Estimating decision tree learnability with polylogarithmic sample complexity

Guy Blanc  
Stanford University  
gblanc@cs.stanford.edu  

Neha Gupta  
Stanford University  
nehagupta@cs.stanford.edu  

Jane Lange  
Massachusetts Institute of Technology  
jlange@mit.edu  

Li-Yang Tan  
Stanford University  
liyang@cs.stanford.edu  

Abstract

We show that top-down decision tree learning heuristics (such as ID3, C4.5, and CART) are amenable to highly efficient learnability estimation: for monotone target functions, the error of the decision tree hypothesis constructed by these heuristics can be estimated with polylogarithmically many labeled examples, exponentially smaller than the number necessary to run these heuristics, and indeed, exponentially smaller than information-theoretic minimum required to learn a good decision tree. This adds to a small but growing list of fundamental learning algorithms that have been shown to be amenable to learnability estimation.

En route to this result, we design and analyze sample-efficient minibatch versions of top-down decision tree learning heuristics and show that they achieve the same provable guarantees as the full-batch versions. We further give “active local” versions of these heuristics: given a test point $x^*$, we show how the label $T(x^*)$ of the decision tree hypothesis $T$ can be computed with polylogarithmically many labeled examples, exponentially smaller than the number necessary to learn $T$.

1 Introduction

We study the problem of estimating learnability, recently introduced by Kong and Valiant [KV18] and Blum and Hu [BH18]. Consider a learning algorithm $A$ and a dataset $S$ of unlabeled examples. Can we estimate the performance of $A$ on $S$—that is, the error of the hypothesis that $A$ would return if we were to label the entire dataset $S$ and train $A$ on it—by labeling only very few of the examples in $S$? Are there learning tasks and algorithms for which an accurate estimate of learnability can be obtained with far fewer labeled examples than the information-theoretic minimum required to learn a good hypothesis?

Motivating applications. Across domains and applications, the labeling of datasets is often an expensive process, requiring either significant computational resources or a large number of person-hours. There are therefore numerous natural scenarios in which an efficient learnability estimation procedure could serve as a useful exploratory precursor to learning. For example, suppose the error estimate returned by this procedure is large. This tells us that if we were to label the entire dataset $S$ and run $A$ on it, the error of the hypothesis $h$ that $A$ would return is large. With this information, we may decide that $h$ would not have been of much utility anyway, thereby saving ourselves the resources and effort to label the entire dataset $S$ (and to run $A$). Alternatively, we may decide to collect more data or to enlarge the feature space of $S$, in hopes of improving the performance of $A$. The learnability estimation procedure could again serve as a guide in this process, telling us how
We study the problem of estimating learnability in the context of decision tree learning where the expectations and probabilities are with respect to \( D \). We focus on top-down decision tree learning heuristics such as ID3 \([Qui86]\), C4.5 \([Qui93]\), and CART \([Bre17]\). These classic and simple heuristics continue to be widely employed in everyday machine learning applications and enjoy significant empirical success. They are also the core subroutine in modern, state-of-the-art ensemble methods such as random forests \([Bre01]\) and gradient boosted trees \([CG16]\).

We briefly describe how these top-down heuristics work, deferring the formal description to the main body of this paper. Each such heuristic \( \text{TOPDOWN}_g \) is defined by impurity function \( \mathcal{G} : [0, 1] \rightarrow [0, 1] \) which determines its splitting criterion. \( \text{TOPDOWN}_g \) takes as input a labeled dataset \( S \subseteq \mathcal{X} \times \{0, 1\} \) and a size parameter \( t \in \mathbb{N} \), and constructs a size-\( t \) decision tree for \( S \) in a greedy, top-down fashion. It begins by querying \( 1[x_i \geq \theta] \) at the root of the tree, where \( x_i \) and \( \theta \) are chosen to maximize the purity gain with respect to \( \mathcal{G} \):

\[
\mathcal{G}(E[y]) - (\Pr[x_i \geq \theta] \cdot \mathcal{G}(E[y | x_i \geq \theta]) + \Pr[x_i < \theta] \cdot \mathcal{G}(E[y | x_i < \theta])),
\]

where the expectations and probabilities are with respect to \((x, y) \sim S\). More generally, \( \text{TOPDOWN}_g \) grows its current tree \( T \) by splitting a leaf \( \ell \in T^\circ \) with a query to \( 1[x_i \geq \theta] \), where \( \ell, x_i, \) and \( \theta \) are chosen to maximize:

\[
\text{PurityGain}_g,S(\ell, i, \theta) := \Pr[x \text{ reaches } \ell] \cdot \text{LocalGain}_g,S(\ell, i, \theta),
\]

where

\[
\text{LocalGain}_g,S(\ell, i, \theta) := \mathcal{G}(E[y | x \text{ reaches } \ell]) - (\Pr[x_i \geq \theta] \cdot \mathcal{G}(E[y | x \text{ reaches } \ell, x_i \geq \theta]) + \Pr[x_i < \theta] \cdot \mathcal{G}(E[y | x \text{ reaches } \ell, x_i < \theta])).
\]

**Provable guarantees for monotone target functions \([BLT20a]\).** Motivated by the tremendous popularity and empirical successes of these top-down heuristics, there has been significant interest and efforts in establishing provable guarantees on their performance \([Kea96, DKM96, KM99, FP04, Lec09, BDM19b, BDM19a, BLT20b, BLT20a]\). The starting point of our work is a recent result of Blanc et al. \([BLT20a]\), which provides a guarantee on their performance when run on monotone target functions, with respect to the uniform distribution:

\[\mathcal{G}(1/2) = 1.\]
Theorem (Theorem 2 of [BLT20a]). Let \( f : \{\pm 1\}^d \to \{0, 1\} \) be a monotone target function and \( \mathcal{G} \) be any impurity function. For \( s \in \mathbb{N} \) and \( \varepsilon, \delta \in (0, \frac{1}{2}) \), let \( t = \frac{s \Theta(\log s)^2}{\varepsilon^2} \) and \( S \) be a set of \( n \) labeled training examples \( (x, f(x)) \) where \( x \sim \{\pm 1\}^d \) is uniform random, and \( n = \tilde{O}(t) \cdot \text{poly}(d, \log(1/\delta)) \).

With probability at least \( 1 - \delta \) over the randomness of \( S \), the size-\( t \) decision tree hypothesis constructed by \( \text{TOPDOWNS}_G(t, S) \) satisfies \( \text{error}_f(T) := \mathbb{P}_{x \sim \{\pm 1\}^d}[T(x) \neq f(x)] \leq \text{opt}_s + \varepsilon \), where \( \text{opt}_s \) denotes the error of the best size-\( s \) decision tree for \( f \).

We refer the reader to the introduction of [BLT20a] for a discussion of why assumptions on the target function are necessary in order to establish provable guarantees. Briefly, as had been noted by Kearns [Kea95], there are examples of simple non-monotone target functions \( f : \{\pm 1\}^d \to \{0, 1\} \), computable by decision trees of constant size, for which any impurity-based heuristic may build a complete tree of size \( \Omega(2^d) \) before achieving any non-trivial accuracy. Monotonicity is a natural way of excluding these adversarial functions, and for this reason it is one of the most common assumptions in learning theory. Results for monotone functions tend to be good proxies for the performance of learning algorithms on real-world datasets, which also do not exhibit these adversarial structures.

Our contributions. We give strengthened provable guarantees on the performance of top-down decision tree learning heuristics, focusing on sample complexity. Our three main contributions are as follows:

1. **Minibatch top-down decision tree learning.** We introduce and analyze \( \text{MINIBATCHTOPDOWNS}_G \), a minibatch version of \( \text{TOPDOWNS}_G \), where the purity gain associated with each split is estimated with only polylogarithmically many samples within the dataset \( S \) rather than all of \( S \). For all impurity functions \( \mathcal{G} \), we show that \( \text{MINIBATCHTOPDOWNS}_G \) achieves the same provable guarantees that those that [BLT20a] had established for the full-batch version \( \text{TOPDOWNS}_G \).

2. **Active local learning.** We then study \( \text{MINIBATCHTOPDOWNS}_G \) within the recently-introduced active local learning framework of Backurs, Blum, and Gupta [BBG20], and show that it admits an efficient active local learner. Given active access to an unlabeled dataset \( S \) and a test point \( x^* \), we show how \( T(x^*) \) can be computed by labeling only polylogarithmically many of the examples in \( S \), where \( T \) is the decision tree hypothesis that \( \text{MINIBATCHTOPDOWNS}_G \) would construct if we were to label all of \( S \) and train \( \text{MINIBATCHTOPDOWNS}_G \) on it.

3. **Estimating learnability.** Building on both our results above, we show that \( \text{MINIBATCHTOPDOWNS}_G \) is amenable to highly-efficient learnability estimation. Given active access to an unlabeled dataset \( S \), we show that the error of \( T \) with respect to any test distribution can be approximated by labeling only polylogarithmically many of the examples in \( S \), where \( T \) is the decision tree hypothesis that \( \text{MINIBATCHTOPDOWNS}_G \) would construct if we were to label all of \( S \) and train \( \text{MINIBATCHTOPDOWNS}_G \) on it.

### 1.2 Formal statements of our results

**Feature space and distributional assumptions.** We work in the setting of binary attributes and binary classification, i.e. we focus on the task of learning a target function \( f : \{\pm 1\}^d \to \{0, 1\} \). We will assume the learning algorithm receives uniform random examples \( x \sim \{\pm 1\}^d \), either labeled or unlabeled. The error of a decision tree hypothesis \( T : \{\pm 1\}^d \to \{0, 1\} \) with respect to \( f \) is defined to be \( \text{error}_f(T) := \mathbb{P}_{x \sim \{\pm 1\}^d}[T(x) \neq f(x)] \) where \( x \sim \{\pm 1\}^d \) is uniform random. We write \( \text{opt}_s(f) \) to denote \( \min \{\text{error}_f(T) : T \text{ is a size-}\( s \) decision tree\}; when \( f \) is clear from context we simply write \( \text{opt}_s \). We will also be interested in the error of \( T \) with respect to general test sets \( \mathbb{P}_{x \sim S_{\text{test}}}[T(x) \neq y] \) and general test distributions \( \mathbb{P}_{x \sim D_{\text{test}}}[T(x) \neq y] \).

**Notation and terminology.** For any decision tree \( T \), we say the size of \( T \) is the number of leaves in \( T \). We refer to a decision tree with unlabeled leaves as a partial tree, and write \( T^0 \) to denote such trees. For a leaf \( \ell \) of a partial tree \( T^0 \), we write \( |\ell| \) to denote its depth within \( T^0 \), the number of attributes queried along the path that leads to \( \ell \). We say that an input \( x \in \{\pm 1\}^d \) is consistent with a leaf \( \ell \) if \( x \) reaches \( \ell \) within \( T^0 \), and we write \( \ell_{T^0}(x) \) to denote the (unique) leaf \( \ell \) of \( T^0 \) that \( x \) is consistent with. A function \( f : \{\pm 1\}^d \to \{0, 1\} \) is said to be monotone if for every coordinate \( i \in [d] \), it is either non-decreasing with respect to \( i \) (i.e. \( f(x) \leq f(y) \) for all \( x, y \in \{\pm 1\}^d \) such...
We are now ready to describe new algorithms and state our main results. We use Theorem 1 (Provable guarantees for $\text{MINIBATCH}_{\text{MINIBATCH}}$ (Minibatch completion of partial trees)) uniform random, and let those that $[\text{BLT20a}]$ had established for the full-batch version $T$ in Section 1.1 and include its full pseudocode in Section 5. $\text{MINIBATCH}$ never splits a leaf of depth greater than $D$ and hence constructs a decision tree of small size and small depth, rather than just small size. (Looking ahead, both optimizations will be crucial for the design of our sample-efficient active local learning and learnability estimation procedures.)

Our first result shows that $\text{MINIBATCH}_{\text{MINIBATCH}}$ achieves the same performance guarantees as those that $[\text{BLT20a}]$ had established for the full-batch version $\text{TOPDOWN}_{\text{FULL}}$:

**Theorem 1** (Provable guarantees for $\text{MINIBATCH}_{\text{MINIBATCH}}$; informal version). Let $f : \{\pm 1\}^d \to \{0, 1\}$ be a monotone target function and fix an impurity function $\mathcal{G}$. For any $s \in \mathbb{N}$, $\varepsilon, \delta \in (0, \frac{1}{2})$, let $t = s^{\Theta(\log s)/\varepsilon^2}$, and $S$ be a set of $n$ labeled training examples $(x, f(x))$ where $x \sim \{\pm 1\}^d$ is uniform random, and

$$n = \tilde{O}(t) \cdot \text{poly}(\log d, \log(1/\delta)).$$
If the minibatch size is at least
\[ b = \text{polylog}(t) \cdot \text{poly}(\log d, \log(1/\delta)), \]
then with probability at least \( 1 - \delta \) over the randomness of \( S \) and the draws of minibatches from within \( S \), the size-\( t \) decision tree hypothesis constructed by \textsc{MiniBatchTopDown}_{\delta}(t, b, S) satisfies error \( f(T) \leq \text{opt}_s + \varepsilon \).

Theorem 1 shows that it suffices for the minibatch size \( b \) of \textsc{MiniBatchTopDown}_{\delta} to depend polylogarithmically on \( t \); in contrast, the full-batch version \textsc{TopDown}_{\delta} uses the entire set \( S \) to compute purity gains and determine its splits, and \( |S| = n \) has a superlinear dependence on \( t \).

Our next algorithm is an implementation of \textsc{MiniBatchTopDown}_{\delta} within the active local learning framework of Backurs, Blum, and Gupta [BBG20]:

\begin{algorithm}
\textbf{LOCALLEARNER}_{\delta}(t, b, S^0, x^*):
\begin{enumerate}
\item Initialize \( T^0 \) to be the empty tree.
\item Define \( D := \log t + \log \log t \).
\item Initialize \( e := 1 \) and let \( B^0_{\text{strands}} \) be \( b \) uniform random points from \( \{\pm 1\}^d \).
\item while \((e < t)\) {
\begin{enumerate}
\item Score: For each leaf \( \ell \in \{\ell_{T^0}(x) : x \in B^0_{\text{strands}} \cup \{x^*\}\} \) of depth at most \( D \), draw \( B^\ell \sim \text{Batch}_b(S^0, \ell) \), query \( f \)'s values on these points. For each coordinate \( i \in [d] \), compute:
\[ \text{PurityGain}_{\delta, B^\ell}(\ell, i) := 2^{-|\ell|} \cdot \text{LocalGain}_{\delta, B^\ell}(\ell, i), \]
where
\[ \text{LocalGain}_{\delta, B^\ell}(\ell, i) := |\mathbb{E}[f(x)]| \]
\[
- \frac{1}{2} |\mathbb{E}[f(x) | x_i = -1]| + \frac{1}{2} |\mathbb{E}[f(x) | x_i = 1]|, \]
where the expectations are with respect to \( x \sim B^\ell \).
\item Split: Let \((\ell^*, i^*)\) be the tuple that maximizes \( \text{PurityGain}_{\delta, B^\ell}(\ell, i) \). Grow \( T^0 \) by splitting \( \ell^* \) with a query to \( x_{i^*} \).
\item Estimate size: Update our size estimate to
\[ e = \mathbb{E}_{x \sim B^0_{\text{strands}}} \left[ 2^{\ell_{T^0}(x)} \right]. \]
\end{enumerate}
\item Draw \( B^0 \sim \text{Batch}_b(S^0, \ell_{T^0}(x^*)) \) and query \( f \)'s values on these points.
\item Output \( \text{round}(\mathbb{E}_{x \sim B^0}[f(x)]) \).
\end{enumerate}
\end{algorithm}

Figure 2: \textsc{LocalLearner}_{\delta} takes as input a size parameter \( t \), a minibatch size \( b \), an unlabeled dataset \( S^0 \), and an input \( x^* \). It selectively queries \( f \)'s values on a few points within \( S^0 \) and outputs \( T(x^*) \), where \( T \) is a tree of size approximately \( t \) that \textsc{MiniBatchTopDown}_{\delta} would return if we were to label all of \( S^0 \) and train \textsc{MiniBatchTopDown}_{\delta} on it.

Theorem 2 (Active local version of \textsc{MiniBatchTopDown}; informal version). Let \( f : \{\pm 1\}^d \to \{0, 1\} \) be a target function, \( \mathcal{G} \) be an impurity function, and \( S^0 \) be an unlabeled training set.

For all \( t \in \mathbb{N} \), \( \eta, \delta \in (0, \frac{1}{2}) \), if the minibatch size is at least \( b = \text{poly}(\log t, \log d, 1/\eta, \log(1/\delta)) \), then with probability at least \( 1 - \delta \) over the randomness of \( B^0_{\text{strands}} \), we have that for all \( x^* \in \{\pm 1\}^d \), \textsc{LocalLearner}_{\delta}(t, b, S^0, x^*) \) labels
\[ q = O(b^2 \log t) = \text{polylog}(t) \cdot \text{poly}(\log d, 1/\eta, \log(1/\delta)) \]
points within \( S^0 \) and returns \( T(x^*) \), where \( T \) is the size-\( t^* \) decision tree hypothesis that \textsc{MiniBatchTopDown}_{\delta}(t', b, S) would construct, \( t' \in t(1 \pm \eta) \), and \( S \) is the labeled dataset obtained by labeling all of \( S^0 \) with \( f \)'s values\(^2\).

\(^2\)To ensure that \textsc{LocalLearner}_{\delta} consistently labels all \( x^* \) according to the same tree \( T \), we run all invocations of \textsc{LocalLearner}_{\delta} with the same outcomes of randomness for \( B^0_{\text{strands}} \) and draws of minibatches.
Theorem 2 yields, as a fairly straightforward consequence, our learnability estimation procedure $\text{Est}_{\mathcal{G}}$ that estimates the performance of $\text{MINIBATCHTOPDOWN}_{\mathcal{G}}$ with respect to any test set $S_{\text{test}}$:

**Theorem 3** (Estimating learnability of $\text{MINIBATCHTOPDOWN}$: informal version). Let $f : \{\pm 1\}^d \rightarrow \{0, 1\}$ be a target function, $\mathcal{G}$ be an impurity function, $S^0$ be an unlabeled training set, and $S_{\text{test}}$ be a labeled test set.

For all $t \in \mathbb{N}$ and $\eta, \delta \in (0, \frac{1}{2})$, if the minibatch size $b$ is as in Theorem 2 then with probability at least $1 - \delta$ over the randomness of the draws of minibatches from within $S^0$, $\text{Est}_{\mathcal{G}}(t, b, S^0, S_{\text{test}})$ labels

$$q = O(|S_{\text{test}}| \cdot b \log t + b^2 \log t) = |S_{\text{test}}| \cdot \text{polylog}(t) \cdot \text{poly}(\log d, 1/\eta, \log(1/\delta))$$

points within $S^0$ and returns the error of $T$ with respect to $S_{\text{test}},$

$$\text{error}_{S_{\text{test}}}(T) := \Pr_{(x, y) \sim S_{\text{test}}}[T(x) \neq y],$$

where $T$ is as in Theorem 2.

We remark that Theorem 1 requires the training set be composed of independent draws of $(x, f(x))$ where $x$ is drawn uniformly from $\{\pm 1\}^d$. On the other hand, in Theorems 2 and 3, the high probability guarantees hold for any fixed choice of training set $S^0$. Similarly, in Theorem 3 $S_{\text{test}}$ can be arbitrarily chosen. Indeed, as an example application of Theorem 3, we can let $S_{\text{test}}$ be $\Theta((1/\delta)/\varepsilon^2)$ many labeled examples $(x, y)$ drawn from an arbitrary test distribution $D_{\text{test}}$ over $\{\pm 1\}^d \times \{0, 1\}$, where the marginal over $\{\pm 1\}^d$ need not be uniform and the the labels need not be consistent with $f$. With probability at least $1 - \delta$, the output of $\text{Est}_{\mathcal{G}}$ will be within $\pm \varepsilon$ of $\Pr(x, y) \sim D_{\text{test}}[T(x) \neq y]$.

## 2 Proof overview for Theorem 1

Our proof of Theorem 1 builds upon and extends the analysis in [BLT20a]. (Recall that [BLT20a] analyzed the full-batch version TOPDOWN$_{\mathcal{G}}$, which we have included in Section 5 of this paper, and their guarantee concerning its performance is their Theorem 2, which we have stated in Section 1.1 of this paper). In this section we give a high-level overview of both [BLT20a]'s and our proof strategy, in tandem with a description of the technical challenges that arise as we try to strengthen [BLT20a]'s Theorem 2 to our Theorem 1.

Let $f : \{\pm 1\}^d \rightarrow \{0, 1\}$ be a monotone function and fix an impurity function $\mathcal{G}$. Let $T^0$ be a partial tree that is being built by either TOPDOWN$_{\mathcal{G}}$ or MINIBATCHTOPDOWN$_{\mathcal{G}}$. Recall that TOPDOWN$_{\mathcal{G}}$ and MINIBATCHTOPDOWN$_{\mathcal{G}}$ compute, for each leaf $\ell \in T^0$ and coordinate $i \in [d]$, PurityGain$_{\mathcal{G}}(\ell, i)$ and PurityGain$_{\mathcal{G}, B}(\ell, i)$ respectively. Both these quantities can be thought of as estimates of the true purity gain:

$$\text{PurityGain}_{\mathcal{G}, f}(\ell, i) := 2^{-|\ell|} \cdot \text{LocalGain}_{\mathcal{G}, f}(\ell, i)$$

where

$$\text{LocalGain}_{\mathcal{G}, f}(\ell, i) := \mathcal{G}(\mathbb{E}[f(x) | x \text{ reaches } \ell])$$

$$- \left( \frac{1}{2} \mathcal{G}(\mathbb{E}[f(x) | x \text{ reaches } \ell, x_i = -1]) + \frac{1}{2} \mathcal{G}(\mathbb{E}[f(x) | x \text{ reaches } \ell, x_i = 1]) \right),$$

where here and throughout this section, all expectations are with respect to a uniform random $x \sim \{\pm 1\}^d$. The fact that MINIBATCHTOPDOWN$_{\mathcal{G}}$'s estimates of this true purity gain are based on minibatches $B$ of size exponentially smaller than that of the full sample set $S$—and hence could be exponentially less accurate—is a major source of technical challenges that arise in extending [BLT20a]'s guarantees for TOPDOWN$_{\mathcal{G}}$ to MINIBATCHTOPDOWN$_{\mathcal{G}}$.

[BLT20a] considers the potential function:

$$\mathcal{G}-\text{impurity}_f(T^0) := \sum_{\text{leaves } \ell \in T^0} 2^{-|\ell|} \cdot \mathcal{G}(\mathbb{E}[f_\ell]).$$

The following fact about this potential function $\mathcal{G}-\text{impurity}_f$ is straightforward to verify (and is proved in [BLT20a]):

Similarly, if one then wished to actually construct this tree $T$, they would run MINIBATCHTOPDOWN$_{\mathcal{G}}$ with these same outcomes of randomness.
Fact 2.1. For any partial tree \( T^\circ \), leaf \( \ell \in T^\circ \), and coordinate \( i \in [d] \), let \( \tilde{T}^\circ \) be the tree obtained from \( T^\circ \) by splitting \( \ell \) with a query to \( x_i \). Then,

\[
\mathcal{G}\text{-impurity}_f(\tilde{T}^\circ) = \mathcal{G}\text{-impurity}_f(T^\circ) - \text{PurityGain}_{\mathcal{G},f}(\ell, i).
\]

A key ingredient in [BLT20a]'s analysis is a proof that as long as error\(_f(T^\circ_S) > \text{opt}_s + \varepsilon\) (where \( T^\circ_S \) denotes the completion of \( T^\circ \) with respect to the full batch \( S \); see Section 5), there must be a leaf \( \ell \in T^\circ \) and coordinate \( i \) with high true purity gain, \( \text{PurityGain}_{\mathcal{G},f}(\ell, i) \geq \text{poly}(\varepsilon/t) \). Since TopDown\(_{\mathcal{G}}\)'s estimates \( \text{PurityGain}_{\mathcal{G},f,S}(\ell, i) \) are with respect to a sample of size \( |S| \geq \text{poly}(t/\varepsilon) \), it follows that TopDown\(_{\mathcal{G}}\) will make a split for which the true purity gain is indeed \( \text{poly}(\varepsilon/t) \). By Fact 2.1 such a split constitutes good progress with respect to the potential function \( \mathcal{G}\text{-impurity}_f \). Summarizing, [BLT20a] that shows until error\(_f(T^\circ_S) < \text{opt}_s + \varepsilon \) is achieved, every split that TopDown\(_{\mathcal{G}}\) makes has high true purity gain, and hence constitutes good progress with respect to the potential function \( \mathcal{G}\text{-impurity}_f \).

The key technical difficulty in analyzing MiniBatchTopDown\(_{\mathcal{G}}\) instead of TopDown\(_{\mathcal{G}}\) is that MiniBatchTopDown\(_{\mathcal{G}}\) is not guaranteed to choose a split with high true purity gain: it can make splits for which its estimate \( \text{PurityGain}_{\mathcal{G},B}(\ell, i) \) is high, but the true purity gain \( \text{PurityGain}_{\mathcal{G},f}(\ell, i) \) is actually tiny. In fact, unless we use batches of size \( b \geq \text{poly}(t) \), exponentially larger than the \( b = \text{polylog}(t) \) of Theorem 1, MiniBatchTopDown\(_{\mathcal{G}}\) could make splits that result in zero true purity gain, and hence constitute zero progress with respect to the potential function \( \mathcal{G}\text{-impurity}_f \).

To overcome this challenge, we instead show that most splits MiniBatchTopDown\(_{\mathcal{G}}\) makes have high true purity gain. We first show that with high probability over the draws of minibatches \( B \), if MiniBatchTopDown\(_{\mathcal{G}}\) splits a leaf that is neither too shallow nor too deep within \( T^\circ \), then this split has high true purity gain (Lemma 6.5). We then show the following two lemmas:

1. Lemma B.6: If MiniBatchTopDown\(_{\mathcal{G}}\) splits a leaf of \( T^\circ \) that is sufficiently deep, then it must be the case that error\(_f(T^\circ_{\text{Batch}}(S)) \leq \text{opt}_s + \varepsilon \), i.e. the current tree already achieves sufficiently small error. With this Lemma, we are able to define MiniBatchTopDown\(_{\mathcal{G}}\) to never split a leaf that is too deep, while retaining guarantees on its performance.

2. Lemma B.7: This lemma shows that only a small fraction of splits made by MiniBatchTopDown\(_{\mathcal{G}}\) can be too shallow.

Combining the above Lemmas, we are able to prove Theorem 1. We defer the proof to Appendix 6.

3 Proof overviews for Theorems 2 and 3

We begin with a proof overview for Theorem 2. Let \( T \) be the decision tree hypothesis that MiniBatchTopDown\(_{\mathcal{G}}\) would construct if we were to all of \( S^\circ \) and train MiniBatchTopDown\(_{\mathcal{G}}\) on it. Our goal is to efficiently compute \( T(x^*) \) for a given \( x^* \) by selectively labeling only \( q \) points within \( S^\circ \), where \( q \) is exponentially smaller than the sample complexity of learning and constructing \( T \).

Intuitively, we would like LocalLearner\(_{\mathcal{G}}\) to only grow the single “strand” within \( T \) required to compute \( T(x^*) \) instead of the entire tree \( T \)—this “strand” is simply the root-to-leaf path of \( T \) that \( x^* \) follows. The key challenge that arises in implementing this plan is: how does LocalLearner\(_{\mathcal{G}}\) know when to terminate this strand (i.e. how does it know when it has reached a leaf of \( T \)? MiniBatchTopDown\(_{\mathcal{G}}\), the “global” algorithm that LocalLearner\(_{\mathcal{G}}\) is trying to simulate, terminates when the tree is of size \( t \). As LocalLearner\(_{\mathcal{G}}\) grows the strand corresponding to \( x^* \), how could it estimate the size of the overall tree without actually growing it? In other words, it is not clear how one would define the stopping criterion of the while loop in the following pseudocode:

Roughly speaking, we want “stopping criterion” to answer the following question: if we grew a size-\( t \) tree using MiniBatchTopDown\(_{\mathcal{G}}\) (on the labeled version of \( S^\circ \)), would \( \ell \) be a leaf of the resulting tree, or would it be an internal node? Nearly equivalently, with access to just a single strand of a tree, we wish to estimate the size of that tree. If that size is \( t \), then we stop the while loop.

It is not possible to accurately estimate the size of a tree using just a single strand. However, by computing a small number of random strands, we can get an accurate size estimator. In Section 7, we show that for \( x_1, \ldots, x_m \) chosen uniformly at random from \( \{\pm 1\}^d \), the estimator \( e := \frac{1}{m} \sum_{i=1}^{m} 2^{|f(x_i)|} \)
We have given strengthened provable guarantees on the performance of popular and empirically
accurately estimates the size of $T$, as long as the depth of $T$ is not too large. Therefore, rather than
growing only the root-to-leaf path for $x^\star$, $\text{LOCALLEARNER}_g$ samples random additional inputs,
$x_1, \ldots, x_m$. Then, it simultaneously grows the strands for the root-to-leaf paths of $x^\star$ as well as
$x_1, \ldots, x_m$. These strands do not all grow at the same “rate”, as we want $\text{LOCALLEARNER}_g$ to
make splits in the same order as $\text{MINIBATCHTOPDOWN}_g$ does. As long as it does this, we can use
the size estimator to, at any step, accurately estimate the size of tree $\text{MINIBATCHTOPDOWN}_g$ would
need to build for all the current strands to end at leaves. $\text{LOCALLEARNER}_g$ terminates when its
estimate of this size is $t$.

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{figure.png}
\caption{Rather than growing the entire tree $T$ (depicted on the LHS) as $\text{MINIBATCHTOPDOWN}_g$
does, $\text{LOCALLEARNER}_g$ only grows $m + 1$ strands within $T$ (depicted on the RHS), corresponding
to the given input $x^\star$ and $m$ additional random inputs $x_1, \ldots, x_m \sim \{\pm 1\}^d$.}
\end{figure}

We back the above intuition for $\text{LOCALLEARNER}_g$ with proofs. In Section 8, we show that the
output of $\text{LOCALLEARNER}_g$ for size parameter $t$ is $T(x^\star)$, where $T$ is size-$t'$ tree produced by
$\text{MINIBATCHTOPDOWN}_g$ where $t \in (1 \pm \eta)$. We also show that $\text{LOCALLEARNER}_g$ needs to only
label polylogarithmic many points within $S^g$ to compute $T(x^\star)$. This completes our proof overview
for [Theorem 2] and [Theorem 3] is a straightforward consequence of [Theorem 2].

4 Conclusion

We have given strengthened provable guarantees on the performance of popular and empirically
successful top-down decision tree learning heuristics such as ID3, C4.5, and CART, focusing
on sample complexity. First, we designed and analyzed \textit{minibatch} versions of these heuristics,
\textit{MINIBATCHTOPDOWN}_g, and proved that they achieve the same performance guarantees as the
full-batch versions. We then gave an implementation of $\text{MINIBATCHTOPDOWN}_g$ within the recently-
introduced active local learning framework of [BBG20]. Building on these results, we showed that
$\text{MINIBATCHTOPDOWN}_g$ is amenable to highly efficient learnability estimation [KV18] [BH18]: its
performance can be estimated accurately by selectively labeling very few examples.

As discussed in [KV18] [BH18], this new notion of learnability estimation opens up a whole host of
theoretical and empirical directions for future work. We discuss several concrete ones that are most
relevant to our work. Our algorithm $Est_g$ efficiently and accurately estimates the quality, relative to
a test set $S_{\text{test}}$, of the hypothesis that $\text{MINIBATCHTOPDOWN}_g$ would produce if trained on a set $S^g$.
Could $Est_g$ be more broadly useful in assessing the quality of the \textit{training data} $S^g$ itself, relative to
$S_{\text{test}}$? Could its estimates provide guarantees on the performance of other algorithms when trained
in $S^g$ and tested on $S_{\text{test}}$? It would also be interesting to explore applications of our algorithms to
the design of training sets. Given training sets $S_1, \ldots, S_m$, $Est_g$ allows us to efficiently determine
the $S_i$ for which $\text{MINIBATCH}\text{TOPDOWN}_g$ would produce a hypothesis that achieves the smallest error with respect to $S_{\text{test}}$. Could $\text{Est}_g$ or extensions of it be useful in efficiently creating an $S^*$, comprising data from each $S_i$, that is of higher quality than any $S_i$ individually? Finally, while we have focused on top-down heuristics for learning a single decision tree in this work, a natural next step would be to design and analyze learnability estimation procedures for ensemble methods such as random forests and gradient boosted trees.

**Broader Impact**

This work does not present any foreseeable societal consequence.

**Acknowledgements and Disclosure of Funding**

Guy, Jane, and Li-Yang were supported by NSF award CCF-192179 and NSF CAREER award CCF-1942123. Neha was supported by NSF award 1704417 and Moses Charikar’s Simons Investigator grant.

**References**


