Apple Tasting

David P. Helmbold
Computer Science Department
University of California
Santa Cruz, CA 95064 USA

Nicholas Littlestone
NEC Research Institute
4 Independence Way
Princeton, NJ 08540 USA

Philip M. Long
Department of Computer Science
National University of Singapore
Singapore 119260, Republic of Singapore

Abstract

In the standard on-line model the learning algorithm tries to minimize the total number of mistakes made in a series of trials. On each trial the learner sees an instance, makes a prediction of its classification, then finds out the correct classification. We define a natural variant of this model ("apple tasting") where

- *the classes are interpreted as the good and bad instances,*
- *the prediction is interpreted as accepting or rejecting the instance, and*
- *the learner gets feedback only when the instance is accepted.*

We use two transformations to relate the apple tasting model to an enhanced standard model where false acceptances are counted separately from false rejections. We apply our results to obtain a good general purpose apple tasting algorithm as well as nearly optimal apple tasting algorithms for a variety of standard classes, such as conjunctions and disjunctions of n *boolean variables. We also present and analyze a simpler transformation useful when the instances are drawn at random rather than selected by an adversary.*

1 Introduction

Consider the task of learning to visually identify tasty apples. We suppose that the learner encounters apples one by one, and decides whether or not to sample each apple. (For the purpose of this paper, we suppose that a final decision about each apple must be made before the next apple is encountered.) The goal is to avoid sampling too many bad apples and to avoid missing too many good apples. Let us say that an apple that is sampled has been *accepted.* We call each acceptance of a bad apple and each rejection of a good apple a mistake and attempt to keep the number of mistakes small. This is similar to the on-line learning task that has been considered by a variety of researchers in computational learning theory [Ang88] [Blu90b] [Blu90a] [BHL91] [HSW90] [Lit88] [Lit89] [LW89] [Maa91] [MT89] [MT90]. As in that task, learning can be thought of as proceeding in a sequence of trials. In each trial, first the learner observes an object, situation, or event—we call the observation an *instance.* Next the learner makes a prediction of the correct classification of the instance. (Equivalently, in this paper, we will sometimes speak of the learner accepting or rejecting the instance.) In the standard on-line model, the learner is assumed to receive a label indicating the correct classification of the instance at the end of each trial (this label may be corrupted) [Lit89,LW89]. By contrast, here we consider a model in which the learner only receives information about the correct classification if the learner chooses the accept action. Thus in this model, unlike in the standard model, the classification information received by the learner is directly controlled by the action of the learner. If there are instances that the learner thinks, with some uncertainty, are bad, then the learner may need to accept such instances just to learn more about them. This model contains a trade-off, sometimes referred to as the exploration-exploitation trade-off, that is absent in the standard on-line model. We refer to the model that we consider here as the *apple tasting model.*

This can be formalized using the language of decision theory (c.f., [Ber80]) as follows. We assume that at each trial the learner suffers a loss that depends on the learner's action and the category of the instance as in the following loss matrix (where $a < b < c$):

	good	bad
accept	a	c
reject	b	b

and the learner desires to achieve small loss. We assume that the learner knows the loss matrix, and that at the end of each trial the learner learns the value of the loss incurred in that trial. This provides information about the classification of the instance if the learner's action was accept, but not if it was reject.

We measure the learner's total loss on a sequence of trials by simply summing the per-trial losses. The least loss that the learner can achieve on a good instance is a and on a bad instance is b. The *regret* of the learner on a particular trial is the amount by which the loss incurred exceeds the best possible. The total regret, the amount by which the total loss exceeds the best possible (for a given sequence of instances and classifications), is easily seen to be just the sum of the per-trial regrets. For the loss matrix given above, the per trial regrets are given by the following regret matrix:

	good	bad
accept	0	$c - b$
reject	$b - a$	0

Suppose that the instances are each chosen from some domain X. We assume that for any particular learning task that we consider there exists some hidden function $f : X \to \{0,1\}$ such that an instance x is good if and only if $f(x) = 1$. We look for algorithms that are able to do well for arbitrary hidden functions chosen from particular function classes and arbitrary sequences of instances from X. It will turn out that frequently the total regret will grow with the number of trials. If we fix the number of trials and a function class, then we can define an optimal algorithm to be one that minimizes the maximum total regret, where the maximum is over all sequences of instances of the chosen length, and over all possible hidden functions from the chosen class. In the terminology of decision theory, this is a *minimax regret* strategy. We will consider randomized apple tasting algorithms. For these we will look at expected regret, and attempt to minimize the maximum expected regret. In this paper, we will focus our attention on the case in which $b - a = c - b$, and we may assume that both quantities are 1 without further loss of generality. Nevertheless, our techniques appear to generalize to the case in which $b - a \neq c - b$.

This model represents but one of a variety of possible modifications of the standard on-line learning model that seem likely to be important in practice. The study of this model can be viewed partly as the first step of an investigation to see to what extent previous results from computational learning theory carry over to these models and to what extent new research is needed. We show that simple generally applicable transformations of existing algorithms for the standard on-line model yield bounds in the apple tasting model that are in some cases close to the best possible. However, the best apple tasting bounds that we obtain require more work. Existing research concerning the standard mistake-bound model has typically lumped the two types of mistakes (corresponding to accepting the bad and rejecting the good) together, counting them as equally costly. Because in the apple tasting model the learner does not receive immediate feedback regarding mistakes involving rejecting the good, it can help to start with algorithms (for the standard on-line model) that make a small number of mistakes of this type at the expense of a larger number of mistakes of the other type. In a companion paper [HLL], we consider the standard model, in which the learner is told the correct label at the end of each trial, and study the trade-off in the number of mistakes of the two types. From these we obtain upper bounds for the apple tasting model that for some function classes are nearly optimal. In fact, we show that, in a sense, one can always do nearly as well as is possible in the apple tasting model by applying our general-purpose transformation to algorithms in the standard model which differentiate between the two types of mistakes appropriately (Theorem 12).

There are many variants and extensions of this model that it would be interesting to study. One variant considered in Section 6 assumes that the sequence of instances is generated by independent random draws from some distribution on the domain. Another example (not considered here) is a common extension of this model and bandit processes. If one considers the case that the domain X is of size 1 and the classification is stochastic, then we have a kind of bandit process. (This can be thought of as modeling the case that the instances are indistinguishable to the learner, except via the loss they lead to if they are accepted, and they are presented to the learner in random order.) By extending our model to allow the classification to depend stochastically on the instance, we get a model with some of the flavor of bandit processes.

Subsequent to the publication of an earlier version of this paper [HLL92], study of worst-case learning models containing an exploration-exploitation tradeoff continued. Auer, Cesa-Bianchi, Freund and Schapire [ACBFS95] studied the bandit problem in a worst-case framework, and learning linear functions in an extension of the apple tasting model in which there are more than two choices and all choices potentially have variable cost was studied by Long [Lon97].

Recall that the standard model differs from the apple tasting model by giving the algorithm feedback after each prediction – even rejected instances. Another difference, convenient for proving lower bounds, is that we consider only deterministic algorithms for the standard model, while algorithms in the apple tasting model have access to a source of randomness.

Our apple tasting results are based on two simple transformations. The first transformation converts any algorithm for the standard on-line, mistake bounded model into an apple tasting algorithm. If M_+ is a bound on the number of false positives (i.e., incorrect acceptances) and M_- bounds the number of false negatives (incorrect rejections) made by the original algorithm, then the expected number of mistakes made by the resulting apple tasting algorithm on any sequence of T trials is at most $M_+ + 2\sqrt{TM_-}$ (see Corollary 2).

The second transformation converts any apple tasting algorithm into a (deterministic) algorithm in the standard model. This transformation is used to obtain lower bounds for problems in the apple tasting model, and is of added interest because it uses results for deterministic algorithms to bound the performance of randomized algorithms. Using the second transformation, we show that if for some M_+ and M_- each algorithm in the standard model can be forced to make *either* M_+ false positive mistakes *or* M_- false negative mistakes, then every algorithm in the apple tasting model expects to make at least $\frac{1}{2}\min\left\{M_- \left\lfloor \frac{T}{M_- + M_+ - 1} \right\rfloor, M_+\right\}$ mistakes on some sequence of T trials (Lemma 8). Further analysis gives a lower bound on the (worst case) expected number of apple tasting mistakes which is (discounting a short initial interval) $\frac{1}{2}\min\{\frac{1}{2}\sqrt{TM_-}, M_+\}$ (Theorem 9). It is worth noting that these transformations are very effective – one loses only a constant factor by taking an apple tasting algorithm, converting it into a standard model algorithm, and then converting back into an apple tasting algorithm.

One of the interesting features of the apple tasting model is the dependence of the bounds on T, the number of trials. In the standard model, algorithms can be forced to make all of their mistakes on the first trials [Lit89]. The $\sqrt{T}$ factors in our bounds indicate that this is not the case in the apple tasting model.

As an example, consider the function class of singletons which consists of those functions that map only a single element of the domain to 1 (so $f \in$ singletons iff $|f^{-1}(1)| = 1$). In the standard model, any function in this class can be learned with at most one mistake. The *straightforward algorithm* simply predicts 0 on all instances until a positive feedback is seen. Once that occurs, the algorithm has identified the single positive instance and can infer that all other instances will be negative. The straightforward algorithm works poorly in the apple tasting setting as it would never get any feedback; repeatedly presenting the positive instance forces it to make a mistake on every trial.

Thus the learner must predict 1 sometimes in order to get some feedback. The main tool of this paper is a general randomized strategy that uses a succesful standard-model algorithm and then randomly changes some of its predictions to 1 in order to obtain more feedback. Before introducing that general strategy, we will describe in some detail a simple deterministic strategy that works for the case of singletons. This strategy lets one see clearly how balancing the need for feedback and the cost of feedback leads to mistake bounds that grow as $\sqrt{T}$ over the most interesting range of values for T. We will also discuss how the strategy can be modified for large T to make the bound independent of T. Thus the behavior of the bound as a function of T goes through two phases as T grows. There is also a third phase, when T is very small, where the growth is linear in T, though in the case of singletons this phase is nearly absent. The existence of these three phases, with a $\sqrt{T}$ dependence in the middle phase, is a characteristic feature of apple-tasting mistake bounds.

We call this deterministic strategy the *counting algorithm* because it counts how many times each instance has been seen. It starts by predicting 0, but will predict 1 on those trials where the current instance is being seen for the r^{th} time, where r is a parameter to be optimized. (If the current instance has been seen more than r times, then the learner has seen the correct prediction.) On any sequence of T trials, this learner makes at most $r-1$ false negative mistakes and at most T/r false positive mistakes. Setting r to one of $\lfloor\sqrt{T}\rfloor$ or $\lceil\sqrt{T}\rceil$ minimizes its total number of mistakes, $r-1+T/r$. Using $r=\lceil\sqrt{T}\rceil$ allows us to bound the total number of mistakes by $2\sqrt{T}$. It is easy to see that an adversary can force more mistakes if the learner is either too eager to obtain feedback, choosing a small r, or too cautious about obtaining feedback, choosing a large r (provided that the domain from which the instances are chosen is sufficiently large).

When T is very small or very large (in comparison to the size of the domain), this $2\sqrt{T}$ bound is pessimistic. For small T, we note that one can make no more than T mistakes. (When we turn to randomized strategies, we will want to predict with random coin flips for small T, thus expecting to make at most $T/2$ mistakes.) When the domain contains N instances, the counting algorithm makes at most $N-1$ false positive mistakes, since it never makes more than one false positive mistake for any given instance. Thus the total number of mistakes can be bounded by $r-1+N-1$. This is minimized when r takes its minimum value of 1, and is less than $2\sqrt{T}$ for (very) large values of T.

These considerations lead us to an an apple tasting algorithm whose behavior (when given T) goes through different phases as T grows. The algorithm uses $r=\lceil\sqrt{T}\rceil$ for $T<(N-1)^2/4$ and $r=1$ for larger T. The mistake bound for this algorithm is $\min(T, 2\sqrt{T}, N-1)$. It turns out that this bound is within a constant factor of optimal.

A randomized prediction strategy can obtain a similar $O(\sqrt{T})$ expected mistake bound for the middle range of values of T without counting the number of times each instance has been seen. It simply predicts 1 with probability $1/\sqrt{T}$ at each trial until it discovers the positive instance.

This randomized strategy is a special case of our general transformation strategy. That general strategy can transform an arbitrary deterministic standard model algorithm into a randomized algorithm suitable for use in the apple-tasting setting. It randomly changes 0 predictions to 1's with an appropriate probability. We give two variants of our general strategy that we call *STtoAP* and *STAP*. The STtoAP variant is simpler and has a better constant, while the STAP variant does not require advance knowledge of T.

The STtoAP and STAP conversions are generic. For example, an apple tasting algorithm making an expected number of mistakes bounded by $M_+ + 2\sqrt{TM_-}$ is produced whenever STtoAP is given a standard model algorithm making at most M_+ false positive predictions and M_- false negative predictions. Similar bounds also hold for STAP.

As in the case of our singletons example, to obtain good mistake bounds for all values of T, the algorithm usually must behave differently for very small or very large values of T than for the middle range. The mistake bounds for the apple tasting model that arise typically make two phase transitions as a function of T:

1. growing as $T/2$ for small values of T as random prediction is best,
2. growing as $\sqrt{T}$ for intermediate values of T,
3. constant when T is large enough that exploitation can be deferred until the hidden function has been deduced.

Each different phase requires a different approach, and that selecting the best approach requires advance knowledge of T. Even if T is known to lie in the middle phase, some of our algorithms perform better when given T's actual value (however our best algorithm for singletons does not need this information).

The optimal algorithm for the first phase is not of particular interest as it does not do any learning. For most of the function classes and domains that we have explored, the transition between the second and third phases occurs at extremely large T and the constant bound is exponential in the size of the instances (a notable exception is the function class of boolean disjunctions, where the constant bound is of the same order as the size of the instances). Therefore we concentrate on the more interesting middle phase.

The remainder of the paper is organized as follows. The next section contains a formal description of the apple tasting model and introduces the notation used throughout the paper. Section 3 describes and analyzes our transformations between standard model algorithms and apple tasting algorithms.

In Section 4 we study what can be said about the apple-tasting mistake bound for a concept class $\mathcal{F}$ when one pays attention to nothing about the class other $|\mathcal{F}|$. The results we obtain are analogous to the well known result that in the standard model one can learn a concept class of size n with at most $\log n$ mistakes (using the Halving algorithm [BF72,Lit88]). They are in this regard also similar in spirit to the Occam algorithm results for PAC learning [BEHW87]. We show that, for any concept class $\mathcal{F}$, there is an apple tasting algorithm whose expected number of mistakes on any sequence of T trials is $O\left(\min\left\{T, \sqrt{\frac{T\ln|\mathcal{F}|}{\ln(1+T/(\ln|\mathcal{F}|))}}, |\mathcal{F}|\right\}\right)$ (Theorem 14).[1] We obtain this bound using our transformation from standard-model algorithms to apple-tasting algorithms. We apply the transformation to a standard-model algorithm described in the companion paper [HLL] that can be tuned to vary the trade-off between false positive and false negative mistakes. Lower bounds (see Theorem 15) show that the mistake bound for the resulting apple-tasting algorithm is within a constant factor of the best possible bound depending only on $|\mathcal{F}|$ and T.

In Section 5 we look at a variety of specific classes. We show that for some natural concept classes the upper bounds that result from the approach of Section 4 are within a constant factor of the best possible bounds for those classes (where we now are paying attention to full information about the class, not just its size). However, the resulting algorithms are often not computationally efficient. For each concept class that we study, we also give the expected mistake bounds for the best efficient algorithm that we know of. Table 1 summarizes the results of Section 5.

Section 6 examines the apple tasting model when the instances are drawn at random from an unknown distribution rather than selected by an adversary. Our conclusions and further thoughts appear in Section 7.

2 Preliminaries

If X is a set and D is a probability distribution over X, we denote the expectation of function $\phi : X \to \mathbf{R}$ with respect to D by $\mathbf{E}_{x\in D}(\phi(x))$ and we define $\mathbf{Pr}_{x\in D}$ analogously. We will assume throughout that X is finite. The results also apply to infinite domains if suitable measurability and computability assumptions are made.

Choose a set X, which is called the *domain*. The elements of X are called *instances*. Let f be a function from X to $\{0,1\}$. An *example* for f is either a pair $(x, f(x))$ or $(x, *)$, where $x \in X$. Here $*$ is intended to mean that the value of the hidden function was not obtained. A sample is a finite sequence of examples (we often use $\emptyset$ to denote the empty sample). A function g is *consistent* with a sample if for each (x, ρ) in the sample for which $\rho \neq *$, $g(x) = \rho$. We denote the set of all samples

[1]Bounds that are a minimum of two or three terms will often occur. We will usually arrange them so that if parameters other than T are held fixed, the terms are listed in the order in which they apply (are smallest) as T increases.

by $\mathcal{S}$. A learning algorithms uses a sequence of *trials* to learn some *hidden function* (or target) f. On each trial the learning algorithm is given the new instance x, generates a prediction, and then receives the *feedback* for that trial (either $f(x)$ if the prediction was 1, or the uninformative feedback $*$ if the prediction was 0). It can use the instance and the feedback to update its internal state.

The prediction of a randomized learning algorithm is a computable function from $\mathcal{S} \times X \times [0,1]^\infty$ to $\{0,1\}$. The first parameter is a sample consisting of the past instances and the corresponding feedback values. The second parameter is the current instance, and the third is a random sequence used for randomization. We will sometimes use the name of the learning algorithm to denote this function. Elements of the random sequence will be generated one by one independently at random as needed by the algorithm. We are particularly interested in the random source R which generates real numbers by sampling from the uniform distribution on $[0,1]$. We assume that R takes unit time to generate each random number.

If $\sigma = \langle x_t \rangle_{t=1}^T$ is a sequence of elements of X, f is a function from X to $\{0,1\}$, and A is a learning algorithm, we define the predictions $\langle \lambda_t \rangle_{t=1}^T$ and feedbacks $\langle \rho_t \rangle_{t=1}^T$ inductively as follows:

- $\lambda_1 = A(\emptyset, x_1, R)$
- $\rho_1 = \begin{cases} f(x_1) & \text{if } \lambda_1 = 1 \\ * & \text{otherwise} \end{cases}$
- $\lambda_t = A(((x_1, \rho_1), ..., (x_{t-1}, \rho_{t-1})), x_t, R)$
- $\rho_t = \begin{cases} f(x_t) & \text{if } \lambda_t = 1 \\ * & \text{otherwise.} \end{cases}$

Note that the dependence of the λ_t's and the ρ_t's on σ, f, and the outputs of R is not explicitly written in the notation, as their values can be understood from context throughout the paper. The apple tasting performance of A on σ and f, written as $\mathrm{AL}(A, \sigma, f)$, is defined by

$$\mathrm{AL}(A, \sigma, f) = \mathbf{E}(|\{t : \lambda_t \neq f(x_t)\}|)$$

where the expectation is over the outputs of R. We denote the worst case performance of an algorithm A learning a function from some class $\mathcal{F}$ of functions from X to $\{0,1\}$ on an input sequence of length T by

$$\mathrm{AL}(A, \mathcal{F}, T) = \sup_{f \in \mathcal{F}, \sigma \in X^T} \mathrm{AL}(A, \sigma, f).$$

Finally, we define the apple tasting learning complexity (analogous to standard learning complexity [MT89]) of $\mathcal{F}$ on sequences of T trials as

$$\mathrm{ALC}(\mathcal{F}, T) = \inf_A \mathrm{AL}(A, \mathcal{F}, T).$$

Note that the above definition allows a different algorithm for each T, and therefore, loosely speaking, we might view our apple tasting algorithms as "knowing" T.

As it is often easy to quickly calculate $|\mathcal{F}|$ for natural concrete $\mathcal{F}$, we are interested in general bounds on $\mathrm{ALC}(\mathcal{F}, T)$ in terms of $|\mathcal{F}|$ as well as T. Let

$$\mathrm{ALC}(n, T) = \max_{\mathcal{F} : |\mathcal{F}| = n} \mathrm{ALC}(\mathcal{F}, T)$$

denote the best possible such bounds. Note that the definitions trivially imply that $\mathrm{AL}(A, \mathcal{F}, T)$, $\mathrm{ALC}(\mathcal{F}, T)$, and $\mathrm{ALC}(n, T)$ are all non-decreasing in T.

We will make use of algorithms which get the value of $f(x_t)$ on each trial as subroutines for our apple tasting algorithms. Throughout, we will refer to the model in which the value of the hidden function is always received as the *standard model.* We develop here a notation for this model. If standard model algorithm B maps $\mathcal{S} \times X$ to $\{0,1\}$ and $\sigma = \langle x_t \rangle_{t=1}^{T}$ as above, define the predictions $\langle \xi_t \rangle_{t=1}^{T}$ by

- $\xi_1 = B(\emptyset, x_1)$.
- $\xi_t = B(((x_1, f(x_1)), ..., (x_{t-1}, f(x_{t-1}))), x_t)$.

Again, the dependence on σ and f is not made explicit. If $\xi_t = 1$ we say interchangeably that in trial t the algorithm: predicts 1; makes a positive prediction; or accepts the instance (the same terminology is used for λ_t). If $\xi_t = 1$ and $f(x_t) = 0$, we say that the algorithm makes a false positive mistake. Analogous definitions are made for negative predictions (rejections) and false negative mistakes.

Note that we require the standard model algorithm to be deterministic; it is not given a source of random numbers. This will be useful in proving lower bounds. Continuing, let

$$\begin{aligned}
&\mathrm{L}_+(B,\sigma,f) = |\{t : \xi_t = 1 \text{ and } f(x_t) = 0\}| \\
&\mathrm{L}_-(B,\sigma,f) = |\{t : \xi_t = 0 \text{ and } f(x_t) = 1\}| \\
&\mathrm{L}_+(B,\mathcal{F}) = \sup\nolimits_{\sigma, f\in\mathcal{F}} L_+(B,\sigma,f) \\
&\mathrm{L}_-(B,\mathcal{F}) = \sup\nolimits_{\sigma, f\in\mathcal{F}} L_-(B,\sigma,f) \\
&\mathrm{L}_+^k(\mathcal{F}) = \inf\nolimits_{B:L_-(B,\mathcal{F})\le k} L_+(B,\mathcal{F}).
\end{aligned}$$

Informally, $\mathrm{L}_+(B,\sigma,f)$ is the number of false positive mistakes made by B learning f with σ, $\mathrm{L}_-(B,\sigma,f)$ is the number of false negative mistakes, and $L_+^k(\mathcal{F})$ is the best bound on the number of false positive mistakes obtainable by an algorithm which makes at most k false negative mistakes.

We say that a function class $\mathcal{F}$ is *amply splittable* if for every subclass $\mathcal{F}' \subseteq \mathcal{F}$ and every positive integer $k < |\mathcal{F}'|$ there exists an $x \in X$ such that $|\{f \in \mathcal{F}' : f(x) = 0\}| = k$. Note that if a class $\mathcal{F}$ is amply splittable then every subclass of $\mathcal{F}$ is also amply splittable. For example, the class of initial segments ($X = \{1, 2, \ldots, n\}$ and $\mathcal{F} = \{\{1, \ldots, k\} : 1 \le k \le n\}$) is amply splittable.

3 Transformations

In this section we show how a standard model algorithm B can be used to obtain an apple tasting algorithm A. If the standard model algorithm can appropriately trade between false positives, $\mathrm{L}_+(B,\mathcal{F})$, and false negatives, $\mathrm{L}_-(B,\mathcal{F})$, then the resulting apple tasting algorithm makes few mistakes. We also demonstrate how to use a good apple tasting algorithm to obtain a good learning algorithm for the standard mistake-bound model. The latter transformation is useful for proving lower bounds in the apple tasting model.

3.1 From the Standard Model to Apple Tasting

Notice that any standard model algorithm which never makes a false negative prediction can be used without modification in the apple tasting setting. Because such an algorithm already "knows" the label will be negative when it makes a negative prediction, it is not necessary to obtain the feedback. If a standard model algorithm can make false negative predictions then our transformations will randomly change some negative predictions to positive ones in order to obtain some feedback on those instances that are rejected.

The first transformation we consider takes two parameters, M and T, in addition to a standard model algorithm B. Parameter M is intended to be a bound on the number of false negative

Notation conventions used throughout the paper	
X	the domain, set of possible instances
x	an instance (element of X)
D	probability distribution on X
$\mathcal{F}$	the set (or class) of possible hidden functions
f	the hidden function from X to $\{0,1\}$ to be learned
$*$	a placeholder indicating the value of f was not obtained
$\mathcal{S}$	the set of all possible samples
R	the random source sampling uniformly from $[0,1]$
A	an apple tasting algorithm
B	a standard model algorithm
σ	a sequence of instances
ρ	the feedback received by the apple tasting algorithm
λ	prediction of an apple tasting algorithm
T	upper bound on the number of trials
M, M_+, M_-	bounds on the the numbers of mistakes, false positive mistakes, and false negative mistakes made by a standard model algorithm.
$\mathrm{AL}(A,\sigma,f), \mathrm{AL}(A,\mathcal{F},T)$	apple tasting performance of algorithm A
$\mathrm{ALC}(\mathcal{F},T)$	apple tasting complexity of function class $\mathcal{F}$
$\mathrm{ALC}(n,T)$	maximum apple tasting complexity over function classes of size n
ξ	prediction of a standard model algorithm
$\mathrm{L}_+(B,\sigma,f)$	number of false positive mistakes made by B
$\mathrm{L}_-(B,\sigma,f)$	number of false negative mistakes made by B
$\mathrm{L}_+(B,\mathcal{F})$	maximum number of false positive mistakes made by B when learning hidden functions in $\mathcal{F}$
$\mathrm{L}_-(B,\mathcal{F})$	maximum number of false negative mistakes made by B when learning hidden functions in $\mathcal{F}$
$\mathrm{L}_+^0(\mathcal{F})$	the minimum number of false positive mistakes made by standard model algorithms which make no false negative mistakes

mistakes (or occasionally the total number of mistakes) made by algorithm B. The T parameter indicates the number of trials on which the resulting apple tasting algorithm will be asked to predict. Later in the section we present a second transformation with similar performance which does not need this information.

Transformation *STtoAP*. Given a standard model algorithm B and parameter M, we form the apple tasting algorithm STtoAP(B, M, T) as follows. On the tth trial:

- STtoAP(B, M, T) receives x_t and passes it on to B.
- B predicts ξ_t.
- STtoAP(B, M, T) calls R obtaining $r \in [0, 1]$ and predicts

$$\lambda_t = \begin{cases} 1 & \text{if } \xi_t = 1 \text{ or } \xi_t = 0 \text{ and } r \leq \sqrt{M/T} \\ 0 & \text{otherwise.} \end{cases}$$

- If $\lambda_t = 0$, then, loosely speaking, B acts as if this trial never happened; i.e., B's future predictions will be functions only of the subsequence of trials in which feedback was received. Otherwise STtoAP(B, M, T) receives $f(x_t)$ from the environment and passes it on to B.

To illustrate this transformation, let B be the straightforward standard model algorithm for singletons that predicts 0 until it sees the positive instance, and then predicts 1 on that instance and zero elsewhere. Note that this algorithm makes only a single false negative mistake. The apple tasting algorithm STtoAP$(B, 1, T)$ starts by predicting 1 with probability $\sqrt{1/T}$ on each instance. If STtoAP$(B, 1, T)$ predicts 1 on the positive instance then the standard model algorithm B identifies the hidden function f. Thereafter STtoAP$(B, 1, T)$ will predict 1 if the positive instance is seen again. However, this generic transformation exploits neither the fact that $f(x) = 0$ for all other instances nor the fact that it may have already seen the function's value on the current instance. Therefore it will continue to make false positive mistakes on the other instances with probability $\sqrt{1/T}$.

In this example, we can easily bound the expected number of mistakes made by STtoAP$(B, 1, T)$. The expected number of false positive mistakes is at most $T\sqrt{1/T} = \sqrt{T}$. The expected number of false negative mistakes is just the expected number of times STtoAP$(B, 1, T)$ predicts zero on the positive instance before predicting 1. The expected length of this run of zeros is bounded by $\sqrt{T} - 1$. Therefore, the expected total number of mistakes is at most $2\sqrt{T} - 1$

The following theorem extends this reasoning to the general case where the standard model algorithm B makes at most M_+ false positive mistakes and M_- false negative mistakes.

Theorem 1 *Let X be a finite set, and let $\mathcal{F}$ be a class of functions from X to $\{0, 1\}$. Suppose B is a standard model learning algorithm for which $L_+(B, \mathcal{F}) \leq M_+$ and $L_-(B, \mathcal{F}) \leq M_-$. Then for all nonnegative integers $T \geq M_-$, the learning algorithm STtoAP(B, M_-, T) has*

$$AL(\mathit{STtoAP}(B, M_-, T), \mathcal{F}, T) \leq M_+ - M_- + 2\sqrt{TM_-}.$$

Proof: Choose $T \geq M_-, f \in \mathcal{F}$ and a sequence $x_1, ..., x_T$ in X^T. Let A denote algorithm STtoAP(B, M_-, T). Consider a fixed sequence of numbers generated by R and divide up the trials into classes, where the class of the tth trial depends on B's prediction ξ_t, A's prediction λ_t, and $f(x_t)$. For each $\vec{b} \in \{0, 1\}^3$, let

$$S_{\vec{b}} = \{t : \xi_t = b_1, \lambda_t = b_2, f(x_t) = b_3\}.$$

Note that the $S_{\vec{b}}$'s are functions of the random numbers generated by R, which for now have been fixed. From the definition of B, $|S_{1,0,0}| = |S_{1,0,1}| = 0$. Thus,

$$\begin{aligned} \mathrm{AL}(A, \sigma, f) &= \mathbf{E}(|S_{1,1,0}| + |S_{0,1,0}| + |S_{0,0,1}|) \\ &\leq M_+ + \mathbf{E}(|S_{0,1,0}|) + \mathbf{E}(|S_{0,0,1}|) \end{aligned} \tag{1}$$

where the expectations are over the random numbers generated by R.

We can trivially bound $|S_{0,1,0}|$ by the number of times that A changes the prediction made by B, which A does with probability at most $\sqrt{M_-/T}$ on any given trial. Thus,

$$\mathbf{E}(|S_{0,1,0}|) \leq (\sqrt{M_-/T})T = \sqrt{TM_-}. \tag{2}$$

Next, by the bounds on the performance of B, we have that, for all sequences of random bits, $|S_{0,1,1}| \leq M_-$, as B receives $f(x_t)$ for these trials.

This implies that $|S_{0,0,1}|$ is at most the number of failures before M_- successes in independent Bernoulli trials with probability $\sqrt{M_-/T}$ of success, so (see, e.g. [Fel68]),

$$\mathbf{E}(|S_{0,0,1}|) \leq M_- \sqrt{\frac{T}{M_-}} - M_- = \sqrt{TM_-} - M_-. \tag{3}$$

Combining this with (2) and (1) yields the desired result. □

This trivially yields the following Corollary, in which we remove the restriction that $T \geq M_-$.

Corollary 2 *Let X be a finite set, and let $\mathcal{F}$ be a class of functions from X to $\{0,1\}$. Suppose B is a learning algorithm for which $L_+(B, \mathcal{F}) \leq M_+$ and $L_-(B, \mathcal{F}) \leq M_-$. Then for all nonnegative integers T*

$$AL(STtoAP(B, M_-, T), \mathcal{F}, T) \leq M_+ + 2\sqrt{TM_-}.$$

Proof: The expected number of STtoAP(B, M_-, T)'s mistakes is certainly at most T. If $T \leq M_-$, then this is at most $\sqrt{TM_-}$, satisfying the claimed bound. The case in which $T > M_-$ follows immediately from Theorem 1, completing the proof. □

We may also derive an apple tasting bound in terms of the total number of (standard) mistakes of any kind.

Corollary 3 *Let X be a finite set, and let $\mathcal{F}$ be a class of functions from X to $\{0,1\}$. If Learning Algorithm B makes at most M mistakes of any kind, then the algorithm A which behaves like STtoAP(B, M, T) if $T > 20M$ and predicts randomly if $T \leq 20M$ has*

$$AL(A, \mathcal{F}, T) \leq \sqrt{5TM} \leq 2.24\sqrt{TM}.$$

Proof: Corollary 2 implies that if $T > 20M$

$$\mathrm{AL}(A, \mathcal{F}, T) \leq M + 2\sqrt{TM}.$$

Because $T > 20M$, we have $TM > 20M^2$ and $M < \sqrt{TM/20}$. Now,

$$\mathrm{AL}(A, \mathcal{F}, T) \leq (2 + 1/\sqrt{20})\sqrt{TM} \leq \sqrt{5TM}$$

in this case. On the other hand, if $T \leq 20M$

$$\mathrm{AL}(A, \mathcal{F}, T) \leq T/2 \leq \sqrt{20TM}/2 = \sqrt{5TM}.$$

□

Note that STtoAP needs to know both the total number of trials T and an upper bound M on the number of incorrect rejections made by B in order to compute the needed probability. The following transformation adaptively adjusts these parameters, eliminating the need to know them in advance. Unfortunately, not knowing the M_- parameter can change the asymptotics of the bound when $M_- = 0$. The essential difference is that the STtoAP transformation can trust the standard model algorithm's rejections when told that no false negative mistakes will be made, while the STAP transformation does not have this knowledge, and must probabilistically check the correctness of these rejections.

Transformation *STAP* Given a standard model algorithm B, we form the apple tasting algorithm STAP(B) as follows.

Algorithm STAP(B) initializes a variable m_- to 0. On the tth trial:

- STAP(B) receives x_t and passes it on to B.
- B predicts ξ_t.
- STAP(B) calls R obtaining $r \in [0,1]$ and predicts

$$\lambda_t = \begin{cases} 1 & \text{if } \xi_t = 1, \text{ or if } \xi_t = 0 \text{ and } r \leq \sqrt{(m_- + 1)/t} \\ 0 & \text{otherwise.} \end{cases}$$

- If $\lambda_t = 0$, then, loosely speaking, B acts as if this trial never happened; i.e., B's future predictions will be functions only of the subsequence of trials in which reinforcement was received. Otherwise STAP(B) receives $f(x_t)$ from the environment and passes it on to B. In addition, if $\xi_t = 0$, $\lambda_t = 1$, and $f(x_t) = 1$ then STAP(B) increments m_-.

Although STtoAP(B, M, T) "flips" the zero predictions of algorithm B with the fixed probability $\sqrt{M/T}$, algorithm STAP(B) estimates the mistake bound M and the number of trials T. The total number of trials T is estimated by the current trial number, t. The mistake bound M is estimated by $m_- + 1$, where m_- is the number of previous trials where B has been observed make a false negative prediction – i.e. B predicted 0, the conversion predicted 1, and the observed outcome was 1. (If we estimated the mistake bound M by m_- rather than $m_- + 1$, then the conversion would never randomly change a prediction and m_- would remain stuck at 0.)

Note that the probabilities used by STAP(B) and STtoAP(B, M, T) are incomparable[2].

As with STtoAP, we separately bound the expected number of false positive and false negative predictions made by the STAP(B) conversion. This leads to the following theorem.

Theorem 4 *Let X be a finite set, and let $\mathcal{F}$ be a class of functions from X to $\{0,1\}$. Suppose B is a learning algorithm for which $L_+(B, \mathcal{F}) \leq M_+$ and $L_-(B, \mathcal{F}) \leq M_-$. Then for the learning algorithm STAP(B) and all nonnegative integers $T \geq M_-$,*

$$AL(STAP(B), \mathcal{F}, T) \leq M_+ - M_- + 4\sqrt{T(M_- + 1)}.$$

[2]On the first trial, STtoAP(B) flips negative predictions with probability $\sqrt{M/T}$ while STAP(B) always predicts one. Once B has made M false negative predictions, then STAP(B) will again flip negative predictions more aggressively than STtoAP(B, M, T). On the other hand, if few false negatives are observed on the initial trials (so m_- is much less than M), then the probability that STAP(B) flips the prediction can be much less than $\sqrt{M/T}$.

Proof: As in the proof of Theorem 1, choose $T \geq M_-$, $f \in \mathcal{F}$, and a sequence $x_1, ..., x_T$ in X^T. Let A denote algorithm STAP(B). Consider a fixed sequence of numbers generated by R and divide up the trials into classes as before. Again we have

$$\mathrm{AL}(A, \sigma, f) \leq M_+ + \mathbf{E}(|S_{0,1,0}|) + \mathbf{E}(|S_{0,0,1}|)$$

We can trivially bound $|S_{0,1,0}|$ by the number of times that A changes the prediction made by B, which A does with probability at most $\sqrt{(m_- + 1)/t} \leq \sqrt{(M_- + 1)/t}$ on trial t. Thus the expected number of of times that STAP(B) changes the prediction made by B is at most

$$\mathbf{E}(|S_{0,1,0}|) \leq \sum_{t=1}^{T} \sqrt{(M_- + 1)/t} \leq \int_0^T \sqrt{(M_- + 1)/x}\, dx = 2\sqrt{(M_- + 1)T}. \tag{4}$$

Let $m_-(t)$ denote the value of the variable m_- at the beginning of trial t. For $j = 0, ..., M_- - 1$, let $U_j = \{t \in S_{0,0,1} : m_-(t) = j\}$. Note that by the bounds on the performance of B, when $m_- \geq M_-$ there will never be a trial t at which $\xi_t = 0$ and $f(x_t) = 1$, so the union of these sets is all of $S_{0,0,1}$.

In order for $|U_j|$ to be at least i, the choice $\lambda_t = 0$ must be made by $STAP(B)$ in at least i consecutive trials at which $\xi_t = 0$ and $f(x_t) = 1$, starting with the first such trial at which $m_-(t) = j$. Each one of these choices will be made with probability at most $1 - \sqrt{(j+1)/T}$, and each choice is independent when conditioned on previous events. Thus the probability that $|U_j| \geq i$ is at most $\left(1 - \sqrt{(j+1)/T}\right)^i$. The expectation of $|U_j|$ is just the sum for all $i \geq 1$ of the probability that $|U_j| \geq i$, which is bounded by $\sqrt{T/(j+1)} - 1$. Thus

$$\mathbf{E}(|S_{0,0,1})| \leq \sum_{j=0}^{M_- - 1} (\sqrt{T/(j+1)} - 1) \leq \int_0^{M_-} \sqrt{T/x}\, dx - M_- = 2\sqrt{(M_- T)} - M_-.$$

Combining this with (4) and (3.1) yields the desired result. □

A qualitative difference between the bounds of Theorems 1 and 4, is that when $M_- = 0$, the first bound is independent of T, while the second grows with T like $\sqrt{T}$. The difference reflects the fact that if M_- is 0, but it is not known to be 0 by the learner, then the algorithm needs to take into account the possibility that it is not. Our transformation does this by randomly changing some 0 predictions to 1, leading to extra mistakes. It is easy to see that a standard algorithm achieves $M_- = 0$ if and only if it predicts 1 on trial t whenever there is an element f of $\mathcal{F}$ consistent with earlier trials for which $f(x_t) = 1$. Thus, the property that an algorithm never makes an incorrect rejection is often easily tested, and such algorithms trivially make a number of mistakes in the apple tasting model at most the number of incorrect acceptances, without requiring modification.

3.2 From Apple Tasting back to the Standard Model

We now turn to the problem of obtaining lower bounds in the apple tasting model. Toward this end, we introduce two other transformations: one that converts apple tasting algorithms into standard model algorithms, and a second that makes standard model algorithms "conservative." Together, these transformations produce good standard model algorithms from good apple tasting algorithms. We will use these transformations to obtain lower bounds in the apple tasting model from known lower bounds in the standard model (Theorem 9) through the following reasoning.

If there is a lower bound on all standard model algorithms for a given function class, then no apple tasting algorithm for that class can have a mistake bound so low that its conversion breaks the standard model lower bound.

We begin by presenting and analyzing the two transformations. The first, APtoST, creates a standard model algorithm from an apple tasting algorithm. Recall that the apple tasting algorithms often randomize their predictions, but we insist that standard model algorithms be deterministic. Because of the difference in feedback for the two models, this turns out to be more difficult than one might expect.

For example, consider learning the function class of singletons with the good apple tasting algorithm which predicts 1 with probability $\sqrt{1/T}$ on each trial, until it predicts 1 on the positive instance (this is the STtoAP conversion of the straightforward algorithm). Simply rounding this apple tasting algorithm's probability of predicting 1 at each trial leads either to an algorithm that always predicts 1, or an algorithm that always predicts 0, neither of which performs well in the standard model (rounding to 0 means that the algorithm never predicts 1 on the positive instance).

Our way out of this difficulty is to consider what would happen if the instance were presented to the apple tasting algorithm many times rather than just once. Thus we obtain a standard model prediction by rounding the probability that the apple tasting algorithm would predicts 1 on *any* of the next k trials, assuming that the same instance is repeated each time.

Transformation *APtoST*. Given an apple tasting algorithm A we form the standard model algorithm APtoST(A, k) as follows. The prediction of APtoST$(A, k)(((x_1, \rho_1), ..., (x_{t-1}, \rho_{t-1})), x_t)$ is 1 if the following is at least $1/2$:

> The probability (with respect to A's randomization) that if A were presented with the sequence of kt instances consisting of k copies of x_1, followed by k copies of x_2, ..., followed by k copies of x_t, where the feedback after a prediction of 1 on any copy of x_i is ρ_i, algorithm A would predict positively on at least one of the k copies of x_t.

Algorithm APtoST(A, k) predicts 0 whenever this probability is less than $1/2$.

To illustrate this transformation, consider the APtoST conversion of the good apple tasting algorithm for singletons described above. First assume that k is small enough so that the probability of any successes occurring in k Bernoulli experiments, each with probability $1/\sqrt{T}$ of success, is less than $1/2$, but the probability of some success in $2k$ experiments is greater than $1/2$. With this k, the APtoST conversion will never predict positively on a negative instance.

Things become a bit more complicated when considering the positive instance. The first time the positive instance is seen, the APtoST conversion predicts 0. However, when the positive instance is seen the second time, the APtoST conversion will consider the behavior of the apple tasting algorithm on a sequence including $2k$ copies of the positive instance. Once the apple tasting algorithm predicts 1 on any of these instances, it will predict 1 on all following ones. Therefore the probability of predicting 1 on any of the last k copies of the positive instance is the same as the probability of one or more successes in the $2k$ Bernoulli experiments, and is greater than $1/2$. This means that the APtoST conversion will also predict 1 the second time the positive instance is received.

In effect, we have managed to complete the circle – using the STtoAP transformation to convert the straightforward standard model algorithm for singletons into an apple tasting algorithm, and then using the APtoST transformation to convert this apple tasting algorithm back into the straightforward algorithm. Of course, singletons are a particularly nice case, and things don't work out quite as well in general. However, Theorem 12 at the end of this section shows that making a circuit around the circle degrades the mistake bounds by only a small constant factor.

The next lemma forms the basis for our analysis of Transformation APtoST. It bounds the apple tasting loss in terms of the standard model performance of APtoST conversions.

Lemma 5 *Let X be a finite set, $\mathcal{F}$ be a class of functions from X to $\{0,1\}$ and A be an apple tasting algorithm. For all non-negative integers l, $\sigma \in X^l$, $f \in \mathcal{F}$, and non-negative integers k,*

$$L_+(APtoST(A,k),\sigma,f) + kL_-(APtoST(A,k),\sigma,f) \leq 2AL(A,\mathcal{F},lk).$$

Proof: Pick any X, $\mathcal{F}$, and apple tasting algorithm A. If σ is the empty sequence ($l = 0$) then the inequality is trivial. Pick any positive integer l, $\sigma = x_1, x_2, \ldots, x_l \in X^l$, non-negative integer k and $f \in F$. Form the length kl sequence $\sigma' = z_1, \ldots, z_{kl}$ by setting $z_1, \ldots, z_k$ all equal to x_1, setting $z_{k+1}, \ldots, z_{2k}$ all equal to x_2, and so on.

Suppose APtoST(A,k) makes at least M_- false negative mistakes on the sequence σ. Then, by definition, A expects to make at least $(kM_-/2)$ false negative mistakes on σ', because for any x_t on which APtoST(A,k) predicts 0, there is probability at least 1/2 that A will never predict 1 on any of the k corresponding z_j's. Similarly, if APtoST(A,k) makes at least M_+ false positive mistakes, apple tasting algorithm A expects to make at least $M_+/2$ false positive mistakes on σ'.

Therefore,

$$L_+(\text{APtoST}(A,k),\sigma,f) + kL_-(\text{APtoST}(A,k),\sigma,f) \leq 2\text{AL}(A,\sigma',f) \leq 2\text{AL}(A,\mathcal{F},lk)$$

completing the proof. □

Note that the bound of Lemma 5 depends on l, the number of trials on which the standard model algorithm must make predictions. In the following we argue that only short sequences matter. In particular, Lemma 7 below states that if any standard model algorithm makes few mistakes on all short sequences, then the conservative version of that algorithm makes few mistakes on all sequences.

We now present the standard transformation to a conservative algorithm which essentially ignores those trials where the algorithm predicts correctly. This transformation is known to preserve mistake bounds of deterministic algorithms (see, for example, [Lit88]). Here we will need a slightly stronger property, given in Lemma 7 below.

Transformation CONSERVE. Given a standard model algorithm B, we form the standard model algorithm CONSERVE(B), which simulates B as follows. At the start of each trial, CONSERVE(B) saves the current state of algorithm B. Algorithm CONSERVE(B) then gives the trial's instance to the simulation of B, and uses the prediction made by the simulation as its prediction. If the prediction turns out to be correct then CONSERVE(B) restores the simulation of B to the state saved at the beginning of the trial. If the prediction is incorrect, then CONSERVE(B) gives the correct response to the simulation of B and uses the resulting state of B in the next trial (discarding the previously saved state of B).

Note that if B is a polynomial-time algorithm, then so is CONSERVE(B).

Lemma 6 *Let B be a deterministic standard model algorithm. For any sequence σ and function f from X to $\{0,1\}$, there exists a subsequence σ' such that $L_+(B,\sigma',f) = L_+(CONSERVE(B),\sigma,f)$, $L_-(B,\sigma',f) = L_-(CONSERVE(B),\sigma,f)$, and the length of σ' equals $L_+(B,\sigma',f) + L_-(B,\sigma',f)$.*

Proof: This follows immediately from the construction of CONSERVE(B). The sequence consists of those elements of σ on which CONSERVE(B) predicts incorrectly. □

Lemma 7 *Let X be a finite set, $\mathcal{F}$ be a class of functions from X to $\{0,1\}$, and B be a deterministic standard model algorithm. For any positive integer l and any $f \in \mathcal{F}$, if $L_+(B,\sigma,f)+L_-(B,\sigma,f) < l$ for all $\sigma \in X^l$ then $L_+(CONSERVE(B),\sigma,f) + L_-(CONSERVE(B),\sigma,f) < l$ for all $\sigma \in X^*$.*

Proof: If the conclusion fails to hold, then there exists a $\sigma \in X^*$ for which $L_+(\text{CONSERVE}(B),\sigma,f) + L_-(\text{CONSERVE}(B),\sigma,f) = l$. Applying Lemma 6 to this σ gives us a contradiction. □

We are now ready to prove the key lower bound lemma which states that if any standard model algorithm can be forced to make either many false positive mistakes or many false negative mistakes, then mistake bounds in the apple tasting model are also large. As an example, consider learning singletons on a domain of cardinality N. Any standard model algorithm must either predict negatively on an unknown instance (and can be forced to make a false negative mistake) or predicts positively on all unknown instances (making $N-1$ false negative mistakes in the worst case). Now consider Lemma 5 with $k = l < N$ (recall that the number of apple tasting trials is $T = kl$). If the conversion makes a false negative prediction, then the apple tasting algorithm makes at least $k/2 = \sqrt{T}/2$ mistakes. On the other hand, if the conversion makes l false positive predictions then the apple tasting algorithm also makes at least $l/2 = \sqrt{T}/2$ mistakes. In general, the two cases will lead to different bounds, and we can only guarantee that the apple tasting complexity is at least the smaller of the two bounds.

Lemma 8 *Let X be a finite set, and $\mathcal{F}$ be a class of functions from X to $\{0,1\}$. If M_-, M_+ are positive integers such that for any standard model algorithm B, either $L_-(B,\mathcal{F}) \geq M_-$ or $L_+(B,\mathcal{F}) \geq M_+$, then for all nonnegative integers T,*

$$ALC(\mathcal{F},T) \geq \frac{1}{2}\min\left\{M_-\left\lfloor\frac{T}{M_- + M_+ - 1}\right\rfloor, M_+\right\}.$$

Proof: Set $k = \lfloor T/(M_+ + M_- - 1)\rfloor$. The choice of M_+ and M_- ensures that for all apple tasting algorithms A there exists a $\sigma_A \in X^*$ and $f_A \in \mathcal{F}$ such that either

$$M_+ \leq L_+(\text{CONSERVE}(\text{APtoST}(A,k)),\sigma_A,f_A)$$

or

$$M_- \leq L_-(\text{CONSERVE}(\text{APtoST}(A,k)),\sigma_A,f_A).$$

If

$$\begin{aligned}L_+(\text{CONSERVE}(\text{APtoST}(A,k)),\sigma_A,f_A) + L_-(\text{CONSERVE}(\text{APtoST}(A,k)),\sigma_A,f_A)\\ > M_- + M_+ - 1,\end{aligned}$$

then truncate σ_A so that

$$\begin{aligned}L_+(\text{CONSERVE}(\text{APtoST}(A,k)),\sigma_A,f_A) + L_-(\text{CONSERVE}(\text{APtoST}(A,k)),\sigma_A,f_A)\\ = M_- + M_+ - 1.\end{aligned}$$

It will still be the case that either

$$M_+ \leq L_+(\text{CONSERVE}(\text{APtoST}(A,k)),\sigma_A,f_A)$$

or

$$M_- \leq L_-(\text{CONSERVE}(\text{APtoST}(A,k)),\sigma_A,f_A).$$

By Lemma 6, for each apple tasting algorithm there is a σ'_A of length at most $M_+ + M_- - 1$ such that either

$$M_+ \leq L_+(\mathrm{APtoST}(A,k), \sigma'_A, f_A)$$

or

$$M_- \leq L_-(\mathrm{APtoST}(A,k), \sigma'_A, f_A).$$

Applying Lemma 5 (recall that $T \geq k(M_+ + M_- - 1)$) shows that for each algorithm A, either $M_+ \leq 2\mathrm{AL}(A, \mathcal{F}, T)$ or $kM_- \leq 2\mathrm{AL}(A, \mathcal{F}, T)$. Because this holds for every apple tasting algorithm A,

$$\frac{1}{2}\min(M_+, kM_-) \leq \inf_A \mathrm{AL}(A, \mathcal{F}, T) = \mathrm{ALC}(\mathcal{F}, T)$$

as desired. □

It turns out that with a little bit of work the lower bound of Lemma 8 can be into the $\sqrt{M_- T}$ form appearing in our upper bounds. When T is very large or very small, simple arguments lead to lower bounds for the first and third phases. These results are combined in the following theorem that gives a generic apple tasting lower bound in a form matching our upper bounds.

Theorem 9 *Let X be a finite set, $\mathcal{F}$ be a class of functions from X to $\{0,1\}$, and M_-, M_+ be nonnegative integers such that for any standard model algorithm B, either $L_-(B, \mathcal{F}) \geq M_-$ or $L_+(B, \mathcal{F}) \geq M_+$. If $T \leq \min\{M_+, M_-\}$ then*

$$ALC(\mathcal{F}, T) \geq T/2. \tag{5}$$

If $T \geq \min\{M_+, M_-\}$ then

$$ALC(\mathcal{F}, T) \geq \frac{1}{2}\min\left\{\frac{1}{2}\sqrt{M_- T}, M_+\right\}. \tag{6}$$

Proof: If either M_+ or M_- is zero, then the theorem holds trivially. For the remainder of the proof we assume that both M_+ and M_- are positive. If $T \leq \min\{M_+, M_-\}$ then by Lemma 6, for every standard model algorithm B there is a sequence σ_B of length T and $f_B \in \mathcal{F}$ causing B to make T mistakes. In particular, for every apple tasting algorithm A, there is a sequence σ_A of length T and function $f_A \in \mathcal{F}$ where the standard model algorithm $\mathrm{APtoST}(A, 1)$ makes T mistakes. Therefore, by Lemma 5, for every apple tasting algorithm A we have $\mathrm{AL}(A, \mathcal{F}, T)$ is at least $T/2$, giving $\mathrm{ALC}(\mathcal{F}, T) \geq T/2$.

We now use a case analysis (and the precondition $T \geq \min\{M_+, M_-\}$) to show (6). By the above reasoning,

$$\mathrm{ALC}(\mathcal{F}, T) \geq \frac{1}{2}\min\{M_+, M_-\}. \tag{7}$$

For the first case we assume that $T \leq 4M_-$, so that $\frac{1}{2}\sqrt{M_- T} \leq M_-$ and Inequality 7 implies

$$\mathrm{ALC}(\mathcal{F}, T) \geq \frac{1}{2}\min\{\frac{1}{2}\sqrt{M_- T}, M_+\}.$$

For the second case we assume that $M_- T > (M_- + M_+)^2$ so

$$\frac{M_- T}{M_- + M_+ - 1} - M_- > M_+$$

and Lemma 8 gives $\mathrm{ALC}(\mathcal{F}, T) \geq \frac{1}{2}M_+ \geq \frac{1}{2}\min\{\frac{1}{2}\sqrt{M_- T}, M_+\}$.

For the final case, assume $T > 4M_-$ and $\sqrt{M_-T} - M_- \leq M_+$. Set $b = \lceil \sqrt{M_-T} - M_- \rceil$ and apply Lemma 8 (with M_+ set to b) to obtain

$$\mathrm{ALC}(\mathcal{F}, T) \geq \frac{1}{2} \min \left\{ M_- \left\lfloor \frac{T}{M_- + b - 1} \right\rfloor, b \right\}.$$

Working on the first term of the min,

$$M_- \left\lfloor \frac{T}{M_- + b - 1} \right\rfloor \geq \frac{M_-T}{M_- + b - 1} - M_- \geq \sqrt{M_-T} - M_-.$$

Therefore, using $T > 4M_-$,

$$\mathrm{ALC}(\mathcal{F}, T) \geq \frac{1}{2}(\sqrt{M_-T} - M_-) \geq \frac{1}{4}\sqrt{M_-T} \geq \frac{1}{2}\min\{\frac{1}{2}\sqrt{M_-T}, M_+\}.$$

This completes the proof. □

The following relationship between apple tasting and one-sided learning follows immediately from Corollary 2 and Theorem 9. Recall that $\mathrm{L}^0_+(\mathcal{F})$ is the minimum $L_+(A, \mathcal{F})$ over those algorithms A with $L_-(A, \mathcal{F}) = 0$.

Theorem 10 *Choose a finite set X, and a set $\mathcal{F}$ of $\{0, 1\}$-valued functions defined on X. Then for all T,*

$$\frac{1}{2}\min\{\sqrt{T}/2, L^0_+(\mathcal{F})\} \leq ALC(\mathcal{F}, T) \leq L^0_+(\mathcal{F}).$$

We next show that if one is willing to accept a constant factor worsening of the bound, then (in all non-trivial cases) any apple tasting algorithm can be replaced with one obtained from a deterministic standard-model algorithm by means of our standard conversion. Details are given in Theorem 12. This result helps us to answer one of the questions motivating this research. It tells us that in some sense we don't have to worry about the apple tasting model when devising learning algorithms for specific tasks. If we have an adequate collection of standard model algorithms then these will suffice for apple tasting as well, provided we are willing to accept a constant factor worsening of the bounds over the best possible. In more detail, suppose we manage to come up with a toolbox of standard model algorithms that come within a constant factor of any achievable bounds (that is, there is a constant c such that for any standard model algorithm B there exists an algorithm B' in the toolbox such that $L_+(B', \mathcal{F}) \leq cL_+(B, \mathcal{F})$ and $L_-(B', \mathcal{F}) \leq cL_-(B, \mathcal{F})$). Then there exists a constant c' such that, for any achievable apple tasting bound, one gets an apple tasting algorithm with a performance guarantee of c' times the bound by applying our standard conversion to the appropriate algorithm from this toolbox.

We start with the following preliminary theorem.

Theorem 11 *Let X be a finite set and $\mathcal{F}$ be a class of functions from X to $\{0, 1\}$ with $|\mathcal{F}| \geq 2$. Let A be any apple tasting algorithm and T be any integer such that $AL(A, \mathcal{F}, T) < T/2$. If $k = \lfloor T / \lfloor 2AL(A, \mathcal{F}, T) + 1 \rfloor \rfloor$ then*

$$L_+(CONSERVE(APtoST(A, k)), \mathcal{F}) \leq 2AL(A, \mathcal{F}, T)$$

and

$$L_-(CONSERVE(APtoST(A, k)), \mathcal{F}) \leq \frac{12AL(A, \mathcal{F}, T)^2}{T}.$$

Proof: Let $l = \lfloor 2\mathrm{AL}(A, F, T) + 1 \rfloor$. Note that l is the least integer strictly greater than $2\mathrm{AL}(A, \mathcal{F}, T)$. Thus $T \geq l$, implying $k \geq 1$. Note also that our assumption regarding T trivially implies $T \geq 1$, and that our definitions of k and l imply $kl \leq T$.

Because $kl \leq T$, Lemma 5 implies that for any $\sigma \in X^l$ and $f \in \mathcal{F}$ we have

$$L_+(\mathrm{APtoST}(A, k), \sigma, f) + kL_-(\mathrm{APtoST}(A, k), \sigma, f) \leq 2\mathrm{AL}(A, \mathcal{F}, T)$$

implying

$$L_+(\mathrm{APtoST}(A, k), \sigma, f) + L_-(\mathrm{APtoST}(A, k), \sigma, f) \leq 2\mathrm{AL}(A, \mathcal{F}, T) < l.$$

Thus Lemma 7 implies that

$$\begin{aligned} &L_+(\mathrm{CONSERVE}(\mathrm{APtoST}(A, k)), \mathcal{F}) + L_-(\mathrm{CONSERVE}(\mathrm{APtoST}(A, k)), \mathcal{F}) \\ &\qquad \leq 2\mathrm{AL}(A, \mathcal{F}, T) < l. \end{aligned}$$

This gives the first inequality of the theorem. It also implies that

$$L_-(\mathrm{CONSERVE}(\mathrm{APtoST}(A, k)), \mathcal{F}) \leq 2\mathrm{AL}(A, \mathcal{F}, T).$$

We prove the second inequality by considering two cases. If $T < 2l$ then $T \leq 4\mathrm{AL}(A, \mathcal{F}, T) + 1$, and we have

$$\begin{aligned} L_-(\mathrm{CONSERVE}(\mathrm{APtoST}(A, k))) &\leq 2\mathrm{AL}(A, \mathcal{F}, T) \\ &\leq 2\mathrm{AL}(A, \mathcal{F}, T)(4\mathrm{AL}(A, \mathcal{F}, T) + 1)/T \\ &\leq 12\mathrm{AL}(A, \mathcal{F}, T)^2/T. \end{aligned}$$

(Here we use the fact that $\mathrm{AL}(A, \mathcal{F}, T) \geq 1/2$ when $|\mathcal{F}| \geq 2$ and $T \geq 1$.)

For the second case, $T \geq 2l$. Thus

$$k = \left\lfloor \frac{T}{l} \right\rfloor \geq \frac{2}{3} \frac{T}{2\mathrm{AL}(A, F, T) + 1} \geq \frac{T}{6\mathrm{AL}(A, F, T)}$$

where the last inequality uses the fact that $\mathrm{AL}(A, \mathcal{F}, T) \geq 1/2$. Recall from the proof of the first inequality that $\mathrm{CONSERVE}(\mathrm{APtoST}(A, k))$ makes strictly fewer than l mistakes. Thus Lemma 6 implies that for any $\sigma \in X^*$ and $f \in \mathcal{F}$ there is a σ' of length at most l such that

$$L_-(\mathrm{APtoST}(A, k), \sigma', f) = L_-(\mathrm{CONSERVE}(\mathrm{APtoST}(A, k)), \sigma, f).$$

Applying Lemma 5 yields

$$kL_-(\mathrm{CONSERVE}(\mathrm{APtoST}(A, k)), \sigma, f) = kL_-(\mathrm{APtoST}(A, k), \sigma', f) \leq 2\mathrm{AL}(A, \mathcal{F}, T).$$

Thus

$$L_-(\mathrm{CONSERVE}(\mathrm{APtoST}(A, k)), \sigma, f) \leq 2\mathrm{AL}(A, \mathcal{F}, T)/k \leq 12\mathrm{AL}(A, \mathcal{F}, T)^2/T,$$

as desired.

□

The following theorem shows that our transformations are very effective. In particular, if we start with an apple tasting algorithm A, convert it to a standard model algorithm B, and then convert back into an apple tasting algorithm A', the performance of algorithm A' is at most nine times worse than the performance of algorithm A. The only cases not covered by this theorem are the trivial cases in which the target class contains only one function or the apple tasting algorithm performs no better than random coin flips.

Theorem 12 *Let X be a finite set and $\mathcal{F}$ be a class of functions from X to $\{0,1\}$ such that $|\mathcal{F}| \geq 2$. Let A be any apple tasting algorithm, T be any positive integer for which $AL(A,\mathcal{F},T) < T/2$, and $c \geq 1$ be a real-valued constant. Then for any standard model algorithm B such that $L_+(B,\mathcal{F}) \leq 2cAL(A,\mathcal{F},T)$ and $L_-(B,\mathcal{F}) \leq 12cAL(A,\mathcal{F},T)^2/T$, we have*

$$AL\left(STtoAP\left(B, \frac{12cAL(A,\mathcal{F},T)^2}{T}, T\right), \mathcal{F}, T\right) \leq 9cAL(A,\mathcal{F},T).$$

Proof:
A straightforward application of Theorem 1 shows that for any standard model algorithm B with $L_+(B,\mathcal{F}) \leq 2c\mathrm{AL}(A,\mathcal{F},T)$ and $L_-(B,\mathcal{F}) \leq 12c\mathrm{AL}(A,\mathcal{F},T)^2/T$ we have

$$\mathrm{AL}\left(STtoAP\left(B, \frac{12c\mathrm{AL}(A,\mathcal{F},T)^2}{T}, T\right), \mathcal{F}, T\right) \leq 9c\mathrm{AL}(A,\mathcal{F},T).$$

□

The following corollary shows that one can obtain nearly optimal algorithms for the apple tasting setting by transforming standard model algorithms.

Corollary 13 *Let X be a finite set and $\mathcal{F}$ be a class of functions from X to $\{0,1\}$. Then there is a standard model algorithm B such that for any positive integer T, there is a nonnegative integer k such that*

$$AL\left(STtoAP\left(B,k,T\right), \mathcal{F}, T\right) \leq 9ALC(\mathcal{F},T).$$

Proof: Applying Theorem 11, if $k = \lfloor T/\lfloor 2\mathrm{AL}(A,\mathcal{F},T)+1\rfloor\rfloor$, the algorithm CONSERVE(APtoST(A,k)) satisfies the constraints on B of Theorem 12 with $c=1$. □

4 General bounds for apple tasting

In this section we derive bounds on $\mathrm{ALC}(\mathcal{F},T)$ that depend only on $|\mathcal{F}|$ and T. The upper bound bound given in Theorem 14 provides a useful general upper bound on the complexity of learning various natural classes. We also give a lower bound, in Theorem 15 and Corollary 16, showing that our upper bound is within a constant factor of the best possible bound of this form. Let

$$\mathrm{ALC}(n,T) = \max_{\mathcal{F}:|\mathcal{F}|=n} \mathrm{ALC}(\mathcal{F},T)$$

denote the best possible such upper bound.

We obtain these results from mistake bounds for standard-model algorithms, via the conversions between apple-tasting and standard model algorithms developed in Section 3. To obtain optimal apple-tasting bounds, we must consider standard-model bounds that distinguish between false positive and false negative mistakes. In the companion paper [HLL] we look at the trade off between false positive and false negative mistakes. There we obtain upper and lower bounds on the total loss of the learner if the loss for each false positive mistake is some arbitrary $a > 0$ and the loss for each false negative mistake is some arbitrary $b > 0$. (Note that the special case $a = b = 1$ corresponds to the standard mistake bound model). There we also examine how many false positive mistakes must be made if the allowed number of false negative mistakes is constrained (or equivalently, the number of false negative mistakes that must be made if the number of false positive mistakes is constrained), and show that these two generalizations of the standard model are closely related. The main results from the companion paper that we need here are given below in Lemmas 17 and 18.

We next state the main results of this section, Theorems 14 and 15, and Corollary 16, as well as the lemmas we need from the companion paper [HLL]. The remainder of this section presents the proofs of these theorems.

Our upper bound on $\mathrm{ALC}(n,T)$ is given by the following theorem.

Theorem 14 *For $n \geq 1$,*

$$ALC(n,T) \leq \min\left\{T/2, 8\sqrt{\frac{T\ln n}{\ln(1+\frac{T}{\ln n})}}, (n-1)\right\}.$$

The next theorem and its corollary give our lower bound. The lower bound of Corollary 16 matches the upper bound of the Theorem 14 to within a constant factor. The theorem states a slightly stronger result that will be useful for analyzing concrete classes in Section 5.

Theorem 15 *For any set X, and any nonempty, amply splittable set $\mathcal{F}$ of functions from X to $\{0,1\}$,*

$$ALC(\mathcal{F},T) \geq \min\left\{\frac{T}{2}, \frac{1}{8}\sqrt{\frac{T\ln|\mathcal{F}|}{\ln(1+\frac{T}{\ln|\mathcal{F}|})}}, \frac{|\mathcal{F}|-1}{2\sqrt{2}}\right\}.$$

As it is easy to find amply splittable classes of cardinality n this theorem immediately implies the following corollary.

Corollary 16

$$ALC(n,T) \geq \min\left\{\frac{T}{2}, \frac{1}{8}\sqrt{\frac{T\ln n}{\ln(1+\frac{T}{\ln n})}}, \frac{n-1}{2\sqrt{2}}\right\}.$$

The main results of [HLL] that we use are stated in the following two lemmas. These give upper and lower bounds on how well we can trade-off between false positive and false negative mistakes.

Lemma 17 ([HLL]) *Choose a set X and a class $\mathcal{F}$ of $\{0,1\}$-valued functions defined on X. Let $n = |\mathcal{F}|$. For any real $b \geq 1$ there is a (standard) learning algorithm B for $\mathcal{F}$ such that the number M_+ of false positive mistakes made by B, and the number M_- of false negative mistakes made by B always satisfy*

$$M_+ + bM_- \leq \frac{2b\ln n}{\ln(1+b)}.$$

Lemma 18 ([HLL]) *Choose a set X and a positive integer n. Choose an amply splittable[3] set $\mathcal{F}$ of n functions defined on X. Choose $b \geq 1$. Then for any (standard) learning algorithm B for $\mathcal{F}$ there exists a sequence σ of elements of X and $f \in \mathcal{F}$ such that*

$$L_+(B,\sigma,f) + bL_-(B,\sigma,f) \geq \min\left\{(n-1), \frac{b\ln n}{2\ln(1+b)}\right\}.$$

The bound of Lemma 17 trivially implies the following which is more useful for our purposes.

[3]Recall that a class $\mathcal{F}$ is amply splittable if for every subclass $\mathcal{F}' \subseteq \mathcal{F}$ and every integer $0 < k < |\mathcal{F}'|$ there is an $x \in X$ such that $|\{f \in \mathcal{F}' : f(x) = 0\}| = k$.

Lemma 19 *Choose a set X and a set $\mathcal{F}$ of $\{0,1\}$ valued functions defined on X. Let $n = |\mathcal{F}|$. For any $b \geq 1$ there is a (standard) learning algorithm A for $\mathcal{F}$ such that the number M_+ of false positive mistakes made by A, and the number M_- of false negative mistakes made by A always satisfy*

$$\begin{aligned} M_+ &\leq \frac{2b \ln n}{\ln(1+b)} \\ M_- &\leq \frac{2 \ln n}{\ln(1+b)}. \end{aligned}$$

We now turn to the technical details of applying these lemmas to prove the theorems of this section. We begin with the upper bound on $\mathrm{ALC}(n, T)$.

We will need the following easily verified lemma.

Lemma 20 *For all $x \geq 0$,*

$$\ln(1+\sqrt{x}) \geq \frac{\ln(1+x)}{2}.$$

Now we are ready to for the proof of the upper bound theorem.

Proof of Theorem 14: The $T/2$ bound is achieved by the algorithm which predicts randomly. The $n-1$ bound is achieved by the algorithm which predicts 1 on x_t whenever there is an $f \in \mathcal{F}$ such that $f(x_t) = 1$ and f is consistent with the previous examples. If $T < (e-1)\ln n$ then

$$8\sqrt{\frac{T \ln n}{\ln(1+\frac{T}{\ln n})}} \geq (8/\sqrt{e-1})T \geq T/2,$$

so we assume hereafter that $T \geq (e-1)\ln n$.

Choose any function class $\mathcal{F}$ for which $|\mathcal{F}| = n$. Combining Corollary 2 and Lemma 19, for all $b \geq 1$,

$$\mathrm{ALC}(\mathcal{F}, T) \leq \frac{2b \ln n}{\ln(1+b)} + 2\sqrt{\frac{2T \ln n}{\ln(1+b)}}.$$

Suppose

$$b = \sqrt{\frac{T}{\ln n} \ln\left(1 + \frac{T}{\ln n}\right)}.$$

As $T \geq (e-1)\ln n$, $b \geq 1$, and therefore

$$\begin{aligned} \mathrm{ALC}(\mathcal{F}, T) &\leq \frac{2\sqrt{(T \ln n)\ln\left(1+\frac{T}{\ln n}\right)}}{\ln(1+\sqrt{\frac{T}{\ln n}\ln\left(1+\frac{T}{\ln n}\right)})} + 2\sqrt{\frac{2T \ln n}{\ln(1+\sqrt{\frac{T}{\ln n}\ln\left(1+\frac{T}{\ln n}\right)})}} \\ &\leq \frac{2\sqrt{(T \ln n)\ln\left(1+\frac{T}{\ln n}\right)}}{\ln(1+\sqrt{\frac{T}{\ln n}})} + 2\sqrt{\frac{2T \ln n}{\ln(1+\sqrt{\frac{T}{\ln n}})}} \\ &\leq \frac{4\sqrt{(T \ln n)\ln\left(1+\frac{T}{\ln n}\right)}}{\ln(1+\frac{T}{\ln n})} + 4\sqrt{\frac{T \ln n}{\ln\left(1+\frac{T}{\ln n}\right)}} \qquad \text{(Lemma 20)} \\ &= 8\sqrt{\frac{T \ln n}{\ln\left(1+\frac{T}{\ln n}\right)}}. \end{aligned}$$

Since $\mathcal{F}$ was chosen arbitrarily, this completes the proof. □

Next, we turn to the proof of the lower bound. We will make use of the following lemma, which is a direct consequence of Lemma 18.

Lemma 21 *Choose a set X and a positive integer n. Choose an amply splittable set $\mathcal{F}$ of n functions defined on X. Choose $b \geq 1$ for which $\frac{b \ln n}{2 \ln(1+b)} \leq n-1$. Then for any (standard) learning algorithm B for $\mathcal{F}$ there exists a sequence σ of elements of X and $f \in \mathcal{F}$ such that*

$$\begin{array}{rrcl} & L_+(B,\sigma,f) & \geq & \frac{b \ln n}{4 \ln(1+b)} \\ or & L_-(B,\sigma,f) & \geq & \frac{\ln n}{4 \ln(1+b)}. \end{array}$$

We will also use the following.

Lemma 22 ([HLL]) *Choose a set X and a positive integer n. Choose an amply splittable set $\mathcal{F}$ of n functions defined on X. Then for any (standard) learning algorithm B for $\mathcal{F}$ there exists a sequence σ of elements of X and $f \in \mathcal{F}$ such that $L_+(B,\sigma,f) \geq (n-1)$ or $L_-(B,\sigma,f) \geq 1$.*

In addition, we will use of a couple of technical lemmas, whose proofs are omitted.

Lemma 23 *If $f : [1,\infty) \to \mathbf{R}$ is defined by*

$$f(x) = \frac{x}{\ln(1+x)},$$

then f is increasing over its domain.

Lemma 24 *For all $x \geq 2$,*

$$x^2 \geq 1 + \frac{2(x-1)^2}{\ln x}.$$

Now we are ready to give the proof of the lower bound theorem.

Proof of Theorem 15: Choose X and $\mathcal{F}$ as in the statement of the theorem. Let $n = |\mathcal{F}|$. If $n = 1$, the minimum is trivially 0. Assume $n \geq 2$.

We will show that, for each T, one of the terms of the minimum above is a lower bound on $\mathrm{ALC}(|\mathcal{F}|, T)$. This will establish the theorem. We will divide our analysis into cases, based on the relative size of T and n.

Case 1 ($T \leq \log_2 n$): The proof of this case uses techniques from [Lit89,LW94]. Let $\mathcal{F}'$ be a subclass of $\mathcal{F}$ of size $2^T \leq n$. Let A be an apple tasting algorithm for $\mathcal{F}$.

Construct a sequence $x_1, ..., x_T$ of elements of X and $\rho_1, ..., \rho_T \in \{0,1\}$ recursively as follows. Let $\mathcal{F}'_t = \{f \in \mathcal{F}' : f(x_i) = \rho_i \text{ for } i = 1, \ldots, t-1\}$ Let x_1 be such that

$$|\{f \in \mathcal{F}' : f(x_1) = 0\}| = |\{f \in \mathcal{F}' : f(x_1) = 1\}| = 2^{T-1}.$$

Let ρ_1 be such that the probability that A predicts $1 - \rho_1$ on the first trial is at least 1/2.

On trial t, choose x_t such that

$$|\{f \in \mathcal{F}'_t : f(x_t) = 0\}| = |\{f \in \mathcal{F}'_t : f(x_t) = 1\}| = 2^{T-t}.$$

By the ample splittability of F', it is easily verified by induction that this is possible. Choose ρ_t such that the probability over all of A's randomization on all trials, that

if the first $t-1$ trials of some fixed sequence have $(x_1,\rho_1),...,(x_{t-1},\rho_{t-1})$
then A predicts $1-\rho_t$ if given x_t on the tth trial

is at least $1/2$. Notice that the above probability does not depend on future trials. We have that $|\mathcal{F}'_{T+1}| = 1$, and therefore, there is a function f in $\mathcal{F}'$ for which $f(x_t) = \rho_t$ for all $t \le T$. By construction, the expected number of mistakes made by A on $(x_1,\rho_1),...,(x_T,\rho_T)$ is at least $T/2$, and therefore,

$$\mathrm{ALC}(\mathcal{F},T) \ge T/2.$$

Case 2 ($T \ge 2(n-1)^2$): Combining Theorem 9 and Lemma 22, we have

$$\begin{aligned}\mathrm{ALC}(\mathcal{F},T) &\ge \frac{1}{2}\min\left\{\frac{1}{2}\sqrt{T}, n-1\right\}\\ &\ge \frac{n-1}{2\sqrt{2}}.\end{aligned}$$

Case 3 ($\log_2 n \le T \le 2(n-1)^2$): Let

$$b = \sqrt{\frac{T}{\ln n}\ln\left(1+\frac{T}{\ln n}\right)}.$$

Since $T \ge \log_2 n$, we have that $b \ge 1$. Furthermore,

$$\begin{aligned} b &= \sqrt{\frac{T}{\ln n}\ln\left(1+\frac{T}{\ln n}\right)}\\ &\le \sqrt{\frac{2(n-1)^2}{\ln n}\ln\left(1+\frac{2(n-1)^2}{\ln n}\right)} \qquad \text{(since } T \le 2(n-1)^2\text{)}\\ &= 2(n-1)\sqrt{\frac{\ln\left(1+\frac{2(n-1)^2}{\ln n}\right)}{2\ln n}}\\ &\le 2(n-1),\end{aligned}$$

by Lemma 24. Thus

$$\begin{aligned}\frac{b\ln n}{2\ln(1+b)} &\le \frac{(n-1)\ln n}{\ln(1+2(n-1))} \qquad \text{(Lemma 23)}\\ &\le n-1.\end{aligned}$$

Applying Theorem 9 and Lemma 21, this implies

$$\mathrm{ALC}(\mathcal{F},T) \ge \frac{1}{2}\min\left\{\frac{1}{4}\sqrt{\frac{T\ln n}{\ln(1+b)}}, \frac{b\ln n}{4\ln(1+b)}\right\}. \tag{8}$$

We have

$$\frac{b\ln n}{\ln(1+b)} = \frac{\sqrt{\frac{T}{\ln n}\ln\left(1+\frac{T}{\ln n}\right)}\ln n}{\ln\left(1+\sqrt{\frac{T}{\ln n}\ln\left(1+\frac{T}{\ln n}\right)}\right)}$$

$$
\begin{aligned}
&\geq \frac{\sqrt{\frac{T}{\ln n}\ln\left(1+\frac{T}{\ln n}\right)}\ln n}{\ln\left(1+\frac{T}{\ln n}\right)} \\
&= \sqrt{\frac{T\ln n}{\ln\left(1+\frac{T}{\ln n}\right)}}
\end{aligned}
$$

and

$$
\begin{aligned}
\sqrt{\frac{T\ln n}{\ln(1+b)}} &= \sqrt{\frac{T\ln n}{\ln(1+\sqrt{\frac{T}{\ln n}\ln\left(1+\frac{T}{\ln n}\right)})}} \\
&\geq \sqrt{\frac{T\ln n}{\ln\left(1+\frac{T}{\ln n}\right)}}.
\end{aligned}
$$

Plugging into (8), in this case ($\log_2 n \leq T \leq 2(n-1)^2$)

$$
\mathrm{ALC}(\mathcal{F},T) \geq \frac{1}{8}\sqrt{\frac{T\ln n}{\ln(1+\frac{T}{\ln n})}}.
$$

This completes the proof. □

Theorem 14 and Corollary 16 give us upper and lower bounds on the apple tasting learning complexity of class $\mathcal{F}$ in terms of the cardinality of $\mathcal{F}$. In the next section we apply these results (along with the bounds of Section 3 to a number of natural concept classes).

5 Applications

The previous sections have presented ways to bound the apple tasting learning complexity of an arbitrary class $\mathcal{F}$ in terms of the numbers of false positive and false negative mistakes made by standard model algorithms for $\mathcal{F}$ or the cardinality of $\mathcal{F}$. In this section we use these results to obtain reasonable bounds on the apple tasting learning complexity of several natural concept classes, including:

- The class DIS_n of disjunctions of literals defined on n boolean variables,
- The class $\mathrm{MDIS}_{k,n}$ of monotone disjunctions of k of n boolean variables.
- The class CON_n of conjunctions of literals (possibly negated) defined on n boolean variables,
- The class MCON_n of monotone conjunctions of variables defined on n boolean variables,
- The class $\mathrm{MCON}_{k,n}$ of monotone conjunctions of k of n boolean variables.
- The class $\mathrm{SVAR}_n = \{f_i : i \leq n\}$ (single variable concepts) of functions of n boolean variables defined by $f_i(\vec{x}) = x_i$,
- The class $\mathrm{SMALL}_{k,n}$ of all functions f from $\{1,...,n\}$ to $\{0,1\}$ such that $|f^{-1}(1)| = k$,
- The class INSEG_n of indicator functions for initial segments of $\{1,...,n\}$, i.e., $\{\{1,...,k\} : 1 \leq k \leq n\}$.

The class of disjunctions includes the empty formula which is always false, and non-monotone disjunctions can contain both a literal and its negation (making the formula always true). Similarly, the empty conjunction is always true and a conjunction containing both a literal and its negation is always false. Our results for these classes are summarized in Table 1.

In this section we first determine $L^0_+(\mathcal{F})$ (the number of false positive mistakes that can be forced on any standard model algorithm for $\mathcal{F}$ which never makes a false negative mistake) for each of these classes so we can apply Theorem 10. We then show upper bounds (Theorem 30) on the apple tasting learning complexity of these classes. The final part of this section is devoted to obtaining lower bounds on the apple tasting complexity of the various classes.

The following lemma states trivial upper bounds on $L^0_+(\mathcal{F})$ for any class.

Lemma 25 *For any finite concept class $\mathcal{F}$ over a finite domain X,*

$$L^0_+(\mathcal{F}) \leq |\mathcal{F}| - 1.$$

Also,

$$L^0_+(\mathcal{F}) \leq \max\{|f^{-1}(0)| : f \in \mathcal{F}\}.$$

Proof: The first inequality follows from the fact that each mistake eliminates at least one function from the consistent functions and the second from the fact that each mistake reduces by at least 1 the number of points of the domain at which a false negative mistake can be made. This number is initially bounded by the number of points at which the hidden (target) function is 0. □

We now determine $L^0_+(\mathcal{F})$ for the all of the classes $\mathcal{F}$ defined above.

Theorem 26

$$\begin{aligned}
L^0_+(DIS_n) &= n+1 && (9)\\
L^0_+(MDIS_{k,n}) &= n-k && (10)\\
L^0_+(MCON_n) &= 2^n - 1 && (11)\\
L^0_+(CON_n) &= 2^n && (12)\\
L^0_+(MCON_{k,n}) &= \binom{n}{k} - 1 && (13)\\
L^0_+(SMALL_{k,n}) &= n-k && (14)\\
L^0_+(INSEG_n) &= n-1 && (15)\\
L^0_+(SVAR_n) &= n-1 && (16)
\end{aligned}$$

Proof: First note that any algorithm which never makes a false negative mistake must predict 1 whenever some concept (which has not been contradicted by previous examples) in $\mathcal{F}$ maps the current $x \in X$ to 1. Of course, if every $f \in \mathcal{F}$ which is consistent with the sample maps x to 0, every sensible algorithm will predict zero. These two properties define a standard model algorithm for each of our classes which makes the fewest possible false positive mistakes of those algorithms making no false negative mistakes. We now analyze this algorithm for each of our classes.

Except for (9) and (10), Lemma 25 implies that L^0_+ is in each case at most the stated value; with these exceptions, it remains only to demonstrate corresponding lower bounds.

For (9), we bound L^0_+ above by observing that the first false positive mistake eliminates n literals as candidates for inclusion in the hidden disjunction, and each subsequent false positive eliminates at least one additional literal. The upper bound of (10) is proved similarly.

For the lower bound for (9), consider the following sequence of examples.

$\vec{x}_t$	ρ_t
$(0,0,0,0,...,0)$	0
$(1,0,0,0,...,0)$	0
$(0,1,0,0,...,0)$	0
$\vdots$	$\vdots$
$(0,0,0,0,...,1)$	0

The algorithm predicts 1 on each instance of this sequence. Furthermore, the above sequence is consistent with the empty disjunction. The lower bound for (10) is proved similarly, except using the first $n-k$ examples from

$\vec{x}_t$	ρ_t
$(1,0,0,0,...,0)$	0
$(0,1,0,0,...,0)$	0
$\vdots$	$\vdots$
$(0,0,0,0,...,1)$	0

and remaining consistent with the disjunction of the last k variables.

For (11) suppose the elements of $\{0,1\}^n$ are given to the algorithm one-by-one as follows. First, the algorithm is given $(0,0,...,0)$. Next, the algorithm is given all vectors with a single 1 in them, then all vectors with two 1's, and so on, until the algorithm is given all those vectors with $1's$ in exactly $n-1$ places. Suppose further that the response on each trial is 0. The algorithm predicts 1 on each trial t, since the conjunction of those variables corresponding to the components of $\vec{x}_t$ which are 1 is satisfied by $\vec{x}_t$, but by no $\vec{x}_{t'}$ for $t' < t$. Even when all responses are 0, the sequence is still consistent with the conjunction of all variables. This gives the desired lower bound.

For (12) we use the previous example sequence together with an additional trial in which $x_{2^n} = (1,1,...,1)$ and $\rho_{2^n} = 0$. The algorithm must predicts 1 on the first $2^n - 1$ trials by the above argument, and it also predicts 1 on the last trial as well since that instance is consistent with the conjunction of all n variables. Note that the responses of 0 are consistent with any conjunction containing both a variable with its negation.

The argument for (13) is similar, but for the lower bound the adversary presents those vectors with exactly k 1s in them, responding 0 until the final such vector is presented.

For (14), the lower bound is obtained using an adversary that presents the elements of the domain in order. The adversary responds 0 the first $n-k$ trials.

The lower bound for (15) follows immediately from Lemma 22. The lower bound for (16) follows from this same lemma, and also as a special case of (13). $\Box$

We now upper bound the apple tasting learning complexity of the classes defined above. We will use the following lemma, which bounds the number of mistakes of the different kinds made by the WINNOW algorithm.

Lemma 27 ([Lit88]) *For all positive integers n, k, with $k \leq n$, all $\alpha > 1$, $\theta \geq 1/\alpha$, there is an algorithm WINNOW that makes its predictions in polynomial in n and k time such that*

$$L_+(\mathit{WINNOW}, \mathit{MDIS}_{k,n}) \leq \frac{n}{\theta} + k(\alpha - 1)(\log_\alpha \theta + 1).$$

and

$$L_-(\mathit{WINNOW}, \mathit{MDIS}_{k,n}) \leq k(\log_\alpha \theta + 1).$$

The following lemma also will be useful.

Lemma 28 ([HLL]) *Choose $b \geq 1$. Then there is a polynomial p such that, for any n, there is a (standard) learning algorithm B for $INSEG_n$ which makes each of its predictions in time $p(\log n)$ and such that the number M_+ of false positive mistakes made by B, and the number M_- of false negative mistakes made by B satisfy*

$$\begin{aligned} M_+ &\leq \frac{2b \ln n}{\ln(1+b)} \\ M_- &\leq \frac{2 \ln n}{\ln(1+b)}. \end{aligned}$$

We will also use a technical lemma, which can be verified using calculus.

Lemma 29 *For all $x > 0$ and $0 < y < 1$,*

$$1 + xy \geq (1+x)^y.$$

Now we are ready to prove the theorem stating our upper bounds on the apple tasting learning complexity of the classes defined above.

Theorem 30 *There is a polynomial p such that, for each positive n, k, and T, the following all hold:*

1. *There is an apple tasting algorithm A_1 that requires time $p(n,1)$ for each prediction, and*

$$AL(A_1, DIS_n, T) \leq n+1.$$

2. *There is an apple tasting algorithm A_2 that requires time $p(n,k)$ for each prediction, and*

$$AL(A_2, MDIS_{k,n}, T) \leq \sqrt{Tk} + (e-1)k \ln \frac{en}{\sqrt{Tk}} + 2\sqrt{Tk \ln \frac{en}{\sqrt{Tk}}}$$

3. *There is an apple tasting algorithm A_3 for which*

$$AL(A_3, MDIS_{k,n}, T) \leq 8\sqrt{\frac{T \ln \binom{n}{k}}{\ln(1 + T/(\ln \binom{n}{k}))}}$$

4. *There is an apple tasting algorithm A_4 that requires time $p(n,1)$ for each prediction, and*

$$AL(A_4, CON_n, T) \leq 2.24\sqrt{(n+1)T}.$$

5. *There is an apple tasting algorithm A_5 for which*

$$AL(A_5, CON_n, T) \leq 11.2\sqrt{\frac{nT}{\ln(1+T/n)}}.$$

6. *There is an apple tasting algorithm A_6 that requires time $p(n,k)$ for each prediction, and*

$$AL(A_6, MCON_{k,n}, T) \le 2.24\sqrt{ekT(1+\ln(n/k))}.$$

7. *There is an apple tasting algorithm A_7 for which*

$$AL(A_7, MCON_{k,n}, T) \le 8\sqrt{\frac{T\ln\binom{n}{k}}{\ln(1+T/(\ln\binom{n}{k}))}}$$

8. *There is an apple tasting algorithm A_8 that requires time $p(\log n, k)$ for each prediction, and*

$$AL(A_8, SMALL_{k,n}, T) \le 2\sqrt{kT}.$$

9. *There is an apple tasting algorithm A_9 for which*

$$AL(A_9, SMALL_{k,n}, T) \le \min\{2\sqrt{kT}, n-k\}.$$

10. *There is an apple tasting algorithm A_{10} that requires time $p(\log n, 1)$ for each prediction and*

$$AL(A_{10}, INSEG_n, T) \le 8\sqrt{\frac{T\ln n}{\ln(1+T/(\ln n))}}.$$

Proof:

1. The algorithm [Val84] that maintains a list of literals that might possibly be in the hidden disjunction, and predicts one whenever any of those literals evaluates to 1, makes at most $n+1$ false positive mistakes, and never makes a false negative mistake, as proved in Theorem 26. Since this algorithm is efficient, Statement 1 follows from Theorem 10.

2. Plugging into Lemma 27 with $\alpha = e$ and $\theta = \frac{n}{\sqrt{Tk}}$, and applying Corollary 2, yields this statement.

3. Statement 3 follows directly from Theorem 14.

4. For conjunctions, the dual of the algorithm of (1), which maintains a list of literals that could possibly be in the hidden conjunction, and predicts 0 whenever any of those literals evaluates to 0, makes at most $n+1$ mistakes. Using this algorithm and applying Corollary 3 yields Statement 4.

5. Applying Theorem 14, there is an algorithm A_5 such that

$$\begin{aligned}\mathrm{AL}(A_5, \mathrm{CON}_n, T) &\le 8\sqrt{\frac{T\ln 2^{2n}}{\ln(1+T/(\ln 2^{2n}))}}\\ &= 8\sqrt{\frac{nT\ln 4}{\ln(1+T/(n\ln 4))}}\end{aligned}$$

Applying Lemma 29 with $y = 1/\ln 4$, we get

$$\begin{aligned}\mathrm{AL}(A_5, \mathrm{CON}_n, T) &\leq 8\sqrt{\frac{nT(\ln 4)^2}{\ln(1+T/n)}} \\ &\leq 11.12\sqrt{\frac{nT}{\ln(1+T/n)}}.\end{aligned}$$

6. A transformation of the vanilla version of WINNOW [Lit88] learns conjunctions in time polynomial in n and k while making at most $ek(1+\ln(n/k))$ mistakes. Statement 6 then follows from an application of Corollary 3.

7. Statement 7 follows directly from Theorem 14.

8. Statement 8 follows from Corollary 2 and the fact that the algorithm which maintains a list of previously seen points x for which $f(x) = 1$, and predicts 0 whenever it encounters a point not of the list, makes at most k false negative mistakes, and never makes a false positive mistake.

9 Statement 9 follows from Theorem 26, Theorem 10, and Statement 8.

10. Statement 10 follows by substituting Lemma 28 for Lemma 19 in the proof of Theorem 14.

□

Notice that the order of quantifiers in the statement of the above theorem effectively enables the algorithms to "know" k and n. The need for this knowledge can be removed through the application of standard doubling tricks. Also, note that one can obtain upper bounds for non-monotone conjunctions and disjunctions of k of n boolean variables from the above using standard variable substitution techniques [Lit88,KLPV87]. The settings of α and θ in the proof of (2.) have not been optimized; they were chosen to give a readable bound.

We now turn to lower bounds, beginning with disjunctions.

Theorem 31 *For all n and T*

$$ALC(DIS_n, T) \geq \frac{1}{2}\min\{T, n\}.$$

Proof: This proof is a straightforward modification of the standard lower bound argument for disjunctions. Assume as a first case that $T \geq n$. The adversary gives the learner the following sequence of instances:

$$\begin{gathered}(1, 0, ..., 0)\\(0, 1, ..., 0)\\\vdots\\(0, 0, ..., 1).\end{gathered}$$

If each variable is included in the disjunction with probability 1/2, the expected number of mistakes (over the random choice of hidden disjunction and the algorithm's prediction) is trivially at least $n/2$. Therefore, there must be a particular choice of hidden disjunction for which the expected number of mistakes over the algorithm's randomization on the last n trials is at least $n/2$. The bound for the case in which $T \leq n$ can be trivially obtained by truncating the above example sequences to be length T. This completes the proof. □

Our next goal is a lower bound on $\mathrm{ALC}(\mathrm{CON}_n, T)$. To do this, we will first prove a lower bound on $\mathrm{ALC}(\mathrm{MCON}_n, T)$. We start with a lemma which relates the complexity of learning MCON_n to the complexity of learning s different copies of $\mathrm{MCON}_{n/s}$.

Lemma 32 *Choose positive integers n, T, and s, where $s \le n$. Then*

$$ALC(MCON_{k,n}, T) \ge sALC(MCON_{\lfloor k/s \rfloor, \lfloor n/s \rfloor}, \lfloor T/s \rfloor),$$

and

$$ALC(MCON_n, T) \ge sALC(MCON_{\lfloor n/s \rfloor}, \lfloor T/s \rfloor).$$

Proof: Let $X = \{0,1\}^n$ represent the possible truth values for the n variables. We split the variables into s groups, $G_1, G_2, \dots, G_s$, of $\lfloor n/s \rfloor$ variables each (ignoring any leftover variables). For each group G_j we define a corresponding subset $X_j \subset X$. An $\vec{x} \in \{0,1\}^n$ is in X_j if and only if $x_i = 1$ for every i *not* in G_j. Thus if $\vec{x} \in X_j$ then every variable set to false (zero) by $\vec{x}$ is in G_j. For $1 \le j \le s$, let $\mathcal{F}_j$ be the set of all conjunctions of at most $\lfloor k/s \rfloor$ variables in G_j. Each $\mathcal{F}_j$ is isomorphic to $\mathrm{MCON}_{\lfloor k/s \rfloor, \lfloor n/s \rfloor}$.

Our goal is to create for any apple tasting algorithm A a sequence $\sigma \in X^T$ and $f \in \mathrm{MCON}_{k,n}$ such that $AL(A, \sigma, f) \ge s\mathrm{ALC}(\mathrm{MCON}_{\lfloor k/s \rfloor, \lfloor n/s \rfloor}, \lfloor T/s \rfloor)$. To create this sequence we run A in s phases, where phase j has A learning some f_j from $\mathcal{F}_j$. In particular, we inductively construct from A a sequence $A_1, ..., A_s$ of apple tasting algorithms, a sequence $\sigma_1, ..., \sigma_s$ elements of $(\{0,1\}^n)^{\lfloor T/s \rfloor}$, and a sequence of functions $f_1 \in \mathcal{F}_1, ..., f_s \in \mathcal{F}_s$. Algorithm A_1 is just A, and $\sigma_1 \in X_1^{\lfloor T/s \rfloor}$ and $f_1 \in \mathcal{F}_1$ are chosen such that

$$\mathrm{AL}(A_1, \sigma_1, f_1) \ge \mathrm{ALC}(\mathcal{F}_1, \lfloor T/s \rfloor) = \mathrm{ALC}(\mathrm{MCON}_{\lfloor k/s \rfloor, \lfloor n/s \rfloor}, \lfloor T/s \rfloor).$$

For each $j > 1$ let A_j be the algorithm obtained by simulating A_{j-1} on σ_{j-1} and f_{j-1}, and then continuing A_{j-1} from that state on whatever trials A_j encounters. Thus each A_j is algorithm A starting from a different initial state[4]. As above, $\sigma_j \in G_j^{\lfloor T/s \rfloor}$ and $f_j \in \mathcal{F}_j$ are chosen such that

$$\mathrm{AL}(A_j, \sigma_j, f_j) \ge \mathrm{ALC}(\mathrm{MCON}_{\lfloor k/s \rfloor, \lfloor n/s \rfloor}, \lfloor T/s \rfloor).$$

Let σ be the concatenation of $\sigma_1, ..., \sigma_s$, and let f be the conjunction of $f_1, ..., f_s$. For each j and each $t \le \lfloor T/s \rfloor$, $f(\sigma_{j,t}) = f_j(\sigma_{j,t})$ (by the definition of X_j). Furthermore, for each set of random inputs a trivial induction shows that the sequence of predictions obtained by concatenating A_1's predictions on σ_1 with hidden function f_1, A_2's predictions on σ_2 with hidden function f_2, and so on, is the same as the sequence of predictions made by A on σ with hidden function f. Thus

$$\mathrm{AL}(A, \sigma, f) \ge s\mathrm{ALC}(\mathrm{MCON}_{\lfloor k/s \rfloor, \lfloor n/s \rfloor}, \lfloor T/s \rfloor).$$

Since σ is of length at most T and $f \in \mathrm{MCON}_{k,n}$ this completes the proof. □

For future reference, we record the following lemma, which can be proved in essentially the same manner.

[4]The result of simulating A_{j-1} on σ_{j-1} is likely to depend on the random inputs given to A_{j-1}, and thus the simulation will generally need to make calls to the random number generator. The choice of σ_j is made so the expected number of mistakes made by A_j is large, where the expectation is over the calls to the random number generator made for the simulation as well as any calls made to help determine predictions.

Lemma 33 *Choose positive integers n, T, and s, where $s \leq n$. Then*

$$ALC(MDIS_{k,n}, T) \geq s ALC(MDIS_{\lfloor k/s \rfloor, \lfloor n/s \rfloor}, \lfloor T/s \rfloor).$$

Proof: The proof is the same as that of Lemma 32, except that X_j is defined to consist of those $\vec{x} \in \{0,1\}^n$ for which $x_i = 0$ for each i not in G_j, ensuring that any variable set to true (one) by an $\vec{x} \in X_j$ is in G_j. The proof then goes through without modification by substituting MDIS for MCON throughout. □.

We will also make use of the following inequality.

Lemma 34 *For all $x \geq 1$,*

$$\log_2(1 + x \log_2(1 + x)) < 2 \log_2(1 + x).$$

Proof: Choose $x \geq 1$. Then

$$\begin{aligned} 1 + x \log_2(1 + x) &\leq (1 + x) \log_2(1 + x) \quad \text{(since } x \geq 1) \\ &< (1 + x)^2. \end{aligned}$$

Taking logs completes the proof. □

Now we prove the lower bound for monotone conjunctions. No attempt has been made to optimize the constants.

Theorem 35 *Choose positive integers n and T. Then*

$$ALC(MCON_n, T) \geq \min\left\{\frac{T}{6}, \frac{1}{16}\sqrt{\frac{nT}{\log_2(1 + T/n)}}, \frac{1}{8}2^n\right\}.$$

Proof: First, note that, by Theorem 26 and Theorem 10, for all n and T,

$$\text{ALC}(\text{MCON}_n, T) \geq \frac{1}{2} \min\left\{\frac{1}{2}\sqrt{T}, 2^n - 1\right\}. \tag{17}$$

To prove the lower bound claimed in this theorem, we divide our analysis into cases based on the relative sizes of T and n.

Case 1 ($T \leq n$): In this case a trivial adaptation of the proof of Theorem 31 shows that

$$\text{ALC}(\text{MCON}_n, T) \geq T/2.$$

Case 2 ($n < T \leq 3n$): Here

$$\begin{aligned} \text{ALC}(\text{MCON}_n, T) &\geq \text{ALC}(\text{MCON}_n, n) \\ &\geq n/2 \\ &\geq T/6 \end{aligned}$$

since $n \geq T/3$.

Case 3 ($T \geq 2^{2n-2}$): In this case, by (17),

$$\begin{aligned} \text{ALC}(\text{MCON}_n, T) &\geq \frac{1}{2} \min\left\{\frac{1}{2}\sqrt{T}, 2^n - 1\right\} \\ &\geq \frac{1}{2} \min\left\{\frac{1}{2}\sqrt{2^{2n-2}}, 2^n - 1\right\} \\ &\geq \frac{1}{8}2^n. \end{aligned}$$

Case 4 ($3n < T < 2^{2n-2}$): Let

$$s = \left\lfloor \frac{2n}{\log_2\left(1 + \frac{T}{n}\log_2\left(1+\frac{T}{n}\right)\right)} \right\rfloor .$$

We begin by establishing some facts about s that will be useful. First, we have

$$\begin{aligned}
T &\leq 2^{2n-2} \\
T\log_2(1+T) &\leq 2^{2n-2}\log_2(2^{2n}) \\
T\log_2(1+T) &\leq n(2^{2n-1}) \\
T\log_2\left(1+\frac{T}{n}\right) &\leq n(2^{2n}-1) \\
\log_2\left(1+\frac{T}{n}\log_2\left(1+\frac{T}{n}\right)\right) &\leq 2n \\
1 &\leq s.
\end{aligned}$$

We prove another useful fact through the following sequence of implications:

$$\begin{aligned}
3n &\leq T \\
4 &\leq 1+\frac{T}{n} \\
2 &\leq \log_2\left(1+\frac{T}{n}\right) \\
2 &\leq \log_2\left(1+\frac{T}{n}\log_2\left(1+\frac{T}{n}\right)\right) \\
\frac{2n}{\log_2\left(1+\frac{T}{n}\log_2\left(1+\frac{T}{n}\right)\right)} &\leq n \\
s &\leq n.
\end{aligned}$$

We will also use the fact implied by this that $s \leq T/2$. Since $s \geq 1$ and $s \leq n$, by Lemma 32,

$$\mathrm{ALC}(\mathrm{MCON}_n, T) \geq s\mathrm{ALC}(\mathrm{MCON}_{\lfloor n/s \rfloor}, \lfloor T/s \rfloor).$$

By (17), this implies

$$\mathrm{ALC}(\mathrm{MCON}_n, T) \geq \min\left\{\frac{s}{4}\sqrt{\lfloor T/s \rfloor}, \frac{s}{2}(2^{\lfloor n/s \rfloor}-1)\right\}. \tag{18}$$

First, since $s \leq T/2$,

$$\begin{aligned}
\frac{s}{4}\sqrt{\lfloor T/s \rfloor} &\geq \frac{1}{4}\sqrt{\frac{sT}{2}} \\
&\geq \frac{1}{4}\sqrt{\frac{nT}{2\log_2\left(1+\frac{T}{n}\log_2\left(1+\frac{T}{n}\right)\right)}} \qquad (s \geq 1) \\
&\geq \frac{1}{8}\sqrt{\frac{nT}{\log_2\left(1+\frac{T}{n}\right)}},
\end{aligned}$$

by Lemma 34. Now for the second term of (18). We have

$$\begin{aligned}
\frac{s}{2}(2^{\lfloor n/s \rfloor} - 1) &\geq \frac{s}{8}(2^{\frac{n}{s}}) \\
&\geq \frac{1}{8}\left\lfloor \frac{2n}{\log_2\left(1+\frac{T}{n}\log_2\left(1+\frac{T}{n}\right)\right)} \right\rfloor \sqrt{1+\frac{T}{n}\log_2\left(1+\frac{T}{n}\right)} \\
&\geq \frac{n}{8\log_2\left(1+\frac{T}{n}\log_2\left(1+\frac{T}{n}\right)\right)} \sqrt{1+\frac{T}{n}\log_2\left(1+\frac{T}{n}\right)} \quad (s \geq 1) \\
&\geq \frac{n}{16\log_2\left(1+\frac{T}{n}\right)} \sqrt{1+\frac{T}{n}\log_2\left(1+\frac{T}{n}\right)} \quad \text{(Lemma 34)} \\
&\geq \frac{1}{16}\sqrt{\frac{nT}{\log_2(1+T/n)}}.
\end{aligned}$$

This completes the proof. □

Next, we turn to intervals and SVAR$_n$.

Theorem 36 *For all $n \geq 1$ and T,*

$$\begin{aligned}
ALC(INSEG_n, T) &\geq \min\left\{\frac{T}{2}, \frac{1}{8}\sqrt{\frac{T\ln n}{\ln(1+\frac{T}{\ln n})}}, \frac{n-1}{2\sqrt{2}}\right\} \\
ALC(SVAR_n, T) &\geq \min\left\{\frac{T}{2}, \frac{1}{8}\sqrt{\frac{T\ln n}{\ln(1+\frac{T}{\ln n})}}, \frac{n-1}{2\sqrt{2}}\right\}
\end{aligned}$$

Proof: Follows immediately from Theorem 15 together with the fact that INSEG$_n$ and SVAR$_n$ are amply splittable function classes. □

Now we prove the lower bound for SMALL$_{k,n}$.

Theorem 37 *For all k, n, and T,*

$$ALC(SMALL_{k,n}, T) \geq \frac{1}{2}\min\{T, \sqrt{kT}/2, n-k\}.$$

Proof: The adversary which presents the elements of the domain in order, always answering the opposite of the learner, forces any learner to make at least k false negative mistakes or at least $n-k$ false positive mistakes in the standard model. The lower bound then follows immediately from Theorem 9. □

Now we turn to proving a lower bound for conjunctions and disjunctions of a bounded number of literals.

Theorem 38 *For all T, k, and n such that $1 < k \leq n/2$,*

$$ALC(MCON_{k,n}, T) \geq \frac{1}{2}\min\left\{\frac{1}{2}\sqrt{T}, \binom{n}{k} - 1\right\} \tag{19}$$

$$ALC(MDIS_{k,n}, T) \geq \frac{1}{2}\min\left\{\frac{1}{2}\sqrt{T}, n-k\right\} \tag{20}$$

$$\mathit{ALC}(\mathit{MCON}_{k,n},T) \geq \min\left\{\frac{T}{4}, \frac{1}{8\sqrt{2}}\sqrt{\frac{Tk\ln\lfloor n/k\rfloor}{\ln(1+\frac{T}{2k\ln\lfloor n/k\rfloor})}}, \frac{k(\lfloor n/k\rfloor-1)}{2\sqrt{2}}\right\} \tag{21}$$

$$\mathit{ALC}(\mathit{MDIS}_{k,n},T) \geq \min\left\{\frac{T}{4}, \frac{1}{8\sqrt{2}}\sqrt{\frac{Tk\ln\lfloor n/k\rfloor}{\ln(1+\frac{T}{2k\ln\lfloor n/k\rfloor})}}, \frac{k(\lfloor n/k\rfloor-1)}{2\sqrt{2}}\right\} \tag{22}$$

Proof: First, applying Theorem 26 and Theorem 10, we get (19) and (20).

To prove (21) and (22), we divide our analysis into cases, based on the relative sizes of T and k.

Case 1 ($T \leq k$): For $\mathrm{MCON}_{k,n}$, suppose the T instances are

$$\begin{array}{c}(0,1,1,...,1)\\(1,0,1,...,1)\\(1,1,0,...,1)\\\vdots\end{array}$$

until the Tth trial. For any algorithm A, if each of the first k variables is included in the conjunction with probability 1/2 (with variables added arbitrarily from the last $n-k$ to pad out the conjunction to include k variables), the expected number of mistakes (over the random choice of hidden function and the algorithm's randomness) is trivially at least $T/2$. Thus, there must exist a hidden conjunction of k variables such that the expected number of mistakes made by A with the above instances is at least $T/2$, establishing the bound in this case. This case is handled similarly for $\mathrm{MDIS}_{k,n}$, except using

$$\begin{array}{c}(1,0,0,...,0)\\(0,1,0,...,0)\\(0,0,1,...,0)\\\vdots\end{array}$$

and adding each of the first k variables to the disjunction with probability 1/2.

Case 2 ($k < T \leq 2k$): Here we have

$$\begin{aligned}\mathrm{ALC}(\mathrm{MCON}_{k,n},T) &\geq \mathrm{ALC}(\mathrm{MCON}_{k,n},k) \quad (\text{since } T > k)\\ &\geq k/2 \quad (\text{by Case 1})\\ &\geq T/4\end{aligned}$$

since $k \geq T/2$. This case is handled similarly for $\mathrm{MDIS}_{k,n}$.

Case 3 ($T > 2k$): Lemma 32 with $s = k$ together with the bound of Theorem 36 (and the observation that $\mathrm{MCON}_{1,n} = \mathrm{SVAR}_n$) gives us

$$\mathrm{ALC}(\mathrm{MCON}_{k,n},T) \geq k\min\left\{\frac{T}{4k}, \frac{1}{8\sqrt{2}}\sqrt{\frac{T\ln\lfloor n/k\rfloor}{k\ln(1+\frac{T}{2k\ln\lfloor n/k\rfloor})}}, \frac{\lfloor n/k\rfloor-1}{2\sqrt{2}}\right\},$$

and therefore

$$\mathrm{ALC}(\mathrm{MCON}_{k,n},T) \geq \min\left\{\frac{T}{4}, \frac{1}{8\sqrt{2}}\sqrt{\frac{Tk\ln\lfloor n/k\rfloor}{\ln(1+\frac{T}{2k\ln\lfloor n/k\rfloor})}}, \frac{k(\lfloor n/k\rfloor-1)}{2\sqrt{2}}\right\}.$$

Similarly, using Lemma 33, we get

$$\mathrm{ALC}(\mathrm{MDIS}_{k,n},T) \geq \min\left\{\frac{T}{4}, \frac{1}{8\sqrt{2}}\sqrt{\frac{Tk\ln\lfloor n/k\rfloor}{\ln(1+\frac{T}{2k\ln\lfloor n/k\rfloor})}}, \frac{k(\lfloor n/k\rfloor - 1)}{2\sqrt{2}}\right\},$$

completing the proof. □

These results show the most interesting parts of Table 1; the remainder of the table follows easily.

6 Apple Tasting and Random Draws

In this section we consider the apple tasting problem where an adversary picks a distribution from which the instances are drawn rather than selecting the sequence of instances directly. In this setting the adversary first chooses the hidden function from the function class and the distribution on the possible instances. In each trial: an instance is drawn at random from the distribution, the apple tasting algorithm predicts either 0 (reject) or 1 (accept), and then if the algorithm predicts 1, it is told whether or not the drawn instance is labeled 1 by the hidden function. As before, the loss of the algorithm is the expected number of incorrect predictions made on a series of trials.

We first give a simple way to convert standard algorithms into apple tasting algorithms. Instead of using mistake-bounded algorithms, this conversion uses algorithms for the standard model having low probability of error when the instances are drawn at random. It is well known that the VC-dimension [VC71] of the concept class helps characterize the difficulty of learning the concept class from random draws in the standard model. The VC-dimension [VC71] of concept class $\mathcal{F}$ over domain X is defined to be the size of the largest set $S = \{s_1, ..., s_k\} \subseteq X$ for which $\{(f(s_1), ..., f(s_k)) : f \in \mathcal{F}\} = \{0,1\}^{|S|}$. We use d to denote the VC-dimension of the class $\mathcal{F}$ under discussion.

Our simple conversion restricts learning to an initial period of $\sqrt{dT}$ trials. Since the apple tasting algorithm responds 1 on every trial of the initial period, without regard to the data received, the examples gained during this period are independent. The resulting apple tasting algorithms have expected mistake bounds that grow with $\sqrt{T}$, as before. To get this bound, however, we use a standard algorithm which is not in general computationally efficient. Thus we mention other methods from the standard model that can sometimes be used to construct efficient apple tasting algorithms (with differing bounds).

Our simple conversion fails to take advantage of algorithms that make few false-negative mistakes relative to the number of false-positive mistakes. For some function classes this is of little or no importance — we give lower bounds indicating that the conversion is within a constant factor of optimal for several natural concept classes. Certain other concept classes can be learned in the standard model by algorithms that make no false-negative mistakes. These (standard model) algorithms determine the unique largest $f \in \mathcal{F}$ consistent with the examples and expect to make a number $O(d\ln T)$ mistakes (see [Nat91,HSW90,HLW94]). We can easily take full advantage of this fact, obtaining algorithms with bounds that grow as $\ln T$ rather than $\sqrt{T}$. Taking full advantage of the possible trade-offs between false negatives and false positives remains an open problem. The analysis is complicated by the fact that the algorithm only sees the labels of some of the examples, and, under some obvious conversion schemes, these examples are not chosen independently.

Any of the algorithms described in previous sections can still be used under the stronger assumption that the examples are chosen independently at random. If the mistake bounds of a class for adversarially chosen examples are close to the best expected mistake bounds for random examples, then we can use the methods of the previous section to take advantage of the trade-off between false-negative and false-positive mistakes. On the other hand, finite adversarial mistake bounds cannot be obtained for function classes such as rectangles defined over continuous domains, so the methods of the previous sections cannot be directly applied.

When the instances are drawn at random we use $\mathrm{RALC}(\mathcal{F}, T)$ instead of $\mathrm{ALC}(\mathcal{F}, T)$ to denote the expected total loss. Formally, for a set X and a class $\mathcal{F}$ of $\{0,1\}$-valued functions defined on X, we define $\mathrm{RALC}(\mathcal{F}, T)$ to be

$$\inf_A \left(\sup_D \sup_{f \in \mathcal{F}} \mathbf{E}_{\sigma \in D^T}(\mathrm{AL}(A, \sigma, f)) \right)$$

where D ranges over probability distributions on X.

6.1 Upper Bounds

The conversion simply samples the first $\sqrt{dT}$ instances (by predicting 1) and uses a normal PAC algorithm to produce a hypothesis with low expected error. This hypothesis is then used to predict on the rest of the examples, ignoring any additional feedback. Standard doubling techniques can be used when T is unknown (see Section 6.2).

We will make use of the 1-inclusion graph learning strategy given by Haussler, Littlestone, and Warmuth [HLW94]. This algorithm gives a mapping B from sequences of examples to hypotheses such that the expected error of the hypothesis is bounded as described in the following lemma.

Lemma 39 ([HLW94]) *There is a computable mapping B from sequences of elements in $X \times \{0,1\}$ to functions from X to $\{0,1\}$ with the following property. Choose any $f \in \mathcal{F}$. If for nonnegative integers T and $\sigma \in X^T$, we let $h_{B,\sigma} = B((\sigma_1, f(\sigma_1)), ..., (\sigma_T, f(\sigma_T)))$, then for all probability distributions D on X,*

$$\mathbf{E}_{\sigma \in D^T}(\mathbf{Pr}_{x \in D}(h_{B,\sigma}(x) \neq f(x))) \leq d/(T+1).$$

Using the mapping of Lemma 39 as indicated above yields the following theorem.

Theorem 40 *For any concept class $\mathcal{F}$ of VC-dimension d, when T is known,*

$$RALC(\mathcal{F}, T) \leq 2\sqrt{dT}.$$

Proof: Consider the algorithm which "predicts" 1 on the first $\lfloor \sqrt{dT} \rfloor$ instances to create the sequence of examples $(\sigma_1, f(\sigma_1)), ..., (\sigma_{\lfloor \sqrt{dT} \rfloor}, f(\sigma_{\lfloor \sqrt{dT} \rfloor}))$. The apple tasting algorithm then passes this sequence of examples to algorithm B, which produces a hypothesis h having expected error at most $d/(\lfloor \sqrt{dT} \rfloor + 1)$. The apple tasting algorithm then uses h to generate its predictions on the remaining trials. Trivially, for any distribution D, the expected number of mistakes is bounded by

$$\begin{aligned} \lfloor \sqrt{dT} \rfloor + (T - \lfloor \sqrt{dT} \rfloor) \left(\frac{d}{\lfloor \sqrt{dT} \rfloor + 1} \right) &\leq \sqrt{dT} + T\left(\frac{d}{\sqrt{dT}}\right) \\ &= 2\sqrt{dT}, \end{aligned}$$

completing the proof. □

Unfortunately, the mapping B of Lemma 39 cannot in general be computed efficiently. We briefly discuss a method for constructing apple tasting algorithms that in some cases leads to efficient algorithms. It uses a consistency oracle for the concept class as described in the following lemma.

Lemma 41 ([HLW94]) *Suppose B is a mapping from sequences of elements in $X \times \{0,1\}$ to $\mathcal{F}$ such that the function output by B is consistent with B's input sequence. Choose $f \in \mathcal{F}$. For nonnegative integer T and $\sigma \in X^T$, let $h_{B,\sigma} = B((\sigma_1, f(\sigma_1)), ..., (\sigma_T, f(\sigma_T)))$, then for all probability distributions D on X,*

$$\mathbf{E}_{\sigma \in D^T}(\mathbf{Pr}_{x \in D}(h_{B,\sigma}(x) \neq f(x))) \in O((d/T)\log(T/d)).$$

We can apply this lemma using the same conversion as above to show that if there is an algorithm which efficiently finds a function in $\mathcal{F}$ consistent with a given sample, then there is an efficient apple tasting algorithm for $\mathcal{F}$ with an expected mistake bound of $O(\sqrt{dT}\log(T/d))$.

6.2 When T is unknown

If the number of trials in the sequence is not known in advance, then we cannot simply sample the first $\sqrt{dT}$ instances. We must interleave the sampling and prediction processes. This leads to bounds of the same form as those in Theorem 40, but with a slightly increase in the constant (from 2 to 3).

Here σ is the sample set used for predicting and t is the trial index. In the following, $h_{B,\sigma}(x)$ is the hypothesis generated by the algorithm B of Lemma 39.

```
Algorithm $A_1$
$\sigma := \emptyset$;
for $t := 1$ to $\infty$ do
    get instance $x$;
    if $|\sigma| < \sqrt{td} - 1$
      then predict 1 and add sample to $\sigma$ (explore)
      else predict $h_{B,\sigma}(x)$ (exploit)
end do;
```

Note that whenever the algorithm exploits at trial t, the current size of σ is at least $\sqrt{td} - 1$. Also, by induction, after each trial t, $|\sigma| < \sqrt{td}$. Thus, if the algorithm is run for T trials, the final size of σ is at most $\sqrt{Td}$.

Theorem 42 *For any concept class $\mathcal{F}$ of VC-dimension d, when T is not known in advance,*

$$RALC(\mathcal{F}, T) \leq 3\sqrt{dT}.$$

Proof: Consider the performance of Algorithm A_1 on a sequence of T trials.

$$\begin{aligned}
\sup_{f \in \mathcal{F}} \mathbf{E}_{\sigma \in D^T}(\mathrm{AL}(A_1, \sigma, f)) &\leq \text{number of explorations} + \text{expected mistakes exploiting} \\
&\leq \sqrt{Td} + \sum_{t=1}^{T} \frac{d}{\sqrt{td}} \quad \text{(Lemma 39)} \\
&= \sqrt{Td} + \sum_{t=1}^{T} \sqrt{\frac{d}{t}} \\
&\leq \sqrt{Td} + \sqrt{d}\int_{t=0}^{T} \sqrt{1/t} \\
&= \sqrt{Td} + \sqrt{d}(2\sqrt{T}) \\
&= 3\sqrt{Td}
\end{aligned}$$

□

6.3 Lower bounds

To obtain lower bounds we consider k-STONS$_n$, the class of n^k functions mapping $\{1, 2, \ldots, kn\}$ to $\{0, 1\}$ defined as follows. A function f is in k-STONS$_n$ iff for each $i \in \{0, 1, ..., k-1\}$, $|f^{-1}(1) \cap \{in+1, in+2, ..., in+n\}| = 1$. Thus, the domain is partitioned into k parts each containing n elements and k-STONS$_n$ contains those functions which map exactly one element from each part to 1. Class k-STONS$_n$ is a subclass of SMALL$_{k,kn}$, can be viewed as k copies of STONS$_n$, and has VC-dimension k when $n > 1$. The following theorem shows that there are natural classes $\mathcal{F}$ where RALC$(\mathcal{F}, T)$ grows as $\sqrt{Td}$. Our lower bound argument is similar to a lower bound argument used in [HLW94].

Theorem 43 *If $T = 4n^2k$, then*

$$RALC(k\text{-}STONS_n, T) \geq \frac{1}{6}\sqrt{kT} - \frac{k}{3}$$

Proof: Let A be any prediction algorithm for k-STONS$_n$. We lower bound the expected number of mistakes made by algorithm A when the hidden function is selected uniformly at random from the n^k functions in k-STONS$_n$ and the distribution D on the instances is the uniform distribution on $\{1, 2, \ldots, kn\}$. Clearly, RALC(k-STONS$_n, T$) is at least this lower bound.

To learn a hidden function from k-STONS$_n$, algorithm A must learn k nearly independent copies of STONS$_n$. The copies are not quite independent tasks as the number of instances falling within each copy are dependent functions of the sequence of instances.

Consider a sequence of T instances. We say an instance is *sparse* wrt the sequence if it occurs in the sequence less than $n/2$ times. We say a copy of STONS$_n$ is *rich* if none of its n instances are sparse, i.e. all of its instances occur at least $n/2$ times in the sequence.

The proof proceeds by showing the following for a randomly selected hidden function and randomly generated sequence of instances.

- The expected number of sparse points is small, so most of the k copies of STONS$_n$ are rich.
- Any algorithm expects to make a large number of mistakes on each copy of STONS$_n$ that is rich.
- Therefore, the total expected number of mistakes made by A is large.

Claim: When $T = 4n^2k$ instances of k-STONS$_n$ are drawn uniformly at random, the expected number of copies of STONS$_n$ that are rich is at least $33k/49$.

Proof of claim: Let SPARSE$_i$ for $i \in \{1, 2, \ldots, kn\}$ be the event that instance i is sparse in the randomly drawn sequence of instances.

We will make use of the following consequence of Chebyshev's inequality[5]:

Consider a sequence of j Bernoulli trials with success probability p, and let S_j be the random variable denoting the number of successes. Then:

$$\mathbf{Pr}\left(\left|\frac{S_j}{j} - p\right| \geq \epsilon\right) \leq \frac{p(1-p)}{j\epsilon^2} \leq \frac{p}{j\epsilon^2}.$$

[5]See equation (5) on page 47 of A.N. Shiryayev, **Probability** from the series Graduate Texts in Mathematics by Springer-Verlag, 1984 (Trans. R.P. Boas).

We apply the Chebyshev bound as follows, with n_i denoting the number of times instance i appears in the sequence.

$$\begin{aligned}
\mathbf{Pr}(\textsc{Sparse}_i) &= \mathbf{Pr}\left(\frac{n_i}{T} \leq \frac{n}{2T}\right) \\
&= \mathbf{Pr}\left(\frac{1}{nk} - \frac{n_i}{T} \geq \frac{1}{nk} - \frac{n}{2T}\right) \\
&\leq \mathbf{Pr}\left(\left|\frac{n_i}{T} - \frac{1}{nk}\right| \geq \frac{1}{nk} - \frac{n}{2T}\right) \\
&\leq \frac{1/(nk)}{T(\frac{1}{nk} - \frac{n}{2T})^2} \\
&= \frac{1}{4n^3k^2(\frac{7}{8nk})^2} \\
&= \frac{16}{49n}
\end{aligned}$$

As the probability of $\textsc{Sparse}_i$ for each instance $i \in \{1,..,nk\}$ is at most $\frac{16}{49n}$, the expected number of sparse instances is bounded by $\frac{16k}{49}$. Since each non-rich copy of STONS_n contains at least one sparse instance, the expected number of rich copies of STONS_n is at least $k - \frac{16k}{49} = 33k/49$, proving the claim.

Claim: If A' is any prediction algorithm for STONS_n and σ is any sequence of instances where each of the n instances occur at least $n/2$ times then

$$\mathbf{E}_{f \in \mathrm{STONS}_n, r \in U}(\mathrm{AL}(A', \sigma, f)) \geq \frac{n-1}{2},$$

where r is the randomization of A'.

Proof of Claim: Let σ be any sequence of STONS_n instances where each instance occurs at least $n/2$ times and A' be any prediction algorithm for STONS_n. Fix the random input to A' so that A' is deterministic. As we will prove the bound on the loss for each random input r to A', the bound holds for the expectation over r.

Let $I = \{i_1, i_2, \ldots, i_j\}$ be the set of distinct instances on which Algorithm A' predicts 1 when given the sequence σ and the feedback 0 after every 1 prediction. Let f be the randomly chosen hidden function and x_f the single point mapped to 1 by f. Finally, let $\textsc{Hit}$ be the event that $x_f \in I$ and $\textsc{Miss}$ be the event that $x_f \notin I$,

We first consider $\mathbf{E}_{f \in \mathrm{STONS}_n}(\mathrm{AL}(A', \sigma, f)|\textsc{Hit})$. Since the hidden function f is chosen at random, each of the j instances in I are equally likely to be x_f. If the first instance on which A' predicts 1 is x_f then A' may make no mistakes. If the second instance on which A' predicts 1 is x_f then A' makes at least one false positive mistake, and so on. With this reasoning,

$$\mathbf{E}_{f \in \mathrm{STONS}_n}(\mathrm{AL}(A', \sigma, f)|\textsc{Hit}) \geq \frac{1}{j}\sum_{i=1}^{j}(i-1) = \frac{j-1}{2}.$$

We now consider $\mathbf{E}_{f \in \mathrm{STONS}_n}(\mathrm{AL}(A', \sigma, f)|\textsc{Miss})$. In this case the algorithm makes at least j incorrect 1 predictions and incorrectly predicts 0 on each x_f instance. Thus

$$\mathbf{E}_{f \in \mathrm{STONS}_n}(\mathrm{AL}(A', \sigma, f)|\textsc{Miss}) \geq j + (n/2).$$

Now, $\mathbf{Pr}_{f\in \mathrm{STONS}_n}(\textsc{Hit}) = \frac{j}{n}$ and $\mathbf{Pr}_{f\in \mathrm{STONS}_n}(\textsc{Miss}) = \frac{n-j}{n}$, so

$$\begin{aligned}\mathbf{E}_{f\in \mathrm{STONS}_n}(\mathrm{AL}(A',\sigma,f)) &\geq \frac{j}{n}\frac{j-1}{2} + \frac{n-j}{n}(j+(n/2)) \\ &= \frac{n^2+nj-j^2-j}{2n}.\end{aligned}$$

For $j \in [0,n]$, the right-hand-side is minimized when $j = n$ (as can be seen by taking derivatives). Thus for all $j \in \{0,...,n\}$, we have the bound

$$\mathbf{E}_{f\in \mathrm{STONS}_n}(\mathrm{AL}(A',\sigma,f)) \geq \frac{n-1}{2},$$

proving the claim.

We are now ready to bound RALC(k-STONS$_n, T$) for $T = 4n^2k$. It is easy to see, since the choice of the target and the randomization of the algorithm are independent of the choice of the instances, that RALC(k-STONS$_n, T$) is at least the expected number of rich copies times the expected mistake bound per rich copy. Obtaining these values from the claims,

$$\begin{aligned}\mathrm{RALC}(k\text{-}\mathrm{STONS}_n, T) &\geq \frac{33k}{49}\frac{n-1}{2} \\ &\geq \frac{nk}{3} - \frac{k}{3} \\ &\geq \sqrt{\frac{4n^2k^2}{36}} - \frac{k}{3} \\ &\geq \frac{1}{6}\sqrt{kT} - \frac{k}{3}\end{aligned}$$

□

The following lower bound shows that for $n \geq 3$ and $T \leq 4n^2k$, the class k-STONS$_n$ has a mistake bound that is $\Omega(\sqrt{kT})$.

Corollary 44 *If $n \geq \sqrt{\frac{T}{4k}} \geq 3$ then RALC(k-STONS$_n, T$) $\geq \frac{5}{72}\sqrt{kT}$.*

Proof: Let $n' = \lfloor\sqrt{\frac{T}{4k}}\rfloor$, and $T' = 4n'^2k$. Since $n' \leq n$, $T' \leq T$, and k-STONS$_{n'}$ can be trivially embedded in k-STONS$_n$, we have (using Theorem 43)

$$\begin{aligned}\mathrm{RALC}(k\text{-}\mathrm{STONS}_n, T) &\geq \mathrm{RALC}(k\text{-}\mathrm{STONS}_{n'}, T') \\ &\geq \frac{1}{6}\sqrt{kT'} - \frac{k}{3}.\end{aligned}$$

Now $4n'^2k = T' \leq T < 4(n'+1)^2k$, so

$$\frac{T'}{T} > \frac{4n'^2k}{4(n'+1)^2k}$$

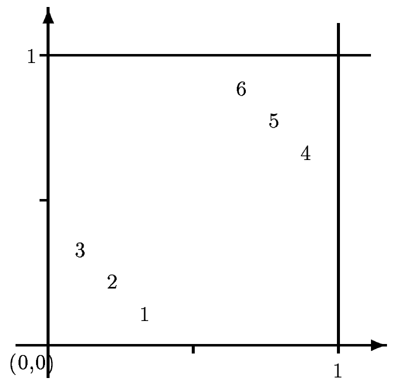

Figure 1: Mapping of the 2-STONS$_3$ domain, $\{1, 2, 3, \; 4, 5, 6\}$, into the domain of RECT$_2$.

Recalling that $n' \geq 3$, we conclude that $T' > 9T/16$ and $\sqrt{T'} \geq 3\sqrt{T}/4$. Plugging this into the above bound yields

$$\text{RALC}(k\text{-STONS}_n, T) \geq \frac{1}{8}\sqrt{kT} - \frac{k}{3}.$$

Note also that $\sqrt{kT} \geq \sqrt{4k^2n'^2} \geq 6k$. Thus $\sqrt{kT}/18 \geq k/3$ and

$$\text{RALC}(k\text{-STONS}_n, T) \geq \left(\frac{1}{8} - \frac{1}{18}\right)\sqrt{kT} = \frac{5}{72}\sqrt{kT}.$$

□

Using the lower bounds on k-STONS$_n$, we can show lower bounds for learning the more natural classes of k-dimensional halfspaces and k-dimensional axis-parallel rectangles that grow as $\sqrt{T}$.

Corollary 45 *Consider the class SMALL$_{k,kn}$ (defined in Section 5), the class RECT$_k$ of k-dimensional axis-parallel rectangles (RECT$_k$ = $\{f_{\bar{l},\bar{h}} : \bar{l}, \bar{h} \in \mathbf{R}^k\}$ where $f_{\bar{l},\bar{h}}(\bar{x}) = 1$ iff for all $1 \leq i \leq k$, $l_i \leq x_i \leq h_i$), and the class of HALF$_k$ of k-dimensional halfspaces (HALF$_k$ = $\{f_{\bar{v},z} : z \in \mathbf{R}, v \in \mathbf{R}^k\}$ where $f_{\bar{v},z}(\bar{x}) = 1$ iff $\bar{x} \cdot \bar{v} \geq z$). We have the following bounds on these classes. If $\sqrt{\frac{T}{4k}} \geq 3$ then*

$$\begin{aligned} \mathit{RALC}(\mathit{RECT}_{2k}, T) &\geq \frac{5}{72}\sqrt{2kT} \\ \mathit{RALC}(\mathit{HALF}_{2k}, T) &\geq \frac{5}{72}\sqrt{kT} \end{aligned}$$

and if $n \geq \sqrt{\frac{T}{4k}} \geq 3$ then

$$\mathit{RALC}(\mathit{SMALL}_{k,nk}, T) \geq \frac{5}{72}\sqrt{kT}.$$

Proof: For the last bound, simply note that the class SMALL$_{k,nk}$ contains k-STONS$_n$, and thus the bound of Corollary 44 applies to SMALL$_{k,nk}$ as well.

For the first bound, we show how to simulate the class $2k$-STONS$_n$ (for any n) with the class RECT$_{2k}$. Each $2k$-STONS$_n$ instance $(i-1)n + j$ (for $1 \leq i \leq 2k$ and $1 \leq j \leq n$) is mapped to RECT_{2k} instance $(x_1, x_2, ..., x_{2k})$. Figure 1 shows this mapping for 2-STONS_3. If $i \leq k$ then $x_{2i} = j/3n$, $x_{2i-1} = (n-j+1)/3n$, and the other components are set to 1/2. If $i > k$ then $x_{2(i-k)-1} = (3n-j)/3n$, $x_{2(i-k)} = (2n+j-1)/3n$, and the other components are set to 1/2. Now each $2k$-STONS_n concept containing instances $j_1, n+j_2, 2n+j_3, \ldots (2k-1)n+j_{2k}$ is mapped to the RECT_{2k} concept $f_{\bar{l},\bar{h}}$ where

$$\bar{l} = ((n-j_1+1)/3n, j_1/3n, (n-j_2+1)/3n, j_2/3n, \ldots (n-j_k+1)/3n, j_k/3n),$$

and

$$\bar{h} = ((3n-j_{k+1})/3n, (2n+j_{k+1}-1)/3n, (3n-j_{k+2})/3n, (2n+j_{k+2}-1)/3n, \\ \ldots (3n-j_{2k})/3n, (2n+j_{2k}-1)/3n).$$

It is easily, if tediously, verified that for any possible target f in $2k$-STONS_n, for the corresponding target $g \in \mathrm{RECT}_{2k}$ described above, the value of f on some instance is that same as the value of g on the instance's transformation. Thus, we can view a copy of $2k$-STONS_n as being embedded in RECT_{2k}.[6]

For the bound on HALF_{2k}, we embed k-STONS_n into HALF_{2k}. Here, the k-STONS_n instance $(i-1)n+j$ (where here, $1 \leq i \leq k$ and $1 \leq j \leq n$) is mapped to the instance $(x_1, x_2, ..., x_{2k})$ defined by letting $x_{2i} = j/n$, and $x_{2i-1} = \sqrt{1 - j^2/n^2}$, with all other components set to 0. A concept in k-STONS_n containing instances $j_1, n+j_2, 2n+j_3, \ldots (k-1)n+j_k$ is mapped to the halfspaces defined by $\bar{w} \cdot \bar{x} \geq 1$, where

$$\bar{w} = (\sqrt{1 - \frac{j_1^2}{n^2}}, \frac{j_1}{n}, \sqrt{1 - \frac{j_2^2}{n^2}}, \frac{j_2}{n}, \ldots \sqrt{1 - \frac{j_k^2}{n^2}}, \frac{j_k}{n})$$

Choose a k-STONS_n instance $(i-1)n+j$, let $\bar{x}$ be defined as above. Assume for now that $i = 1$. (Other cases can be handled similarly.) Choose a concept f in k-STONS_n, and let $j_1, n+j_2, 2n+j_3, \ldots (k-1)n+j_k$ be the elements f maps to 1. Let $\bar{w}$ be defined from f as above. Then

$$\begin{aligned}
&\bar{w} \cdot \bar{x} \geq 1 \\
&\Leftrightarrow (\sqrt{1 - \tfrac{j_1^2}{n^2}}, \tfrac{j_1}{n}, \sqrt{1 - \tfrac{j_2^2}{n^2}}, \tfrac{j_2}{n}, \ldots \sqrt{1 - \tfrac{j_k^2}{n^2}}, \tfrac{j_k}{n}) \cdot (\sqrt{1 - \tfrac{j^2}{n^2}}, \tfrac{j}{n}, 0, 0, \ldots 0, 0) \geq 1 \\
&\Leftrightarrow (\sqrt{1 - \tfrac{j_1^2}{n^2}}, \tfrac{j_1}{n}, 0, 0, \ldots 0, 0) \cdot (\sqrt{1 - \tfrac{j^2}{n^2}}, \tfrac{j}{n}, 0, 0, \ldots 0, 0) \geq 1 \\
&\Leftrightarrow j_1 = j,
\end{aligned}$$

since for unit length vectors $\bar{u}$ and $\bar{v}$, $\bar{u} \cdot \bar{v} \geq 1$ iff $\bar{u} = \bar{v}$.

This completes the proof. □

The bounds of Theorems 43 and 40 match to within a small constant factor on the function classes for which the lower bound applies.

Certain function classes can be effectively learned in the standard on-line model by algorithms that make no false-negative mistakes [Nat91,HSW90,HLW94]. These are classes for which there is a unique largest member consistent with any finite set of examples generated by a class member. As these algorithms are always correct when they predict 0, they can be used without modification as apple tasting algorithms. For such a function class $\mathcal{F}$, known results [Nat91,HSW90,HLW94] imply that $\mathrm{RALC}(\mathcal{F}, T)$ is $O(d \ln T)$. The class INSEG_n (defined in Section 5) is such a class; since it has VC-dimension one, we have $\mathrm{RALC}(\mathrm{INSEG}_n, T)$ is $O(\ln T)$. In contrast, Table 1 shows that

[6]Pitt and Warmuth used this sort of embedding argument to prove hardness results for computationally efficient learning [PW90].

ALC(INSEG_n, T) is $\Omega(\sqrt{T})$ for $T < n^2$. Thus this class has a vastly improved expected mistake bound when the instances are drawn at random.

In contrast to this case, there are cases in which learning is nearly as hard given independent random examples as it is when an adversary chooses the examples. For example, the lower bounds of Theorem 43 (for randomly drawn examples) nearly match the upper bounds on $\text{SMALL}_{k,kn}$ in Table 1 (for adversarial sequences). In particular the dependence on $\sqrt{T}$ seen in the mistake bounds for adversarial sequences also occurs for some classes when the instances are drawn at random.

7 Conclusion

We have studied the apple tasting model, where the information obtained by the learner is affected by the learner's actions. This model contains an interesting exploration/exploitation tradeoff, since effective learning in the apple tasting model requires that some instances be accepted for the sole purpose of obtaining the necessary data to improve future performance. Our main result is a series of algorithm transformations that relate the apple tasting model to a slight extension of the standard on-line model where false negative and false positive mistakes are counted separately.

One of the questions we address is whether a developer of learning algorithms needs to design special algorithms for the apple tasting model. That is, suppose that one has already developed an optimal (or nearly optimal) algorithm for learning a class of functions in the standard model. Is there a routine modification of this algorithm so that its performance in the apple tasting setting is close to optimal? We do not obtain an affirmative answer to this question for the standard model as it has normally been studied. However, when the examples are generated by an adversary and the standard model is extended so that false positive and false negative mistakes have different costs, then we do obtain an affirmative answer (see Theorem 12 for the details). Of the specific applications we have looked at, the function class for which manipulating the trade-off between false positive and false negative mistakes is most important is the class of disjunctions. When learning disjunctions in the apple tasting model it is desirable to use a standard model algorithm that makes no false negative mistakes at all – this results in an apple tasting algorithm whose bounds are independent of the number of trials. In the other cases, the improvement from paying attention to the trade-off amounts to a log factor.

It is often difficult to design efficient algorithms that fully respond to cost differences between false positive and false negative mistakes. Therefore, the bounds obtained for algorithms in the original standard model remain important. Further development of efficient algorithms for the generalized model (where false negatives and false positive mistakes have different costs) would be of interest.

Theorem 12 shows that, in all cases, transforming the appropriate standard model algorithm yields an apple tasting algorithm with mistake bounds that are within a constant factor of the best possible. This theorem is proven by converting an algorithm for the apple tasting model into a standard model algorithm and back while increasing the (expected) number of mistakes made by only a constant factor. However, the conversions do not fully address the issue of computational efficiency. When an efficient standard model algorithm is converted to an apple tasting algorithm, the resulting apple tasting algorithm is efficient. On the other hand, when we convert an apple tasting algorithm A into a standard model algorithm B, algorithm B must examine the probabilities that A will accept instances in different circumstances. It is conceivable that these probabilities might not be efficiently computable even when A itself is efficient. An obvious approach is to simulate several copies of algorithm A and thus obtain estimates of its prediction probabilities. Unfortunately, this approach seems to only postpone the difficulty until the (now randomized) standard-model algorithm is converted back into an apple tasting algorithm. Further research is needed.

A key feature of the apple tasting model is that effective learners must accept some instances solely in order to obtain data for improving future performance. One might expect that the probability of accepting an instance should depend on both how much information would be gained by finding out its classification and how likely it appears that "accept" is the proper action. Although detailed analysis of these two quantities might be necessary if one is to obtain optimal algorithms, we have shown that algorithms within a constant factor of optimal can be obtained when the probability of accepting an instance depends only on its classification by a standard-model algorithm. We actually do two things leading to the acceptance of additional instances. The first, increasing the cost of false negative mistakes, tends to cause the standard model algorithm to accept additional instances based on their identities. However, even after the cost is optimally chosen, our transformation involves accepting additional instances obliviously at random.

We briefly examined the apple tasting setting when the instances are drawn from an unknown distribution on the domain rather than selected by an adversary. In this case our transformation uses standard model algorithms with low probability of error. It turns out that we have not been able to exploit standard model algorithms which trade-off false positive and false negative mistakes as fully as in the case when an adversary directly selects the instances. The degree to which this trade-off can be exploited when the instances are randomly drawn remains an open problem.

A final important open problem is to investigate an extension of the apple tasting model where there is noise in the data. If the noise is mild enough so that mistake bounded algorithms exist, then they can be transformed into good apple tasting algorithms. It appears that additional ideas are needed when the noise corrupts an unbounded number of examples.

Acknowledgements

David Helmbold gratefully acknowledges the support of NSF grants CCR-9700201 and CCR-9704348. Nick Littlestone benefited from discussions of this model with Naoki Abe and Avrim Blum. This work was done while Phil Long was at UC Santa Cruz supported by a UC Santa Cruz Chancellor's dissertation-year fellowship, at Technische Universitaet Graz supported by a Lise Meitner Postdoctoral Fellowship from the Fonds zur Förderung der wissenschaftlichen Forschung (Austria), and at Duke supported by Air Force Office of Scientific Research grant F49620–92–J–0515.

References

[ACBFS95] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. *Proceedings of the 36th Annual Symposium on the Foundations of Computer Science*, 1995.

[Ang88] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[BEHW87] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

[Ber80] J. O. Berger. *Statistical Decision Theory and Bayesian Analysis.* Springer Verlag, 1980.

[BF72] J. M. Barzdin and R. V. Freivald. On the prediction of general recursive functions. *Soviet Mathematics-Doklady*, 13:1224–1228, 1972.

[BHL91] A. Blum, L. Hellerstein, and N. Littlestone. Learning in the presence of finitely many or infinitely many irrelevant attributes. *Proceedings of the 1991 Workshop on Computational Learning Theory*, 1991.

[Blu90a] A. Blum. Learning boolean functions in an infinite attribute space. *Proceedings of the 22nd ACM Symposium on the Theory of Computing*, 1990.

[Blu90b] A. Blum. Separating PAC and mistake-bound learning models over the boolean domain. *Proceedings of the 31st Annual Symposium on the Foundations of Computer Science*, 1990.

[Fel68] W. Feller. *An Introduction to Probability and its Applications*, volume 1. John Wiley and Sons, third edition, 1968.

[HLL] D. P. Helmbold, N. Littlestone, and P. M. Long. On-line learning with linear loss constraints. Submitted.

[HLL92] D. P. Helmbold, N. Littlestone, and P. M. Long. Apple tasting and nearly one-sided learning. *Proceedings of the 33rd Annual Symposium on the Foundations of Computer Science*, 1992.

[HLW94] D. Haussler, N. Littlestone, and M. K. Warmuth. Predicting $\{0,1\}$-functions on randomly drawn points. *Information and Computation*, 115(2):129–161, 1994. Preliminary version in FOCS'88.

[HSW90] D. Helmbold, R. Sloan, and M. K. Warmuth. Learning lattices and reversible, commutative regular languages. *Proceedings of the 1990 Workshop on Computational Learning Theory*, 1990.

[KLPV87] M. Kearns, M. Li, L. Pitt, and L. G. Valiant. On the learnability of Boolean formulae. *Proceedings of the 19th Annual Symposium on the Theory of Computation*, pages 285–295, 1987.

[Lit88] N. Littlestone. Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[Lit89] N. Littlestone. *Mistake Bounds and Logarithmic Linear-threshold Learning Algorithms.* PhD thesis, UC Santa Cruz, 1989.

[Lon97] P. M. Long. On-line evaluation and prediction using linear functions. *Proceedings of the 1997 Conference on Computational Learning Theory*, 1997.

[LW89] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Proceedings of the 30th Annual Symposium on the Foundations of Computer Science*, pages 256–261, 1989.

[LW94] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.

[Maa91] W. Maass. On-line learning with an oblivious environment and the power of randomization. *Proceedings of the 1991 Workshop on Computational Learning Theory*, pages 167–175, 1991.

[MT89] W. Maass and G. Turán. On the complexity of learning from counterexamples. *Proceedings of the 30th Annual Symposium on the Foundations of Computer Science*, pages 262–267, 1989.

[MT90] W. Maass and G. Turán. On the complexity of learning from counterexamples and membership queries. *Proceedings of the 31st Annual Symposium on the Foundations of Computer Science*, 1990.

[Nat91] B. K. Natarajan. *Machine Learning: A Theoretical Approach.* Morgan Kaufmann, San Mateo, California, 1991.

[PW90] L. Pitt and M. K. Warmuth. Prediction preserving reducibility. *Journal of Computer and System Sciences*, 41(3), 1990.

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

[VC71] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

$\mathcal{F}$	Efficient Algorithm	Inefficient Algorithm	Main Lower Bound Term	$\lvert\mathcal{F}\rvert$	$L_+^0(\mathcal{F})$
CON_n	$2.24\sqrt{nT}$	$11.2\sqrt{\frac{nT}{\ln(1+T/n)}}$	$\frac{1}{16}\sqrt{\frac{nT}{\ln(1+T/n)}}$	3^n+1	2^n
DIS_n	$n+1$	$n+1$	$\frac{n}{2}$	3^n	$n+1$
$\mathrm{MCON}_{k,n}$	$2.24\sqrt{ekT(1+\ln(n/k))}$	$8\sqrt{\frac{T\ln\binom{n}{k}}{\ln(1+T/(\ln\binom{n}{k}))}}$	see Theorem 37	$\binom{n}{k}$	$\binom{n}{k}-1$
$\mathrm{MDIS}_{k,n}$	$\sqrt{Tk}+(e-1)k\ln\frac{en}{\sqrt{Tk}}+2\sqrt{Tk\ln\frac{en}{\sqrt{Tk}}}$	$8\sqrt{\frac{T\ln\binom{n}{k}}{\ln(1+T/(\ln\binom{n}{k}))}}$	see Theorem 37	$\binom{n}{k}$	$n-k$
SVAR_n	$8\sqrt{\frac{T\ln n}{\ln(1+T/(\ln n))}}$	$8\sqrt{\frac{T\ln n}{\ln(1+T/(\ln n))}}$	$\frac{1}{8}\sqrt{\frac{T\ln n}{\ln(1+T/(\ln n))}}$	n	$n-1$
$\mathrm{SMALL}_{k,n}$	$2\sqrt{kT}$	$2\sqrt{kT}$	$\frac{1}{4}\sqrt{kT}$	$\binom{n}{k}$	$n-k$
INSEG_n	$8\sqrt{\frac{T\ln n}{\ln(1+T/(\ln n))}}$	$8\sqrt{\frac{T\ln n}{\ln(1+T/(\ln n))}}$	$\frac{1}{8}\sqrt{\frac{T\ln n}{\ln(1+T/(\ln n))}}$	n	$n-1$

Table 1: **Summary of apple tasting results for specific function classes**

The efficient algorithms all have running times bounded by a polynomial in the total length of the instances seen, even when k might grow with n. For those classes above the double line, each instance has size n and for those classes below the double line, each instance has size $\log n$.

The actual lower bounds are within a constant factor of $\min\{W, L_+^0(\mathcal{F})/2, T/2\}$ where W is the main lower bound term given in the table. Our lower bounds for $\mathrm{MCON}_{k,n}$ and $\mathrm{MDIS}_{k,n}$ are not of this form; see Theorem 37 for these bounds.

Similarly, our actual upper bounds may be obtained by replacing each algorithm's upper bound U with $\min\{U, L_+^0(\mathcal{F}), T/2\}$, except for the efficient $\mathrm{MCON}_{k,n}$ algorithm. Our upper bound for this algorithm is $\min\{U, T/2\}$ as the only algorithm that we know for this class that guarantees the $L_+^0(\mathcal{F})$ bound has a running time exponential in k.

The algorithm used to attain a particular upper bound depends on which term of the min applies in a given situation. When the $T/2$ term is least, it suffices to predict based on random coin flips.

When the least term is the bound given in the table, a standard-model algorithm is converted as described in the proof of Theorem 1. The following standard-model algorithms are used: the bounds containing $\ln T$ (except the $\mathrm{MDIS}_{k,n}$ bound) come from a general purpose trading algorithm from [HLL]. The bound independent of T (for DIS_n) comes from an algorithm making no false-negative mistakes described by Valiant [Val84]. The efficient algorithm used for $\mathrm{MCON}_{k,n}$ and $\mathrm{MDIS}_{k,n}$ is Winnow [Lit88]. The bounds for $\mathrm{SMALL}_{k,n}$ come from the algorithm that predicts 1 on points known to be 1, and 0 otherwise. When the smallest term in the min is $L_+^0(\mathcal{F})$, an algorithm is used that predicts 0 on points that must be 0, and 1 otherwise. This is straightforward except for $\mathrm{MCON}_{k,n}$, for which we do not have efficient algorithms to achieve this bound. Additional comments regarding efficient implementation for this case: For SVAR_n, the size of the class is the same as the size of each example, so efficient implementation is trivial. DIS_n is learned with the algorithm already given. For $\mathrm{SMALL}_{k,n}$ and STONS_n an appropriate search tree can be used to recall the points labeled 0 in time $O(\log n)$. For INSEG_n, it suffices to remember the smallest example known to be labeled 0.

Upper bounds for nonmonotone disjunctions and conjunctions of k of n boolean variables can be obtained from the bounds for $\mathrm{MDIS}_{k,n}$ and $\mathrm{MCON}_{k,n}$ using standard variable substitution techniques.