lemma, we know the all-pair KL-divergence of all the scenarios after all queries are finished is at most $3td$. Combining this fact with Lemma 7, we know that

$$3td \geq \mathsf{APKL}(S_q^1, S_q^2, \ldots S_q^d) \geq 0.2d^2$$

which implies our lemma. ■

3.5 Non-ideal Queries

Now we consider the situation where the algorithm A makes non-ideal queries.

Recall that a query function g is ideal, if $|\,E_D[g(X, f_i(X))]\,| \leq 1/2$ for all i's. For an ideal query g, the probability that $g(X, f_i(X)) = +1$ is between $1/4$ and $3/4$ for every i, which is "reasonably unbiased". In this case, the KL-divergence is a very nice measure for the "differences" between scenarios. However it doesn't work very well for very biased distributions, since with probabilities close to 0 or 1, even a tiny difference in the probabilities can result in an huge KL-divergence. Therefore we need to treat non-ideal queries differently.

We divide the non-ideal queries into two classes. For a non-ideal query function g, recall that C_g is the inherent bias of g from Lemma 1. If $|C_g| \leq 1/3$, we call it a *semi-ideal query*; otherwise we call g a *bad query*. We develop different techniques for these 2 classes of non-ideal queries.

[4]Notice that Q_x^1 and Q_x^2 may depend on x since the learning algorithm A may make adaptive queies, and in that case, the query function may depend on the state of A.

3.5.1 Semi-ideal queries

We first assume that all non-ideal queries are actually semi-ideal. For a semi-ideal query function g, if a concept f_i makes $|E_D[g(X, f_i(X))]| > 1/2$, we say f_i is an *abnormal concept for g*. Abnormal concepts are very few, since if f_i is abnormal, then

$$|\beta_i| \geq |\sqrt{1-\lambda} \cdot \beta_i| = |E_D[g(X, f_i(X))] - C_g| \geq |E_D[g(x, f_i(x))]| - |C_g| \geq \frac{1}{6}.$$

Notice by Lemma 1, we have $\sum_{i=1}^{d} \beta_i^2 \leq 1 + 100/d$. Thus there are at most 60 abnormal concepts for each semi-ideal query for $d > 200$. This is the reason we call such query concepts "semi-ideal": they behave almost like ideal queries, except for very few abnormal concepts.

Now, instead of measuring the all-pair KL-divergence of all the scenarios, we exclude the scenarios for abnormal concepts for each query. Then we measure the all-pair KL-divergence for the remaining scenarios. We obtain the following lemma:

Lemma 10 *Suppose $\mathcal{F}$ is a concept class of SQ-dimension $d > 200$ with correlation λ. Let A be a learning algorithm that learns $\mathcal{F}$ with accuracy ϵ and confidence δ with respect to an honest SQ-oracle, and all the queries A makes are semi-ideal queries. If $24(\epsilon+\delta) \leq 1 - \lambda - 1/d^3$, then the total sample count of A is at least $d/100$.*

Proof: We first consider the simple case that A is deterministic and non-adaptive, i.e., the queries A makes are fixed a priori. We assume that A makes a total of q queries. If $q \geq d/100$, we are already done. Otherwise, since we assume all the queries are semi-ideal, there are at most $60q$ candidate concepts that are abnormal for some of the queries. We exclude all these abnormal concepts, and there are at least $d - 60q \geq 2d/5$ remaining concepts. They correspond to at least $2d/5$ scenarios. We measure the all-pair KL-divergence of these remaining scenarios, and all the analysis for ideal queries work here. So the total sample count is at least $(1/15) \cdot 2d/5 \geq d/100$.

Next, we consider the case that the A is probabilistic and adaptive. In this case, there can be many potentially possible queries, and we cannot eliminate all the abnormal concepts for each query. However, we may still assume A always makes exactly q queries during its execution. Again, if $q \geq d/100$, the lemma is already proved. Otherwise, we modify the algorithm A slightly so that after making all the queries, A writes down the names of all the abnormal concepts to the queries A have made[5]. We denote this output by AB. Notice that AB is a random variable that consists of at most $60q$ concepts. Let R denote the number of the concepts that are not abnormal for any of the queries. We have $R \geq d - 60q \geq 2d/5$. We also require A to write down the all-pair KL-divergence of the scenarios corresponding to all concepts excluding the abnormal ones, and denote this by $\mathsf{KL_{REM}}$. Then $\mathsf{KL_{REM}}$ is also a random variable. Lemma 6 and Lemma 8 still work for $\mathsf{KL_{REM}}$, and we have

$$\Pr[\mathsf{KL_{REM}} \leq 3t \cdot R] = 1. \tag{13}$$

where t is the total sample count of A. Next we bound $\mathsf{KL_{REM}}$ from below. Overall, A has an accuracy ϵ and confidence δ. So there exists an s such that conditioned on $\mathsf{AB} = s$, A still has an accuracy ϵ and confidence δ. By Lemma 7, we have

$$\Pr[\mathsf{KL_{REM}} \geq 0.2R^2 \mid \mathsf{AB} = s] = 1. \tag{14}$$

By putting (13) and (14) together, we have

$$t \geq \frac{0.2R^2}{3R} = R/15 \geq d/100. \tag{15}$$

■

[5]Notice that A knows the queries it makes and all the candidate concepts. So A can compute the distributions of the replies for all the scenarios, and decide which candidate concepts are abnormal. This modification might not preserve the efficiency of A, but this is not a concern here.

3.5.2 Bad Queries

Now we extend the lower bound to algorithms that make bad queries as well. Recall that a query function g is bad, if $|C_g| > 1/3$. For a bad g, the probability that $g(X, f_i(X)) = +1$ can be very biased for a lot of target concepts f_i. The KL-divergence won't work very well under this circumstance.

However, intuitively, a highly biased query won't be very efficient since a biased query carries less information than an unbiased one. We shall prove this intuition is correct to some extend. In fact, we show how to efficiently simulate a biased query using an unbiased one.

First, we define a variance of the honest SQ-oracle, namely the semi-honest SQ-oracles.

Definition 7 (Semi-honest SQ-oracle) *A query to a* semi-honest SQ-oracle *for concept f is a pair (g, M), where $g : \Sigma \times \{-1, +1\} \to \{-1, +1\}$ is a boolean function, called the* query function*, and M is a positive integer written in* unary*, called the* sample count*. The oracle outputs a pair (r', M'). Here M' is an integral random variable satisfying $\Pr[M' < M] = 1$ and $r' = \frac{1}{M'}\sum_{i=1}^{M'} g(X_i, f(X_i))$, where each X_i is a random variable independently chosen according to D. We call M' the* actual sample count.

If the distribution of the actual sample count is independent of the choice of the concept f, we say this oracle is oblivious.

Intuitively, a semi-honest SQ-oracle behaves almost like an honest SQ-oracle, except that it is allowed to decrease the sample count. However the oracle must report the actual sample count it uses. Obviously an honest SQ-oracle is also a semi-honest SQ-oracle.

We notice that the lower bounds for honest SQ-oracles also works for oblivious semi-honest SQ-oracles. Lemma 6 and Lemma 7 are unaffected since they are not concerned about oracles. Lemma 8 is still true, thanks to the obliviousness. The only difference an oblivious semi-honest SQ-oracle makes is from the query functions, not the actual sample count, and decreasing the sample count will not increase the all-pair KL-divergence. There are different instantiations of semi-honest SQ-oracles since the oracle has the freedom to choose the distribution of M'. However, Lemma 10 is true for *any* instantiation of an oblivious semi-honest SQ-oracle.

Now we are ready to prove Theorem 1.

Proof: (of Theorem 1)

Given a learning algorithm A, we shall construct a new algorithm A' that has almost the same accuracy, confidence, and comparable efficiency as A. Furthermore, A' only makes semi-ideal queries. So Lemma 10 can be applied to A', which in turn gives a lower bound for the running time of A.

We first extend the domain Σ by one bit: we define $\tilde{\Sigma} = \Sigma \times \{-1, +1\}$. We write each element in $\tilde{\Sigma}$ as (x, e), where $x \in \Sigma$, while $e \in \{-1, +1\}$ is the extended bit.

We extend all the concepts to the new domain by defining new concepts $\tilde{f}_1, ..., \tilde{f}_d$:

$$\tilde{f}_i(x, +1) = \tilde{f}_i(x, -1) = f_i(x).$$

We also extend the distribution D such that it is the uniform distribution over the extended bit.

$$\tilde{D}(x, -1) = \tilde{D}(x, +1) = D(x)/2.$$

The new algorithm A' works with the extended concept class $\tilde{\mathcal{F}} = \{\tilde{f}_1, \tilde{f}_2, ..., \tilde{f}_d\}$. It is easy to verify that the problem of learning $\tilde{\mathcal{F}}$ is exactly the original problem of learning $\mathcal{F}$, since one can simply ignore the extended bit. Obviously, the SQ-dimension of $\tilde{\mathcal{F}}$ is d as well.

The new algorithm A' is constructed as follows: A' behaves exactly like A except for the queries. For every query A makes to the honest SQ-oracle, A' simulates it by querying a semi-honest SQ-oracle.

Suppose A makes a query (g, M) and the reply is r. We show how A' replaces it by making a query (g', M^*) and estimates r, where $M^* = 2M \cdot \lceil \log d \rceil + 1$. We shall describe the query function g', the semi-honest oracle A' uses, and how A' estimates r.

- **The query function g'**
  The new query concept g' works in the extended domain $\tilde{\Sigma} \times \{-1, +1\}$, and we write it as $g'(x, e, y)$, where $x \in \Sigma$ and e is the extended bit. It is defined as follows:
  - If g is semi-ideal, then
  $$g'(x, -1, y) = g(x, y) \text{ and } g'(x, +1, y) = g(x, y);$$
  - If g is bad and $C_g > 1/3$, then
  $$g'(x, -1, y) = g(x, y) \text{ and } g'(x, +1, y) = -1;$$
  - If g is bad and $C_g < -1/3$, then
  $$g'(x, -1, y) = g(x, y) \text{ and } g'(x, +1, y) = +1.$$

  Roughly speaking, g' behaves exactly like g on half of its inputs (when the extended bit e is -1), and on the other half, it is a constant, creating an "artificial bias".

  Lemma 11 *The query function g' is semi-ideal.*

  Proof: Let's compute the inherent bias $C_{g'}$ of g', per Lemma 1.
  First we consider the case that g is already semi-ideal:
  $$\begin{aligned} E_{\tilde{D}}[g'(x, e, \tilde{f}_i(x, e))] &= \frac{1}{2}\left(E_D[g'(x, -1, \tilde{f}_i(x, -1))] + E_D[g'(x, +1, \tilde{f}_i(x, +1))]\right) \\ &= C_g + \sqrt{1-\lambda} \cdot \beta_i \end{aligned}$$

  So $C_{g'} = C_g \in [-1/3, 1/3]$, and g' is semi-ideal. Similarly, when g is bad and $C_g > 1/3$, $C_{g'} = (C_g - 1)/2 \in [-1/3, 0]$; when g is bad and $C_g < -1/3$, $C_{g'} = (C_g + 1)/2 \in [0, 1/3]$. Thus g' is always semi-ideal. ■

- **The semi-honest SQ-oracle**
  We describe the semi-honest SQ-oracle A' uses. Intuitively, g' is like g with added "artificial bias". By adding this bias, g' becomes semi-ideal. But g' also becomes less efficient than g, since about half of its inputs are not "useful". Therefore we need to increase the sample count to make sure g' gets enough "useful" inputs. The actual construction is a bit complicated.
  On a query (g', M^*), the oracle repeatedly draws independent examples (x_i, e_i) according to $\tilde{D}$. An example (x_i, e_i) is a *useful example*, if $e_i = -1$; otherwise it is *useless*. The oracle keeps drawing random examples until it has exactly M useful examples, or it has sampled M^* times. Finally the oracle uses all its examples (both the useful ones and useless ones) to compute the average of g'.
  We comment that this semi-honest SQ-oracle is indeed oblivious. In fact, the actual sample count is a random variable that only depends on the probability that a random example (x_i, e_i) is a useful one. According to the distribution $\tilde{D}$, this probability is $1/2$, independent of the target concepts.

- **Estimating r**
  Suppose the reply from the semi-honest SQ-oracle is (r', M'). We show how A' computes a value s, and uses s to simulate r, the reply from the honest SQ-oracle. The value s is defined as follows:
  $$s = \begin{cases} \frac{1}{M} M' \cdot r' & \text{if } g \text{ is semi-ideal;} \\ \frac{1}{M}(M' \cdot r' - M + M') & \text{if } g \text{ is bad and } C_g > 1/3; \\ \frac{1}{M}(M' \cdot r' + M - M') & \text{if } g \text{ is bad and } C_g < -1/3. \end{cases}$$

We prove that s is very close to r.

Lemma 12 *The statistical distance between s and r is at most M/d^2.*

Proof: First, we prove that in the case $M' < M^*$, s and r have identical distributions. Notice if $M' < M^*$, then the semi-honest SQ-oracle has exactly M useful examples and $M' - M$ useless ones. It is easy to verify that in this case, s is the average of the M useful examples (since g' outputs constants on the useless examples, these constants can be easily removed from r'). On useful examples, g' behaves identically as g. So s has identical distribution as r in this case.

Therefore, the statistical distance between s and r is at most the probability that $M = M^*$. But $M = M^*$ means that the first $2M \cdot \lceil \log d \rceil$ examples contain less than M useful ones. We group these examples into M groups, each containing $2\lceil \log d \rceil$ examples. For any group, the probability that it doesn't contain any useful example is at most $2^{-2\lceil \log d \rceil} < 1/d^2$, since a random example is useful with probability $1/2$. So the probability that one of the groups doesn't contain any useful example is at most $M \cdot 1/d^2$. But if each group contains at least one useful example, we have at least M useful examples. Therefore, the probability that there are less than M useful examples is at most M/d^2. ■

Now, putting everything together: we have constructed an algorithm A' that simulates A using only semi-ideal queries. For each query (g, M), A' can simulate it up to an error of M/d^2. Let t be the total sample count of A. If $t \geq d$, the theorem is already true. Otherwise the total error A' makes in simulating A is at most $t/d^2 < 1/d$, and thus A' has an accuracy ϵ and confidence $\delta + 1/d$. By Lemma 10, the total sample count of A' is at least $d/60$. Since A' replaces a query of sample count M by a query of sample count $2M \cdot \lceil \log d \rceil + 1 \leq 3M \log d$, the total sample count of A is at least $d/100/(3 \log d) = d/(300 \log d)$. ■

3.5.3 Remarks

Our result isn't directly comparable with that in [J00]. The lower bound in [J00] is for a slightly different SQ model (the "SQ-based PAC algorithms"), and it doesn't direct translate to a lower bound in our model. Furthermore, Jackson proved that lower bound only for learning PARITY functions, while our lower bound holds for a broader class, namely, the class of "uniformly correlated" concepts. On the other hand, his lower bound is tighter: for PARITY functions over n bits, he proved that an $\Omega(2^n)$ running time is needed, while our result only gives $\Omega(2^n/n)$. However, our lower bound is strong enough to separate the SQ-learning from the PAC-learning of PARITY functions with a high $(1/2 - 1/\mathsf{poly}(n))$ noise rate, as in [J00].

It is also interesting to compare our result to that in [Y01]. We improved the lower bound from $\Omega(\sqrt{d})$ to $\Omega(d/\log d)$. Nevertheless, the result in [Y01] is a "continuous" lower bound in that it shows a continuous trade-off between the number of queries needed and the advantage a learning algorithm has. Our result, on the other hand, is a single lower bound for the running time if a certain accuracy and confidence is needed.

4 The Second Lower Bound: Arbitrary Concept Classes

The second lower bound is for classes of concepts that are arbitrarily correlated. This lower bound is against an adversarial SQ-oracle.

4.1 Notations and Definitions

We introduce some notations and definitions to be used in this section, most of which are also used in [Y01].

For a learning algorithm A, we denote its *advantage* in learning a target concept f_i to be the probability it makes a correct prediction on a random example minus the probability that it makes an incorrect prediction. The probability is taken over the randomness from A and the random example. The advantage is also known as "correlation" in literature.

Notice that after making all the queries, the algorithm is ready to make a prediction on any example x. We can describe the state of this algorithm by its *characteristic function*, which is defined to be a real-valued function over the same domain. $\psi_A : \Sigma \to [-1, +1]$, such that $\psi_A(x) = 2 \cdot \mathsf{Pr}\,[A \text{ outputs } +1 \text{ on } x] - 1$, where the probability is taken over the randomness A uses and the randomness from the oracles.

It is not hard to see that if the target concept is $\hat{f}$, then the advantage of A is $\langle \hat{f}, \psi_A \rangle$.

Since we are dealing with a class of concepts that are arbitrarily correlated, we need a means to describe the "correlations" among the concepts.

Definition 8 (Correlation Matrix) *Let $\mathcal{F} = \{f_1, f_2, ..., f_s\}$ be a concept class. Its* correlation matrix*, denoted by $\mathcal{C}^{\mathcal{F}}$ is a $s \times s$ matrix such that $\mathcal{C}^{\mathcal{F}}_{ij} = \langle f_i, f_j \rangle$.*

Clearly $\mathcal{C}^{\mathcal{F}}$ is a semi-positive definite matrix, and it has 1's on the diagonal. When there is no danger of confusion, we omit the superscript $\mathcal{F}$.

Now we are ready to state the theorem.

Theorem 2 *Suppose $\mathcal{F}$ is a concept class of cardinality s with correlation matrix $\mathcal{C}$. Let $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_d$ be the eigenvalues of $\mathcal{C}$. Let A be a learning algorithm that learns $\mathcal{F}$ with advantage at least ξ using any adversarial SQ-oracle of error tolerance τ. If A makes q queries, then we have*

$$\sum_{i=1}^{q+1} \lambda_i \geq s \cdot \min\{\xi^2, \tau^2\}. \tag{16}$$

4.2 Why the Previous Technique Does not Work

Before we actually prove the theorem, we first provide some intuition why previous technique doesn't work here.

The technique of the all-pair KL-divergence from the previous section doesn't work very well here. Here is an intuitive argument. In the case that each pair of concepts has the same correlation, it suffices to upper bound how much "difference" one single query makes. In a sense, no query function is significantly more efficient than others, since the concepts are sort of "uniform". In the case that the concepts are arbitrarily correlated, it is quite possible that there exists some query function that makes a large difference while others don't. In other words, we no longer have the uniformity among the concepts and some queries can be much more efficient than others. So a single upper bound on one query doesn't necessarily scale well to a good upper bound on a sequence of queries.

Here is an example: let Σ be the set of all n-bit strings, and D be the uniform distribution over Σ. Let $\mathcal{F}_0$ be the family of 2^n functions that are PARITY functions if the Least Significant Bit (LSB) of the input is 0 and are constant $+1$ if the LSB is 1. Let $\mathcal{F}_1$ be the family of 2^n functions that are PARITY functions if the LSB of the input is 0 and are constant -1 if the LSB is 1. Let the concept class $\mathcal{F}$ be the union of $\mathcal{F}_0$ and $\mathcal{F}_1$.

It is very easy to tell concepts in $\mathcal{F}_0$ and concepts in $\mathcal{F}_1$ apart, since they have difference biases. Therefore, there exists a very efficient query (just ask for the bias of the target concept) that can learn one bit about the target concept, and there exists a learning algorithm that makes a single query and learns the target concept with accuracy 3/4. If every query were as efficient as this one, then only $O(n)$ queries would have been needed to learn the target function exactly. This is also the best lower bound we can hope for if we use the technique from the previous section.

However, any SQ algorithm needs exponential time to learn the target concept with accuracy higher than 7/8. This is true since in having an accuracy higher than 7/8 implies learning the unique target function, which implies learning a parity function [J00]. This sharp contrast is due to the fact that although the query that distinguishes $\mathcal{F}_0$ from $\mathcal{F}_1$ is very efficient, others are not.

4.3 Proof of Theorem 2

We prove the Theorem 2 here. But we first state a lemma that is very similar to Lemma 1.

Lemma 13 *For every query function $g(x, y)$, there exists a vector χ_g and a constant C_g such that $||\chi_g|| \leq 1$ and*

$$E_D[g(x, f_i(x))] = C_g + \langle \chi_g, f_i \rangle \tag{17}$$

for every concept f_i.

We call the vector χ_g the *characteristic vector* of g.

Proof: We essentially follow the same outline as in [BFJ+94]. Let $\mathcal{L}$ be the subspace spanned by the concepts $f_1, f_2, ..., f_s$. Let $\{u_1, u_2, ..., u_s\}$ be an orthonormal basis in $\mathcal{L}$. We regard each u_i as a function from Σ to reals. Next we extend these functions to over the domain $\Sigma \times \{-1, +1\}$ by defining

$$\begin{aligned} \tilde{u}_i(x, y) &= u_i(x); \\ \tilde{v}_i(x, y) &= u_i(x) \cdot y. \end{aligned}$$

We also extend the distribution D to over $\Sigma \times \{-1, +1\}$ by defining

$$\tilde{D}(x, -1) = \tilde{D}(x, +1) = D(x).$$

It is easy to check that $\{\tilde{u}_i, \tilde{v}_i\}_i$ forms an orthonormal basis for functions over $\Sigma \times \{-1, +1\}$. Then we decompose the query function $g(x, y)$ as

$$g(x, y) = \sum_j \alpha_j \tilde{u}_i(x, y) + \sum_j \beta_j \tilde{v}_j(x, y).$$

Since $||g|| = 1$, we have $\sum_j \alpha_j^2 + \sum_j \beta_j^2 = 1$.
Then

$$\begin{aligned} E_D[g(X, f_i(X))] &= \sum_j \alpha_j E_D[\tilde{u}_j(X, f_i(X))] + \sum_j \beta_j E_D[\tilde{v}_j(X, f_i(X))] \\ &= \sum_j \alpha_j E_D[u_j(X)] + E_D[\sum_j \beta_j u_j(X) \cdot f_i(X)] \end{aligned}$$

We set $C_g = \sum_j \alpha_j E_D[u_j(X)]$ and $\chi_g(x) = \sum_j \beta_j u_j(x)$, and then we have

$$E_D[g(x, f_i(x))] = C_g + \langle \chi_g, f_i \rangle.$$

Next we prove that $||\chi_g|| \leq 1$. Since u_j's are orthonormal, we have

$$||\chi_g||^2 = \sum_j \beta_j^2 ||u_j||^2 = \sum_j \beta_j^2 \leq 1.$$

■

Now we are ready for proving the theorem.

Proof: (of Theorem 2)

Consider a particular adversarial SQ-oracle that doesn't commit to any target concept. On each query, this oracle returns a special value that is consistent with a large number of concepts. As a result, a small number of "inconsistent" concepts will be "eliminated". So long as there are many concepts remaining (not eliminated), the learning algorithm won't be able to decide which remaining concept is the true target concept. Therefore the learning algorithm will not learn the target concept with high accuracy and high confidence.

We consider the oracle that on a query (g, τ) returns C_g as defined in (17). This result would be valid for all concepts that have inner products at least τ with χ_g. All other concepts will be eliminated. After all the queries are made, we consider the remaining concepts that haven't been eliminated yet. We claim that each of them has an inner product of at least ξ with ψ_A, the characteristic function of A. This is because each of the remaining concept can be the target concept since it is consistent with all the queries. Because A has advantage ξ, the inner product of the remaining concepts and ψ_A must be at least ξ.

We denote the q queries made by A by $g_1, g_2, ..., g_q$, and denote their corresponding characteristic vector by $v_1, v_2, ..., v_q$. We define the vector v_{q+1} to be ψ_A. Then we have a total of $q+1$ vectors, $v_1, v_2, ..., v_{q+1}$, all of norm at most 1. For any concept f, if it is eliminated in one of the queries, then it has an inner product of at least ξ with one of the vectors $v_1.v_2, ..., v_q$. If f is not eliminated, then it is a possible target concept and thus has an inner product of at least ξ with the vector v_{q+1}. Therefore, all concepts have inner products of at least $\min\{\xi, \tau\}$ with one of the vectors in $v_1, v_2, ..., v_{q+1}$.

Let Q be the subspace spanned by the vectors $v_1, v_2, ..., v_{q+1}$. Then every candidate concept f, when projected to Q, has a length of at least ξ since it has an inner product of at least $\min\{\xi, \tau\}$ with a unit vector in Q. We use f^Q to denote the projection of f into subspace Q. And we define S to be

$$S = \sum_{f \in \mathcal{F}} ||f^Q||^2.$$

Obviously, we have $S \geq s \cdot \min\{\xi^2, \tau^2\}$.

On the other hand, since Q is a subspace of dimension at most $(q+1)$, the maximum possible value of S is bounded by the sum of the $(q+1)$ largest eigenvalues of $\mathcal{C}$. This problem of approximating vectors in high dimension by vectors in lower dimensions is can be solved by the Singular Value Decomposition (SVD), sometimes also known as the Principal Component Analysis (PCA). The readers are referred to [S88, CC80, B95] for comprehensive discussions.

Therefore, we have

$$\sum_{i=1}^{q+1} \lambda_i \geq S \geq s \cdot \min\{\xi^2, \tau^2\}$$

■

Immediately from the theorem, we see a continuous trade-off between the advantage an algorithm can achieve and the number of queries it needs to make. This trade-off can be very non-linear: it completely depends on how the eigenvalues of the correlation matrix are distributed. If all eigenvalues are similar, then the advantage of an algorithm will increase almost linearly with the number of queries. If some eigenvalues are much larger than others, then an algorithm might gain a lot of advantage in the first few queries. But after that, it has to make a lot of queries to make a tiny progress — interested readers can verify that this is exactly the case for the example given in Section 4.2.

4.4 Applications

We discuss several applications of Theorem 2 by proving two corollaries.

Corollary 1 *Let $\mathcal{F} = \{f_1, f_2, ..., f_d\}$ be a concept class such that for every pair $i \neq j$, $|\langle f_i, f_j \rangle| \leq 1/d$. Any learning algorithm that makes q queries of tolerance $1/d^{1/3}$ to an adversarial SQ-oracle and has an advantage of $1/d^{1/3}$ in learning $\mathcal{F}$ satisfies that*

$$q \geq d^{1/3}/2 - 1.$$

Proof: The correlation matrix $\mathcal{C}$ is close to the identity matrix I. Actually each entry of $\mathcal{C} - I$ is bounded by $1/d$. By the Geršgorin Theorem [HJ85], all eigenvalues of $\mathcal{C} - I$ are bounded by 1. So all eigenvalues of $\mathcal{C}$ are bounded by 2. So (16) becomes $2(q+1) \geq d \cdot \xi^2$. Setting $\xi = 1/d^{1/3}$ completes the proof. ■

Comparing this corollary to the negative result in [BFJ+94], we used a weaker condition on the SQ-dimension (that $|\langle f_i, f_j \rangle| \leq 1/d$ instead of $1/d^3$) to achieve asymptotically identical bounds, and the proof is simple.

Corollary 2 *Let $\mathcal{F} = \{f_1, f_2, ..., f_d\}$ be a concept class such that for every pair $i \neq j$, $|\langle f_i, f_j \rangle - \lambda| \leq 1/d^2$, where λ is a constant. If a learning algorithm doesn't make any queries, it has an advantage of at most $\sqrt{\frac{1+(d-1)\lambda+1/d}{d}}$ in approximating all concepts in $\mathcal{F}$. To achieve an advantage ϵ with tolerance ϵ, an algorithm needs to make $d(\epsilon^2 - \lambda) - 1$ queries to an adversarial SQ-oracle.*

Proof: We first cite a result that is a corollary to the Geršgorin Theorem[HJ85].

Lemma 14 ([HJ85, Corollary 6.3.4, p.p. 367]) *Let $A \in M_n$ be a normal matrix and let $E \in M_n$. If $\hat{\lambda}$ is an eigenvalue of $A + E$, then there is some eigenvalue λ_i of A for which $|\hat{\lambda} - \lambda_i| \leq |||E|||_2$.* ■

The correlation matrix $\mathcal{C}$ is close to the matrix M that has 1's on the diagonals and λ on the off-diagonal entries. Let $E = \mathcal{C} - M$, and then each entry of E is bounded by $1/d$, and thus we have $|||E|||_2 \leq 1/d$. Obviously M is a norm matrix. The largest eigenvalue of M is $\lambda_1 = 1 + (d-1)\lambda$ and all the rests are $1 - \lambda$.

Let $\lambda_1 \geq \cdots \geq \lambda_n$ be the eigenvalues of $\mathcal{C}$. By Lemma 14, we know that $\lambda_1 \leq 1+(d-1)\lambda+1/d$, and $\lambda_i \leq 1 - \lambda + 1/d$ for $i = 2, ..., d$.

So if an algorithm doesn't make any queries, then its maximum advantage is bounded by

$$\sqrt{\frac{\lambda_1}{d}} = \sqrt{\frac{1 + (d-1)\lambda + 1/d}{d}}.$$

Suppose q queries are made, then we have

$$d \cdot \epsilon^2 \leq 1 + (d-1)\lambda + 1/d + (1 - \lambda + 1/d)q,$$

or

$$q \geq \frac{(\epsilon^2 - \lambda) \cdot d - 1 + \lambda - 1/d}{1 - \lambda + 1/d} \geq d(\epsilon^2 - \lambda) - 1.$$

■

This result is similar to that in [Y01]. Particularly, in the case that no query is made, the two are almost identical. However, in general they are not directly comparable. In [Y01], a more restrictive condition is used: one requires that $|\langle f_i, f_j \rangle - \lambda| \leq 1/d^3$ instead of $1/d^2$. On the other hand, their result is against an honest SQ-oracle, which is stronger than one against an adversarial SQ-oracle. Nevertheless, the Theorem 2 makes our proof very simple.

5 Conclusion and Future Work

We proved two lower bounds for SQ-learning algorithms. The first lower bound is for a uniformly correlated concept class and works against the honest SQ-oracle. This lower bound is almost tight up to a logarithmic factor, and is strong enough to imply the separation of SQ- and PAC- learning of noisy PARITY functions with a high noise rate. This lower bound improves previous results by both extending the range of concept classes and using a stronger SQ-oracle model. The second lower bound is for any concept classes and against an adversarial SQ-oracle. This lower bound applies to a much wider range of concept classes than previous results. It also shows a continuous trade-off between the advantage an algorithm has and the number of queries it needs to make. This trade-off could be useful in designing learning algorithms. As demonstrated in Section 4.4, this lower bound almost immediately implies some previous results in the literature, and sometimes yields even stronger ones.

Some techniques used in the proofs may be of independent interest. In proving the first lower bound, the "all-pair KL-divergence" is introduced, which plays a very important role in the proof. We also used a technique to simulate biased queries ("bad" queries) using unbiased ones ("semi-ideal" queries). In proving the second lower bound, a connection to the Singular Value Decomposition (SVD) was made.

There are still many open problems remaining.

- **Tighter Lower Bounds.**
  The first lower bound we proved in this paper is $\Omega(d/\log d)$, which seems still a logarithmic factor short of being tight. It would be interesting to have a truly tight lower bound. A tight lower bound would directly imply that the "SQ-based PAC algorithm" model is essentially the same as the "honest SQ-oracle" model, at least for certain concept classes. Also a better lower bound translates to a better separation of the SQ model from the PAC model. We conjecture that an $\Omega(d)$ lower bound exists.
  We don't know if the second lower bound we proved is tight either. Actually we conjecture that it is not. However, new techniques might be needed to prove a tighter bound.
- **More General SQ-oracle Model.**
  The second lower bound is only for the adversarial SQ-oracle model. Can we prove a similar result for the honest SQ-oracle model?
- **Complete Characterization of SQ-leaning.**
  The SQ-dimension as defined by Blum et. al. [BFJ+94] characterizes the *weak learnability* of a concept class. If the SQ-dimension is high, then the class is not weakly learnable, and if the dimension is low, then the class is (non-uniformly) weakly learnable. However, the situation is less clear for strong learning. There doesn't exist a quantity that completely characterizes strong learnability. The second lower bound in this paper suggests that the eigenvalues of the correlation matrix might be a useful quantity. In this paper we proved that these eigenvalues lead to a lower bound. If we can prove a matching upper bound using the eigenvalues, then we would have a much better understanding of general SQ learning (rather than only for weak learning).

Acknowledgment

Jeff Jackson initiated the (still on-going) quest for a tight lower bound on SQ models, and remains a main driving force behind all the work that lead to the first lower bound in this paper. It is he who first suggested that the KL-divergence could be useful to look at. Many stimulating discussions with him greatly helped the author understand the (quite subtle) differences between various models of SQ-oracle. He proved to be a wonderful error corrector, too.

The author thanks Avrim Blum for his many brilliant suggestions, and in particular, the suggestions of looking into eigenvalues for useful bounds in the arbitrary concept class case. This suggestion contributed significantly to the second lower bound.

The author thanks Ioannis Koutis, Francisco Pereira and Xiaojin Jerry Zhu for useful discussions and comments and Bartosz Przydatek for careful proofreading.

References

[AS95] Javed A. Aslam and Scott E. Decatur, *Specification and Simulation of Learning Algorithms for Efficiency and Noise-Tolerance*, In COLT 1995, pages 437-446. ACM Press, July 1995.

[B95] Christopher Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[BFJ+94] Avrim Blum, Merrick Furst, Jeffrey Jackson, Michael Kearns, Yishay Mansour, and Steven Rudich. *Weakly Learning DNF and Characterizing Statistical Query Learning Using Fourier Analysis.* STOC 1994, pages 253–262, 1994.

[CC80] Christopher Chatfield and Alexander Collins, *Introduction to Multivariate Analysis*, Chapman and Hall, 1980.

[CT91] Thomas Cover and Joy Thomas, *Elements of Information Theory*, John Wiley and Sons, Inc., 1991.

[FW95] Sally Floyd and Manfred Warmuth, *Sample Compression, Learnability, and the Vapnik-Chervonenkis Dimension*, in *Machine Learning*, Vol.21 (3), pp. 269–304, December 1995.

[HJ85] Roger Horn and Charles Johnson, *Matrix Analysis*, Cambridge University Press, 1985.

[K59] S. Kullback, *Information Theory and Statistics*, New York: Dover Publications, 1959.

[KL51] S. Kullback and R. A. Leibler, *On Information and Sufficiency*, Annals of Mathematical Statistics **22**, pp. 79-86, 1951.

[J00] Jeff Jackson *On the Efficiency of Noise-Tolerant PAC Algorithms Derived from Statistical Queries.* In *Proceedings of the 13th Annual Workshop on Computational Learning Theory*, 2000.

[K93] Michael Kearns. *Efficient noise-tolerant learning from statistical queries.* In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pp. 392–401, 1993.

[S88] Gilbert Strang, *Linear Algebra and Its Applications*, third Edition, Harcourt Bruce Jovanovich Inc., 1988.

[V84] Leslie Valiant, *A theory of the Learnable.* In *Communications of the ACM*, 27(11): 1134–1142, November 1984.

[Y01] Ke Yang, *On Learning Correlated Boolean Concepts Using Statistical Query*, In the Proceedings of the 12th International Conference on Algorithmic Learning Theory (ALT'01), LNAI 2225, pp. 59-76, 2001.

[Y02] Ke Yang, *New Lower Bounds for Statistical Query Learning*, in the proceeding of the 15th Annual Conference on Computational Learning Theory, COLT 2002, Sydney, Australia, July 8-10, LNCS 2375, pp. 229-243, 2002.

A Proofs to the properties of the KL-divergence

We list several proofs to the properties of the KL-divergence stated in Section 3.2.

Proof: (to Lemma 3)
Simple computation:

$$KL(P^1 Q^1_{P^1})||P^2 Q^2_{P^2}) \quad = \quad \sum_x \sum_y P^1(x) Q^1_x(y) \log\left(\frac{P^1(x)Q^1_x(y)}{P^2(x)Q^2_x(y)}\right)$$

$$\begin{aligned}
&= \sum_x \sum_y P^1(x) Q^1_x(y) \left[\log\left(\frac{P^1(x)}{P^2(x)}\right) + \log\left(\frac{Q^1_x(y)}{Q^2_x(y)}\right)\right] \\
&= \sum_x \sum_y P^1(x) Q^1_x(y) \log\left(\frac{P^1(x)}{P^2(x)}\right) + \sum_x \sum_y P^1(x) Q^1_x(y) \log\left(\frac{Q^1_x(y)}{Q^2_x(y)}\right) \\
&= \mathsf{KL}(P^1 || P^2) + \sum_x P^1(x) \mathsf{KL}(Q^1_x || Q^2_x) \\
&\leq \mathsf{KL}(P^1 || P^2) + s
\end{aligned}$$

■

Proof: (to Lemma 4)

We first prove a very simple inequality:

$$(p_1 + p_2) \log\left(\frac{p_1 + p_2}{q_1 + q_2}\right) \leq p_1 \log\left(\frac{p_1}{q_1}\right) + p_2 \log\left(\frac{p_2}{q_2}\right) \tag{18}$$

The proof to this inequality is simple: since $\log(x)$ is a concave function, we have

$$\frac{p_1}{p_1 + p_2} \cdot \log\left(\frac{q_1}{p_1}\right) + \frac{p_2}{p_1 + p_2} \cdot \log\left(\frac{q_2}{p_2}\right) \leq \log\left(\frac{p_1}{p_1 + p_2} \cdot \frac{q_1}{p_1} + \frac{p_2}{p_1 + p_2} \cdot \frac{q_2}{p_2}\right) = \log\left(\frac{q_1 + q_2}{p_1 + p_2}\right)$$

or

$$p_1 \log\left(\frac{p_1}{q_1}\right) + p_2 \log\left(\frac{p_2}{q_2}\right) \geq (p_1 + p_2) \log\left(\frac{p_1 + p_2}{q_1 + q_2}\right)$$

In general, we have

$$\left(\sum_i p_i\right) \cdot \log\left(\frac{\sum_i p_i}{\sum_i q_i}\right) \leq \sum_i p_i \cdot \log\left(\frac{p_i}{q_i}\right) \tag{19}$$

Intuitively, this inequality implies that when we combine several probabilistic events into one, the KL-divergence will not increase. Now consider the distribution of $\phi(P)$ and $\phi(Q)$. Since ϕ is a deterministic function, it essentially combines several events into "larger" events. Therefore, intuitively, ϕ shouldn't increase the KL-divergence. Below is a more formal proof:

$$\begin{aligned}
\mathsf{KL}(\phi(P) || \phi(Q)) &= \sum_y \mathsf{Pr}\left[\phi(P) = y\right] \cdot \log\left(\frac{\mathsf{Pr}\left[\phi(P) = y\right]}{\mathsf{Pr}\left[\phi(Q) = y\right]}\right) \\
&= \sum_y \left(\sum_{\phi(x) = y} P(x)\right) \cdot \log\left(\frac{\sum_{\phi(x)=y} P(x)}{\sum_{\phi(x)=y} Q(x)}\right) \\
&\leq \sum_x P(x) \log\left(\frac{P(x)}{Q(x)}\right) \\
&= \mathsf{KL}(P || Q)
\end{aligned}$$

■

Proof: (to Lemma 5)

Simple computation:

$$\begin{aligned}
\mathsf{KL}(P_n || Q_n) &= \sum_{k=0}^{n} P(k) \log\left(\frac{P(k)}{Q(k)}\right) \\
&= \sum_{k=0}^{n} \binom{n}{k} p^k (1-p)^{n-k} \cdot \log\left(\frac{p^k (1-p)^{n-k}}{q^k (1-q)^{n-k}}\right)
\end{aligned}$$

$$\begin{aligned}
&= \sum_{k=0}^{n} \binom{n}{k} p^k (1-p)^{n-k} \cdot \left[k \cdot \log\left(\frac{p}{q}\right) + (n-k) \cdot \log\left(\frac{1-p}{1-q}\right) \right] \\
&= \log\left(\frac{p}{q}\right) \cdot \sum_{k=0}^{n} k \binom{n}{k} p^k (1-p)^{n-k} + \log\left(\frac{1-p}{1-q}\right) \cdot \sum_{k=0}^{n} (n-k) \binom{n}{k} p^k (1-p)^{n-k} \\
&= \log\left(\frac{p}{q}\right) \cdot np + \log\left(\frac{1-p}{1-q}\right) \cdot n(1-p) \\
&= n \cdot \mathsf{KL}(P||Q)
\end{aligned}$$

■