Technical Appendices and Supplementary Material

A Preliminaries

A.1 Basic Notation

Notation A.1. $\mathbb{N} = \{1, 2, 3, \ldots\}$, *i.e.,* $0 \notin \mathbb{N}$. $\log(\cdot)$ *and* $\ln(\cdot)$ *denote logarithm to base* 2 *and* e, *respectively.*

Notation A.2 (Sequences)**.** *Let $\mathcal{X}$ be a set and $n, k \in \mathbb{N}$. For a sequence $x = (x_1, \ldots, x_n) \in \mathcal{X}^n$, we write $x_{\leq k}$ to denote the subsequence $(x_1, \ldots, x_k)$. If $k \leq 0$ then $x_{\leq k}$ denotes the empty sequence, which is also denoted by $\lambda = \mathcal{X}^0$. We use the notation $\mathcal{X}^{\leq n} = \cup_{i=0}^{n} \mathcal{X}^i$.*

A.2 Standard Online Learning

Let $\mathcal{X}$ be a set, and let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ be a collection of functions called a *hypothesis class*. A *learner strategy* or simply *learner* for the standard online learning game (Game 1) is a function

$$L : \bigcup_{i=0}^{n-1} (\mathcal{X} \times \{0,1\})^i \times \mathcal{X} \to \{0,1\},$$

where $n \in \mathbb{N}$ is the number of rounds in the game. The set of all such learner strategies is denoted $\mathcal{L}_n$. An *adversary strategy* or simply *adversary* for the standard online learning game is a pair of functions

$$A_{\mathsf{instance}} : \bigcup_{i=0}^{n-1} (\mathcal{X} \times \{0,1\} \times \{0,1\})^i \to \mathcal{X}, \text{ and}$$
$$A_{\mathsf{label}} : \bigcup_{i=1}^{n-1} (\mathcal{X} \times \{0,1\} \times \{0,1\})^i \times \{0,1\} \to \{0,1\}.$$

The set of all such adversary strategies is denoted $\mathcal{A}_n$.

Semantically, the interpretation of these strategies is that in each round $t \in [n]$ of Game 1, the adversary selects an instance

$$x_t = A_{\mathsf{instance}}(x_1, \hat{y}_1, y_1, \ldots, x_{t-1}, \hat{y}_{t-1}, y_{t-1}) \in \mathcal{X},$$

then the learner makes a prediction

$$\hat{y}_t = L(x_1, y_1, \ldots, x_{t-1}, y_{t-1}, x_t) \in \{0,1\},$$

and finally, the adversary assigns a label

$$y_t = A_{\mathsf{label}}(x_1, \hat{y}_1, y_1, \ldots, x_{t-1}, \hat{y}_{t-1}, y_{t-1}, \hat{y}_t) \in \{0,1\}.$$

The adversary's function A_{label} must satisfy *realizability*, meaning that there exists $h \in \mathcal{H}$ such that

$$\forall t \in [n]: \ y_t = h(x_t).$$

The number of mistakes is in a game with n rounds and hypothesis class $\mathcal{H}$ between learner L and adversary A is

$$M_{\mathsf{std}}(\mathcal{H}, n, L, A) = |\{t \in [n] : \ \hat{y}_t \neq y_t\}|.$$

A.3 Transductive Online Learning

Given $\mathcal{X}$ and $\mathcal{H}$ as in Appendix A.2, a learner strategy for the *transductive online learning setting* (Game 2) is a function

$$L : \mathcal{X}^n \times \bigcup_{i=0}^{n-1} \{0,1\}^i \to \{0,1\},$$

where $n \in \mathbb{N}$ is the number of rounds in the game. An adversary strategy consists of a sequence $x \in \mathcal{X}^n$ and an *adversary labeling strategy*, which is a function

$$A : \left(\bigcup_{i=0}^{n-1} \{0,1\}^{2i}\right) \times \{0,1\} \to \{0,1\}.$$

The sets of all such learner and adversary strategies are denoted $\mathcal{L}_n$ and $\mathcal{A}_n$ respectively.

Semantically, the interpretation of these strategies is that at the start of Game 2, the adversary selects the sequence x. Then, in each round $t \in [n]$, the learner makes a prediction

$$\hat{y}_t = L(x, y_1, \ldots, y_{t-1}) \in \{0, 1\},$$

and then the adversary assigns a label

$$y_t = A(\hat{y}_1, y_1, \ldots, \hat{y}_{t-1}, y_{t-1}, \hat{y}_t) \in \{0, 1\}.$$

Exactly as in Appendix A.2, the adversary's function A must satisfy realizability, namely,

$$\exists h \in \mathcal{H}\ \forall t \in [n]:\ y_t = h(x_t),$$

and the number of mistakes is in a game with sequence length n and hypothesis class $\mathcal{H}$ between learner L and adversary A is

$$M_{\mathsf{tr}}(\mathcal{H}, n, L, A) = |\{t \in [n]:\ \hat{y}_t \neq y_t\}|.$$

A.4 Mistake Bounds

In this paper, we study *optimal mistake bounds*, or the *optimal number of mistakes*, which is the value of Games 1 and 2. For $M \in \{M_{\mathsf{std}}, M_{\mathsf{tr}}\}$, the optimal number of mistakes in a game with hypothesis class $\mathcal{H}$ and sequence length n is,

$$M(\mathcal{H}, n) = \sup_{A \in \mathcal{A}_n} \inf_{L \in \mathcal{L}_n} M(\mathcal{H}, n, L, A).$$

The optimal number of mistakes for hypothesis class $\mathcal{H}$ is

$$M(\mathcal{H}) = \sup_{n \in \mathbb{N}} M(\mathcal{H}, n).$$

Remark A.3. As is common in learning theory literature, in both Game 1 and Game 2, we take the sets $\mathcal{L}_n$ and $\mathcal{A}_n$ to be the sets of all (deterministic) functions. In this paper, we do not consider randomized strategies. By allowing arbitrary functions, we ignore issues relating to computability.

A.5 Trees

Definition A.4 (Notation for binary trees)**.** *Let $d \in \mathbb{N} \cup \{0\}$. A perfect binary tree of depth d is a collection of $2^{d+1} - 1$ nodes, which we identify with the collection of binary strings*

$$T_d = \left\{\{0, 1\}^k :\ k \in \{0, 1, 2, \ldots, d\}\right\}.$$

The empty string, denoted $\lambda = \{0, 1\}^0$, is a member of T_d and is called the root of the tree. Every string $u \in \{0, 1\}^d$ is called a leaf. The depth of a node $u \in T_d$, denoted $|u|$, is the length of u as a string, namely, the integer k such that $u \in \{0, 1\}^k$.

For two nodes $u, v \in T_d$, we say that u is a parent of v, and that v is a child of u, if $v = u \circ 0$ or $v = u \circ 1$, where $\circ$ denotes string concatenation. More fully, for $b \in \{0, 1\}$, we say that v is a b-child of u if $v = u \circ b$.

Recursively, we define that u is an ancestor of v and that v is a descendant of u, and write $u \preccurlyeq v$, if one of the following holds:

- $v = ufunction \in T_d :\ u \preccurlyeq w \preccurlyeq v.$

For $b \in \{0, 1\}$, we say that v is a b-descendant of u, denoted $u \preccurlyeq_b v$, if v is a descendant of the b-child of u.

Definition A.5 (Paths in a binary tree)**.** *Let $d, k \in \mathbb{N}$, $k \leq d$. Let $u \in \{0, 1\}^k$ be a node in T_d. The path to u is the unique sequence $\mathrm{path}(u) = (u_0, u_1, u_2, \ldots, u_k)$ such that $u_0 = \lambda$ is the root, $u_k = u$, and u_i is a child of u_{i-1} for all $i \in [k]$.*

Let $f : T_d \to \{0, 1\}$ be a function. The path of f is the unique sequence $\mathrm{path}(f) = (u_0, u_1, u_2, \ldots, u_d)$ such that $u_0 = \lambda$ is the root, and for each $i \in [d]$, $u_i = u_{i-1} \circ f(u_{i-1})$. Namely, u_i is the $f(u_{i-1})$-child of u_{i-1}.

For a hypothesis class $\mathcal{H} \subseteq \{0, 1\}^{T_d}$, we define $\mathrm{path}(\mathcal{H}) = \mathrm{path}(f_{\mathsf{maj}})$, where $f_{\mathsf{maj}} :\ T_d \to \{0, 1\}$ is given by

$$f_{\mathsf{maj}}(u) = \mathbb{1}\left(\frac{|\{h \in \mathcal{H} :\ h(u) = 1\}|}{|\mathcal{H}|} \geq \frac{1}{2}\right).$$

A.6 Littlestone Dimension

Definition A.6 (Littlestone, 1987). *Let $\mathcal{X}$ be a set, let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$, and let $d \in \mathbb{N} \cup \{0\}$. We say that $\mathcal{H}$ shatters the binary tree T_d if there exists a mapping $T_d \to \mathcal{X}$ given by $u \mapsto x_u$ such that for every $u \in \{0,1\}^{d+1}$ there exists $h_u \in \mathcal{H}$ such that*

$$\forall i \in [d+1] : \; h(x_{u_{\leq i-1}}) = u_i.$$

The Littlestone dimension of $\mathcal{H}$, denoted $\mathsf{LD}(\mathcal{H})$, is the supremum over all $d \in \mathbb{N}$ such that there exists a Littlestone tree of depth $d-1$ that is shattered by $\mathcal{H}$.

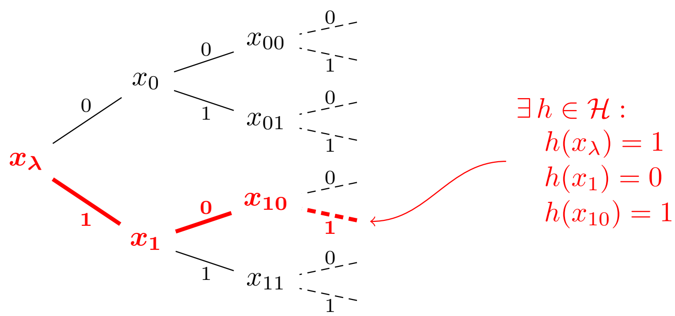

Figure 2: A shattered Littlestone tree of depth 2. The empty sequence is denoted by λ.
(Source: Bousquet et al., 2021)

Theorem A.7 (Littlestone, 1987). *Let $\mathcal{X}$ be a set and let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ such that $d = \mathsf{LD}(\mathcal{H}) < \infty$. Then there exists a strategy for the learner that guarantees that the learner will make at most d mistakes in the standard (non-transductive) online learning setting, regardless of the adversary's strategy and of the number n of instances to be labeled. Furthermore, there exists an adversary that forces every learner to make at least $\min\{n, d\}$ mistakes.*

B Lower Bound

B.1 Statement

Our $\Omega\left(\sqrt{d}\right)$ lower bound states the following.

Theorem B.1 (Lower bound). *There exists a constant $d_0 \geq 0$ as follows. Let $d \in \mathbb{N}$, $d \geq d_0$, let $\mathcal{X}$ be a set, and let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ be a hypothesis class with $\mathsf{LD}(\mathcal{H}) = d$. Then for every $k \in [d]$ there exist a sequence $x \in \mathcal{X}^n$ of length $n = O\left(k \cdot 2^{\sqrt{k}}\right)$ such that for every learning rule L there exists $h \in \mathcal{H}$ such that*

$$M_{\mathsf{tr}}(L, x, h) \geq \sqrt{k}. \tag{3}$$

Moreover, this is witnessed by a simple adaptive labeling strategy for the adversary (as in Algorithm 1) that forces every learning rule to make at least $\sqrt{k}$ mistakes on the (same) sequence x. Furthermore, for any integer $n \in \mathbb{N}$,

$$M_{\mathsf{tr}}(\mathcal{H}, n) \geq \min\left\{\sqrt{d}, \lfloor \log(n) \rfloor\right\}. \tag{4}$$

See Section 2.2 for a general overview of this result and the main proof ideas. In the following subsections we prove Theorem B.1. Algorithm 1 gives an explicit construction of the adversary that witnesses the lower bound, using Algorithm 2 as a subroutine. We start with presenting some initial observations about the behavior of these algorithms in Appendix B.2.

Assumptions:

- $d \in \mathbb{N}$, $\varepsilon = 2^{-\sqrt{d}/2}$.
- $T = T_d$ is a perfect binary tree of depth d.
- $\mathcal{H} \subseteq \{0,1\}^T$ is a class that shatters T.

TRANSDUCTIVEADVERSARY $(\mathcal{H})$:

$(x_1, x_2, \ldots, x_n) \leftarrow$ CONSTRUCTSEQUENCE $(\mathcal{H})$ ▷ See Algorithm 2.
send $(x_1, x_2, \ldots, x_n)$ to learner
$\mathcal{H}_0 \leftarrow \mathcal{H}$
for $t \in [n]$:
  receive $\hat{y}_t$ from learner
  $r_t \leftarrow \dfrac{|\{h \in \mathcal{H}_{t-1} : h(x_t) = 1\}|}{|\mathcal{H}_{t-1}|}$
  $y_{\mathsf{maj}} \leftarrow \mathbb{1}(r_t \geq 1/2)$
  $y_t \leftarrow \begin{cases} y_{\mathsf{maj}} & r_t \notin [\varepsilon, 1-\varepsilon] \\ 1 - \hat{y}_t & \text{otherwise} \end{cases}$
  send y_t to learner
  $\mathcal{H}_t \leftarrow \{h \in \mathcal{H}_{t-1} : h(x_t) = y_t\}$

Algorithm 1: The strategy for the adversary that achieves the lower bound in Theorem B.1. Note that while the construction of the sequence x is not entirely trivial, the adversary's strategy for labeling this sequence is very simple.

B.2 Analysis of the Adversary

Claim B.2. *Let $d \in \mathbb{N}$, let $M = \sqrt{d}/10$, and let $\mathcal{H} \subseteq \{0,1\}^{T_d}$ be a hypothesis class. Consider an execution of* CONSTRUCTSEQUENCE $(\mathcal{H})$ *as in Algorithm 2 that produces a sequence $x_1, x_2, \ldots, x_t$. Then:*

(a) For all $i \in [t]$, $\mathrm{path}(x_i)$ is a subsequence of $x_0, x_1, \ldots, x_i$.

(b) The length t of the sequence satisfies $t < n_d$, where $n_d = (d+1) \cdot 2^{M+1}$.

Proof.

(a) Fix $i \in [t]$. It suffices to show that for all $u \in T_d$, if $u \preccurlyeq x_i$ then $u \in (x_1, x_2, \ldots, x_i)$. Proceed by induction on i. For the base case $i = 1$, the claim holds because $x_1 = \lambda$.

For the induction step, assume the claim holds for $i \in [t-1]$. Let $u \preccurlyeq x_{i+1}$, we prove that $u \in (x_1, x_2, \ldots, x_{i+1})$. Assume $x_{i+1} \neq \lambda$ (otherwise, there is nothing to prove).

Because x_{i+1} appears in the sequence x, it must have been added to $\mathcal{Q}$ before it was added to x. The only place where items that are not λ are added to $\mathcal{Q}$ is in the line $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{x_t \circ y\}$. Namely, there exist an index $j \in [i]$ and a bit $y \in \{0,1\}$ such that $x_{i+1} = x_j \circ y$ (note that $j < i+1$ because x_j was added to the sequence before x_{i+1}). If $x_j = u$ we are done. Otherwise, note that x_j is the parent of x_{i+1}, and therefore $u \preccurlyeq x_j$. By the induction hypothesis, $u \in (x_1, x_2, \ldots, x_j)$. This concludes the proof.

(b) Items are added to the the sequence x only if they were previously added to $\mathcal{Q}$. Therefore, the length of the sequence x is $t = U + 1$, where U is the number of times that the line "$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{x_t \circ y\}$" was executed (we add 1 because λ is added to $\mathcal{Q}$ elsewhere).

Consider a function f that maps a node u in the sequence x to the value of the index b' at the time that u was added to $\mathcal{Q}$. Namely, if $u = x_t \circ y$ and the index b' had some value β when the line "$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{x_t \circ y\}$" was executed, then $f(u) = \beta$.

Notice that "$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{x_t \circ y\}$" is executed only if the condition $x_t \in \mathrm{path}(\mathcal{H}_{b'})$ is satisfied in the previous line. By Item (a), the line "$\mathcal{H}_{b'} \leftarrow \{h \in \mathcal{H}_b : h(x_t) = y\}$", and the fact that items in $\mathcal{Q}$ are processed in lexicographic order, it follows that there exists some assignment of labels $\ell : \ \mathrm{path}(x_t) \to \{0,1\}$ such that

$$\mathcal{H}_{b'} \subseteq \{h \in \mathcal{H} : \ (\forall u \in \mathrm{path}(x_t) : \ h(u) = \ell(u))\}\,.$$

Assumptions:

- $d \in \mathbb{N}$, $M = \sqrt{d}/10$, $\varepsilon = 2^{-\sqrt{d}/2}$.
- $T = T_d$ is a perfect binary tree of depth d.
- λ, the empty string, is the root of T.
- $\mathcal{H} \subseteq \{0,1\}^T$ is a class that shatters T.

CONSTRUCTSEQUENCE ($\mathcal{H}$):

$\mathcal{H}_\lambda \leftarrow \mathcal{H}$

$\mathbb{H}_0 \leftarrow \{\mathcal{H}_\lambda\}$ ▷ A set of classes indexed by bit strings.

$\mathcal{Q} \leftarrow \{\lambda\}$ ▷ A set of nodes to be processed.

$t \leftarrow 0$

while $|\mathcal{Q}| > 0$:

  $t \leftarrow t + 1$

  $x_t \leftarrow \min_{\leq_{\text{lex}}} \mathcal{Q}$ ▷ Pop the lexicographically first string from $\mathcal{Q}$ and add it to the output sequence.
  $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{x_t\}$

  $\mathbb{H}_t \leftarrow \varnothing$

  for $\mathcal{H}_b \in \mathbb{H}_{t-1}$:

$$r \leftarrow \frac{|\{h \in \mathcal{H}_b : \ h(x_t) = 1\}|}{|\mathcal{H}_b|}$$

$$\mathcal{Y} \leftarrow \begin{cases} \{0,1\} & \big(r \in [\varepsilon, 1-\varepsilon]\big) \wedge \big(|b| \leq M\big) \\ \{\mathbb{1}(r \geq 1/2)\} & \text{otherwise} \end{cases}$$ ▷

Adversary will force mistakes on the first M balanced nodes.

   for $y \in \mathcal{Y}$:

$$b' \leftarrow \begin{cases} b & |\mathcal{Y}| = 1 \\ b \circ y & |\mathcal{Y}| = 2 \end{cases}$$ ▷ Restrict class to agree with y. If splitting the class in two to force a mistake then create new indices.

    $\mathcal{H}_{b'} \leftarrow \{h \in \mathcal{H}_b : h(x_t) = y\}$

    $\mathbb{H}_t \leftarrow \mathbb{H}_t \cup \{\mathcal{H}_{b'}\}$

    if $x_t \in \mathrm{path}(\mathcal{H}_{b'}) \ \wedge \ |x_t| < d$: ▷ If x_t is on-path for $\mathcal{H}_{b'}$ and it has a y-child, add that child to $\mathcal{Q}$.
     $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{x_t \circ y\}$

return $(x_1, x_2, \ldots, x_t)$

Algorithm 2: A subroutine of Algorithm 1 for selecting the sequence x.

Consequently, $x_t \in \mathrm{path}(\mathcal{G})$ for any class $\mathcal{G}$ that is a restriction of $\mathcal{H}_{b'}$; in particular, because the only way that $\mathcal{H}_{b'}$ might be modified later during the execution of Algorithm 2 is by restricting it further, it follows that $x_t \in \mathrm{path}(\mathcal{H}_{b'})$ when the line "$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{x_t \circ y\}$" is executed and in all subsequent times.

However, $|\mathrm{path}(\mathcal{G})| = d + 1$ for any class $\mathcal{G} \subseteq \{0,1\}^{T_d}$. This implies that f maps at most $(d+1)$ nodes to each bit string.[19] In other words, for any bit string b, the size of the preimage satisfies $|f^{-1}(b)| \leq d+1$.

Thus,

$$\begin{aligned} t &= 1 + |\{2,3,\ldots,t\}| \\ &= 1 + \sum_{\substack{b \in \{0,1\}^k \\ k \leq M}} |\{i \in \{2,3,\ldots,t\} : \ f(x_i) = b\}| \\ &= 1 + \sum_{\substack{b \in \{0,1\}^k \\ k \leq M}} |f^{-1}(b)| \\ &\leq 1 + \sum_{\substack{b \in \{0,1\}^k \\ k \leq M}} (d+1) \\ &\leq 1 + (d+1) \cdot (2^{M+1} - 1). \\ &< (d+1) \cdot 2^{M+1}, \end{aligned}$$

as desired. □

Claim B.3. *Let $d \in \mathbb{N}$, and let $\mathcal{H} \subseteq \{0,1\}^{T_d}$ be a hypothesis class. Consider an execution of* TRANSDUCTIVEADVERSARY $(\mathcal{H})$ *as in Algorithm 1. Let*

$$\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_n$$

be the sequence of hypothesis classes created by TRANSDUCTIVEADVERSARY*, and let*

$$\mathbb{H}_0, \mathbb{H}_1, \ldots, \mathbb{H}_n$$

be the sequence of collections created by the subroutine CONSTRUCTSEQUENCE *(Algorithm 2). Then*

$$\forall t \in \{0,1,\ldots,n\} : \ \mathcal{H}_t \in \mathbb{H}_t.$$

Proof. Proceed by induction on $t \in \{0,1,\ldots,n\}$. The base case $t = 0$ is satisfied, because $\mathcal{H}_0 = \mathcal{H} \in \{\mathcal{H}\} = \mathbb{H}_0$. For the induction step, assume that $\mathcal{H}_{i-1} \in \mathbb{H}_{i-1}$ for some $i \in [n]$. We prove that $\mathcal{H}_i \in \mathbb{H}_i$.

Let y_i be the label assigned to x_i by TRANSDUCTIVEADVERSARY. Then

$$\mathcal{H}_i = \{h \in \mathcal{H}_{i-1} : \ h(x_i) = y_i\}.$$

Consider the iteration of the while loop in CONSTRUCTSEQUENCE that starts with $t \leftarrow i$. By the induction hypothesis, $\mathcal{H}_{i-1} \in \mathbb{H}_{i-1}$. Therefore, in this iteration of the while loop, there will be an iteration of the "for $\mathcal{H}_b \in \mathbb{H}_{t-1}$" loop where $\mathcal{H}_b = \mathcal{H}_{i-1}$. In that iteration, $y_i \in \mathcal{Y}$ by construction of y_i and $\mathcal{Y}$. Therefore, in the iteration of the "for $y \in \mathcal{Y}$" loop in which $y = y_i$,

$$\mathcal{H}_{b'} = \{h \in \mathcal{H}_b : h(x_t) = y\} = \{h \in \mathcal{H}_{i-1} : h(x_i) = y_i\} = \mathcal{H}_i.$$

The class $\mathcal{H}_{b'}$ is then added to $\mathbb{H}_i = \mathbb{H}_t$ in the line "$\mathbb{H}_t \leftarrow \mathbb{H}_t \cup \{\mathcal{H}_{b'}\}$". Furthermore, no class is ever removed from $\mathbb{H}_t$. So $\mathcal{H}_i \in \mathbb{H}_i$, as desired. □

[19] Note that $x_t \circ y$ could be added to $\mathcal{Q}$ twice: once with $y = 0$ and a second time with $y = 1$. However, when this happens, the index b is replaced by two indices of the form $b' = b \circ y$; for each such b', there just is a single node of the form $x_t \circ y$ such that $f(x_t \circ y) = b'$.

Claim B.4. *Let $d \in \mathbb{N}$, let $k \in \{0,1,\dots,d\}$, and let $\mathcal{H} \subseteq \{0,1\}^{T_d}$ be a hypothesis class. Consider an execution of* TRANSDUCTIVEADVERSARY $(\mathcal{H})$ *as in Algorithm 1 where the adversary constructs a sequence of nodes $x_1, x_2, \dots, x_n \in T_d$ and a sequence of classes $\mathcal{H}_0, \mathcal{H}_1, \dots, \mathcal{H}_n \subseteq \{0,1\}^{T_d}$. Then there exists $i \in [n]$ such that*

1. $|x_i| = k$, *and*
2. $x_i \in \mathrm{path}(\mathcal{H}_{i-1})$.

Proof. Proceed by induction on k. For the base case $k = 0$, notice that $x_1 = \lambda$, $|\lambda| = 0$, and $\lambda \in \mathrm{path}(\mathcal{H}_0)$.

For the induction step, assume the claim holds for some $k \in \{0,1,\dots,d-1\}$, and let $i_k \in [n]$ such that $|x_{i_k}| = k$ and $x_{i_k} \in \mathrm{path}(\mathcal{H}_{i_k - 1})$; we prove that the claim holds for $k+1$ as well.

Consider the iteration of the while loop in CONSTRUCTSEQUENCE in which x_{i_k} is added to the sequence (i.e., the iteration starting with $t \leftarrow i_k$). By Claim B.3, $\mathcal{H}_{i_k-1} \in \mathbb{H}_{i_k-1}$. Hence, within this iteration of the while loop, there is an iteration of the "for $\mathcal{H}_b \in \mathbb{H}_{t-1}$" loop such that $\mathcal{H}_b = \mathcal{H}_{i_k-1}$. By construction, the set $\mathcal{Y}$ always contains the label predicted by the adversary, so $y_{i_k} \in \mathcal{Y}$. Because $x_{i_k} \in \mathrm{path}(\mathcal{H}_{i_k-1}) = \mathrm{path}(\mathcal{H}_b)$, it follows that $x_{i_k} \in \mathrm{path}(\mathcal{H}_{b'})$. Seeing as $|x_{i_k}| < d$, in the last line of the iteration of the "for $y \in \mathcal{Y}$" loop with $y = y_{i_k}$, the node $x_{i_{k+1}} := x_{i_k} \circ y_{i_k}$ is added to $\mathcal{Q}$. This guarantees that $x_{i_{k+1}}$ will eventually be popped from $\mathcal{Q}$ and added to the sequence returned by CONSTRUCTSEQUENCE. Once a node has been added to the sequence, it is never removed.

Notice that $|x_{i_{k+1}}| = |x_{i_k}| + 1 = k+1$, satisfying Item 1. Therefore, it remains to show Item 2, namely, to show that $x_{i_{k+1}} \in \mathrm{path}\big(\mathcal{H}_{i_{k+1}-1}\big)$.

Let $(\lambda = u_0, u_1, u_2, \dots, u_k = x_{i_k}) = \mathrm{path}(x_{i_k})$ be the root-to-node path to x_{i_k}. By Item (a) in Claim B.2, $\mathrm{path}(x_{i_k})$ is a subsequence of $x_{\leq i_k}$. By the construction of the version space $\mathcal{H}_{i_k}$, $\mathcal{H}_{i_k}$ is restricted to agree with the root-to-node path to x_{i_k}, as well as with the label y_{i_k}.[20] So $\mathcal{H}_{i_k}$ is restricted to agree with the root-to-node path to $x_{i_k} \circ y_{i_k} = x_{i_{k+1}}$. This implies that,

$$\forall \mathcal{G} \subseteq \mathcal{H}_{i_k} : \; x_{i_{k+1}} \in \mathrm{path}(\mathcal{G}).$$

In particular, because $\mathcal{H}_{i_{k+1}-1} \subseteq \mathcal{H}_{i_k}$, it follows that $x_{i_{k+1}} \in \mathrm{path}\big(\mathcal{H}_{i_{k+1}-1}\big)$, as desired. □

B.3 Proof

Claim B.5. *Let $d \in \mathbb{N}$, $d \geq 800$, and let $\mathcal{H} \subseteq \{0,1\}^{T_d}$ with cardinality $|\mathcal{H}| = 2^{d+1}$ be a hypothesis class that shatters T_d. Consider an execution of* TRANSDUCTIVEADVERSARY $(\mathcal{H})$ *as in Algorithm 1. Let $x_1, \dots, x_n$ and $y_1, \dots, y_n$ be nodes and labels selected by the adversary, let $\mathcal{H}_0, \mathcal{H}_1, \dots, \mathcal{H}_n$ be the hypothesis classes defined by the adversary, and let $\hat{y}_1, \dots, \hat{y}_n$ be the labels selected by the learner.*

Let

$$S = \{s_0, s_1, s_2, \dots\} = \big\{t \in [n] : \; r_t \in [\varepsilon, 1-\varepsilon]\big\}$$

be the set of indices where the adversary forces a mistake.

For each $k \in \{0,1,2,\dots,\lfloor\sqrt{d}\rfloor\}$, define

$$s_{\mathsf{max}}(k) = \max\left\{t \in [n] : \; |x_t| \leq k\sqrt{d}\right\}.$$

Then $|S| \geq \lfloor\sqrt{d}\rfloor + 1$, and for all $k \in \{0,1,2,\dots,\lfloor\sqrt{d}\rfloor\}$:

1. $s_k \leq s_{\mathsf{max}}(k)$, *and*
2. $|\mathcal{H}_{s_k}| \geq 2^{d-k\sqrt{d}}$.

[20] Formally, there exists a mapping $j : \; \mathrm{path}(x_{i_k}) \to [i_k]$ such that for each $u \in \mathrm{path}(x_{i_k})$, $u = x_{j(u)}$, and $\mathcal{H}_{i_k} \subseteq \big\{h \in \mathcal{H} : \; \big(\forall u \in \mathrm{path}(x_{i_k}) : \; h(u) = y_{j(u)}\big)\big\}$.

Proof. Proceed by induction on k. For the base case $k = 0$, $s_{\mathsf{max}}(0) = 1$ because $x_1 = \lambda$ is the root, and all other nodes in the sequence are strictly deeper than the root. Because $\mathcal{H}$ shatters T_d and $|\mathcal{H}| = 2^{d+1}$, $\mathcal{H}$ is perfectly balanced on the root λ, meaning that for all $\upsilon \in \{0, 1\}$, if we denote

$$\mathcal{H}_{(\upsilon)} = \{h \in \mathcal{H} : \; h(\lambda) = \upsilon\} \,,$$

then

$$\left|\mathcal{H}_{(\upsilon)}\right| = \frac{1}{2} \cdot 2^{d+1} = 2^d. \tag{5}$$

So $r_1 = 1/2$, and therefore the adversary forces a mistake on index $s_0 = s_{\mathsf{max}}(0) = 1$, satisfying Item 1. Furthermore, Eq. (5) implies,

$$|\mathcal{H}_{s_0}| = |\mathcal{H}_1| = \left|\mathcal{H}_{(y_1)}\right| = 2^d = 2^{d-k\sqrt{d}},$$

satisfying Item 2.

For the induction step, assume for some $k \in \{0, 1, 2, \ldots, \left\lfloor\sqrt{d}\right\rfloor - 1\}$ that $|S| \geq k + 1$ and Items 1 and 2 hold for k. We show that $|S| \geq k + 2$ and Items 1 and 2 hold for $k + 1$ as well. To establish $|S| \geq k + 1$ and Item 1 for $k + 1$, it suffices to show that there exists an index t such that $s_k < t \leq s_{\mathsf{max}}(k + 1)$ and $r_t \in [\varepsilon, 1 - \varepsilon]$. Assume for contradiction that no such index exists. Then,

$$\begin{aligned}
\left|\mathcal{H}_{s_{\mathsf{max}}(k+1)}\right| &\geq |\mathcal{H}_{s_k}| \cdot \prod_{t=s_k+1}^{s_{\mathsf{max}}(k+1)} \max\{r_t, 1 - r_t\} && \begin{array}{l}(r_t \notin [\varepsilon, 1-\varepsilon] \text{ implies } y_t = y_{\mathsf{maj}}, \\ \text{so larger subclass survives})\end{array} \\
&\geq 2^{d-k\sqrt{d}} \cdot \prod_{t=s_k+1}^{s_{\mathsf{max}}(k+1)} \max\{r_t, 1 - r_t\} && (\text{Induction hypothesis Item 2}) \\
&\geq 2^{d-k\sqrt{d}} \cdot (1 - \varepsilon)^{n_d} && (r_t \notin [\varepsilon, 1-\varepsilon]\text{; Item (b) in Claim B.2}) \\
&\geq 2^{d-k\sqrt{d}-1},
\end{aligned} \tag{6}$$

where the last inequality holds because

$$(1 - \varepsilon)^{n_d} = \left(1 - 2^{-\sqrt{d}/2}\right)^{(d+1)\cdot 2^{\sqrt{d}/10+1}} \geq \frac{1}{2} \tag{7}$$

for our choice of $d \geq 800$.

On the other hand, by Claim B.4, there exists $t \in [s_{\mathsf{max}}(k + 1)]$ such that $|x_t| = (k + 1)\sqrt{d}$ and $x_t \in \mathrm{path}(\mathcal{H}_{t-1})$. By Item (a) in Claim B.2 and the construction of $\mathcal{H}_t$, the class $\mathcal{H}_t$ agrees with the root-to-node path to x_t. Namely, there exists an assignment of labels $\ell : \; \mathrm{path}(x_t) \to \{0, 1\}$ such that

$$\mathcal{H}_t \subseteq \left\{h \in \mathcal{H} : \; \left(\forall u \in \mathrm{path}(x_t) : \; h(u) = \ell(u)\right)\right\}.$$

Seeing as $|\mathrm{path}(x_t)| = |x_t| + 1 = (k + 1)\sqrt{d} + 1$, this implies that

$$\left|\mathcal{H}_{s_{\mathsf{max}}(k+1)}\right| \leq |\mathcal{H}_t| \leq 2^{d+1} 2^{-(k+1)\sqrt{d}-1} = 2^{d-(k+1)\sqrt{d}}. \tag{8}$$

Combining Eqs. (6) and (8) gives

$$2^{d-k\sqrt{d}-1} \leq \left|\mathcal{H}_{s_{\mathsf{max}}(k+1)}\right| \leq 2^{d-(k+1)\sqrt{d}}.$$

Namely,

$$d - k\sqrt{d} - 1 \leq d - (k + 1)\sqrt{d} \quad \Longrightarrow \quad d \leq 1,$$

which is a contradiction to our choice of $d \geq 800$. This implies that $|S| \geq k + 2$ and Item 1 holds for $k + 1$, namely, the index $s_{k+1} \in S$ satisfies $s_{k+1} \leq s_{\mathsf{max}}(k + 1)$.

Item 2 follows by a calculation similar to Eq. (6). Seeing as s_{k+1} is the first index after s_k where a mistake is forced,

$$\begin{aligned}
\left|\mathcal{H}_{s_{k+1}}\right| &\geq \varepsilon \cdot (1 - \varepsilon)^{s_{k+1}-s_k} \cdot |\mathcal{H}_{s_k}| \\
&\geq \varepsilon \cdot (1 - \varepsilon)^{n_d} \cdot |\mathcal{H}_{s_k}| && (\text{Item (b) in Claim B.2})
\end{aligned}$$

$$\begin{aligned}
&\geq \varepsilon \cdot (1-\varepsilon)^{n_d} \cdot 2^{d-k\sqrt{d}} && \text{(Induction hypothesis Item 2)}\\
&\geq 2^{-\sqrt{d}/2} \cdot 2^{d-k\sqrt{d}-1} && \text{(Eq. (7) and choice of } \varepsilon\text{)}\\
&\geq 2^{d-(k+1)\sqrt{d}},
\end{aligned}$$

as desired. □

Finally, we complete the proof of the lower bound.

Proof of Theorem B.1. Fix $d_0 = 800$ and assume $d \geq d_0$. Fix $k \leq d$. Seeing as $\mathsf{LD}(\mathcal{H}) = d$, $\mathcal{H}$ shatters the tree T_k. By replacing $\mathcal{H}$ with a suitable subset of $\mathcal{H}$ as necessary, assume without loss of generality that $\mathcal{H} \subseteq \{0,1\}^{T_k}$ and $|\mathcal{H}| = 2^{k+1}$.

Consider an execution of TRANSDUCTIVEADVERSARY $(\mathcal{H})$ as in Algorithm 1. By Item (b) in Claim B.2, Algorithm 1 constructs a sequence $x = (x_1, x_2, \ldots, x_n)$ of length $n < n_k$ for

$$n_k = (d+1) \cdot 2^{\sqrt{d}/10+1} = O\Big(d \cdot 2^{\sqrt{d}}\Big).$$

By Claim B.5, for every learning rule, the adversary forces at least $\left\lfloor \sqrt{k} \right\rfloor + 1 \geq \sqrt{k}$ mistakes on the sequence x, establishing Eq. (3) in Theorem B.1 and the "moreover" sentence that follows it.

Fix a length $n \in \mathbb{N}$. Let k be the largest integer such that $2^{\lceil \sqrt{k} \rceil} \leq n$ and $k \leq d$. By Eq. (3), there exists some sequence on which the adversary can force every learning rule to make at least $\sqrt{k}$ mistakes. By Theorem C.2, this implies that there exists a sequence of length $2^{\lceil \sqrt{k} \rceil} \leq n$ on which the adversary can force every learning rule to make at least $\sqrt{k} = \min\left\{\sqrt{d}, \lfloor \log(n) \rfloor\right\}$ mistakes. Namely,

$$M_{\mathsf{tr}}(\mathcal{H}, n) \geq \min\left\{\sqrt{d}, \lfloor \log(n) \rfloor\right\},$$

as in Eq. (4). □

C Sequence Length

In this section, we show that if there exists a sequence on which the adversary can force M mistakes, then a sequence of length $2^M - 1$ is sufficient, and this upper bound is tight for some classes.[21]

Definition C.1 (Minimal sequence)**.** *Let $\mathcal{X}$ be a set, let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ be a class, and let $M \in \mathbb{N}$. The <u>minimal sequence length for forcing M mistakes for the class $\mathcal{H}$</u>, denoted* $\mathsf{MinLen}(\mathcal{H}, M)$ *is*

$$\mathsf{MinLen}(\mathcal{H}, M) = \inf\ \{n \in \mathbb{N} :\ (\exists x \in \mathcal{X}^n :\ M_{\mathsf{tr}}(\mathcal{H}, x) \geq M)\}.$$

In words, $\mathsf{MinLen}(\mathcal{H}, M)$ *is the smallest integer n for which there exists a sequence of length n on which the adversary can force at least n mistakes; if no such sequence exists, then* $\mathsf{MinLen}(\mathcal{H}, M) = \infty$.

Theorem C.2 (Minimal sequence bound)**.** *Let $\mathcal{X}$ be a set, and fix $M \in \mathbb{N}$. Then for any class $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$, if* $\mathsf{MinLen}(\mathcal{H}, M) < \infty$ *then*

$$\mathsf{MinLen}(\mathcal{H}, M) \leq 2^M - 1.$$

Furthermore, there exists a class $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ for which $\mathsf{MinLen}(\mathcal{H}, M) = 2^M - 1$.

Theorem C.2 is a corollary of the tree rank characterization of M_{tr} from Ben-David et al. (1997). For completeness, we present a direct proof of Theorem C.2 that does not directly invoke that characterization. Roughly, given an adversary A_0 that forces every learner to make at least M mistakes on a (possibly long) sequence x, we apply two modifications to obtain new adversaries

$$A_0 \rightsquigarrow A_1 \rightsquigarrow A_2.$$

A_1 forces M mistakes and has a specific structure that we call 'rigidity', but it still uses the same (possibly long) sequence x. Capitalizing on the rigid structure, A_2 selects a subsequence of x of length at most $2^M - 1$, and forces M mistakes on that subsequence.

[21] Of course, there also exist classes for which a shorter sequence is sufficient. For instance, if the class shatters (in the VC sense) a subset of the domain of cardinality M, then a sequence of length M suffices.

C.1 Rigid Adversary

Definition C.3 (Rigid adversary)**.** *Let $n \in \mathbb{N}$, let $\mathcal{X}$ be a set, and let*

$$A: \left(\bigcup_{k=0}^{n-1} \{0,1\}^{2k}\right) \times \{0,1\} \to \{0,1\}$$

be an adversary strategy for some fixed sequence $x \in \mathcal{X}^n$. We say that A is rigid if there exists a function

$$f: \bigcup_{k=0}^{n-1} \{0,1\}^k \to \{0,1,\star\}$$

such that for all $k \in \{0,1,\ldots,n-1\}$ and all $y, \hat{y} \in \{0,1\}^k$,

$$A(\hat{y}_1, y_1, \ldots, \hat{y}_k, y_k, \hat{y}_{k+1}) = \begin{cases} f(y_1,\ldots,y_k) & f(y_1,\ldots,y_k) \in \{0,1\} \\ 1 - \hat{y}_{k+1} & f(y_1,\ldots,y_k) = \star \end{cases}.$$

Note that if an adversary is rigid, then the function f that witnesses this is uniquely determined.

Claim C.4 (Rigid adversary exists)**.** *Let $n, M \in \mathbb{N}$, let $\mathcal{X}$ be a set, let $x \in \mathcal{X}^n$, and let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ be a class. Let A be an adversary strategy that forces every learner to make at least M mistakes on x. Then there exists an adversary strategy A^* such that:*

1. *A^* forces every learner to make at least M mistakes on x and A^* is rigid.*
2. *Let f be the function that witnesses the rigidity of A^*. Then for every $y \in \{0,1\}^n$, the sequence*
$$f(y_{\leq 0}), f(y_{\leq 1}), f(y_{\leq 2}), \ldots, f(y),$$
has at least M members equal to $\star$.

Proof of Claim C.4. For Item 1, consider the adversary strategy A^* that simulates an execution of A, as in Algorithm 3. In broad strokes, A^* functions as a middle-man between the learner and A. As the learner makes a sequence of predictions $\hat{y} \in \{0,1\}^n$, the adversary A^* generates a sequence of (possibly different) predictions $\tilde{y} \in \{0,1\}^n$, and sends those to the adversary A. Adversary A sees only the predictions $\tilde{y}$, and assigns labels $y \in \{0,1\}^n$, which are relayed back to the learner by A^* with no modifications.

First, observe that A^* satisfies the realizability requirement. Indeed, A^* simulates an execution of A such that the sequence of labels $y_1, \ldots, y_n$ sent by A^* to the learner is exactly the sequence of labels selected by A. Seeing as A is realizable, every sequence of labels selected by A is realizable, and therefore every sequence of labels selected by A^* must be realizable as well.

Second, observe that A^* forces every leaner to make at least M mistakes. To see this, notice that in Algorithm 3,

$$\sum_{t \in [n]} \mathbb{1}(\tilde{y}_t \neq y_t) \geq M. \tag{9}$$

Indeed, A forces every learner to make at least M mistakes, and in particular this applies to a learner that makes predictions $\tilde{y}$ as in the simulation. Furthermore, observe that A^* only alters the predictions it receives from the learner in cases when it selects a label that is accepted by A, namely,

$$\forall t \in [n]: \ \tilde{y}_t \neq \hat{y}_t \implies \tilde{y}_t = y_t. \tag{10}$$

Therefore, if $E = \{t \in [n]: \ \tilde{y}_t = \hat{y}_t\}$, then

$$\begin{aligned} \sum_{t \in [n]} \mathbb{1}(\tilde{y}_t \neq y_t) &= \sum_{t \in E} \mathbb{1}(\tilde{y}_t \neq y_t) + \sum_{t \in [n] \setminus E} \mathbb{1}(\tilde{y}_t \neq y_t) \\ &= \sum_{t \in E} \mathbb{1}(\tilde{y}_t \neq y_t) + 0 && \text{(By Eq. (10))} \\ &= \sum_{t \in E} \mathbb{1}(\hat{y}_t \neq y_t) && \text{(Defintion of } E) \end{aligned}$$

Assumptions:

- $n \in \mathbb{N}$, $\mathcal{X}$ is a set, $x \in \mathcal{X}^n$ is a fixed sequence of instances.
- $A : \left(\bigcup_{k=0}^{n-1} \{0,1\}^{2k}\right) \times \{0,1\} \to \{0,1\}$ is an adversary labeling strategy for x.

RIGIDADVERSARY:

send $x_1, \dots, x_n$ to the learner

for $t = 1, 2, \dots, n$:

  receive prediction $\hat{y}_t$ from learner

  if $A(\tilde{y}_1, y_1, \dots, \tilde{y}_{t-1}, y_{t-1}, 0) = 0$:
    $\tilde{y}_t \leftarrow 0$
  else if $A(\tilde{y}_1, y_1, \dots, \tilde{y}_{t-1}, y_{t-1}, 1) = 1$:
    $\tilde{y}_t \leftarrow 1$
  else:
    $\tilde{y}_t \leftarrow \hat{y}_t$

  send prediction $\tilde{y}_t$ to A

  receive label y_t from A

  send label y_t to learner

Algorithm 3: Construction of a rigid adversary, by simulating a given adversary A.

$$\leq \sum_{t \in [n]} \mathbb{1}(\hat{y}_t \neq y_t). \tag{11}$$

Combining Eqs. (9) and (11) implies that A forces at least M mistakes.

Third, we show that A^* is rigid. We claim that there exists a function $g : \{0,1\}^{\leq n-1} \to \{0,1\}^{\leq n-1}$ such that for every $t \in \{0, 1, 2, \dots, n-1\}$,

$$(\tilde{y}_1, \dots, \tilde{y}_t) = g(y_1, \dots, y_t).$$

Proceed by induction on t. For the base case $t = 0$ there is nothing to prove. For the induction step, we assume the claim holds for some $t = k < n - 1$, and show that it holds for $t = k + 1$. From Algorithm 3, $\tilde{y}_{k+1}$ satisfies

$$\tilde{y}_{k+1} = \begin{cases} 0 & A(\tilde{y}_1, y_1, \dots, \tilde{y}_k, y_k, 0) = 0 \\ 1 & A(\tilde{y}_1, y_1, \dots, \tilde{y}_k, y_k, 0) = A(\tilde{y}_1, y_1, \dots, \tilde{y}_k, y_k, 1) = 1 \\ 1 - y_{k+1} & \text{otherwise} \end{cases}. \tag{12}$$

The first two cases in Eq. (12) are immediate from Algorithm 3, and the remaining case occurs when A forces a mistake at time $k+1$, namely, when A selects $y_{k+1} = 1 - \tilde{y}_{k+1}$. Thus, $\tilde{y}_{k+1}$ is a function of $y_{\leq k+1}$ and $\tilde{y}_{\leq k}$. By the induction hypothesis, $\tilde{y}_{\leq k} = g(y_{\leq k})$, so $\tilde{y}_{k+1}$ is simply a function of $y_{\leq k+1}$. This establishes the existence of the desired function g.

Hence, A^* is rigid, as witnessed by the function

$$f(y_1, \dots, y_k) = \begin{cases} 0 & A(\tilde{y}_1, y_1, \dots, \tilde{y}_k, y_k, 0) = 0 \\ 1 & A(\tilde{y}_1, y_1, \dots, \tilde{y}_k, y_k, 0) = A(\tilde{y}_1, y_1, \dots, \tilde{y}_k, y_k, 1) = 1 \\ \star & \text{otherwise} \end{cases},$$

where f is a well-defined function because $\tilde{y}_{\leq k} = g(y_{\leq k})$.

We have seen that A^* is a valid (realizable) adversary that forces every learner to make at least M mistakes, and it is rigid. This concludes the proof of Item 1.

Finally, For Item 2, note that $\tilde{y}_t \neq y_t$ only if A forces a mistake at time t in the sense that A selects $y_t = 1 - b$ for any prediction $b \in \{0, 1\}$ provided at time t. If A forces a mistake at time t, then A^*

forces a mistake at time t as well. Therefore, if $\tilde{y}_t \neq y_t$, then $f(y_{<t}) = \star$, namely, $\tilde{y}_t$ makes mistakes only when the value of f is $\star$. By Eq. (9), $\tilde{y}_t$ makes at least M mistakes throughout the game, so there must be at least M rounds where f outputs $\star$, as desired. $\square$

C.2 Essential Indices

Definition C.5. *Let $n, M \in \mathbb{N}$, let $\mathcal{X}$ be a set, let $x \in \mathcal{X}^n$, and let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ be a class. Let A be a rigid adversary strategy witnessed by function f. We say that an index $t \in [n]$ is essential for A for forcing M mistakes on x if there exists a sequence $y \in \{0,1\}^{t-1}$ such that $f(y) = \star$ and the sequence*

$$f(y_{\leq 0}), f(y_{\leq 1}), f(y_{\leq 2}), \ldots, f(y_{\leq t-1})$$

contains at most $M-1$ members equal to $\star$.

Claim C.6. *Let $n, M \in \mathbb{N}$, let $\mathcal{X}$ be a set, let $x \in \mathcal{X}^n$, and let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ be a class. Let A be a rigid adversary strategy. Then $[n]$ contains at most $2^M - 1$ indices that are essential for A for forcing M mistakes on x.*

Proof. For each essential index $t \in [n]$, there exists a label sequence $y \in \{0,1\}^{t-1}$ that witnesses that t is essential, as in Definition C.5. Each label sequence y is a witness for at most one index (the index $|y|+1$), so it suffices to show that the set $Y \subseteq \{0,1\}^{\leq n-1}$ of all witness label sequences is of cardinality at most $2^M - 1$.

Think of Y as a collection of nodes in the binary tree T_{n-1} (Definition A.4). By Definition C.5, if $y \in Y$, then the collection of all ancestors of y in Y has cardinality

$$\left|\left\{y_{\leq i} : \ i \in \{0,1,2,\ldots,|y|-1\}\right\} \ \cap \ Y\right| \leq M-1.$$

Namely, Y is a subtree of depth at most $d = M-1$ in the binary tree T_{n-1}.[22] Hence, the number of nodes in Y is at most

$$2^{d+1} - 1 = 2^M - 1,$$

as desired. $\square$

[22] The depth of a subtree is s if the longest root-to-node path contains $s+1$ nodes from the subtree.

C.3 Proof

Assumptions:

- $n, M \in \mathbb{N}$, $\mathcal{X}$ is a set, $x \in \mathcal{X}^n$ is a fixed sequence of instances.
- $A_1 : \left(\bigcup_{k=0}^{n-1} \{0,1\}^{2k}\right) \times \{0,1\} \to \{0,1\}$ is a rigid adversary labeling strategy for x that forces every learner to make at least M mistakes on the sequence x, and satisfies Items 1 and 2 in Claim C.4.
- $I = \{i_1, i_2, \ldots, i_k\} \subseteq [n]$ is the set of indices that are essential for A for forcing M mistakes on x, and $i_1 \leq i_2 \leq \cdots \leq i_k$. By Claim C.6, $k \leq 2^M - 1$.

MINIMALADVERSARY:

send $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ to the learner
for $t = 1, 2, \ldots, n$:
  if $t \in I$:
    receive prediction $\hat{y}_t$ from learner
    send prediction $\hat{y}_t$ to A_1
    receive label y_t from A_1
    send label y_t to learner
  else:
    send prediction $\hat{y}_t = 0$ to A_1
    receive label y_t from A_1

Algorithm 4: Construction of an adversary that forces M mistakes using a sequence x of length at most $2^M - 1$. In the proof of Theorem C.2, this adversary is A_2. Internally, it simulates a rigid adversary A_1.

Proof of Theorem C.2. If $\mathsf{MinLen}(\mathcal{H}, M) < \infty$, then there exist a sequence $x \in \mathcal{X}^n$, and an adversary A_0 that forces every learner to make at least M mistakes on x. By Claim C.4, there exists a rigid adversary A_1 that causes every learner to make at least M mistakes on x,[23] and also satisfies Item 2 in Claim C.4. Let f be the function that witnesses the rigidity of A_1. By Claim C.6, the set $I \subseteq [n]$ of indices that are essential for A_1 for forcing M mistakes on x has cardinality $k = |I| \leq 2^M - 1$.

Algorithm 4 defines a new adversary, A_2, which forces every learner to make at least M mistakes on a sequence of length k. A_2 is realizable, because A_1 is realizable.[24]

To see that adversary A_2 forces every learner to make at least M mistakes, let $y_1, \ldots, y_n$ be the sequence of labels assigned by A_2. Seeing as A_2 assigns the same labels as A_1, and A_1 satisfies Item 2 in Claim C.4, it follows that there are at least M indices $j \in [n]$ such that $f(y_{\leq j-1}) = \star$. Fix $J \subseteq [n]$ to be the first M such indices. Then $J \subseteq I$, namely, all the indices in J are essential for A_1 for forcing M mistakes on x (Definition C.5).

Therefore, for each $j \in J$, A_2 includes the instance x_j in the sequence of length k sent to the learner. Then, in round j of the n rounds simulated by A_2:

- The leaner makes a prediction $\hat{y}_j \in \{0,1\}$ corresponding to instance x_j.

[23] This is Item 1 in Claim C.4.
[24] The argument for realizability is the same as in the proof of Claim C.4.

- Adversary A_2 sends prediction $\hat{y}_j$ to adversary A_1. Because $f(y_{\leq j-1}) = \star$, adversary A_1 assigns the label $y_j = 1 - \hat{y}_j$. Adversary A_2 then sends that label y_j to the learner. So the learner makes a mistake on x_j.

Hence, the learner makes at least $|J| = M$ mistakes, as desired. □

D Upper Bound

D.1 Statement

The following result states that the lower bound of Theorem B.1 is tight for some classes.

Theorem D.1 (Upper bound, and separation between standard and transductive online learning)**.** *For every integer $d \geq 23$, there exists a hypothesis class $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ with a domain $\mathcal{X}$ of size $|\mathcal{X}| = 2^d - 1$ such that $\mathsf{LD}(\mathcal{H}) = d$ and the following two conditions hold for all $n \in \mathbb{N}$:*

1. $M_{\mathsf{tr}}(\mathcal{H}, n) \leq 48 \cdot \sqrt{d}$.
2. $M_{\mathsf{std}}(\mathcal{H}, n) = \min\{n, d\}$.

D.2 Hypothesis Class

In this section we construct the hypothesis class for Theorem D.1.

Lemma D.2. *Let $d \in \mathbb{N}$, $d \geq 22$. Let T_d be a perfect binary tree of depth d, as in Definition A.4. Then there exists a collection of functions $\mathcal{H} \subseteq \{0,1\}^{T_d}$ such that $\mathsf{LD}(\mathcal{H}) = d + 1$ and the following two conditions hold for all $H \subseteq \mathcal{H}$ and all $X \subseteq T_d$:*

1. *If* $\forall h \in H\ \forall x \in X : x \notin \mathrm{path}(h) \wedge h(x) = 0$*, then* $\min\{|H|, |X|\} < 2^{2\sqrt{d}}$.
2. *If* $\forall h \in H\ \forall x \in X : x \notin \mathrm{path}(h) \wedge h(x) = 1$*, then* $|H| < 2^{2\sqrt{d}}$ *or* $|X| < 3\sqrt{d}$.

The proof employs the probabilistic method, showing that a hypothesis class sampled randomly from a suitable distribution has the desired properties with very high probability.

Proof. Let $\mathcal{P}$ be a probability distribution over hypothesis classes. Formally, $\mathcal{P} \in \Delta\Big((\{0,1\}^{T_d})^{2^{d+1}}\Big)$ is a distribution over vectors of hypotheses. Each vector $\mathcal{H} \in \mathrm{supp}(\mathcal{P})$ consists of 2^{d+1} hypotheses,

$$\mathcal{H} = (h_b)_{b \in \{0,1\}^{d+1}},$$

where for each $b \in \{0,1\}^{d+1}$, hypothesis h_b is a function $h_b : T_d \to \{0,1\}$ sampled independently as following:

- For each $i \in [d] \cup \{0\}$: $h_b(b_{\leq i}) = b_{i+1}$. (In particular, with probability 1, $\mathrm{path}(h_b) = (b_{\leq 0}, b_{\leq 1}, \ldots, b_{\leq d})$, each entry in the vector $\mathcal{H}$ is unique, and $\mathcal{H}$ shatters T_d.)
- For each $x \in T_d \setminus \mathrm{path}(h_b)$, the bit $h_b(x) \in \{0,1\}$ is sampled $\mathrm{Ber}\Big(2^{-\sqrt{d}}\Big)$ independently of all other bits in $\mathcal{H}$, i.e., $\mathbb{P}[h_b(x) = 1] = \mathbb{P}[h_b(x) = 1 \mid \{h_{b'}\}_{b' \neq b}, \{h_b(x')\}_{x' \neq x}] = 2^{-\sqrt{d}}$.

Fix $B \subseteq \{0,1\}^{d+1}$ and $X \subseteq T_d$, and let $E(B, X, y)$ denote the event

$$\{\forall b \in B\ \forall x \in X : x \notin \mathrm{path}(h_b) \wedge h_b(x) = y\}. \tag{13}$$

Seeing as each off-path label $h_b(x) \in \{0,1\}$ is sampled independently,

$$\begin{aligned} \mathbb{P}_{\mathcal{H} \sim \mathcal{P}}[E(B, X, 0)] &= \prod_{(b,x) \in B \times X} \mathbb{P}_{\mathcal{H} \sim \mathcal{P}}[x \notin \mathrm{path}(h_b) \wedge h_b(x) = 0] \\ &\leq (1 - 2^{-\sqrt{d}})^{|B \times X|}. \end{aligned} \tag{14}$$

Hence,

$$\begin{aligned}
&\mathbb{P}_{\mathcal{H}\sim\mathcal{P}}\left[\exists B \subseteq \{0,1\}^{d+1}\ \exists\, X \subseteq T_d :\ E(B,X,0)\ \wedge\ \min\{|B|,|X|\} \geq 2^{2\sqrt{d}}\right] \\
&\quad \leq \binom{|\{0,1\}^{d+1}|}{\left\lceil 2^{2\sqrt{d}}\right\rceil}\binom{|T_d|}{\left\lceil 2^{2\sqrt{d}}\right\rceil}(1-2^{-\sqrt{d}})^{2^{4\sqrt{d}}} && \text{(union bound, Eq. (14))} \\
&\quad < 2\binom{2^{d+1}}{2^{2\sqrt{d}}+1}(1-2^{-\sqrt{d}})^{2^{4\sqrt{d}}} \\
&\quad < 2\cdot 2^{(d+1)\cdot\left(2^{2\sqrt{d}}+1\right)}\cdot e^{-2^{-\sqrt{d}}\cdot 2^{4\sqrt{d}}} && \left(\binom{n}{k} < n^k \text{ for } k \geq e;\ 1+x \leq e^x \text{ for } x \in \mathbb{R}\right) \\
&\quad < 2^{(d+2)2^{2\sqrt{d}}}\cdot 2^{-2^{-\sqrt{d}}\cdot 2^{4\sqrt{d}}} && (2^{2\sqrt{d}} \geq d+2 \text{ for } d \geq 1) \\
&\quad = 2^{2^{2\sqrt{d}}\cdot(d+2-2^{\sqrt{d}})} \\
&\quad < 2^{-2^{2\sqrt{d}}}. && (d+2-2^{\sqrt{d}} < -1 \text{ for } d \geq 22)
\end{aligned} \tag{15}$$

Similarly,

$$\mathbb{P}_{\mathcal{H}\sim\mathcal{P}}[\forall b \in B\ \forall x \in X :\ x \notin \mathrm{path}(h_b)\ \wedge\ h_b(x) = 1] \leq 2^{-\sqrt{d}\cdot|B\times X|}, \tag{16}$$

so

$$\begin{aligned}
&\mathbb{P}_{\mathcal{H}\sim\mathcal{P}}\left[\exists B \subseteq \{0,1\}^{d+1}\ \exists\, X \subseteq T_d :\ E(B,X,0)\ \wedge\ |H| \geq 2^{2\sqrt{d}}\ \wedge\ |X| \geq 3\sqrt{d}\right] \\
&\quad \leq \binom{|\{0,1\}^{d+1}|}{\left\lceil 2^{2\sqrt{d}}\right\rceil}\binom{|T_d|}{\left\lceil 3\sqrt{d}\right\rceil}\cdot 2^{-\sqrt{d}\cdot 2^{2\sqrt{d}}\cdot 3\sqrt{d}} && \text{(union bound, Eq. (16))} \\
&\quad < 2\binom{2^{d+1}}{2^{2\sqrt{d}}+1}\cdot 2^{-3d\cdot 2^{2\sqrt{d}}} \\
&\quad < 2^{2d\cdot 2^{2\sqrt{d}}}\cdot 2^{-3d\cdot 2^{2\sqrt{d}}} && (\text{for } d \geq 2) \\
&\quad < 2^{-d2^{\sqrt{d}}}.
\end{aligned} \tag{17}$$

Applying a union bound to Eqs. (15) and (17) gives

$$\mathbb{P}_{\mathcal{H}\sim\mathcal{P}}[\mathcal{H} \text{ satisfies Items 1 and 2}] \geq 1 - 2^{-2^{2\sqrt{d}}} - 2^{-d2^{\sqrt{d}}} \geq 1 - 10^{-100}.$$

In particular, there exists a collection $\mathcal{H}$ that satisfies Items 1 and 2. Furthermore, this collection has $\mathsf{LD}(\mathcal{H}) = d+1$ (namely, $\mathsf{LD}(\mathcal{H}) \geq d+1$ because it shatters T_d; and $\mathsf{LD}(\mathcal{H}) \leq d+1$ because $|\mathcal{H}| = 2^{d+1}$). $\square$

D.3 Algorithm

In this section we describe Algorithms 5, 6a and 6b, which together constitute the learning algorithm that achieves the $O\left(\sqrt{d}\right)$ mistake upper bound in the transductive setting, as in Theorem D.1. See Section 2.3 for a general overview of these algorithms.

D.3.1 How Experts Work

We start with some preliminary remarks about experts in Algorithms 5, 6a and 6b.

Experts. A tuple $e = (S, u, H)$ defines an expert that can make predictions using the procedure $\textsc{Expert.Predict}(e, \cdot)$. The tuple e reflects two kinds of information:

1. *Knowledge.* Information that the expert *knows* with certainty. Specifically, this reflects the labels $y_1, y_2, \ldots$ sent by the adversary so far. All experts see the labels sent by the adversary, so this knowledge is the same for all experts.

2. *Assumptions.* At certain times, experts make *assumptions* about things that are not known for certain. Specifically, experts assume that certain nodes x are on-path ($x \in \mathrm{path}(h)$) or off-path ($x \notin \mathrm{path}(h)$) with respect to the correct labeling function $h: T_d \to \{0,1\}$. Assumptions are simply guesses that may be wrong, and therefore when an expert needs to make such an assumption, it splits into two experts (as described below), with one expert assuming $x \in \mathrm{path}(h)$, and the other expert assuming $x \notin \mathrm{path}(h)$. This ensures that there always exists an expert for which all assumptions are correct.

In greater detail, the contents of the state tuple $e = (S, u, H)$ represents the knowledge and assumptions of the expert as follows:

- $u \in T_d$ – This single node encodes everything the expert knows and assumes about which of the nodes labeled so far are on-path. Observe that if $v_1, v_2, \ldots, v_k \in T_d$ are nodes that are assumed to be on-path (and all these assumptions are consistent), then these k assumptions can be represented succinctly by assigning $u = v_{i^*}$ where v_{i^*} is the deepest node among $v_1, v_2, \ldots, v_k$. Therefore, u simply holds the deepest node in the tree that is known or assumed to be on-path. At the start of the algorithm, this value is initialized to be $u = \lambda$, because the root is known to be on-path regardless of the target function.
- $S \subseteq T_d$ – the 'danger zone', as described in Section 2.3.4. This is a collection that contains all nodes in the prefix $x_{\leq t_{\mathsf{max}}} = (x_1, x_2, \ldots, x_{t_{\mathsf{max}}})$ of the sequence to be classified that have not been labeled yet and *might* be on-path for the true labeling function h given what the expert knows and assumes so far. However, S is not required to contain ancestors of nodes that are assumed to be on-path. Initially, S equals the prefix $x_{\leq t_{\mathsf{max}}}$. As information accumulates, nodes that cannot be on-path are removed from S. For instance, if $x_i \in T_d$ is assigned label $y_i \in \{0,1\}$ by the adversary, then any $(1-y_i)$-descendant of x_i (including x_i itself) may safely be removed from S.
- $H \subseteq \{0,1\}^{T_d}$ – the version space of the experts, i.e., the collection of all functions that could be the correct labeling function given everything that the expert knows and assumes. Initially, H contains all functions in $\mathcal{H}$. As information accumulates, some functions are ruled out. Specifically, a function h can be removed from H for two reasons: (i) the adversary assigns a label $y \neq h(x)$ to some node $x \in T_d$; (ii) the expert makes an assumption that some $x \in T_d$ is on-path for the correct labeling function but $x \notin \mathrm{path}(h)$, or vice versa, the expert assumes that x is off-path for the correct labeling function but $x \in \mathrm{path}(h)$.

Updates and splits. An expert can be modified using the procedure EXPERT.UPDATE$(e, \cdot, \cdot)$. This procedure either returns a single modified tuple (S, u, H) (in the first two return statements in the procedure), in which case we think of the expert as being *updated*; or alternatively, the procedure returns two tuples $e_\in = (S_\in, u_\in, H_\in)$ and $e_\notin = (S_\notin, u_\notin, H_\notin)$ (in the third return statement), in which case we think of the expert as being *split* into two experts. $e_\in$ corresponds to adding an assumption that the most recently presented node x_t is on-path for the correct labeling function, and $e_\notin$ corresponds to adding the opposite assumption.

Ancestry. At the end of each iteration of the outer 'for' loop in Algorithm 5, for each expert $e \in E_{t+1}$ there exists a unique *ancestry* sequence $\mathrm{ancestry}(e) = (e_1, e_2, \ldots, e_{t+1})$ such that $e_1 = (\{x_1, \ldots, x_{t_{\mathsf{max}}}\}, \lambda, \mathcal{H})$ is the initial single expert that was created before the start of the outer 'for' loop, $e_{t+1} = e$ is the latest version of the expert, and for each $i \in [t]$, the expert e_{i+1} is either equal to e_i, or it was created by executing EXPERT.UPDATE$(e_i, \cdot, \cdot)$.[25]

[25] Note that in this paper, we use genealogical metaphors in two distinct contexts that should not be confused. First, as is customary, we use "child", "parent", "ancestor" and "descendant" to describe relations between nodes in the binary tree T_d, which constitutes the domain of our hypothesis class. Separately from that, we use "ancestor" and "descendant" to describe relations between experts.

This overlap in terminology can partially be excused by the fact that the history of experts also forms a binary tree. Indeed, initially there is a single expert (the root of the tree), and experts can split into two, corresponding to a node having two children as in a binary tree. Seeing as experts cannot merge, the expert history corresponds precisely to a binary tree. (However, the domain T_d is a *perfect* binary tree, whereas the binary tree corresponding to expert genealogy need not be balanced).

To reduce confusion, we use $\mathrm{path}(\cdot)$ only for nodes in T_d, and $\mathrm{ancestry}(\cdot)$ only for experts, even though these operators are mathematically equivalent (however, $\mathrm{path}(\cdot)$ is defined not only for nodes in T_d but also for functions $T_d \to \{0,1\}$).

D.3.2 Pseudocode

Assumptions:

- $d, n \in \mathbb{N}$, λ is the empty string.
- $\mathcal{H} \subseteq \{0,1\}^{T_d}$ is the class that exists by Lemma D.2.
- $x_1, x_2, \dots, x_n \in T_d$ are points to be classified.

TRANSDUCTIVELEARNER($\mathcal{H}$, d, $(x_1, x_2, \dots, x_n)$):

  $t \leftarrow 0$, $t_{\mathsf{max}} \leftarrow 2^{4\sqrt{d}}$
  $e \leftarrow (\{x_1, \dots, x_{t_{\mathsf{max}}}\}, \lambda, \mathcal{H})$ ▷ The initial expert. An expert is defined by a 3-tuple.
  $w(e) \leftarrow 1$ ▷ Assign the initial expert a weight of 1.
  $E_1 \leftarrow \{e\}$ ▷ E_t is the set of experts used for predicting $\hat{y}_t$.
  $E_2, \dots, E_n, E_{n+1} \leftarrow \varnothing$
  for $t \leftarrow 1, 2, \dots, n$:

    $$\hat{y}_t \leftarrow \mathbb{1}\left(\sum_{e \in E_t} w(S) \cdot \text{EXPERT.PREDICT}(e, x_t) \geq \frac{1}{2}\right)$$ ▷ A weighted majority, using Algorithm 6a.

    send prediction $\hat{y}_t$ to adversary
    receive correct label $y_t \in \{0,1\}$ from adversary

    for $e \in E_t$: ▷ Update the experts.
      if EXPERT.PREDICT$(e, x_t) = y_t$:
        $E_{t+1} \leftarrow E_{t+1} \cup \{e\}$ ▷ If expert e made a correct prediction, keep it and its weight unchanged.
      else:
        $U \leftarrow$ EXPERT.UPDATE(e, x_t, y_t) ▷ If e made a mistake, update e using Algorithm 6b. This might cause e to be split into two experts.
        for $e' \in U$:
          $E_{t+1} \leftarrow E_{t+1} \cup \{e'\}$ ▷ Add updated expert(s) to E_{t+1}.
          $w(e') \leftarrow w(e)/(2 \cdot |U|)$ ▷ When e makes a mistake, its weight is decreased by a factor of 2 and then split equally between its descendants.

Algorithm 5: A transductive online learning algorithm that makes at most $O\left(\sqrt{d}\right)$ mistakes. It is a variant of the multiplicative weights algorithm that employs splitting experts. Namely, we start with a single expert, and when an expert makes a mistake it may split into two experts. The behavior of the experts is defined in Algorithms 6a and 6b.

Assumptions:

- $d \in \mathbb{N}$, $x \in T_d$.
- $e = (S, u, H)$ is a tuple that defines an expert:
  - $S \subseteq T_d$ – a collection of nodes that could be on-path for the true labeling function given what the expert knows and assumes.
  - $u \in T_d$ – the deepest node known or assumed to be on-path by the expert.
  - $H \subseteq \{0,1\}^{T_d}$ – the collection of all functions that could be the correct labeling function given what the expert knows and assumes.

EXPERT.PREDICT(e, x):

$(S, u, H) \leftarrow e$ ▷ Unpack the state that defines the expert.

if $|H| \leq 2^{2\sqrt{d}}$:
 return HALVING.PREDICT(H, x) ▷ Once H becomes small enough, simulate the Halving algorithm (Algorithm 7). [Case I]

if $x \preccurlyeq u$:
 return $b \in \{0,1\}$ such that $x \preccurlyeq_b u$ ▷ u is assumed to be on-path. If u is a b-decendant of x, then the correct label for x must be b. [Case II]

return $\mathbb{1}\left(\frac{|\{x' \in S : \ x \preccurlyeq_1 x'\}|}{|S|} \geq \frac{1}{3}\right)$ ▷ If there is $b \in \{0,1\}$ such that at least a $1/3$ of suspected on-path nodes are b-decendants of x, then output b. Otherwise (when at least $2/3$ of S are non-descendants of x), output 0. [Cases III to VI]

Algorithm 6a: A subroutine of Algorithm 5 that defines how an expert makes predictions.

Assumptions:

- d, x, e, S, u, H – as in Algorithm 6a.
- y – the correct label for x, as selected by the adversary.

EXPERT.UPDATE(e, x, y):

$(S, u, H) \leftarrow e$ ▷ Unpack the state that defines the expert.

$H \leftarrow$ HALVING.UPDATE(H, x, y) ▷ Update the version space, as in the Halving algorithm (Algorithm 7).

if $|H| \leq 2^{2\sqrt{d}}$:
  return $\{(S, u, H)\}$ ▷ If the version space is small, we just simulate the Halving algorithm, so the update is complete. [Case III]

for $b \in \{0, 1\}$:
  $S_b \leftarrow \{x' \in S : x \preccurlyeq_b x'\}$ ▷ Set of suspected on-path nodes that are b-descendant of x.

if $|S_{(1-y)}|/|S| \geq 1/3$:
  $S' \leftarrow S \setminus S_{(1-y)}$
  return $\{(S', u, H)\}$ ▷ At least $1/3$ of suspected on-path nodes were b-decendants of x, and therefore the expert predicted label $\hat{y} = b$. But the correct label was $y = 1 - b$. Remove all b-descendants of x from S. [Case IV]

else:
  $S_{\notin} \leftarrow S$; $u_{\notin} \leftarrow u$
  $H_{\notin} = \{h \in H : x \notin \mathrm{path}(h)\}$
  $e_{\notin} \leftarrow (S_{\notin}, u_{\notin}, H_{\notin})$ ▷ Split e in two. First, construct $e_{\notin}$ to be an updated version of e after adding the assumption that $x \notin \mathrm{path}(h)$ for the correct labeling function h.

  $S_{\in} \leftarrow S_0 \cup S_1$ ▷ Next, construct $e_{\in}$ to be an updated version of e adding the assumption $x \in \mathrm{path}(h)$. $S_{\in}$ contains only nodes that are descendants of x.

  $u_{\in} \leftarrow u$
  if $u_{\in} \preccurlyeq x$:
    $u_{\in} \leftarrow x$ ▷ $u_{\in}$ represents updating the prior assumption that u is on path by adding that x is also on path.

  $H_{\in} = \{h \in H : x \in \mathrm{path}(h)\}$ ▷ $H_{\in}$ is obtained by updating the version space to include only function where x is on path.

  $e_{\in} \leftarrow (S_{\in}, u_{\in}, H_{\in})$
  return $\{e_{\notin}, e_{\in}\}$ ▷ [Cases V and VI]

Algorithm 6b: A subroutine of Algorithm 5 that defines how an expert is updated (and possibly split into two) when it makes a mistake.

Assumptions:

- $\mathcal{X}$ a set, $k \in \mathbb{N}$.
- $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$.
- $x, x_1, \dots, x_k \in \mathcal{X}$, $y \in \{0,1\}$.

HALVING($\mathcal{H}$, $(x_1, x_2, \dots, x_k)$):

$\mathcal{H}_1 \leftarrow \mathcal{H}$

for $i \in [k]$:

$\hat{y}_i \leftarrow$ HALVING.PREDICT$(\mathcal{H}, x_i)$

send prediction $\hat{y}_i$ to adversary

receive correct label $y_i \in \{0,1\}$ from adversary

$\mathcal{H}_{i+1} \leftarrow$ HALVING.UPDATE$(\mathcal{H}_i, x_i, y_i)$

HALVING.PREDICT($\mathcal{H}$, x):

return $\mathbb{1}\left(\sum_{h \in \mathcal{H}} h(x) \geq \frac{1}{2}\right)$

HALVING.UPDATE($\mathcal{H}$, x, y):

return $\left\{h \in \mathcal{H} : \ h(x) = y\right\}$

Algorithm 7: This is the well-known halving algorithm. The experts in Algorithms 6a and 6b simulate this algorithm once their version space becomes small enough.

D.4 Analysis

In this section we prove our main result, Theorem D.1.

D.4.1 Assumption-Consistent Expert

Occasionally, when an expert is updated, it makes an assumption about whether the most-recently presented node x_t is on-path or off-path with respect to the true labeling function h. In these updates, the expert is split into two: one expert assumes that $x_t \in \mathrm{path}(h)$, and the other assumes $x_t \notin \mathrm{path}(h)$. Clearly, by splitting into two in this manner, we preserve the invariant that the set of experts always contains a 'vindicated' expert e^* such that all the assumptions made by e^* are correct. This simple observation is made formal in the following definition and claim.

Definition D.3 (Assumption consistency)**.** *For an expert $e \in E_{t+1}$ with $\mathrm{ancestry}(e) = (e_1, e_2, \dots, e_{t+1})$, and an index $i \in [t]$, we say that the $i \to (i+1)$ update of e was <u>assumption-consistent</u> with a function $h : T_d \to \{0,1\}$ if one of the following conditions hold:*

- $e_{i+1} = e_i$*; or*
- e_{i+1} *was the single expert returned by* EXPERT.UPDATE(e_i, x_i, y_i)*; or*
- EXPERT.UPDATE(e_i, x_i, y_i) *returned two experts* $(S_\in, u_\in, H_\in)$ *and* $(S_\notin, u_\notin, H_\notin)$ *(as in the third return statement of* EXPERT.UPDATE*), and furthermore,*

$$e_{i+1} = \begin{cases} (S_\in, u_\in, H_\in) & x_i \in \mathrm{path}(h) \\ (S_\notin, u_\notin, H_\notin) & x_i \notin \mathrm{path}(h). \end{cases} \tag{18}$$

We say that an expert $e \in E_{t+1}$ is assumption-consistent with h if for all $i \in [t]$, the $i \to (i+1)$ update of e was assumption-consistent with h.

Claim D.4 (Existence of assumption-consistent expert)**.** *Let $d, n, t \in \mathbb{N}$, $t \leq n$, let $\mathcal{H} \subseteq \{0,1\}^{T_d}$, let $x_1, \ldots, x_n \in T_d$, and let $h : T_d \to \{0,1\}$. Consider an execution of*

$$\textsc{TransductiveLearner}(\mathcal{H}, d, (x_1, x_2, \ldots, x_n))$$

as in Algorithm 5. Then, at the end of the t-th iteration of the outer 'for' loop in TRANSDUCTIVE-LEARNER, *there exists a unique expert $e^*_{t+1} \in E_{t+1}$ that is assumption-consistent with h.*

Proof. We prove by induction that, for all $s \in [t+1]$, E_s contains a unique expert that is assumption-consistent with h. The base case $s = 1$ is clear, because E_1 contains only a single expert that was never modified. For the induction step, let e^*_s be the unique assumption-consistent expert in E_s, and consider the $s \to (s+1)$ update. Notice that by Definition D.3,

- For all $e \in E_s \setminus \{e^*_s\}$, every expert $e' \in E_{s+1}$ such that $e' = e$ or e' was created by executing EXPERT.UPDATE(e_s, x_s, y_s) is not assumption-consistent with h; and
- Either EXPERT.UPDATE(e^*_s, x_s, y_s) is not executed and $e^*_s \in E_{s+1}$ (e^*_s is added to E_{s+1} without any modification), or precisely one of the experts created by EXPERT.UPDATE(e^*_s, x_s, y_s) and added to E_{s+1} is assumption-consistent with h.

Seeing as the $s \to (s+1)$ update executes at most EXPERT.UPDATE(e, x_s, y_s) once for each $e \in E_s$, it follows that E_{s+1} contains precisely one expert that is assumption-consistent with h. $\square$

An expert $e = (S, u, H)$ that is assumption-consistent with the correct labeling function enjoys two simple properties. The first property is that the node u in the expert encodes correct information about which previously seen nodes are on-path for the correct labeling function.

The second property is that the set S contains all future nodes that are on-path for the correct labeling function and are also deeper in the tree than all nodes assumed to be on-path so far. These two properties are formalized in the following claim.

Claim D.5 (Properties of assumption-consistent expert)**.** *Let $d, n, t \in \mathbb{N}$, $t \leq n+1$, let $\mathcal{H} \subseteq \{0,1\}^{T_d}$, let $x_1, \ldots, x_n \in T_d$. Consider an execution of*

$$\textsc{TransductiveLearner}(\mathcal{H}, d, (x_1, x_2, \ldots, x_n))$$

as in Algorithm 5. Assume that the adversary selects labels $y_1, y_2, \ldots, y_n \in \{0,1\}$ that are consistent with some function $h : T_d \to \{0,1\}$. Let $e^*_t = (S^*_t, u^*_t, H^*_t) \in E_t$ be the unique expert in E_t that is assumption-consistent with h.[26] *Then the following two properties hold:*

1. $u^*_t \in \mathrm{path}(h)$.
2. $\{x \in \{x_t, x_{t+1}, \ldots, x_{t_{\mathsf{max}}}\} : x \in \mathrm{path}(h) \ \wedge\ x \not\preceq u^*_t\} \subseteq S^*_t$.

Proof of Claim D.5. The proof proceeds by induction on t. For the base case $t = 1$, E_1 contains a single expert $e^*_1 = (S^*_1, u^*_1, H^*_1)$ where $u^*_1 = \lambda$ is the root of T_d. Indeed, $\lambda \in \mathrm{path}(h)$ for any function $h : T_d \to \{0,1\}$. This establishes the base case for Item 1. Additionally, $S^*_1 = \{x_1, x_2, \ldots, x_{t_{\mathsf{max}}}\}$, satisfying the base case for Item 2.

For the induction step, we assume that the claim holds for some integer $t = i$, and show that it holds for $t = i+1$ as well. First, we establish Item 1. If $e^*_{i+1} = e^*_i$, then the claim is immediate. Otherwise, by Definition D.3 and the first first two return statements in EXPERT.UPDATE, either $e^*_{i+1} = (S^*_{i+1}, u^*_{i+1}, H^*_{i+1})$ has $u^*_{i+1} = u^*_i \in \mathrm{path}(h)$, in which case the claim is immediate, or else e^*_{i+1} satisfies Eq. (18), namely,

$$e^*_{i+1} = \begin{cases} (S_{\in}, u_{\in}, H_{\in}) & x_i \in \mathrm{path}(h) \\ (S_{\notin}, u_{\notin}, H_{\notin}) & x_i \notin \mathrm{path}(h). \end{cases}$$

As defined in EXPERT.UPDATE, $u_{\in}$ is equal either to u^*_i or to x_i, so if $x_i \in \mathrm{path}(h)$ then

$$u^*_{i+1} = u_{\in} \in \{u^*_i, x_i\} \subseteq \mathrm{path}(h).$$

[26] Recall that e^*_t exists by Claim D.4.

On the other hand, if $x_i \notin \mathrm{path}(h)$ then we get $u^*_{i+1} = u_{\notin} = u^*_i \in \mathrm{path}(h)$. We see that in all cases, $u^*_{i+1} \in \mathrm{path}(h)$ as desired. This concludes the proof of Item 1.

For Item 2, again, if $e^*_{i+1} = e^*_i$, then the claim is immediate. Otherwise, consider the various ways in which u^*_{i+1} and S^*_{i+1} can be assigned by EXPERT.UPDATE. In the first return statement, $u^*_{i+1} = u^*_i$ and $S^*_{i+1} = S^*_i$, and the claim is immediate.

The second return statement assigns $u^*_{i+1} = u^*_i$ and $S^*_{i+1} = S^*_i \setminus S_{1-y_i}$, where S_{1-y_i} is the set of $(1-y_i)$-descendants of x_i (including x_i itself). Notice that regardless of whether x_i is on-path for the correct labeling function h or not, none of the $(1-y_i)$-descendants of x_i (except possibly x_i itself) can be on-path for h, because h assigns a label y_i to x_i. And seeing as Item 2 only requires that S^*_{i+1} contain nodes from $\{x_{i+1}, x_{i+2}, \dots, x_{t_{\mathsf{max}}}\}$, it is also safe to remove x_i. Therefore, removing S_{1-y_i} preserves Item 2.

For the third return statement, there are two possibilities. The first possibility is that $u^*_{i+1} = u_{\notin} = u^*_i$ and $S^*_{i+1} = S_{\notin} = S^*_i$, in which case the claim is immediate. The second possibility assigns $u^*_{i+1} = u_{\in}$, and $S^*_{i+1} = S_{\in} = S_0 \cup S_1$, namely, S^*_{i+1} is constructed by removing the non-descendants of x_i from S^*_i. By Eq. (18), this happens when $x_i \in \mathrm{path}(h)$, so all non-descendants of x_i or either off-path for h, or they are ancestors of x_i. Seeing as $x_i \in \mathrm{path}(h)$ and $u^*_i \in \mathrm{path}(h)$, and $u_{\in}$ is the deeper node between these two, any node that is an ancestor of x_i is also an ancestor of $u^*_{i+1} = u_{\in}$. Thus, all the nodes removed or either off-path for h, or they are ancestors of u^*_{i+1}, satisfying Item 2. (Similarly, any node that is an ancestor of u^*_i is also an ancestor of u^*_{i+1}, so we do not need to add any new nodes to S^*_{i+1} that are not included in S^*_i.)

We see that in all cases, Item 2 is preserved, as desired. □

D.4.2 Transition to Halving

Claim D.6. *Let $d, n, t \in \mathbb{N}$, $d \geq 16$, let $\mathcal{H} \subseteq \{0,1\}^{T_d}$, and let $x_1, \dots, x_n \in T_d$. Consider an execution of*

$$\textsc{TransductiveLearner}(\mathcal{H}, (x_1, x_2, \dots, x_n))$$

as in Algorithm 5. Let $t > t_{\mathsf{max}} = 2^{4\sqrt{d}}$ and let $e = (S, u, H) \in E_t$ be an expert. Then

$$|H| \leq 2^{2\sqrt{d}}.$$

Proof of Claim D.6. Assume for contradiction that $|H| > 2^{2\sqrt{d}}$. Let $H' \subseteq H$ be an arbitrary subset of size $2^{2\sqrt{d}} + 1$. Let

$$P = \cup_{h \in H'} \mathrm{path}(h).$$

Seeing as each root-to-leaf path contains $d+1$ nodes,

$$|P| \leq |H'| \cdot (d+1) \leq \left(2^{2\sqrt{d}} + 1\right) \cdot (d+1) \leq d 2^{2\sqrt{d}+1}. \tag{19}$$

Let $y_1, y_2, \dots, y_t$ be the labels provided by the adversary in the first t_{max} iterations. The line in EXPERT.UPDATE constructing H using HALVING.UPDATE(H, x, y) ensures that

$$\forall h \in H \; \forall i \in [t_{\mathsf{max}}] : \; h(x_i) = y_i. \tag{20}$$

Consider two cases:

- **Case I.** $\sum_{i=1}^{t_{\mathsf{max}}} y_i \leq t_{\mathsf{max}}/2$. Then the set

  $$X_0 = \{x_i : \; i \in [t_{\mathsf{max}}] \; \wedge \; y_i = 0\}$$

  has cardinality $|X_0| \geq t_{\mathsf{max}}/2$. Let $X_0' = X_0 \setminus P$. By Eq. (19),

  $$|X_0'| \geq \frac{t_{\mathsf{max}}}{2} - d 2^{2\sqrt{d}+1} = 2^{4\sqrt{d}} - d 2^{2\sqrt{d}+1}. \tag{21}$$

  From the choice of X_0', the inclusion $H' \subseteq H$, and Eq. (20),

  $$\forall h \in H' \; \forall x \in X_0' : \; x \notin \mathrm{path}(h) \; \wedge \; h(x) = 0. \tag{22}$$

Seeing as $|H'| > 2^{2\sqrt{d}}$, Eq. (22) and Item 1 from Lemma D.2 imply that

$$|X'_0| \le 2^{2\sqrt{d}}. \tag{23}$$

Combining Eqs. (21) and (23) yields

$$\begin{aligned} 2^{2\sqrt{d}} \ge |X'_0| &\ge 2^{4\sqrt{d}} - d2^{2\sqrt{d}+1} \\ &\ge 2^{4\sqrt{d}-1} && (d \ge 16), \end{aligned}$$

which is a contradiction.

- **Case II.** $\sum_{i=1}^{t_{\mathsf{max}}} y_i > t_{\mathsf{max}}/2$. A similar argument gives a contradiction by defining

$$X_1 = \{x_i :\ i \in [t_{\mathsf{max}}]\ \wedge\ y_i = 1\}, \text{ and } X'_1 = X_1 \setminus P.$$

As before,

$$|X'_1| \ge \frac{t_{\mathsf{max}}}{2} - d2^{2\sqrt{d}+1} \ge 2^{4\sqrt{d}} - d2^{2\sqrt{d}+1}. \tag{24}$$

for all $d \in \mathbb{N}$. However, $|H'| > 2^{2\sqrt{d}}$ and Item 2 imply that

$$|X'_1| < 3\sqrt{d}, \tag{25}$$

which is a contradiction. □

D.4.3 Performance of Best Expert

Claim D.7 (Existence of expert with large weight)**.** *Let $d, n \in \mathbb{N}$, $d \ge 16$, let $\mathcal{H} \subseteq \{0,1\}^{T_d}$, and let $x_1, \ldots, x_n \in T_d$. Consider an execution of*

$$\textsc{TransductiveLearner}(\mathcal{H}, (x_1, x_2, \ldots, x_n))$$

as in Algorithm 5. Then, at the end of the execution, there exists $e \in E_{n+1}$ such that

$$w(e) \ge 2^{-48\sqrt{d}}.$$

Note that the lower bound in Claim D.7 does not depend on n.

Proof. Fix a hypothesis $h \in \mathcal{H}$ such that $h(x_t) = y_t$ for all $t \in [n]$ (such an h exists because the adversary must always select a realizable label).

By Claim D.4, there exists $e^*_{n+1} \in E_{n+1}$ that is assumption-consistent with h. Let $\mathrm{ancestry}\big(e^*_{n+1}\big) = (e^*_1, e^*_2, \ldots, e^*_{n+1})$. We argue that this ancestry sequence makes few mistakes. Specifically, for each $t \in [n]$, let $\hat{y}^*_t = \textsc{Expert.Predict}(e^*_t, x_t)$. We claim that

$$m = \sum_{t=1}^{n} \mathbb{1}(\hat{y}^*_t \ne y_t) \le 24\sqrt{d}.$$

Indeed, let $B = \{t \in [n] :\ \hat{y}^*_t \ne y_t\}$ be the set of m indices where a mistake was made. For each $t \in B$, let $e^*_t = (S, u, H)$, and note that each $t \in B$ has a corresponding execution of $\textsc{Expert.Predict}(e^*_t, x_t)$, and an execution of $\textsc{Expert.Update}(e^*_t, x_t, y_t)$ that produces e^*_{t+1}. We partition the indices in B into six cases (six sets), and bound the number of indices that fall in each.

- **Case I.** *The execution of* $\textsc{Expert.Predict}(e^*_t, x_t)$ *exited via the first return statement in that procedure.* This happens once $|H| \le 2^{2\sqrt{d}}$, and from that point on, the expert and all subsequent experts in the ancestry are exactly simulating the $\textsc{Halving}$ algorithm (Algorithm 7) in both predictions and updates. Hence, by Fact E.1, B contains at most $m_{\text{I}} = 2\sqrt{d}$ such indices.

- **Case II.** *The execution of* $\textsc{Expert.Predict}(e^*_t, x_t)$ *exited via the second return statement in that procedure.* In particular $x \preccurlyeq u$, and the predicted label was $\hat{y}^*_t = b \in \{0,1\}$ such that $x_t \preccurlyeq_b u$. Because e^*_t is assumption-consistent with h, Item 1 in Claim D.5 implies that $u \in \mathrm{path}(h)$. Namely, we see that u is a b-descendant of x_t and $u \in \mathrm{path}(h)$. It follows that $\hat{y}^*_t = b = h(x_t) = y_t$. So no mistakes are made in Case II, and the number of indices $t \in B$ that belong to Case II is simply $m_{\text{II}} = 0$.

In the remaining cases, we assume that $\textsc{Expert.Predict}(e_t^*, x_t)$ exited via the third return statement in that procedure, so the prediction was

$$\hat{y}_t^* = \mathbb{1}\left(\frac{|S_1|}{|S|} \geq \frac{1}{3}\right), \tag{26}$$

where $S_1 = \{x' \in S : \ x_t \preccurlyeq_1 x'\}$. These cases are as follows.

- **Case III.** *The execution of* $\textsc{Expert.Update}(e_t^*, x_t, y_t)$ *exited via the first return statement in that procedure.* Namely, after the update, the resulting expert e_{t+1}^* has $|H| \leq 2^{2\sqrt{d}}$. However, because we are not in Case I, at the beginning of the iteration expert e_t^* had $|H| > 2^{2\sqrt{d}}$. Seeing as the cardinality of H decreases monotonically throughout the ancestry $e_1^*, \ldots, e_{n+1}^*$, this type of mistake can happen at most $m_{\text{III}} = 1$ times.

- **Case IV.** *The execution of* $\textsc{Expert.Update}(e_t^*, x_t, y_t)$ *exited via the second return statement in that procedure.* In this case, $|S_{(1-y_t)}|/|S| \geq 1/3$, and $e_{t+1}^* = (S', u, H)$ with $S' = S \setminus S_{1-y_t}$. So $|S'| \leq 2|S|/3$. Namely, the update causes the cardinality of the set S to be multiplied by a factor of at most $2/3$ and it strictly decreases. Seeing as the initial cardinality is t_{max}, and cardinalities are integers, the number of times this can happen is at most

$$m_{\text{IV}} = \frac{\log(t_{\mathsf{max}})}{\log(3/2)} + 1 = \frac{4\sqrt{d}}{\log(3/2)} + 1. \tag{27}$$

In the remaining cases, we assume that the execution of $\textsc{Expert.Update}(e_t^*, x_t, y_t)$ exited via the third return statement in that procedure. This implies that

$$|S_{\hat{y}_t^*}|/|S| < 1/3 \tag{28}$$

Combining this with Eq. (26), it follows $\hat{y}_t^* = 0$ and therefore $y_t = 1$. The remaining cases are as follows.

- **Case V.** $x_t \in \mathrm{path}(h)$. Let $e_t^* = (S, u, H)$. Seeing as $|H| > 2^{2\sqrt{d}}$ (because we are not in Case I), Claim D.6 (with the assumption $d \geq 16$) implies that $t \leq t_{\mathsf{max}}$. By Item 2 of Claim D.5, the facts $x_t \not\preccurlyeq u$ (we are not in Case II) and $x_t \in \mathrm{path}(h)$ imply that $x_t \in S$. In particular, S is not empty.

  Because the $t \to (t+1)$ update of e_{t+1}^* was assumption-consistent with h, Eq. (18) implies that $e_{t+1}^* = (S_\in, u_\in, H_\in)$, with $S_\in = S_0 \cup S_1$. Observe that

  - $|S_0|/|S| < 1/3$ (plugging $\hat{y}_t^* = 0$ into Eq. (28)); and
  - $|S_1|/|S| < 1/3$ (because otherwise, by Eq. (26), the prediction would have been $\hat{y}_t^* = 1$).

  Therefore,

$$|S_\in| \leq |S_0| + |S_1| \leq 2|S|/3. \tag{29}$$

  As in Case IV, combining Eq. (29) and the fact that S is not empty imply an upper bound m_{V} on the number of times Case V can happen, with the bound being the same number $m_{\text{V}} = m_{\text{IV}}$ as in Eq. (27).

- **Case VI.** $x_t \notin \mathrm{path}(h)$. So (x_t, y_t) is a pair such that $x_t \notin \mathrm{path}(h)$ and $y_t = 1$. Assume for contradiction that this type of mistake can happen strictly more than

$$m_{\text{VI}} = 3\sqrt{d}$$

  times. Let $t_1, t_2, \ldots, t_{m_{\text{VI}}}$ be the indices of the first m_{VI} iterations of the outer 'for' loop of $\textsc{TransductiveLearner}$ in which this type of mistake happened. Note that if at the end of iteration $t_{m_{\text{VI}}}$, we had expert $e_{t_{m_{\text{VI}}}+1}^* = (S_{t_{m_{\text{VI}}}+1}, u_{t_{m_{\text{VI}}}+1}, H_{t_{m_{\text{VI}}}+1})$ such that $|H_{t_{m_{\text{VI}}}+1}| \leq 2^{2\sqrt{d}}$, then from that point onwards, the expert would be simulating the halving algorithm, and in particular, it would not make any further mistake of the type in Case VI (all subsequent mistakes would belong to Case I). Hence, by the assumption that strictly more than m_{VI} mistakes were made, it follows that $|H_{t_{m_{\text{VI}}}+1}| > 2^{2\sqrt{d}}$. Let

$$H^* = \big\{h' \in \mathcal{H} : \ (\forall t \in [m_{\text{VI}}] : \ h'(x_{t_t}) = 1 \ \wedge \ x_t \notin \text{path}(h'))\big\}.$$

Because $e^*_{t_{m_{\text{VI}}}+1}$ is assumption-consistent with h, and from the construction of $H_{t_{m_{\text{VI}}}+1}$ using $H_\in$ and $H_\notin$ in EXPERT.UPDATE, it follows that $H_{t_{m_{\text{VI}}}+1} \subseteq H^*$. So there exist collections $H^* \subseteq \mathcal{H}$ and $X = \{x_{t_t} : \ t \in [m_{\text{VI}}]\} \subseteq T_d$ such that

- $|H^*| \geq |H_{t_{m_{\text{VI}}}+1}| > 2^{2\sqrt{d}}$,
- $|X| = m_{\text{VI}} = 3\sqrt{d}$,
- $\forall h' \in H^* \, \forall x \in X : \ h'(x) = 1.$
- $\forall h' \in H^* \, \forall x \in X : \ x \notin \text{path}(h').$

This is a contradiction to the choice of $\mathcal{H}$, specifically, to Item 2 in Lemma D.2.

Thus, combining the analysis of all cases, we see that the number of mistakes made by the $\text{ancestry}\big(e^*_{n+1}\big)$ is at most

$$\begin{aligned} m &\leq m_{\text{I}} + m_{\text{II}} + m_{\text{III}} + m_{\text{IV}} + m_{\text{V}} + m_{\text{VI}} \\ &\leq 2\sqrt{d} + 0 + 1 + \left(\frac{4\sqrt{d}}{\log(3/2)} + 1\right) + \left(\frac{4\sqrt{d}}{\log(3/2)} + 1\right) + 3\sqrt{d} \\ &\leq 24\sqrt{d}. \end{aligned}$$

The weights satisfy

$$w(e^*_{t+1}) \begin{cases} = \ w(e^*_t) & \hat{y}^*_t = y_t \\ \geq \ \frac{1}{4} \cdot w(e^*_t) & \hat{y}^*_t \neq y_t. \end{cases}$$

This implies that $w(e^*_{n+1}) \geq w(e^*_1) \cdot \prod_{t=1}^n 4^{-\mathbb{1}(\hat{y}_i \neq y_i)} = w(e^*_1) \cdot 4^{-m} \geq 4^{-24\sqrt{d}} = 2^{-48\sqrt{d}}$, as desired. □

D.4.4 Multiplicative Weights Mistake Bound

Claim D.8 (Mistake bound for multiplicative weights)**.** *Let $d, n \in \mathbb{N}$, let $\alpha > 0$, let $\mathcal{H} \subseteq \{0,1\}^{T_d}$, and let $x_1, \ldots, x_n \in T_d$. Consider an execution of*

$$\text{TRANSDUCTIVELEARNER}(\mathcal{H}, (x_1, x_2, \ldots, x_n))$$

as in Algorithm 5. Assume that at the end of the execution, there exists $e^* \in E_{n+1}$ such that

$$w(e^*) \geq 2^{-\alpha}.$$

Then TRANSDUCTIVELEARNER *makes at most α mistakes.*

Proof of Claim D.8. For all $i \in [n+1]$, let $w(E_i) = \sum_{e \in E_i} w(e)$. For each $i \in [n]$, if $\hat{y}_i \neq y_i$, then $w(E_{i+1}) \leq w(E_i)/2$. Hence, if TRANSDUCTIVELEARNER makes m mistakes, then by induction

$$w(E_{n+1}) \leq w(E_1) \cdot \prod_{t=1}^{n} 2^{-\mathbb{1}(\hat{y}_i \neq y_i)} = 2^{-m} \cdot w(E_1).$$

So

$$2^{-\alpha} \leq w(e^*) \leq \sum_{e \in E_{n+1}} w(e) = w(E_{n+1}) \leq 2^{-m} \cdot w(E_1) = 2^{-m}.$$

We conclude that

$$m \leq \alpha,$$

as desired. □

D.5 Proof

Proof of Theorem D.1. Fix an integer $d \geq 23$. Let $\mathcal{H} \subseteq \{0,1\}^{T_{d-1}}$ be the class constructed by invoking Lemma D.2 for the integer $d-1 \geq 22$. We argue that this class satisfies the requirements of Theorem D.1.

By construction, $\mathcal{H}$ is a class of Littlestone dimension precisely d. By Theorem A.7, this implies the equality in Item 2.

We now show the upper bound in Item 1. We argue that TRANSDUCTIVELEARNER (Algorithm 5) satisfies this upper bound. By Claim D.7, at the end of the execution of TRANSDUCTIVELEARNER there exists an expert $e \in E_{n+1}$ such that $w(e) \geq 2^{-48\sqrt{d}}$. By Claim D.8, this implies that the number of mistakes made by TRANSDUCTIVELEARNER is at most $48\sqrt{d}$, as desired. □

E Halving

Fact E.1. *Let $\mathcal{X}$ be a set, and let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ be a hypothesis class. Then for all $n \in \mathbb{N}$, all sequences $x \in \mathcal{X}^n$, and all realizable adversaries,* HALVING *(Algorithm 7) makes at most* $\log(|\mathcal{H}|)$ *mistakes in the transductive online learning (Game 2).*[27] *Namely,*

$$\sup_{n\in\mathbb{N}} \sup_{A\in\mathcal{A}_n} M_{\mathsf{tr}}(\mathcal{H}, n, \textsc{Halving}, A) \leq \log(|\mathcal{H}|).$$

[27] With the suitable syntactic modification, it also makes at most $\log(|\mathcal{H}|)$ mistakes in the standard online learning (Game 1).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [NA]

   Justification: Purely rigorous mathematical results. We explain precisely what our proofs imply (and therefore also what they do not imply).

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: For each theoretical result, the paper provides the full set of assumptions and a complete (and correct) proof.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [NA]

Justification: The paper has no experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [NA]

Justification: The paper does not include experiments requiring code.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [NA]

Justification: The paper does not include experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [NA]

Justification: The paper does not include experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [NA]

   Justification: The paper does not include experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The work is purely theoretical with no immediate direct societal impacts forseeable.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: The paper does not use existing assets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMS as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.