Littlestone Classes are Privately Online Learnable

Noah Golowich
MIT CSAIl
nzg@mit.edu

Roi Livni
Tel Aviv University
rlivni@tauex.tau.ac.il

Abstract

We consider the problem of online classification under a privacy constraint. In this setting a learner observes sequentially a stream of labelled examples (x_t, y_t), for $1 \leq t \leq T$, and returns at each iteration t a hypothesis h_t which is used to predict the label of each new example x_t. The learner's performance is measured by her regret against a known hypothesis class $\mathcal{H}$. We require that the algorithm satisfies the following privacy constraint: the sequence $h_1, \ldots, h_T$ of hypotheses output by the algorithm needs to be an (ϵ, δ)-differentially private function of the whole input sequence $(x_1, y_1), \ldots, (x_T, y_T)$. We provide the first non-trivial regret bound for the realizable setting. Specifically, we show that if the class $\mathcal{H}$ has constant Littlestone dimension then, given an oblivious sequence of labelled examples, there is a private learner that makes in expectation at most $O(\log T)$ mistakes – comparable to the optimal mistake bound in the non-private case, up to a logarithmic factor. Moreover, for general values of the Littlestone dimension d, the same mistake bound holds but with a doubly-exponential in d factor. A recent line of work has demonstrated a strong connection between classes that are online learnable and those that are differentially-private learnable. Our results strengthen this connection and show that an online learning algorithm can in fact be directly privatized (in the realizable setting). We also discuss an adaptive setting and provide a sublinear regret bound of $O(\sqrt{T})$.

1 Introduction

Privacy-preserving machine learning has attracted considerable attention in recent years, motivated by the fact that individuals' data is often collected to train statistical models, and such models can leak sensitive data about those individuals [13, 32]. The notion of *differential privacy* has emerged as a central tool which can be used to formally reason about the privacy-accuracy tradeoffs one must make in the process of analyzing and learning from data. A considerable body of literature on *differentially private machine learning* has resulted, ranging from empirical works which train deep neural networks with a differentially private form of stochastic gradient descent [1], to a recent line of theoretical works which aim to characterize the optimal sample complexity of privately learning an arbitrary hypothesis class [3, 11, 20].

Nearly all of these prior works on differentially private learning, however, are limited to the statistical learning setting (also known as the *offline* setting): this is the setting where the labeled data, (x_t, y_t), are assumed to be drawn i.i.d. from some unknown population distribution. This setting, while very well-understod and readily amenable to analysis, is unlikely to hold in practice. Indeed, the data (x_t, y_t) fed as input into the learning algorithm may shift over time (e.g., as a consequence of demographic changes in a population), or may be subject to more drastic changes which are *adaptive* to the algorithm's prior predictions (e.g., drivers' reactions to the recommendations of route-planning apps may affect traffic patterns, which influence the input data to those apps). For this reason, it is desirable to develop provable algorithms which make fewer assumptions on the data.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

In this work, we do so by studying the setting of *(private) online learning*, in which the sequence of data (x_t, y_t) is allowed to be arbitrary, and we also discuss a certain notion of privacy in a setting where it is even allowed to adapt to the algorithm's predictions in prior rounds. We additionally restrict our attention to the problem of classification, namely where the labels $y_t \in \{0, 1\}$; thus we introduce the problem of differentially private online classification, and prove the following results (see Section 3 for the exact setup):

- In the realizable setting with an oblivious adversary, we introduce a private learning algorithm which, for hypothesis classes of Littlestone dimension d (see Section 2.1) and time horizon T, achieves a mistake bound of $\tilde{O}(2^{O(2^d)} \cdot \log T)$, ignoring the dependence on privacy parameters (Theorem 4.1).
- In the realizable setting with an adaptive adversary, we show that a slight modification of the above algorithm achieves a mistake bound of $\tilde{O}(2^{O(2^d)} \cdot \sqrt{T})$ (Theorem 4.2).

We remark that no algorithm (even without privacy, allowing randomization, and in the oblivious adversary setting) can achieve a mistake bound of smaller than $\Omega(d)$ for classes of Littlestone dimension d [30, 33]. Therefore, a class of infinite Littlestone dimension cannot have any finite mistake bound, and the regret for any algorithm, for any time horizon T, is $\Omega(T)$. Thus, our results listed above, which show a mistake-bound (which is also the regret in the realizable setting) of $\tilde{O}_d(\sqrt{T})$ for classes of Littlestone dimension d, establish that in the realizable setting, finiteness of the Littlestone dimension is necessary and sufficient for online learnability ([31]) with differential privacy.

Recently it was shown by Alon et al. [3] and Bun et al. [11] (later to be improved by Ghazi et al. [20]) that finiteness of the Littlestone dimension is necessary and sufficient for private learnability in the *offline* setting, namely with i.i.d. data (and both in the realizable and agnostic settings). Since, as remarked above, the Littlestone dimension characterizes online learnability (even without privacy), this means that a binary hypothesis class is privately (offline) learnable if and only if it is online learnable. Our result thus strengthens this connection, showing that the equivalence also includes private *online* learnability (in the realizable setting).

1.1 Related work

A series of papers [15, 25, 21, 17, 2] has studied the problem of diferentially private online convex optimization, which includes specific cases such as private prediction from expert advice and, when one assumes imperfect feedback, private non-stochastic multi-armed bandits [35, 36, 18, 24]. These results show that in many regimes privacy is free for such problems: for instance, for the problem of prediction from the expert advice (with N experts), Agarwal and Singh [2] shows that an ϵ-differentially private algorithm (based on follow-the-regularized-leader) achieves regret of $O\left(\sqrt{T} + \frac{N \log^2 T}{\epsilon}\right)$, which matches the non-private regret bound of $O(\sqrt{T \log N})$ when $T \geq \tilde{\Omega}((N/\epsilon)^2)$. Our results can be seen as extending such "privacy is (nearly) free" results to the nonparametric setting where we instead optimize over an arbitrary class of finite Littlestone dimension. Our techniques are different from those of the above papers.

In addition to [11, 20] which establish private learning algorithms for classes with finite Littlestone dimension in the i.i.d. (offline) setting, there has been an extensive line of work on private learning algorithms in the offline setting: [29, 7, 5, 19] study the complexity of private learning with *pure differential privacy*, [26, 9, 10, 4] study the sample complexity of privately learning thresholds, and [27, 28, 6] study the sample complexity of privately learning halfspaces.

2 Preliminaries

In this section we introduce some background concepts used in the paper.

2.1 Online Learning

We begin by revisiting the standard setting of online-learning: We consider a sequential game between a *learner* and an *adversary*. Both learner and adversary know the sets $\mathcal{X}$ and $\mathcal{H}$. The game

proceeds for T rounds (again T is known) and at each round $t \leq T$, the adversary chooses a pair (x_t, y_t) and presents the learner with the example x_t. The learner then must present the adversary with a hypothesis (perhaps randomly) $h_t : \mathcal{X} \to \{0, 1\}$. h_t is not required to lie in $\mathcal{H}$[1]. Finally the adversary presents the learner with y_t, which the learner uses to update its internal state. The performance of the learner is measured by its regret which is its number of mistake vs. the optimal decision in hindsight:

$$\mathbb{E}\left[\sum_{t=1}^{T} 1[h_t(x_t) \neq y_t] - \min_{h^\star \in \mathcal{H}} \sum_{t=1}^{T} 1[h^\star(x_t) \neq y_t]\right]. \tag{1}$$

The adversary is said to be *realizable* if it presents the learner with a sequence of examples (x_t, y_t) so that there is some $h^\star \in \mathcal{H}$ so that for each $t \in [T]$, $h^\star(x_t) = y_t$. In the realizable setting, the regret simply counts the number of mistakes the learner makes. And we measure the performance by its *mistake bound*, namely the maximum, over all possible realizable adversaries, of

$$\mathbb{E}\left[\sum_{t=1}^{T} 1[h_t(x_t) \neq y_t]\right].$$

In the setting with an *agnostic* adversary, we do not require such $h^\star$ to exist; and we measure the learner by its (worst-case) *regret*, as in Eq. (1). In this paper we focus on the realizable setting; the (private) agnostic setting is left as an interesting direction for future work.

Additionally, we normally make a distinction between two types of adversaries: An *oblivious* adversary chooses its sequence in advance and at each iteration (x_t, y_t) is revealed to the learner. In the *adversarial* setting, the adversary may choose (x_t, y_t) as a function of the learner's previous choices: i.e. $h_1, \ldots, h_{t-1}$. This definition follows the standard setup of online learning (see [12] for example). We note though, that in the non-private setting of online binary classification, one can obtain results against an adversary that even gets to observe the learner's prediction at time-step t. However, we will simplify here by considering the more standard setting. It is interesting to find out if we can compete against such a strong adversary in the private setup.

Littlestone dimension We next turn to introduce the Littlestone dimension which is a combinatorial measure that turns out to characterize learnability in the above setting.

Let $\mathcal{H}$ be a class of hypotheses $h : \mathcal{X} \to \{0, 1\}$. To define the Littlestone dimension of $\mathcal{H}$, we first introduce *mistake trees*: a mistake tree of *depth* d is a complete binary tree, each of whose non-leaf nodes v is labeled by a point $x_v \in \mathcal{X}$, and so that the two out-edges of v are labeled by 0 and 1. We associate each root-to-leaf path in a mistake tree with a sequence $(x_1, y_1), \ldots, (x_d, y_d)$, where for each $i \in [d]$, the ith node in the path is labeled x_i and the path takes the out-edge from that node labeled y_i. A mistake tree is said to be *shattered* by $\mathcal{H}$ if for any root-to-leaf path whose corresponding sequence is $(x_1, y_1), \ldots, (x_d, y_d)$, there is some $h \in \mathcal{H}$ so that $h(x_i) = y_i$ for all $i \in [d]$. The Littlestone dimension of $\mathcal{H}$, denoted $\mathrm{Ldim}(\mathcal{H})$, is the depth of the largest mistake tree that is shattered by $\mathcal{H}$.

The Standard Optimal Algorithm (SOA) Suppose $\mathcal{H}$ is a binary hypothesis class with Littlestone dimension d. Littlestone [30] showed that there is an algorithm, called the *Standard Optimal Algorithm* (SOA), which, against an adaptive and realizable adversary, has a mistake bound of d; moreover, this is the best possible mistake bound. We will access the SOA as a black box. The underlying assumption we make is that given a realizable sequence $(x_1, y_1), \ldots, (x_T, y_T)$, the SOA makes at most $\mathrm{Ldim}(\mathcal{H})$ mistakes. We will also assume that whenever the algorithm SOA makes a mistake then it changes it state: namely, if the algorithm makes mistake on example t then $h_{t+1} \neq h_t$, this is in fact true for the SOA algorithm, but it can be seen that any algorithm with mistake bound can be modified to make sure this holds (simply by reiterating the mistake until the algorithm does change state). We refer the reader to [30, 33] for the specifics of it.

2.2 Differential Privacy

We next recall the standard notion of (ϵ, δ)–differential privacy:

[1]This setup is known as the *improper* learning problem. In the *proper* version of the problem, it is required that $h_t \in \mathcal{H}$ and we leave a study of proper private online learning for future work. (see [22] for a discssion on proper online learning in the non-private case

Definition 2.1 (Differential privacy)**.** Let n be a positive integer, $\epsilon, \delta \in (0,1)$, and $\mathcal{W}$ be a set. A randomized algorithm $A : (\mathcal{X} \times \{0,1\})^n \to \mathcal{W}$ is defined to be (ϵ, δ)-*differentially private* if for any two datasets $S, S' \in (\mathcal{X} \times \{0,1\})^n$ differing in a single example, and any event $\mathcal{E} \subset \mathcal{E}$, it holds that

$$\Pr[A(S) \in \mathcal{E}] \leq e^{\epsilon} \cdot \Pr[A(S') \in \mathcal{E}] + \delta.$$

Adaptive Composition The online nature of the problem naturally requires us to deal with adaptive mechanisms that query the data-base. We thus depict here the standard framework of adaptive querying, and we refer the reader to Dwork and Roth [13] for a more detailed exposition.

In this framework we assume a sequential setting, where at step t an adversary chooses two adjacent datasets S_t^1 and S_t^0, and a mechanism $M_t(S)$ from a class $\mathcal{F}$ and receives $z_t^b = M_t(S_t^b)$ for some $b \in \{0,1\}$ (where b does not depend on t).

Definition 2.2. We say that the family $\mathcal{F}$ of algorithms over databases satisfies (ϵ, δ)-*differential privacy under T-fold adaptive composition* if for every adversary A and event $\mathcal{E}$, we have

$$\Pr((z_1^0, \ldots, z_T^0) \in \mathcal{E}) \leq e^{\epsilon} \Pr((z_1^1, \ldots, z_T^1) \in \mathcal{E}) + \delta.$$

3 Problem Setup

We now formally introduce the main problem considered in this paper, namely that of *private online learning*. Let $\mathcal{X}$ be a set, and let $\mathcal{H}$ be a set of *hypotheses*, namely of functions $h : \mathcal{X} \to \{0,1\}$. We consider the setting depicted in Section 2.1 and in this framework we want to study the learnability of *private* learners which are defined next. We make a distinction between the case of an oblivious and an adaptive adversary:

Private online learning vs. an *oblivious* adversary As discussed, in this setting the adversary must choose the entire sequence $(x_1, y_1), \ldots, (x_T, y_T)$ before its interaction with the learner (though it may use knowledge of the learner's *algorithm*). In particular, the samples (x_t, y_t) do not depend on any random bits used by the learner. Thus, in the *private online learning problem* we merely require that the sequence of hypotheses $(h_1, \ldots, h_T)$ output by the learner is (ϵ, δ)-differentially private as a function of the entire input sequence $(x_1, y_1), \ldots, (x_T, y_T)$.

Private online learning vs. an *adaptive* adversary: In the adaptive setting, the adversary may choose each example (x_t, y_t) as a function of all of the learner's hypotheses up to t. This makes the notion of privacy a little bit more subtle, so we need to carefully define what we mean here by (ϵ, δ)-privacy. We consider then the following scenario:

At each round t, the adversary outputs two outcomes (x_t^0, y_t^0) and (x_t^1, y_t^1). The learner then outputs h_t^b and (x_t^b, y_t^b) is revealed to the learner where $b \in \{0,1\}$ is independent of t. We require that the sequences $S_T^0 = \{(x_t^0, y_t^0)\}$ and $S_T^1 = \{(x_t^1, y_t^1)\}$ differ in, at most, a single example. We will say that an adaptive online classification algorithm is (ϵ, δ) differentially private, if for any event $\mathcal{E}$ and any adversary, it holds that

$$\Pr[(h_1^1, \ldots, h_T^1) \in \mathcal{E}] \leq e^{\epsilon} \cdot \Pr[(h_1^0, \ldots, h_T^0) \in \mathcal{E}] + \delta.$$

The notion is similar to privacy under T-fold adaptive composition. Normally, though, for a mechanism to be (ϵ, δ)-differentially private under T-fold adaptive compositions, Dwork et al. [16] requires it to be private under an adversary that may choose at each iteration *any* two adjacent datasets, S_i^0, S_i^1. Note, however that, in the online setup, the utility is dependent only on a single point at each iteration, hence such a requirement will be too strong (in fact, the learner will then be tested on two arbitrary sequences).

4 Main Results

We next state the main results of this paper, we start with a logarithmic regret bound for realizable oblivious learning.

Theorem 4.1 (Private Oblivious online-learning)**.** *For a choice of $k_1 = \tilde{O}(2^{d+1})$, and*

$$k_2 = \tilde{O}\left(\frac{2^{8 \cdot 2^d}}{\epsilon} \ln T/\delta\right),$$

Running DP-SOA (Algorithm 1) for T iterations on any realizable sequence $(x_1, y_1), \ldots, (x_T, y_T)$, the algorithm outputs a sequence of predictors $h_1, \ldots, h_T$ such that

- *The algorithm is (ϵ, δ) differentially private.*
- *The expected number of mistakes the algorithm makes is*

$$\mathbb{E}[\sum_{t=1}^{T} h_t(x_t) \neq y_t] = \tilde{O}\left(\frac{2^{8 \cdot 2^d}}{\epsilon} \ln T/\delta\right).$$

Theorem 4.1 shows that, up to logarithmic factor, the number of mistakes in the private case is comparable with the number of mistakes in the non-private case, when d the Littlestone dimension of the class is constant. We obtain, though, a strong deterioration in terms of the Littlestone dimension – sublinear dependece vs. double exponential dependence. As discussed, Ghazi et al. [20] improved the dependence in the batch case to polynomial, and it remains an open question if similar improvement is applicable in the online case. We next turn to the adversarial case

Theorem 4.2 (Private Adaptive online-learning)**.** *There exists an adaptive online classification algorithm that is (ϵ, δ)-differentially private with expected regret over a realizble seqeunce:*

$$\mathbb{E}[\sum_{t=1}^{T} h_t(x_t) \neq y_t] = \tilde{O}\left(\frac{2^{O(2^d)} \sqrt{T} \log 1/(\delta)}{\epsilon}\right).$$

Theorem 4.2 provides a sublinear regret bound, which is in fact optimal for the agnostic case. However, in the non-private (realizable) case it is known that constant regret can be obtained[2] . We leave it as an open problem whether one can achieve logarithmic regret in the realizable adaptive setting.

5 Algorithm

We next present our main algorithm for an oblivious, realizable online private learning algorithm. The algorithm, DP-SOA, assumes access to a mistake bound algorithm for the class $\mathcal{H}$ (not necessarily private) such as SOA as in [30], which we denote by A,[3] as well as call a procedure HistSparse that is depicted below (Algorithm 2). We can think of DP-SOA as an algorithm that runs several copies of the same procedure, where each copy is working on its own subsequence of $(x_1, y_1), \ldots, (x_T, y_T)$, and the sub sequences form a random partition of the entire sequence.

Each process can be described by a tree whose vertices are labelled by samples that are iteratively constructed. Each tree outputs a predictor according to the state of its vertices. Hence, overall the algorithm can be depicted as a forest, where at each iteration an example is randomly assigned to one of the trees, and that tree, in turn, makes an update.

At each time step, we maintain a set of vertices $\mathcal{V}_t$, which we will call *pertinent* vertices. Each pertinent vertex v holds a sample S_v. At time $t = 1$ only the leaves are in $\mathcal{V}_1$, and each leaf v is assigned the sample $S_v = \emptyset$. Then, at every time-step where an example (x_t, y_t) is assigned to the tree, it is randomly assigned to a pertinent vertex v in $\mathcal{V}$ (in detail, it is first randomly assigned to a leaf and then propagated to a pertinent ancestor), and the sample S_v is updated to $(S_v, (x_t, y_t))$. After that, as we next describe, a process starts that updates the set of pertinent vertices; this process follows the idea of the tournament examples presented in [11].

Whenever two siblings $v, \mathrm{s}(v)$ are pertinent and assigned with sequences S_v and $S_{\mathrm{s}(v)}$, respectively, they stay pertinent as long as $A(S_v) = A(S_{\mathrm{s}(v)})$, and samples are assigned to them at their turn via the process depicted above. Whenever it becomes the case that $A(S_v) \neq A(S_{\mathrm{s}(v)})$, let $\bar{v}$ denote the parent of $v, \mathrm{s}(v)$; we consider an example $x_{\bar{v}}$ on which $A(S_v)$, $A(S_{\mathrm{s}(v)})$ disagree, and guess its label $y_{\bar{v}}$. Then, $v, \mathrm{s}(v)$ are removed from the set of pertinent vertices, their parent $\bar{v}$ becomes pertinent, and we set $S_{\bar{v}}$ to equal $(S_v, (x_v, y_v))$ if $A(S_v)[x_v] \neq y_v$, and $(S_{\mathrm{s}(v)}, (x_v, y_v))$ otherwise. Once this

[2] and as discussed, the adversary may even depend on h_t at round t

[3] In particular, A is required to be an algorithm that achieves a mistake bound of at most d on hypothesis classes of Littlestone dimension d. We will use the following (easily verified) fact about such an algorithm: after making a mistake, the algorithm must change the hypothesis it outputs for the following round.

Algorithm 1 DP-SOA

Input (ϵ, δ), k_1, k_2.
Set $\eta = \frac{2^{-4k_1}}{4k_1}$, and $c = 4k_1/\eta$
Let $G = (V, E)$ be a forest of k_2 full binary trees, each with k_1 leaves.
Let $\pi : T \to \text{Leaves}(V)$ be a random mapping that maps $t \in [T]$ to a random leaf.
Set $S_v = \emptyset$ for each leaf v and $S_u = \perp$ for each non-leaf vertex u (where we define $A(\perp) = \perp$).
Initialize $\mathcal{V}_1$ to be the set of all leaves in the forest.
set $v_1^{(i)}$ be an arbitrary leaf from the tree G_i, for each $i \in [k_2]$
for t=1 to T **do**
  Run $HistSparse_{\epsilon,\delta,\eta,c}(h_{t-1}, L_t)$ on the List $L_t = \{A(S_{v_t^{(i)}})\}_{i=1}^{k_2}$ and receive h_t
  Predict $h_t(x_t) = \hat{y}_t$, and observe y_t.
  Choose $v_1 \in \mathcal{V}_t$ to be an antecedent of leaf $\pi(t)$ %there exists a unique antecedent in $\mathcal{V}_t$
  Set $v_2 = \text{s}(v_1)$ (if v_1 is the root, continue to the next iteration).
  Set $(S_{v_1}, (x_t, y_t)) \to S_{v_1}$.
  while $A(S_{v_1}) \neq A(S_{v_2})$ AND $v_1, v_2 \in \mathcal{V}_t$ **do**
    Set $\bar{v}$ to be the parent of v_1, v_2
    Choose an arbitrary $x_{\bar{v}}$ such that $A(S_{v_1})[x_{\bar{v}}] \neq A(S_{v_2})[x_{\bar{v}}]$ and $y_{\bar{v}}$ randomly
    Set $(S_{v_i}, (x_{\bar{v}}, y_{\bar{v}})) \to S_{\bar{v}}$ where i is such that $A(S_{v_i})[x_v] \neq y_v$.
    Remove v_1,v_2 from $\mathcal{V}_t$ and add $\bar{v}$ to $\mathcal{V}_t$.
    Let v_1 be $\bar{v}_t$
    if v_1 is not the root **then**
      Set v_2 to be the sibling of v_1
    else
      Set $v_1 = v_2$ (and hence exit the loop.)
    end if
  end while
  if The While loop was executed at least once **then**
    Let i be the tree for which $\pi(t)$ belongs to.
    Choose randomly a vertex v in tree i such that $v, \text{s}(v) \in \mathcal{V}_t$ and $A(S_v) = A(S_{\text{s}(v)})$ (break ties by choosing randomly).
    (If no such v exists, let v be the root and set S_v to be some sample for which $A(S_v) = \perp$, add the root to $\mathcal{V}_t$ and remove all other vertices that belong to tree i).
    Set $v_{t+1}^{(i')} = \begin{cases} v & i = i' \\ v_t^{(i')} & i \neq i' \end{cases}$.
  else
    Set $v_{t+1}^{(i')} = v_t^{(i')}$ for all $i' \leq k_2$.
  end if
  Set $\mathcal{V}_{t+1} = \mathcal{V}_t$.
end for

procedure finishes, the tree outputs (randomly) some hypothesis $h = A(S_v)$ where v is a pertinent vertex. The hypothesis will change only when the state of the tree changes (note that at initialization, the tree outputs $A(\emptyset)$).

5.1 Technical Overview

We next give a high level overview of our proof techniques. We focus until the end of this section on the oblivious realizable case. The main procedure of the algorithm, DP-SOA, is Algorithm 1.

Our proof strategy is similar to the approach of Bun et al. [11] for learning privately in the stochastic setting, which we next briefly describe. In the stochastic setup, the idea was to rely on *global stability*. In a nutshell, a randomized algorithm is called globally stable if it outputs a certain function with constant probability (over the random bits of the algorithm as well as the random i.i.d sample). Once we can construct such an algorithm (with sufficiently small error) we run several copies of the algorithm on separate samples, and then we can use any mechanism, such as the one in Theorem 5.1 below, that publishes (privately) an estimated histogram of the frequency of appearance of each

Algorithm 2 `HistSparse`: Receives a sequence of 1-sensitive lists $L_1(D), \ldots, L_T(D)$.

Initialize: parameters $\epsilon, \eta, \delta, c$.
Let $\sigma = 2c/(k\epsilon)$, $\theta = 1 - 3\eta/32$
Let $\theta_0 = \theta + \mathrm{LAP}(\sigma)$.
Let counter = 1
For list L_1 set $h_1 = hist_{\epsilon/(2c,\delta/c,\eta)}(L_1)$.
for $t = 1, \ldots, T$: **do**
  Define query: $Q_t = 1 - \mathrm{freq}_{L_t}(h_{t-1})$.
  Let $\nu_i = \mathrm{LAP}(2\sigma)$
  if $Q_t + \nu_i \geq \theta_{\text{counter}}$ **then**
    Set $h_t = hist_{\epsilon/(2c),\delta/c,\eta}(L_t)$
    counter = counter + 1
    $\theta_{\text{counter}} = \theta + \mathrm{LAP}(\sigma)$.
  else
    Set $h_t = h_{t-1}$
  end if
  if counter $\geq c$ **then**
    ABORT
  end if
end for

function. In detail, given a list $L = \{x_1, \ldots, x_k\}$ we denote by freq_L the mapping

$$\mathrm{freq}_L(f) = \frac{1}{k} \sum_{x \in L} \mathbf{1}[x = f].$$

Theorem 5.1 ([8] essentially Proposition 2.20)**.** *For every ϵ, δ and η, there exists a (ϵ, δ)-DP mechanism $hist_{\epsilon,\delta,\eta}$ that given a list $L = \{x_1, \ldots, x_k\}$, outputs a mapping $\overline{\mathrm{freq}}_L : \mathcal{X} \to [0, 1]$ such that if*

$$k \geq \Theta_{(2)}(\eta, \beta, \epsilon, \delta) := 4/\eta + \frac{\log 1/(\eta^2 \beta\delta)}{\eta\epsilon} = O\left(\frac{\log 1/\eta\beta\delta}{\eta\epsilon}\right), \tag{2}$$

then with probability $(1 - \beta)$:

- *If $\overline{\mathrm{freq}}_L(x) > 0$ then $\mathrm{freq}_L(x) > \frac{\eta}{4}$.*
- *For every x such that $\mathrm{freq}_L(x) > \eta$, we have that $\overline{\mathrm{freq}}_L(x) > 0$.*

Our algorithm follows a similar strategy but certain care needs to taken due to the sequential (and distribution-free) nature of the data, as well as the fact that using `hist` procedure T times may be prohibitive (if we wish to obtain logarithmic regret). We next review these challenges:

Global Stability Our first task is to construct an online version of a globally stable algorithm, which roughly means that different copies of the same algorithm run on disjoint subsequences of $(x_1, y_1), \ldots, (x_T, y_T)$, and output a fixed hypothesis which may depend on the whole sequence but not on the disjoint subsequences. `DP-SOA` does so by assigning each subsequence to a tree which is running the procedure described in Section 5. We now explain how this procedure induces the desired stability.

As in Section 5, recall that a vertex v is pertinent if it is in the set $\mathcal{V}_t$. We will refer to the distance of a vertex to any of its leaves as that vertex's *depth*. Note that for each pertinent vertex v at depth k, the algorithm makes k mistakes on the sequence S_v – indeed, whenever a vertex $\bar{v}$ is made pertinent, we always append to $S_{\bar{v}}$ an example which forces a mistake for the sequence of a child of $\bar{v}$. Also, notice that with probability 2^{-2k_1}, where k_1 is the number of leaves in the tree, all sequences assigned to each pertinent vertex are consistent with the realized hypothesis $h^\star$ (recall that we are considering here the oblivious realizable case, hence $h^\star$ is well-defined). Indeed, this is true as as long as we guessed the label $y_{\bar{v}}$ to equal $h^\star(x_{\bar{v}})$ at each round; the number of guesses is bounded by the number of vertices, which is $2k_1 - 1 < 2k_1$. Ultimately, this allows two cases: in the first case a vertex of depth d is pertinent: in this case the vertex must identify $h^\star$ (indeed, if there are two different

hypotheses that are consistent on a sample with d mistakes, then we can force a $(d+1)$th mistake). So, if there are "many" trees with a d-depth pertinent vertex, then fraction of 2^{-2k_1} of them, are outputting $h^\star$, hence we found a frequent hypothesis. The second case is that in "many" of the trees, for some $k < d$, there are many pairs $v, \mathrm{s}(v)$ of pertinent vertices at depth k so that $A(S_v) = A(S_{\mathrm{s}(v)})$; we will refer to such a pair $v, \mathrm{s}(v)$ as a *collision*.

In the batch case the latter case immediately implies that some hypothesis is outputted frequently (i.e., we get global stability) through a standard concentration inequality that relates the number of collisions between i.i.d random variables, and the frequency of the most probable hypothesis. In the online case it is a little bit more subtle as the examples are not i.i.d, hence the sequences for the pertinent vertices are not i.i.d copies of some random variable. However, suppose that there are many collisions at depth k, and that we now reassign the data by randomly permuting the k-depth subtree (i.e. we reassign a random parent to each vertex at depth k, in order to form a new complete binary tree, and we don't change relations at other depths). Since the assignment of the data (x_t, y_t) to the leaves is invariant under permutation, we can think of this process as randomly picking a new assignment, conditioning on the k-th level structure of the trees. Alternatively, we can also think of this process as randomly picking without replacement the different hypotheses outputed by the k-depth vertices, and counting collisions of siblings.

We now want to relate the number of collisions to their expected mean and obtain a bound on the most frequent hypothesis. We can do this using a variant of Mcdiarmid's inequality for permutations – or sampling without replacement. The observation for this inequality was found in [23] which attributes it to Talagrand [34]. For completeness we provide the proof in the full version.

Lemma 5.2 (Mcdiarmid's without replacement)**.** *Suppose $\bar{Z} = (Z_1, \ldots, Z_n)$ are random variables sampled uniformly from some universe $\mathfrak{Z} = \{z^{(1)}, \ldots, z^{(N)}\}$ without replacement (in particular $n \leq N$). Let $F : Z^n \to [0,1]$ be a mapping such that for $\bar{z} = (z_1, \ldots, z_n)$ and $\bar{z}' = (z'_1, \ldots, z'_n)$ that are of Hamming distance at most 1, $|F(\bar{z}) - F(\bar{z}')| \leq c$. Then:*

$$\mathbb{P}\left(\mathbb{E}(F(\bar{Z})) - F(\bar{Z}) \geq \epsilon\right) \leq e^{-\frac{2\epsilon^2}{9nc^2}}.$$

We use Lemma 5.2 as follows: our function F counts the number of collisions between depth k vertices after a random permutation (where we think here of permutation as sampling without replacement), this function is 1-sensitive to changing a single element, as required. We thus obtain an estimate of the number of collisions for a random permutation, which we can relate to the appearance of the most frequent hypothesis.

The above calculation can be used to obtain a guarantee that there exists an hypothesis that appears at frequency $2^{-O(k_1)}$ (this frequency is roughly the probability that the tree remains consistent with $h^\star$). Since the number of leaves is exponential in the depth, and the depth needs to be at least d (the upper bound on the level at which the algorithm stabilizes for sure), we overall obtain doubly exponential dependence of the frequency on the Littlestone dimension.

Mistake Bound We next turn to bound the number of mistakes. The crucial observation is that every time the algorithm makes a mistake, if example x_t is assigned to tree i then with some positive probability (specifically, the frequency of h_t, lower bounded by $2^{-O(2^d)}$) tree i outputs h_t. Moreover, with probability $1/k_1 > 0$, x_t is assigned to the pertinent vertex that made the mistake. Once the example is assigned to this vertex, we have $A((S_v, (x_t, y_t))) \neq A(S_{\mathrm{s}(v)})$. In particular, the two siblings are taken out of the list of pertinent vertices, and their parent becomes pertinent. In other words, every time the algorithm makes a mistake with some constant probability (roughly $2^{-\tilde{O}(2^d)}$), the set of pertinent vertices diminishes by one. Since we start with finite number of leaves as pertinent vertices, the expected number of mistakes is bounded by the number of leaves in the forest.

It remains to show that the number of leaves in the forest is logarithmic in the sequence size (but doubly exponential in the Littlestone dimension). The number of leaves is roughly k_1 (which is roughly $O(2^d)$) times the number of trees in the forest; this number of trees depends on the sample complexity of the private process in which we output the frequent hypothesis. We now explain why roughly $O(2^{O(2^d)} \ln T)$ trees is sufficient.

Online publishing of a globally stable hypothesis The next challenge we meet is to output the frequent hypothesis. The most straightforward method to do that is to repeat the idea in the batch

setting and use procedure `hist`. We can guarantee a $O(\sqrt{T})$ factor of deterioration in the privacy parameter ϵ (see Lemma 5.4) due to the repeated use of the `hist` procedure T times.

Our main observation though, is that in most rounds, the frequent hypothesis does not change, allowing us to exploit the *sparse vector technique* [14], (see also [13]). The sparse vector technique is a method to answer, adaptively, a stream of queries where: whenever the answer to the query does not exceed a certain threshold the algorithm returns a negative result but *without* any cost in privacy. We pay, though, in each round where the query exceed the threshold.

We will exploit this idea in the following setting: we receive a stream of 1-sensitive lists $L_1(S), \ldots, L_T(S)$: Namely, each list L_t is derived from the data $S = \{(x_1, y_1), \ldots, (x_T, y_T)\}$, and L_t changes by at most one element, given a change in a single (x_t, y_t). We assume that at each iteration t we want to output an element $h_t \in L_t$ with high frequency. Our key assumption is that the lists are related and a very frequent element h_t is also frequent at step $t+1$. Thus in most rounds we just verify that $\text{freq}_{L_t}(h_{t-1})$ is large, and only in rounds where it is too small do we use the stable histogram mechanism, paying for privacy.

Indeed, in our setting, the appearance of the frequent hypothesis may diminish by at most one each round. Once its frequency has diminished by a certain factor, then we have already made a certain fraction of the maximum possible number of mistakes. Thus, in general we only need to verify that the frequency of h_{t-1} in L_t is sufficiently large each round, which can be done via the sparse vector technique without loss of privacy. We next state the result more formally, the proof is provided in the full version

Lemma 5.3. *Consider, the procedure* $\text{HistSparse}_{\eta,c,\epsilon}$ *depicted in Algorithm 2. Given a sample S, suppose Algorithm 2 receives a stream of lists, where each list is a function of S to an array of elements and each list is* 1*-sensitive. Then Algorithm 2 is (ϵ, δ) differentially private and: Set*

$$\Theta_{(3)}(c, \alpha, \beta, \epsilon, \alpha) := \frac{8c(\ln T + \ln 2c/\beta)}{\alpha\epsilon}, \tag{3}$$

and suppose:

$$k \geq \Theta_{(4)}(c, \eta, T, \beta, \epsilon, \delta) := \max\{\Theta_{(3)}(c, \alpha, \beta, \epsilon, \alpha), \Theta_{(2)}(\eta, \beta, \epsilon, \delta)\} = \tilde{O}\left(\frac{c \ln T/\beta\delta}{\eta\epsilon}\right). \tag{4}$$

The procedure then outputs a sequence $\{h_t\}_{t=1}^{T}$, where $h_t \in L_t$ such that if for each list L_t there exists h such that $\text{freq}_{L_t}(h) \geq \eta$ then with probability at least $(1 - 2\beta)$, for all $t \leq T$, either the algorithm aborted before step t or

- $\text{freq}_{L_t}(h_t) \geq \eta/16$.
- *If $h_{t-1} \neq h_t$:*

$$\text{freq}_{L_t}(h_{t-1}) \leq \eta/8 \quad \text{and} \quad \text{freq}_{L_t}(h_t) \geq \eta/4.$$

Adaptive adversaries The proof for the oblivious case relies on the existence of an $h^\star$ that is consistent with the data (and independent of the random bits of the algorithm). In the adaptive case, while the sequence has to be consistent, $h^\star$ need not be determined, and the consistent hypothesis may depend on the algorithm's choices.

However, to obtain a regret bound, we rely on the standard reduction that shows that a randomized learner against oblivious adversary, can attain a similar regret against an adaptive adversary ([12], Lemma 4.1). One issue, though, is that `DP-SOA` uses random bits that are shared through time. Hence for the reduction to work we need to reinitialize the algorithm at every time-step. In this case, though, the assumptions we make for using the sparse vector technique no longer hold. Thus we can run `DP-SOA`, using `hist` (as we no longer obtain any guarantee from `HistSparse`), and we require that each output hypothesis will be $O(\epsilon/\sqrt{T}, O(\delta/T))$-DP. The privacy of the whole mechanism now follows from T-fold composition:

Lemma 5.4. *(see for example Dwork and Roth [13]) Suppose (ϵ', δ') satisfy:*

$$\delta' = \delta/2T, \quad \text{and} \quad \epsilon' = \frac{\epsilon}{2\sqrt{2T\ln(1/\delta)}}. \tag{5}$$

Then, the class of (ϵ', δ')-differentially private mechanisms satisfies (ϵ, δ)-differentialy privacy under T-fold adaptive composition.

Unfortunately though, the above strategy leads to a $\sqrt{T}$ factor in the regret.

Acknowledgments and Disclosure of Funding

The authors would like to thank Uri Stemmer for helpful discussions. N.G is supported by a Fannie & John Hertz Foundation Fellowship and an NSF Graduate Fellowship; R.L is supported by an ISF grant no. 2188/20 and by a grant from Tel Aviv University Center for AI and Data Science (TAD) in collaboration with Google, as part of the initiative of AI and DS for social good.

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 308–318, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341394.

[2] N. Agarwal and K. Singh. The price of differential privacy for online learning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 32–40, 2017.

[3] N. Alon, R. Livni, M. Malliaris, and S. Moran. Private PAC learning implies finite Littlestone dimension. In *STOC*, page 852–860, 2019. ISBN 9781450367059.

[4] A. Beimel, K. Nissim, and U. Stemmer. Private learning and sanitization: Pure vs. approximate differential privacy. In *APPROX-RANDOM*, pages 363–378, 2013.

[5] A. Beimel, H. Brenner, S. P. Kasiviswanathan, and K. Nissim. Bounds on the sample complexity for private learning and private data release. *Machine Learning*, 94:401–437, 2014.

[6] A. Beimel, S. Moran, K. Nissim, and U. Stemmer. Private center points and learning of halfspaces. In *COLT*, pages 269–282, 2019.

[7] A. Beimel, K. Nissim, and U. Stemmer. Characterizing the sample complexity of pure private learners. *JMLR*, 20(146):1–33, 2019.

[8] M. Bun, K. Nissim, and U. Stemmer. Simultaneous private learning of multiple concepts. *arXiv preprint arXiv:1511.08552*, 2015.

[9] M. Bun, K. Nissim, U. Stemmer, and S. P. Vadhan. Differentially private release and learning of threshold functions. In *FOCS*, pages 634–649, 2015.

[10] M. Bun, C. Dwork, G. N. Rothblum, and T. Steinke. Composable and versatile privacy via truncated CDP. In *STOC*, page 74–86, 2018.

[11] M. Bun, R. Livni, and S. Moran. An equivalence between private classification and online prediction. *arXiv preprint arXiv:2003.00563*, 2020.

[12] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

[13] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[14] C. Dwork, M. Naor, O. Reingold, G. N. Rothblum, and S. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 381–390, 2009.

[15] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC '10, page 715–724, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300506.

[16] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[17] C. Dwork, K. Talwar, A. Thakurta, and L. Zhang. Analyze gauss: Optimal bounds for privacy-preserving principal component analysis. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, page 11–20, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450327107.

[18] A. Ene, H. L. Nguyen, and A. Vladu. Projection-free bandit optimization with privacy guarantees. *CoRR*, abs/2012.12138, 2020.

[19] V. Feldman and D. Xiao. Sample complexity bounds on differentially private learning via communication complexity. In *COLT*, pages 1–20, 2014.

[20] B. Ghazi, N. Golowich, R. Kumar, and P. Manurangsi. Sample-efficient proper pac learning with approximate differential privacy. *arXiv preprint arXiv:2012.03893*, 2020.

[21] A. Guha Thakurta and A. Smith. (nearly) optimal algorithms for private online learning in full-information and bandit settings. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[22] S. Hanneke, R. Livni, and S. Moran. Online learning with simple predictors and a combinatorial characterization of minimax in 0/1 games. *arXiv preprint arXiv:2102.01646*, 2021.

[23] K. P. C. (https://mathoverflow.net/users/405/kevin-p costello). Concentration bounds for sums of random variables of permutations. MathOverflow, 2013. URL https://mathoverflow.net/q/120257. URL:https://mathoverflow.net/q/120257 (version: 2013-01-29).

[24] B. Hu, Z. Huang, and N. A. Meta. Optimal algorithms for private online learning in a stochastic environment. *CoRR*, abs/2102.07929, 2021.

[25] P. Jain, P. Kothari, and A. Thakurta. Differentially private online learning. In *Proceedings of the 25th Annual Conference on Learning Theory*, volume 23 of *Proceedings of Machine Learning Research*, pages 24.1–24.34, 2012.

[26] H. Kaplan, K. Ligett, Y. Mansour, M. Naor, and U. Stemmer. Privately learning thresholds: Closing the exponential gap. In *COLT*, pages 2263–2285, 2020.

[27] H. Kaplan, Y. Mansour, U. Stemmer, and E. Tsfadia. Private learning of halfspaces: Simplifying the construction and reducing the sample complexity. In *NeurIPS*, 2020.

[28] H. Kaplan, M. Sharir, and U. Stemmer. How to Find a Point in the Convex Hull Privately. In *SoCG*, pages 52:1–52:15, 2020.

[29] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Rashkodnikova, and A. Smith. What can we learn privately? In *FOCS*, pages 531–540, 2008.

[30] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2(4):285–318, 1988.

[31] A. Rakhlin, K. Sridharan, and A. Tewari. Online learning via sequential complexities. *JMLR*, 16:155–186, 2015.

[32] A. Roth and M. Kearns. *The Ethical Algorithm: The Science of Socially Aware Algorithm Design*. Oxford University Press, 2019.

[33] S. Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and trends in Machine Learning*, 4(2):107–194, 2011.

[34] M. Talagrand. Concentration of measure and isoperimetric inequalities in product spaces. *Publications Mathématiques de l'Institut des Hautes Etudes Scientifiques*, 81(1):73–205, 1995.

[35] A. C. Y. Tossou and C. Dimitrakakis. Algorithms for differentially private multi-armed bandits. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2087–2093. AAAI Press, 2016.

[36] A. C. Y. Tossou and C. Dimitrakakis. Achieving privacy in the adversarial multi-armed bandit. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 2653–2659. AAAI Press, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Proofs

A.1 Proof of Theorem 4.1

Privacy: We begin by proving the privacy guarantees:

Lemma A.1. *Suppose we run Algorithm 1 with parameters (ϵ, δ). Then the output sequence $h_1, \ldots, h_t$ is (ϵ, δ)-DP.*

Proof. Note that at every time step t, changing a single element x_t changes at most one element on the list $L_t = \{A(S_{v_t^{(i)}})\}_{i=1}^{k_2}$ – specifically, the tree i for which $\pi(t)$ assigns the element x_t. Next, note that if we fix the random bits of the algorithm, except for those that are used in the sub-procedure `HistSparse` (i.e. π and the random guessing y_v), then each list is completely determined at step t by the dataset S. Indeed, each $S_{v_t^{(i)}}$ is independent of $h_1, \ldots, h_T$ and the updates of the algorithm are independent of those. As such, we can think of the lists as functions of the dataset S.

The prerequisite assumptions for Algorithm 2 hold then (see Lemma 5.3), and by Lemma 5.3, we have that the list $h_1, \ldots, h_T$ is then (ϵ, δ)-DP. ■

Utility: The core lemma behind our proof is a statement that there exists (at each iteration) a function that is frequently outputted by a fraction of the trees; the proof is deferred to Appendix A.3.

Lemma A.2. *Suppose $(x_1, y_1), \ldots, (x_T, y_T)$ is consistent with some hypothesis $h^\star \in \mathcal{H}$. If*

$$k_1 \geq \max\{2^{d+1}, 20\}, \quad \text{and} \quad k_2 \geq 2^{8k_1+6} k_1^2 \log \frac{5T \log k_1}{\beta} := \Theta_{(6)}(k_1, T, \beta), \tag{6}$$

then with probability at least $1-\beta$, for all iterations $t \leq T$ there exists a predictor $f \neq \perp$ such that:

$$\text{freq}_{L_t}(f) \geq \frac{2^{-4k_1}}{4k_1}.$$

We continue with the proof of Theorem 4.1, assuming Lemma A.2. The proof is an immediate corollary of the following utility lemma.

Lemma A.3. *Suppose Algorithm 1 is run on a sequence $(x_1, y_1), \ldots, (x_T, y_T)$, and assume that there exists $h^\star \in \mathcal{H}$ such that $h^\star(x_i) = y_i$ for all $i \in [T]$. Then, for $\beta = 1/T$, η and c as initialized in Algorithm 1, if:*

$$k_1 \geq \max\{2^{d+1}, 20\}, \quad \text{and} \quad k_2 \geq \max\{\Theta_{(4)}(c, \eta, T, \beta, \epsilon, \delta), \Theta_{(6)}(k_1, T, \beta)\} = \tilde{O}\left(\frac{2^{8\cdot 2^d}}{\epsilon} \ln T/\delta\right).$$

the expected number of mistakes the algorithm makes after T rounds is:

$$\mathbb{E}\left[\sum_{t=1}^{T} \mathbf{1}[h_t(x_t) \neq y_t]\right] \leq \frac{4k_1^3 \cdot 2^{2k_1} k_2}{\eta} + 1 = \tilde{O}\left(\frac{2^{8\cdot 2^d}}{\epsilon} \ln T/\delta\right).$$

Proof of Lemma A.3 First, setting $\beta = 1/T$ we have by assumption that $k_2 \geq \Theta_{(6)}(k_1, T, \beta)$. As such, we can turn to Lemma A.2 and setting $\eta = \frac{2^{-4k_1}}{4k_1}$ we have that, with probability $1 - 1/T$, for each list L_t there is an element f such that $\mathrm{freq}_{L_t}(f) \geq \eta$. We can now apply Lemma 5.3, to obtain that, overall with probability $1 - 3/T$: either the algorithm halted, or for each t:

1. $\mathrm{freq}_{L_t}(h_t) \geq \eta/16$.
2. If $h_{t-1} \neq h_t$, then

$$\mathrm{freq}_{L_t}(h_{t-1}) \leq \eta/8 \quad \text{and} \quad \mathrm{freq}_{L_t}(h_t) \geq \eta/4.$$

Let us denote this event by E_0, and we will assume for now on the E_0 happened.

Next, we want to show (under E_0) that for $c = 4k_1/\eta$, we have that

$$|\{t : \mathrm{freq}_{L_t}(h_{t-1}) \leq \eta/8\}| \leq c.$$

To see the above, let t be a time-step for which $\mathrm{freq}_{L_t}(h_{t-1}) \leq \eta/8$, but the algorithm did not abort before time-step t. Set $t' < t$ be the last iteration where we called `hist` procedure (i.e. the last time we updated counter in `HistSparse`). Observe that $h_{t-1} = h_{t'}$, and note that by Item 2 we have that $\mathrm{freq}_{L_{t'}}(h_{t'}) > \eta/4$. In particular, the Hamming distance between the lists L_t and $L_{t'}$ is at least $\eta \cdot k_2/4$.

Note that for each $i \in [k_2]$, $v_t^{(i)}$ is changed between rounds t and $t+1$ only if we run the While loop in Algorithm 1 at round t. Note also that at each iteration of the While loop, the size of the set $\mathcal{V}_t$ is decreased by 1 (as we remove two siblings and add their parent). So $|\mathcal{V}_{t'}| - |\mathcal{V}_t| \geq \eta \cdot k_2/4$. Let c_t be the number of time steps $t' \leq t$ so that $\mathrm{freq}_{L_{t'}}(h_{t'-1}) \leq \eta/8$. At initialization we have that $|\mathcal{V}_1| = k_2 \cdot k_1$; thus, for all $t \geq 1$,

$$k_2 \cdot k_1 - \eta/4 \cdot k_2 \cdot c_t \geq 0 \Rightarrow c_t \leq 4k_1/\eta.$$

By the choice of $c = 4k_1/\eta$ in Algorithm 1, the algorithm doesn't halt and we have that, under E_0,

$$\forall t = 1, \ldots, T : \mathrm{freq}_{L_t}(h_t) > \frac{\eta}{16}. \tag{7}$$

We next continue to bound the expected number of mistakes conditioned on E_0.

Suppose that $\pi(t)$ belongs to the i-th tree. Note that $\pi(t)$ is independent of h_t as well as $\mathcal{V}_t$. We have, then, that with probability $1/k_1$, $\pi(t)$ is a descendent of $v_t^{(i)}$. One can observe, that for every leaf there exists a unique predecessor that belongs to $\mathcal{V}_t$. Overall then, we obtain that with probability $1/k_1$, $v_t^{(i)} = v_1$. (Recall that v_1 is defined in Algorithm 1 to be the unique antecedent of $\pi(t)$ that is in $\mathcal{V}_t$.)

Also, because $\mathrm{freq}_{L_t}(h_t) > \eta/16$, with probability $\eta/16$ we have $A(S_{v_t^{(i)}}) = h_t$. Taken together we have that whenever the algorithm makes a mistake then $A(S_{v_1})$ makes a mistake with probability at least $\eta/(16k_1)$. Therefore

$$\mathbb{E}\left(\mathbf{1}[h_t(x_t) \neq y_t] \mid E_0\right) \leq \frac{16k_1}{\eta}\,\mathbb{E}\left(\mathbf{1}[A(S_{v_1})[x_t] \neq y_t \mid E_0\right]).$$

Again, notice that if $A(S_{v_1})$ makes a mistake, we have that $|\mathcal{V}_t|$ is reduced by at least 1. (Indeed, in this case we have that both $v_1, v_2 \in \mathcal{V}$ by choice of $v_t^{(i)}$; because we make a mistake, after adding (x_t, y_t) to the sequence S_{v_1}, the algorithm disagrees on these two sequences, hence we run at least one iteration of the While loop that reduces the size of $\mathcal{V}_t$ by at least 1.)

As before, since at the beginning $|\mathcal{V}_1| = k_2 \cdot k_1$:

$$\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^{T}\mathbf{1}[h_t(x_t) \neq y_t] \mid E_0\right] &\leq \frac{16k_1}{\eta}\sum_{t=1}^{T}\mathbb{E}\left(\mathbf{1}\left[A(S_{v_1})[x_t] \neq y_t \mid E_0\right]\right) \\
&= \frac{16k_1}{\eta}\,\mathbb{E}\left(\sum_{t=1}^{T}\mathbf{1}\left[A(S_{v_1})[x_t] \neq y_t\right] \mid E_0\right) \\
&\leq \frac{16k_1|\mathcal{V}_1|}{\eta} \\
&= \frac{16k_1^2k_2}{\eta}.
\end{aligned}$$

Hence, we obtain in expectation

$$\mathbb{E}\left[\sum_{t=1}^{T}\mathbf{1}[h_t(x_t) \neq y_t]\right] \leq \frac{k_1^2k_2}{\eta} + \beta T \leq \frac{k_1^2k_2}{\eta} + 3.$$

A.2 Proof of Theorem 4.2

We consider the following procedure:

- Given ϵ, δ, T, set ϵ', δ' as in Eq. (5).
- At each time-step t, run DP-SOA with privacy parameters (ϵ', δ'), k_1, k_2 on the input sequence $S_t = ((x_1, y_1), \ldots, (x_{t-1}, y_{t-1}))$.
- Receive a sequence $h_1^{(t)}, \ldots, h_t^{(t)}$ from DP-SOA and output $h_t = h_t^{(t)}$.

Now, we assume k_1 and k_2 are chosen so that for an oblivious sequence the conditions of Theorem 4.1 are met, and hence

- Each output $h_1^{(t)}, \ldots, h_t^{(t)}$ is (ϵ', δ')-DP w.r.t to the input sequence $S_t = ((x_1, y_1), \ldots, (x_{t-1}, y_{t-1}))$.
- For any oblivious sequence of length T, we have that the mistake bound is bounded by $O\left(2^{8\cdot 2^d}/\epsilon' \ln T/\delta'\right)$.

Now, for privacy we can use Lemma 5.4. Consider the setting of privacy against an adaptive adversary as introduced in Section 3. Observe that, by our definition of the adaptive adversary, each time we apply DP-SOA, we apply it on either the sample $S_t^0 = (x_1^0, y_1^0), \ldots, (x_t^0, y_t^0)$, or $S_t^1 = (x_1^1, y_1^1), \ldots, (x_t^1, y_t^1)$, which can differ by at most one sample. Therefore, since the mechanism that outputs $h_t^{(t)}$ at step t is (ϵ', δ')-DP, we obtain via Lemma 5.4 that the above adaptive online classification algorithm is (ϵ, δ)-DP.

As for utility, the result follows immediately for the standard reduction from an oblivious online learner to an adaptive one (Lemma 4.1 in [12]). Indeed, note that at step t we predict h_t according to a distribution p_t which is completely defined by the previous sequence of examples $(x_1, y_1), \ldots, (x_{t-1}, y_{t-1})$ (it is the distribution from which the oblivious algorithm DP-SOA chooses its prediction). Thus the precondition of [12, Lemma 4.1] is verified, and we obtain the regret bound:

$$\sum_{t=1}^{T}\mathbb{E}[\mathbf{1}[h_t(x) \neq y]] \leq O\left(2^{8\cdot 2^d}/\epsilon' \ln T/\delta'\right).$$

A.3 Proof of Lemma A.2

Let $h^\star$ be a fixed hypothesis that is consistent with the dataset $(x_1, y_1), \ldots, (x_T, y_T)$. We will call a tree T in the forest G consistent if for every vertex v, S_v is consistent with hypothesis $h^\star$ and we let $\mathcal{G}_c$ be the sub-graph that consists only of consistent trees. With these notations in mind, we now proceed to the proof. We will divide the proof into two claims; the first one, Claim A.4, gives a lower bound on the number of consistent trees.

Claim A.4. *For a fixed time-step $t \le T$, with probability at least, $1 - e^{-\frac{1}{2}k_2 \cdot 2^{-4\cdot k_1}}$, we have that $2^{-2k_1-1} \cdot k_2$ of the trees in G are consistent.*

Proof. Note that for a tree to be consistent we only need that for every $y_{\bar{v}}$ that we guess while running the algorithm, we have that $y_{\bar{v}} = h^\star(x_{\bar{v}})$. If this happens, then all sequences $S_{\bar{v}}$ remain consistent in the tree. For each $\bar{v}$, this happens with probability $1/2$, independent on the sequence and the other labels $y_{\bar{v}}$. Hence each tree is consistent with probability at least $2^{-2\cdot k_1}$ (the number of vertices) and this is independent of the other trees. Thus, applying the Chernoff bound, we obtain that if M_t is the number of conistent trees at time t, then:

$$\mathbb{P}\left(M_t \le (2^{-2k_1} - 2^{-(2k_1+1)}) \cdot k_2\right) \le e^{-2k_2 \cdot 2^{-2(2\cdot k_1+1)}} \tag{8}$$

■

The next step is to prove that (with high probability) there exists a function f that appears frequently in the list $\{A(S_v)\}$ of vertices that belong to consistent trees, which we do next.

First let us denote by $\Xi = (\pi, \{y_v\}_{v\in V})$ the random seed, or internal bits, of `DP-SOA`, not including the random bits of the mechanisms `HistSparse`. Note that, at each time-step, the sets S_v, and $\mathcal{V}_t$ are completely determined by Ξ (and the oblivious sequence). In particular, the state of the forest is completely independent of the output hypotheses picked by `HistSparse`.

Let $\mathcal{G}_c(\Xi; t)$ denote the subgraph of consistent trees given Ξ at time t and let $F_k(\Xi; t)$ be the multiset that consists of all labeled subtrees (at time step t) of consistent trees whose root is a depth-k vertex. We will often, with slight abuse of notation, associate a tree in $F_k(\Xi; t)$ to its S_v-labeled root v, which is a depth-k vertex of some consistent tree; thus we will write, at times, "for each v in $F_k(\Xi; t)$". (Also note that it may be the case that for some depth-k vertices v, $S_v = \emptyset$; the subtrees rooted at such v are still included in $F_k(\Xi; t)$). Also, let us say that the (multi)set F_k is *f-heavy* if, for at least $2^{-k_1}|F_k|$ of the vertices v in F_k we have that $f = A(S_v) \neq \perp$.

Then we have the following claim:

Claim A.5. *For a fixed time-step $t \le T$, let $\mathcal{F}$ denote the event that for some $k \le \log k_1 + 1$ and f, F_k is f-heavy. then,*

$$\mathbb{P}(\mathcal{F}) \ge 1 - 2\log k_1 \cdot e^{-\frac{2^{-4k_1-1}}{9}k_2}. \tag{9}$$

Proof. The crucial observation is that, because the distribution of π is invariant under permutation of the leaves, then given F_k and $\mathcal{G}_c$, the distribution π of the assignments of data points can be viewed as randomly sampling (without replacement) elements from F_k and assigning to each subtree its appropriate depth-k vertex as a root.

Specifically, let us say that a vertex v is *active* if it belongs to a consistent tree. Now, let $V_k(\Xi; t)$ be the set of labeled depth-k active vertices which are right-children of their parents. For each $v \in V_k(\Xi; t)$, denote by X_v the random variable defined as follows: $X_v = 1$ if $A(S_v) = A(S_{s(v)})$ and $v, v_{s(v)} \in \mathcal{V}_t$ at the end of the While loop at step t of Algorithm 1, and $X_v = 0$ otherwise (recall that t is fixed). And further, denote

$$E(\Xi; k) = \mathbb{E}\left[\frac{1}{|V_k|}\sum_{v\in V_k} X_v \;\middle|\; F_k(\Xi; t), \mathcal{G}_c(\Xi; t)\right].$$

We claim the following bound holds for the time-step t:

$$\Pr_{\Xi}\left(\max_k \left\{E(\Xi; k) - \frac{1}{|V_k|}\sum_{v\in V_k} X_v\right\} > 2^{-k_1}\right) \le \log k_1 e^{-\frac{2^{-2k_1}|V_k|}{18}}. \tag{10}$$

To establish Eq. (10), note that for a fixed $k \le \log k_1$ and a set F_k, by symmetry of the distribution of π, the joint distribution of all X_v does not change if we resample the labels S_v for all vertices v in F_k, from this set of all labels, without replacement. Note that changing a single element S_v will change at most one random variable X_v, and as such we get that $\frac{1}{|V_k|}\sum X_v$ is $\frac{1}{|V_k|}$-sensitive. Since we randomly draw $2|V_k|$ elements, we can thus use Lemma 5.2 to obtain that for a fixed k, F_k and $\mathcal{G}_c$:

$$\begin{aligned}
&\Pr_{\Xi}\left(E(\Xi;k) - \frac{1}{|V_k|}\sum_{v\in V_k} X_v > 2^{-k_1} \;\middle|\; F_k(\Xi;t) = F_k, \mathcal{G}_c(\Xi;t) = \mathcal{G}_c\right)\\
=&\Pr_{\Xi}\left(\mathbb{E}\left[\frac{1}{|V_k|}\sum_{v\in V_k} X_v \;\middle|\; F_k, \mathcal{G}_c\right] - \frac{1}{|V_k|}\sum_{v\in V_k} X_v > 2^{-k_1} \;\middle|\; F_k, \mathcal{G}_c\right)\\
\le& e^{-\frac{2^{-2k_1}|V_k|}{18}}.
\end{aligned}$$

Eq. (10) now follows by taking expectation over $F_k, \mathcal{G}_c$ as well as a union bound over the $\log k_1$ possible values of $k \le \log k_1$.

We next observe that for any consistent tree there exists a vertex v, such that $v, \mathrm{s}(v) \in \mathcal{V}_t$ and $A(S_v) = A(S_{\mathrm{s}(v)})$. Indeed, if this is not the case, then one can prove by induction that the tree's root v_r is in $\mathcal{V}$. However, the sequence S_{v_r} makes $\log k_1 \ge d+1$ mistakes, which is a contradiction to the consistency of the tree. Then, what we showed so far is that in any consistent tree there exists v such that $X_v = 1$. Thus, applying pigeon-hole principle, we obtain that for any π there exists a $k \le \log k_1$ such that

$$\frac{1}{|V_k|}\sum_{v\in V_k} X_v \ge \frac{1}{k_1 \log k_1} \ge 2^{-k_1+1}.$$

Together with Eq. (10) we get that, given $\mathcal{G}_c$, with probability at least $1 - \log k_1 \cdot e^{-\frac{2^{-2k_1+1}|V_k|}{18}}$, for some k we have that

$$E(\Xi;k) > 2^{-k_1}.$$

Finally, (where for ease of notation we neglect the dependence of $F_k, \mathcal{G}_c$ in Ξ) we have

$$\begin{aligned}
E(\Xi;k) &= \frac{1}{|V_k|}\sum_{v\in V_k} \mathbb{P}\left(A(S_v) = A(S_{\mathrm{s}(v)}) \neq \perp | F_k, \mathcal{G}_c\right)\\
&= \frac{1}{|V_k|}\sum_{v\in V_k}\sum_{f\neq\perp} \mathbb{P}(A(S_v) = f | F_k, \mathcal{G}_c)\,\mathbb{P}\left(A(S_{\mathrm{s}(v)}) = f | A(S_v) = f, F_k, \mathcal{G}_c\right)\\
&\le \frac{1}{|V_k|}\sum_{v\in V_k}\sum_{f\neq\perp} \mathbb{P}(A(S_v) = f | F_k, \mathcal{G}_c)\,\mathbb{P}\left(A(S_{\mathrm{s}(v)}) = f | F_k, \mathcal{G}_c\right)\\
&= \frac{1}{|V_k|}\sum_{v\in V_k}\sum_{f\neq\perp} (\mathbb{P}(A(S_v) = f | F_k, \mathcal{G}_c))^2\\
&\le \frac{1}{|V_k|}\sum_{v\in V_k} \max_{f\neq\perp} \mathbb{P}(A(S_v) = f | F_k, \mathcal{G}_c)\\
&= \max_{f\neq\perp} \mathbb{P}(A(S_{v_0}) = f | F_k, \mathcal{G}_c),
\end{aligned}$$

where the first inequality follows from the fact that S_v are sampled without replacement, hence the distribution for $A(S_{\mathrm{s}(v)}) = f$ given that we already sampled such an element reduces. The last equality follows from the fact that the distribution of S_v, conditioned on $F_k, \mathcal{G}_c$, is identical for all $v \in V_k$; in the last line we set v_0 to be an arbitrary vertex in V_k.

Finally, using Claim A.4, and noting that $|V_k|$ is at least the number of consistent trees, we have that with probability

$$1 - \log k_1 \cdot e^{-\frac{2^{-2k_1+1}|V_k|}{18}} - e^{-k_2 \cdot 2^{-4\cdot k_1 - 1}} \ge 1 - 2\log k_1 \cdot e^{-\frac{2^{-4k_1-1}k_2}{9}},$$

for some k, we have

$$\max_{f \neq \perp} \mathbb{P}(A(S_{v_0}) = f | F_k, \mathcal{G}_c) \geq 2^{-k_1},$$

where again v_0 is an arbitrary vertex in V_k. Since S_{v_0} is sampled uniformly at random from the set of S_v for $v \in F_k(\Xi; t)$, the left-hand side of the above inequality is simply the fraction of S_v, for $v \in F_k(\Xi; t)$ for which $A(S_v) = f$. In particular, we obtain that $F_k(\Xi; t)$ is heavy. ■

The final claim we will need bounds the number of times we have $A(S_v) = A(S_{s(v)}) = f$ given the F_k is heavy:

Claim A.6. *For a fixed time-step $t \leq T$, recall that $\mathcal{F}$ is the event that F_k is f-heavy for some f and k. Let E be the event that for at least $2^{-2k_1-1}k_2$ of the trees, there exists a vertex v such that $A(S_v) = A(S_{s(v)}) = f$, then if $k_1 \geq 20$:*

$$\mathbb{P}(E|\mathcal{F}) \geq 1 - 2e^{-\frac{2^{-8k_1-6}}{9}k_2}. \tag{11}$$

Proof. Fix the set of consistent trees $\mathcal{G}_c$, and assume that the number of consistent trees is at least $2^{-k_1-1} \cdot k_2$. We can assume that $k_2 \geq 2^{2k_1+2}$ (otherwise, since $k_1 \geq 20$ the bound is trivial), hence $|F_k| \geq 2^{k_1+1}$, for any k (as $|F_k|$ is bounded below by the number of consistent trees).

Let us condition π on the consistent trees $\mathcal{G}_c$ and F_k, which we will assume to be f-heavy. Again, we use the fact that conditioned on $F_k, \mathcal{G}_c$, the joint distribution of all S_v ($v \in F_k$) is unchanged if we randomly resample each S_v-labeled vertex v from F_k, without replacement. In particular we have that, for any k-depth vertex v:

$$\begin{aligned}
\mathbb{P}(A(S_{s(v)}) = f | A(S_v) = f, F_k, \mathcal{G}_c) &\geq 2^{-k_1} - \frac{1}{|F_k|} \\
&\geq 2^{-k_1} - 2^{-k_1-1} && |F_k| \geq 2^{k_1+1} \\
&= 2^{-k_1-1}.
\end{aligned}$$

For $i \in [k_2]$, we now set X_i to be the random variable defined by: $X_i = 1$ if there exists v in the i-th tree such that $A(S_{s(v)}) = A(S_v) = f$, and $X_i = 0$ otherwise. For each consistent tree i, and for any depth-k vertex v of tree i, using the fact that F_k is f-heavy, we have:

$$\begin{aligned}
\mathbb{E}[X_i | F_k, \mathcal{G}_c] &\geq \mathbb{P}(A(S_v) = A(S_{s(v)}) = f | F_k, \mathcal{G}_c) \\
&= \mathbb{P}(A(S_v) = f \mid F_k, \mathcal{G}_c) \cdot \mathbb{P}(A(S_s(v) = f \mid A(S_{s(v)}) = f,\ F_k, \mathcal{G}_c) \\
&\geq 2^{-k_1}\, \mathbb{P}(A(S_s v) = f | A(S_v) = f, F_k, \mathcal{G}_c) \\
&\geq 2^{-k_1} \cdot 2^{-k_1-1} \\
&\geq 2^{-2k_1-1}.
\end{aligned}$$

So if $2^{-2k_1-1} \cdot k_2$ of the trees are in $\mathcal{G}_c$, i.e. are consistent, we have that

$$\mathbb{E}\left[\frac{1}{k_2}\sum_{i=1}^{k_2} X_i \mid\ F_k, \mathcal{G}_c\right] \geq 2^{-2k_1-1} \cdot 2^{-2k_1-1} = 2^{-4k_1-2}. \tag{12}$$

We again exploit the fact that changing the label S_v of a single vertex v in a tree changes at most one random variable X_i, and use Lemma 5.2 to obtain a high probability rate. In particular, for any set of consistent trees $\mathcal{G}_c$ that includes $2^{-2k_1-1} \cdot k_2$ of the trees, and for any heavy F_k:

$$\mathbb{P}\left(\frac{1}{k_2}\sum_{i=1}^{k_2} X_i \leq 2^{-4k_1-3} \mid F_k, \mathcal{G}_c\right) \leq e^{\frac{2^{-8k_1-6}}{9}k_2}. \tag{13}$$

Finally, we take expectation over heavy F_k. Note that F_k determines if F_j is heavy for all $j \leq k$, meaning that we may take the expectation of Eq. (13) over only those F_k for which the determined F_j is not heavy for all $j < k$. And by Claim A.4, $\mathcal{G}_c$ consists of $2^{-2k_1-1} \cdot k_2$ of the trees with probability at least $1 - e^{-k_2 \cdot 2^{-4k_1-2}}$. Hence

$$\mathbb{P}(E|\mathcal{F}) \geq 1 - e^{-\frac{2^{-8k_1-6}}{9}k_2} - e^{-2^{-4k_1-2}k_2} \geq 1 - 2e^{-\frac{2^{-8k_1-6}}{9}k_2}.$$

■

Concluding the proof of Lemma A.2 We are now ready to conclude the proof of Lemma A.2. First note that if a vertex satisfies $A(S_v) = A(S_{s(v)}) \neq \perp$ then we must have $v \in \mathcal{V}_t$. Indeed, since for both $S_v, S_{s(v)} \neq \perp$, they must at some point have been in $\mathcal{V}$ (because every time we initialize S_v we also add v to $\mathcal{V}$). And whenever we take v out of $\mathcal{V}$ then we must also take $s(v)$, but we take them out only if $A(S_v) \neq A(S_{s(v)})$).

As such, for any fixed f, for any tree that contains a vertex v such that $A(S_v) = A(S_{s(v)}) = f$, with probability at least $1/k_1$ we have that $A(v_t^{(i)}) = f$ (as $v_t^{(i)}$ is chosen randomly, at each time-step the tree is updated). Now utlizing Claims A.5 and A.6 we obtain that with probability at least

$$1 - 2e^{-\frac{2^{-8k_1-6}}{9}k_2} - 2\log k_1 e^{-\frac{2^{-4k_1-1}}{9}k_2} \geq 1 - 4\log k_1 e^{-\frac{2^{-8k_1-6}}{9}k_2},$$

at least $2^{-2k_1-1}k_2$ of the trees contain a vertex v such that $A(S_v) = A(S_{s(v)}) = f$ for some $f \neq \perp$ (independent of the tree).

By the Chernoff bound, we obtain that for at least $\frac{2^{-4k_1-2}k_2}{k_1}$ of these trees i, we choose $v_t^{(i)}$ satisfying $A(v_t^{(i)}) = f$, with probability at least $1 - e^{-\frac{2^{-8k_1-4}}{k_1^2}\cdot k_2}$.

To conclude, for any fixed t, with probability at least

$$1 - e^{-\frac{2^{-8k_1-4}}{k_1^2}k_2} - 4\log k_1 e^{-\frac{2^{-8k_1-6}}{9}k_2} \geq 1 - 5\log k_1 e^{\frac{2^{-8k_1-6}}{k_1^2}k_2},$$

for $\frac{2^{-4k_1-2}}{k_1}$ fraction of the trees i we have $A(S_{v_t^{(i)}}) = f$ for some fixed f. The result now follows from a union bound over $t \leq T$.

A.4 Proof of Lemma 5.3

Privacy For privacy, the proof is verbatim the proof that `sparse` is private provided in [13] (but instead of publishing the answer to a linear query everytime a threshold is passed, we output a frequent hypothesis). First, we consider the following variant of the procedure `Above-threshold` introduced in [13]:

Theorem A.7 ([13], Thm 3.26)**.** *There exists a $(\epsilon, 0)$-DP procedure, `Above-threshold`$_{\theta,c,\epsilon}$ (depicted in Algorithm 3, that receives an adaptive sequence of queries $Q_1, \ldots, Q_T$ that are $1/k$ sensitive and outputs a list $\{a_t\}_{t=1}^T$ such that if:*

$$k \geq \Theta_{(3)}(c, \alpha, \beta, \epsilon, \alpha) := \frac{8c(\ln T + \ln 2c/\beta)}{\alpha\epsilon}, \tag{14}$$

then for any sequence $Q_1, \ldots, Q_T$ such that $|\{t : Q_t(D) \geq \theta - \alpha\}| \leq c$, with probability $1 - \beta$:

- *For all $a_i = \top$: $Q_i(D) \geq \theta - \alpha$.*
- *For all $a_i = \perp$: $Q_i(D) \leq \theta + \alpha$.*

Algorithm 3 `Above-threshold`

Initialize: parameters ϵ, θ, c.
Let $\sigma = 2c/(k\epsilon)$
Let $\theta_0 = \theta + \mathrm{LAP}(\sigma)$.
Let counter = 0
for each list L_t **do**
  Receive a $1/k$ sensitive query $Q_t(D)$
  Let $\nu_i = \mathrm{LAP}(2\sigma)$
  if $Q_t(D) + \nu_i \geq \theta$ **then**
    output $\top$.
    Set counter = counter + 1.
    Let $\theta_{\text{counter}} = \theta + \mathrm{LAP}(\sigma)$
  else
    output $\perp$
  end if
  if counter $\geq c$ **then**
    ABORT
  end if
end for

We observe that Algorithm 2 is the adaptive composition of `Above-threshold`, together with the `hist` mechanism with parameters $(\epsilon'/(2c), \delta'/(2c))$. Moreover since each list changes by at most one element if we change a single point in the database, we have that the queries $Q_t(D) = 1 - \mathrm{freq}_{L_t}(h_{t-1})$ are $1/k$ sensitive. Hence by standard composition we obtain that the algorithm is (ϵ, δ)-DP.

Utility As for accuracy, first note that at each round t we choose as a query

$$Q_t(L_t) = 1 - \mathrm{freq}_{L_t}(h_{t-1}).$$

By our choice of parameters (and standard union bound), we have that with probability $(1 - 2\beta)$ the following happens at each round: Whenever the algorithm chooses $h_t = h_{t-1}$ we have that:

$$1 - \mathrm{freq}_{L_t}(h_t) = 1 - \mathrm{freq}_{L_t}(h_{t-1}) = Q_t(D) \leq \theta + \eta/32 = 1 - \eta/16 \Rightarrow \mathrm{freq}_{L_t}(h_t) \geq \eta/16,$$

and at each round that the algorithm calls *hist* we have by the guarantee of *hist* that:

$$\mathrm{freq}_{L_t}(h_t) \geq \eta/4,$$

and moreover

$$1 - \mathrm{freq}_{L_t}(h_{t-1}) = Q_t(L_t) \geq \theta - \eta/32 = 1 - \eta/8 \Rightarrow \mathrm{freq}_{L_t}(h_{t-1}) \leq \eta/8.$$

A.5 Proof of Lemma 5.2

The main observation is that if we let (i, j) be the permutation that switches between i and j, a uniform randomly chosen permutation can be written as

$$\pi = (N, a_N) \circ ((N-1), a_{N-1}) \circ \ldots \circ (3, a_3) \circ (2, a_2),$$

where each a_i is an independent random variable distributed uniformly on the set $\{1, \ldots, i\}$. An equivalent way to generate n random variables $\bar{Z} = (Z_1, \ldots, Z_n)$ sampled without replacement from $\mathfrak{Z} = \{z^{(1)}, \ldots, z^{(N)}\}$ is as follows: first choose a permutation π uniformly at random, then set $(i_1, \ldots, i_n) = (\pi(N), \ldots, \pi(N - n + 1))$, and finally set $(Z_1, \ldots, Z_n) = (z^{(i_1)}, \ldots, z^{(i_n)})$. In particular, the random variable $\bar{Z} = (Z_1, \ldots, Z_n)$ is completely determined by the independent random variables $a_N, \ldots, a_{N-n+1}$. Let us write this mapping from $a_N, \ldots, a_{N-n+1}$ to $Z_1, \ldots, Z_n$ as $(Z_1, \ldots, Z_n) = G(a_N, \ldots, a_{N-n+1})$. Also note that changing a single variable a_i changes at most the position of 3 elements of $G(a_N, \ldots, a_{N-n+1})$. Hence, via the triangle inequality, we obtain that, for any tuples $\bar{a} = (a_N, \ldots, a_{N-n+1})$ and $\bar{a}' = (a'_N, \ldots, a'_{N-n+1})$ that are of Hamming distance at most 1,

$$|F(G(\bar{a})) - F(G(\bar{a}'))| \leq 3c.$$

Thus, considering $F \circ G$ as a function of $a_N, \ldots, a_{N-n+1}$, we obtain the desired result via the standard Mcdiarmid's inequality.