Scalable and provably accurate algorithms for differentially private distributed decision tree learning

Kai Wen Wang[1], **Travis Dick**[2], **Maria-Florina Balcan**[3]
[1]wangkaiwen998@gmail.com, [2]tbd@seas.upenn.edu, [3]ninamf@cs.cmu.edu
[1,3]Carnegie Mellon University, [2]University of Pennsylvania

Abstract

This paper introduces the first provably accurate algorithms for differentially private, top-down decision tree learning in the distributed setting (Balcan et al. 2012). We propose DP-TopDown, a general privacy-preserving decision tree learning algorithm, and present two distributed implementations. Our first method NoisyCounts naturally extends the single machine algorithm by using the Laplace mechanism. Our second method LocalRNM significantly reduces communication and added noise by performing local optimization at each data holder. We provide the first utility guarantees for differentially private top-down decision tree learning in both the single machine and distributed settings. These guarantees show that the error of the privately-learned decision tree quickly goes to zero provided that the dataset is sufficiently large. Our extensive experiments on real datasets illustrate the trade-offs of privacy, accuracy and generalization when learning private decision trees in the distributed setting.

1 Introduction

New technologies for edge computing and the Internet of Things (IoT) are driving computing toward the dispersed setting (Satyanarayanan 2017). A distributed learning algorithm has access to data with more quantity and diversity than learning from one data holder. Today, data entities comprise of mobile phones, schools, hospitals, companies, countries, and more. As an illustrative example, consider training a model for early detection of a new strand of malaria, which has been tested in multiple hospitals across the country. Most hospitals alone would not have enough data to train an accurate model. Even if a hospital could locally train a model, the model would be biased toward and only accurate for that specific city. As such, it would perform poorly as a nationwide diagnostic tool. However, while benefits of distributed learning are evident, privacy concerns can often deter cooperation.

Differential privacy (Dwork et al. 2006) is a strong guarantee of individual privacy with an additional benefit of encouraging generalization (Oneto, Ridella, and Anguita 2017). Differentially private mechanisms formally quantify the amount of leaked information and can greatly incentivize distributed learning with sensitive data. Another attractive feature of differential privacy is its robustness to post-processing; once the model is learned through a differentially private mechanism, any arbitrary applications of the model as well as post-processing to the model can be performed without worrying about additional privacy leaks.

In this work, we investigate learning decision trees while preserving privacy in the distributed setting (Balcan et al. 2012). Decision trees are often used for their efficiency and interpretability. The learned structure of decision trees is much more interpretable than other machine learning models like deep neural networks. Motivated by the fact that constructing optimal decision trees is NP-complete (Laurent and Rivest 1976), greedy top-down learning algorithms are used in practice (i.e. ID3, C4.5 and CART (Mitchell 1997)). Top-down algorithms can be viewed as boosting algorithms that quickly drive their error to zero under the Weak Learning Assumption (Kearns and Mansour 1999).

In the distributed setting, any communicated information between data holders must be made differentially private, which limits the quantity and quality of information that can be published to learn the decision tree. Intermediate computations often cost privacy, which violates the information minimization principle of differential privacy (Dwork and Roth 2014). Despite these constraints, we are able to prove theoretical guarantees that the error of the privately-learned tree quickly goes to zero with high probability provided that we have sufficient data under the Weak Learning Assumption.

We present DP-TopDown, a family of differentially private decision tree learning algorithms based off the TopDown algorithm (Kearns and Mansour 1999). The DP-TopDown procedure uses a differentially private subroutine, called $\mathrm{PrivateSplit}$, to approximate the optimal split chosen by TopDown. Designing a private and distributed mechanism that satisfies the utility requirements of $\mathrm{PrivateSplit}$ is the main challenge of running DP-TopDown. In our first approach *NoisyCounts*, each entity publishes noised counts that are aggregated to find the best splitting functions. Our second approach *LocalRNM* is a heuristic that performs local optimization at each entity to greatly reduce the amount of communication and noise compared to NoisyCounts.

Finally, we perform a comprehensive evaluation of our dif-

ferent implementations of DP-TopDown on real and relevant datasets. We provide plots that illustrate the privacy-accuracy trade-off for our algorithms and empirically show that LocalRNM performs better than NoisyCounts on most datasets. Motivated by our theoretical results, we also show that both the training and testing accuracy of the decision tree increases with more training data. In one dataset, we observe that our private algorithms can actually improve the performance of decision trees, which is similar in spirit to adding a bit of noise in non-convex optimization problems to escape local optimums (Zhang, Liang, and Charikar 2017).

Our contributions are summarized as follows:

1. We propose DP-TopDown, a differentially private, top-down learning algorithm for decision trees, and present two implementations for the distributed setting.
2. We prove boosting-based utility guarantees for DP-TopDown. These are the first utility guarantees for differentially private top-down learning of decision trees in both single machine and distributed settings.
3. We extensively evaluate DP-TopDown on real datasets. The code and instructions to reproduce our experimental results will be made available.

1.1 Related Works

Previous works have explored private top-down learning of decision trees for specific algorithms. (Blum et al. 2005) introduced and used the SuLQ framework to implement ID3. (Friedman and Schuster 2010; Mohammed et al. 2011) demonstrated the effectiveness of using the exponential mechanism for private ID3 and C4.5. Our approach generalizes previous works and provide the first utility guarantees and sample complexity bounds for private, top-down decision tree learning, even in the distributed setting.

Apart from top-down decision tree learning (i.e. ID3 and CART), previous worked have also considered random decision trees, where the construction of the decision tree is entirely random and can be done even before it sees any data (Jagannathan, Pillaipakkamnatt, and Wright 2009). (Bojarski et al. 2014) studied techniques which averaged over collections of random decision trees (a.k.a. Random Forests) and gave guarantees for empirical and generalization errors. In practice, random forests are powerful models that sometimes out-perform vanilla top-down learned decision trees. However, top-down learning of decision trees performs well in many cases and in general leads to more interpretable trees than randomly constructed decision trees.

Previous works have used boosting to design improved differentially private algorithms for answering batches of queries (Dwork, Rothblum, and Vadhan 2010). Our work is different and is about designing differentially private boosting algorithms, in particular top-down decision tree learning.

2 Preliminaries

We study learning problems where the goal is to map input examples from a space X to a finite output space Y. We suppose that there is an unknown distribution P defined over X, together with an unknown target function $f : X \to Y$. Given a sample S of i.i.d. pairs $(x, f(x))$ drawn from the distribution P, our goal is to learn an approximation to f.

We focus on the canonical problem of top-down decision tree learning. Each internal node is labeled by a splitting function $h : X \to \{0,1\}$ that "splits" the data according to $h(x)$, and the leaves of the tree are labeled by a class in Y. These splitting functions route each example $x \in X$ to exactly one leaf of the tree. That is, if the root of the tree has splitting function h, a point x is routed to the left subtree if $h(x) = 0$ and to the right subtree if $h(x) = 1$. The prediction made by a decision tree T for an input x is the label of the leaf that x is routed to. We let $H \subseteq \{0,1\}^X$ denote the class of splitting functions that may be placed at each internal node, and denote the error of a decision tree T by $\epsilon(T) = \Pr(T(x) \neq f(x))$, where x is sampled from P.

2.1 Top-down Decision Tree Learning

In this section we describe the high level structure of (non-private) top-down decision tree learning algorithms. Top-down learning algorithms construct decision trees by repeatedly "splitting" a leaf node in the tree built so far. On each iteration, the algorithm greedily finds the leaf and splitting function that maximally reduces an upper bound on the error of the tree. The upper bound (see Equation (1)) on error is induced by a *splitting criterion* $G : [0,1] \to [0,1]$ that is concave, symmetric about $\frac{1}{2}$, and that satisfies $G(\frac{1}{2}) = 1$, $G(0) = G(1) = 0$. The selected leaf is replaced by an internal node labeled with the chosen splitting function, which partitions the data at the node into two new children leaves. Once the tree is built, the leaves of the tree are labeled by the label of the most common class that reaches the leaf. Many effective and widely-used decision tree learning programs are instances of the TopDown algorithm. For example, C4.5 uses entropy for splitting criterion while CART uses Gini.

The splitting criterion G provides an upper bound on the error of a decision tree T. The error $\epsilon(T)$ can be written as $\epsilon(T) = \sum_{\ell \in \text{leaves}(T)} w(\ell) \min\{q(\ell), 1 - q(\ell)\}$ where $w(\ell) = \Pr_P[x \text{ reaches leaf } \ell]$ is the fraction of data that reaches ℓ and $q(\ell) = \Pr_P[f(x) = 1 | x \text{ reaches leaf } \ell]$ is the fraction of that data with label 1. For all $z \in [0,1]$, the splitting criterion satisfies $G(z) \geq \min\{z, 1-z\}$, and therefore the following potential function is an upper bound on the error $\epsilon(T)$ of a decision tree T:

$$G(T) = \sum_{\ell \in \text{leaves}(T)} w(\ell) G(q(\ell)). \tag{1}$$

With this, we can formally describe one iteration of top-down decision tree learning. For each leaf $\ell \in \text{leaves}(T)$, let $S_\ell \subseteq S$ denote the subset of data that reaches leaf ℓ. For each splitting function $h \in H$, let $T(\ell, h)$ denote the tree obtained from T by replacing ℓ by an internal node that splits S_ℓ into two children leaves ℓ_0, ℓ_1, where any data satisfying $h(x) = i$ goes to ℓ_i. At each round, TopDown chooses the split-leaf pair (h, ℓ) that maximizes the decrease in the potential: $G(T) - G(T(\ell, h)) = w(\ell) J(\ell, h)$ where $J(\ell, h) = G(q(\ell)) - \frac{|S_{\ell_0}|}{|S_\ell|} G(q(\ell_0)) - \frac{|S_{\ell_1}|}{|S_\ell|} G(q(\ell_1))$.

Kearns and Mansour showed that (non-private) top-down decision tree learning can be viewed as a form of boosting (Kearns and Mansour 1999). Whenever the class of splitting functions H used to label internal nodes satisfies the Weak

Learning Assumption (i.e., they can predict the target f better than random guessing for any distribution on X), then top-down learning algorithms amplify these slight advantages and quickly drives the training error of the tree to zero. Formally, the Weak Learning Assumption is as follows:

Definition 1. A function class H $\{0,1\}^X$ -satisfies the *Weak Learning Assumption* for a target function $f : X \to \{0,1\}$ if for any distribution P over X, there exists $h \in H$ such that $\Pr_{x \sim P}(h(x) = f(x)) \geq \frac{1}{2}$. Commonly, is refered to as the advantage.

We restate the result of (Kearns and Mansour 1999) when G is entropy: $G(q) = -q\log(q) - (1-q)\log(1-q)$.

Theorem 2. *Let $G(q)$ be entropy. Under the Weak Learning Assumption, for any $\varepsilon \in (0,1]$, TopDown suffices to make $(1/\varepsilon)^{O(\log(1/\varepsilon)/\gamma^2)}$ splits to achieve training error ε.*

2.2 Differential Privacy

Differential privacy is the *de facto* standard notion of privacy that enjoys composition theorems (Dwork et al. 2006). The formal notion is as follows. We say two datasets S and S' are neighboring if they differ on at most one point. A randomized algorithm A is -differentially private if for any neighboring datasets S, S' and outcomes $O \subseteq \mathrm{Range}(A)$, we have

$$\Pr(A(S) \in O) \leq e^{\varepsilon} \Pr(A(S') \in O)$$

where the probability space is over the coin flips of A.

Our decision tree algorithms will make use of two standard differentially private components. First, the Laplace mechanism (LM) can be used to privately evaluate a function f mapping datasets to real numbers, while preserving -differential privacy. Given a dataset S, the Laplace mechanism with privacy parameter outputs $f(S) + Z$, where Z is drawn from the Laplace distribution with scale parameter $\Delta_f/$ and Δ_f is the *sensitivity* of f:

$$\Delta_f = \max_{S,S' \text{ neighboring datasets}} |f(S) - f(S')|.$$

Next, we will sometimes use the Report Noisy Max (RNM) mechanism to privately find the best scoring function from a collection of k scoring functions $f_1, \ldots, f_k$ mapping datasets to real numbers. Given a dataset S, RNM with privacy parameter computes $\hat{f}_i = f_i(S) + Z_i$, where Z_i is drawn from the Laplace distribution with scale parameter $\max_{i=1,\ldots,k} 2\Delta_{f_i}/$. Then it outputs $i^* = \mathrm{argmax}_i \hat{f}_i$ together with the estimate $\hat{f}_{i^*}$. Importantly, RNM does not publish the estimated scores $\hat{f}_i$ for $i \neq i^*$, allowing it to add less noise than applying k LMs. The RNM mechanism with privacy parameter preserves -differential privacy.

Our decision tree learning algorithms will apply LM and RNM many times, with varying privacy parameters. Standard composition theorems, presented in Appendix A.1, guarantee that the overall algorithm are still -differentially private (Dwork et al. 2006; McSherry 2009).

2.3 Distributed setting with privacy constraints

In this work, we consider the distributed learning setting with privacy constraints (Balcan et al. 2012). We suppose there are k entities (i.e. data holders) and each entity $i \in [k]$ must preserve -differential privacy for their subset of the data S^i. In general, we make no assumptions on how S^i partitions S in terms of size and distribution. An example of the distributed setting is hospitals with private patient data. Patients entrust their sensitive information to be used within their local hospital, but the response to any queries from outside the hospital must preserve privacy on behalf of the patients. Note that this notion is not local differential privacy (Warner 1965; Cormode et al. 2018), where the patients would locally perturb their data before giving them to the hospital.

We remark here that a distributed learning algorithm can also use cryptographic privacy protocols such as multi-party computation to preserve privacy while learning (Evans et al. 2018). Assuming reasonable computational limitations of the adversary, cryptographic privacy leaks zero bits of information during the training process. On the other hand, differential privacy is an information theoretic guarantee that bounds the amount of leaked information and is robust to post-processing. In this work, we focus on differential privacy protocols (Balcan et al. 2012) since they tend to be more communication efficient and feasible in practice than cryptographic techniques.

3 Differentially Private TopDown

In this section, we propose a private, top-down decision tree learning algorithm called DP-TopDown that is based off the TopDown algorithm (Kearns and Mansour 1999). The key step of DP-TopDown that depend on the sensitive dataset S is choosing the optimal split for a given leaf ℓ. Since there are many ways to do this, we encapsulate this with a function called $\mathrm{PrivateSplit}$ that satisfies the following privacy and utility requirements, which are sufficient to prove privacy and utility guarantees for DP-TopDown:

1. $\mathrm{PrivateSplit}(\ell, , ,)$ outputs a pair $(\hat{h}, \hat{J})$ where $h \in H$ and $\hat{J} \in \mathbb{R}$, while preserving -differential privacy;
2. $\forall > 0, \exists N := N(, , ,)$ so that if $|S| \geq N$, then $J(\ell, \hat{h}) \geq \max_h J(\ell, h)$ and $|J(\ell, \hat{h}) - \hat{J}| \leq$ with probability $1-$. In words, this means that $\mathrm{PrivateSplit}$ finds nearly optimal splits w.h.p.

Pseudocode for DP-TopDown is given in Algorithm 1. It takes as input the dataset S, the privacy parameter , the max number M of nodes (excluding leaves) in the tree, the desired error $\varepsilon \in (0,1]$ and failure probability $\in (0,1]$. We use a budget function $B(d)$ to assign a fraction to data queries at depth d in the tree. Since nodes at depth d partition the data at that layer, they together preserve $B(d)$-differential privacy by parallel composition (Theorem 9). These parameters are used in Theorem 3.

Privacy guarantee: The privacy analysis follows from the composition theorems. We use half of the total privacy budget for labeling the leaves at line 14 using the Laplace mechanism, which preserves $\frac{}{2}$-differential privacy by parallel composition (Theorem 9) since the leaves partition S. The rest of DP-TopDown preserves $\frac{}{2}$-differential privacy for an appropriate budget function B that satisfies $\sum_{\text{all depths } d} B(d) \leq 1$.

One privacy budgeting approach is to uniformly split the budget across all possible depths with $B(d) = \frac{1}{M}$. This gives

Algorithm 1 DP-TopDown

Input: dataset S, max size M, error ε, privacy γ, failure probability
Init T to single leaf ℓ and Q to max-priority queue
$(\hat{h}, \hat{J}) \leftarrow \text{PrivateSplit}(\ell, \frac{\gamma}{2}B(1), \frac{\gamma}{2(2M+1)})$
$Q.\text{push}(\hat{J}, (\hat{h}, \ell))$ [add $(\hat{h}, \ell)$ to Q with priority $\hat{J}$]
for $t = 1, 2, \ldots, M$ **do**
  $(\ell, h) \leftarrow Q.\text{pop}()$ [get max element in Q]
  $T \leftarrow T(\ell, h)$ with children leaves ℓ_0, ℓ_1
  for $i = 0, 1$ **do**
    $\gamma_i \leftarrow \frac{\gamma}{2}B(\text{depth}(\ell_i))$ [budget for leaf]
    $w_i \leftarrow \frac{|S_{\cdot i}|}{|S|} + \text{Lap}(\frac{2}{|S|\gamma_i})$ [LM with budget $\frac{\gamma_i}{2}$]
    $(\hat{h}_i, \hat{J}_i) \leftarrow \text{PrivateSplit}(\ell_i, \frac{\gamma_i}{2}, \frac{\gamma}{2(2M+1)})$
    if $w_i \geq \frac{\varepsilon}{M}$ **then**
      $Q.\text{push}(w_i \cdot \hat{J}_i, (\ell_i, \hat{h}_i))$
Label leaves by majority label [RNM with budget $\frac{\gamma}{2}$]
return T

us the tightest sample complexity bounds for Theorem 3.

Another approach is to set $B(d) = \frac{1}{2^d}$. This exponentially decaying budgeting is motivated by the intuition that early splits in the tree are more important than later splits. Empirically, we found that this strategy gives higher accuracy than the uniform budgeting approach.

Utility guarantee: Intuitively for any fixed γ, the noise from preserving γ-differential privacy becomes negligible with high probability as the size of the dataset grows to infinity. The following theorem formalizes this intuition for DP-TopDown by providing general utility guarantees and sample complexity bounds, which are applicable in both the single machine and distributed settings. Under the Weak Learning Assumption, the error of the privately-learned tree quickly goes to zero provided that the dataset is large enough. The required dataset size can be bounded in terms of the properties of the budgeting strategy B and of PrivateSplit.

Theorem 3. *Let $G(q)$ be entropy. Let M, γ, ε, δ be inputs to DP-TopDown, as defined in Algorithm 1. Under the Weak Learning Assumption with advantage γ, DP-TopDown suffices to make $(1/\varepsilon)^{O(\log(1/\varepsilon)/\gamma^2)}$ splits to achieve training error ε with probability $1-\delta$ provided the dataset S has size at least*

$$|S| \geq \max\left(\tilde{\Theta}\left(\frac{M}{\varepsilon^2 \gamma b}\right), \frac{2M}{\varepsilon}N\left(\tilde{\Theta}(\tfrac{\varepsilon^2}{M}), \tfrac{b}{2}, O(\tfrac{\gamma}{M})\right)\right),$$

where $b = \min_{1 \leq d \leq M} B(d)$ and $N(\cdot, \cdot, \cdot)$ is the data requirement of PrivateSplit.

Proof Sketch. Let T_t denote the decision tree constructed by DP-TopDown so far at the beginning of the t^{th} iteration. The boosting analysis of (Kearns and Mansour 1999) has two key steps. First, they argue that during the t^{th} iteration, there must be a leaf ℓ of the tree T_t whose weight is at least $\varepsilon(T_t)/t$. Next, they leverage the weak-learning assumption to guarantee that the optimal split for that leaf significantly reduces the potential G_t. This guarantees that after a small number of iterations, the error cannot be large.

There are two main challenges in extending this analysis to the DP-TopDown algorithm. First, since DP-TopDown only splits leaves whose *estimated* weight is large, there is some risk that it will not consider splitting the leaf used in the analysis of Kearns and Mansour. Second, even if that leaf is considered, the PrivateSplit algorithm will only produce nearly optimal splits for that leaf when there is enough data reaching that leaf (recall that PrivateSplit returns an α-optimal split with probability at least $1-\beta$, provided that there are at least $N(\alpha, \beta, \gamma)$ data points in the leaf).

The term $\tilde{\Theta}(\frac{M}{\varepsilon^2 \gamma b})$ in our dataset size requirements is chosen to ensure that the error in the estimated weight of every leaf during the run of DP-TopDown is bounded by $\frac{\varepsilon^2}{M}$ with high probability. This guarantees that the leaf used in the analysis of Kearns and Mansour will be considered by DP-TopDown with high probability. Moreover, every leaf applying PrivateSplit has non-trivial weight. This term also ensures the total error from labeling leaves is less than $\frac{\varepsilon}{2}$.

A consequence of the above argument is that DP-TopDown only runs the PrivateSplit algorithm on leaves with weight at least $\Omega(\frac{\varepsilon}{M})$. In other words, they contain at least $\Omega(\frac{|S|\varepsilon}{M})$ data points. The term $\frac{2M}{\varepsilon}N\left(\tilde{\Theta}(\frac{\varepsilon^2}{M}), \frac{b}{2}, O(\frac{\gamma}{M})\right)$ in our dataset size requirement is chosen to guarantee the number of data points in any leaf that we run PrivateSplit is large enough to guarantee that it returns a γ-optimal split where γ is at most half the reduction in $G(T)$ guaranteed by the split constructed in the Kearns and Mansour analysis. This ensures that our algorithm reduces the potential $G(T)$ by at least half of the guarantee of the non-private algorithm, which is sufficient to prove our utility guarantee. □

We note that our proof technique generalizes to other splitting criteria analyzed by (Kearns and Mansour 1999), such as the Gini Index $G(q) = 4q(1-q)$ and $G(q) = \sqrt{2q(1-q)}$. We also note that the only assumption on H in Theorem 3 is the Weak Learning Assumption. In particular, there are no assumptions on the size of H. In the rest of the paper, we present implementations of PrivateSplit that iterate through H, which is why our corollaries depend on $|H|$ being finite.

SingleMachine algorithm: RNM. In the single machine setting and when the class of splitting functions H is finite, we can simply use RNM to select an approximately optimal split from H. In particular, if we define $f_i(S_\cdot) = J(\ell, h_i)$ for $h_i \in H$ and use RNM to choose the optimal split, we satisfy the utility requirement of PrivateSplit as follows.

Lemma 4. *Single machine RNM satisfies the* PrivateSplit *requirements with $N(\alpha, \beta, \gamma) = \tilde{\Theta}(\frac{1}{\alpha})$.*

Proof Sketch. In Lemma 11 in the Appendix, we show that the sensitivity of the function $f_i(S_\cdot) = J(\ell, h_i)$ is bounded by $\tilde{\Theta}(1/|S_\cdot|)$. Let $\hat{f}_i = f_i(S) + Z_i$ where Z_i is drawn from the Laplace distribution with scale parameter $O(\gamma/|S_\cdot|)$ to each estimate $\hat{f}_i$. With probability at least $1-\beta$, we have $|Z_i| = O(\frac{\gamma}{|S_\cdot|}\log\frac{|H|}{\beta})$ simultaneously for all i. When $|S_\cdot| \geq \tilde{\Omega}(\frac{1}{\alpha})$, the error in all estimates made by the RNM mechanism are bounded by $\alpha/2$, which proves the claim. □

Corollary 5. *Under uniform budgeting, using RNM as* PrivateSplit *requires* $|S| = \tilde{\Theta}\left(\frac{M^3}{\varepsilon^2\tau^2}\right)$ *for Theorem 3.*

We remark that another possible private implementation is to use the exponential mechanism, which recovers the algorithm in (Friedman and Schuster 2010; Mohammed et al. 2011).

Distributed setting with privacy constraint. Weight estimation in line 10 can be done distributedly by having each entity estimate the number of data points at a particular leaf with LM. Leaf labeling can be done distributedly by first estimating the number of data points at a particular leaf for each label using LM and then finding the label that maximizes the aggregated counts. The sample complexity of Theorem 3 accrues a k factor in the first term, as formally quantified in Corollary 16 of the Appendix. Therefore in the distributed setting, we can run DP-TopDown given a distributed PrivateSplit$(\ell, \cdot, \cdot)$. We now present two methods.

Distributed algorithm 1: NoisyCounts. For each splitting function $h_i \in H$, each entity $j \in [k]$ uses LM to publish noisy estimated counts $\hat{c}^j_{y=a;h_i=b}$ of the number of datapoints at leaf ℓ, with label a and splitting value b. Then, the coordinator can use the aggregated noisy counts to compute each $\hat{J}(\ell, h_i)$ and locate the maximum. The following lemma follows from the utility guarantees of LM.

Lemma 6. *NoisyCounts satisfies the* PrivateSplit *requirements with* $N(\cdot, \cdot, \cdot) = \tilde{\Theta}\left(\frac{k|H|}{\varepsilon}\right)$.

Proof Sketch. The proof has two main steps. First, we bound the total error in the noisy counts collected by the coordinator, and second we argue that when the counts are sufficiently accurate, then the chosen splitting function is nearly optimal. The collection of counts associated with any splitting function $h_i \in H$ form a histogram query, and they can all be approximated while satisfying ε_0-differential privacy using the Laplace mechanism by adding $\mathrm{Lap}(\frac{1}{\varepsilon_0})$ noise to each count. We approximate one histogram query for each $h_i \in H$, so we set $\varepsilon_0 = \frac{1}{|H|}$ so that the total privacy cost is ε. Next, using tail bounds for Laplace random variables, we know that with high probability, the maximum error in any of the counts constructed by a single entity is bounded by $\tilde{\Theta}(\frac{|H|}{\varepsilon})$. Therefore, the each accumulated count obtained by the coordinator has at most $\tilde{\Theta}(\frac{k|H|}{\varepsilon})$ error accumulated from the k entities. Finally, we show that when $|S| \cdot \tilde{\Theta}(\frac{k|H|}{\varepsilon})$, the counts are accurate enough guarantee a γ-optimal split. □

Corollary 7. *Under uniform budgeting, using NoisyCounts as* PrivateSplit *requires* $|S| = \tilde{\Theta}\left(\frac{M^3 k|H|}{\varepsilon^2\tau^2}\right)$ *for Theorem 3.*

Compared to single machine, the bound on N for NoisyCounts increased by a factor of $k|H|$. This is the consequence of having each entity publish all the counts so that $\hat{J}(\ell, h_i)$ can be approximated for every $h_i \in H$ whereas we only needed the maximum. Therefore, each of the k entities calls LM with $|H|$ factor more noise. Note that unlike weight estimation and leaf labeling, which easily extended to the distributed setting while accruing only a k factor, the natural method of NoisyCounts accrues also a $|H|$ factor since there are $|H|$ splitting functions to try, which is not ideal since $|H|$ can be a lot larger than k in practice.

Distributed algorithm 2: LocalRNM. To reduce communication and to increase the privacy budget for each data query, we propose the following heuristic improvement over NoisyCounts. With budget $\frac{\varepsilon}{2}$, each entity i uses RNM to compute the locally-best splitting function $\hat{h}_i$ for their own data S^i at leaf ℓ. Then, with budget $\frac{\varepsilon}{2}$, the coordinator runs the NoisyCounts procedure to find the best splitting function over the set of locally-best splitting functions $\hat{H} = \{\hat{h}_i\}_{i \in [k]}$ rather than the entire H. In LocalRNM, the noise of each data query scales with the number of entities k instead of the number of splitting functions $|H|$. Since the amount of noise injected is in general much less in LocalRNM, this heuristic have the possibility of achieving higher accuracy than NoisyCounts. However, locally-best splitting functions, especially from entities with small datasets, can have poor performance across the union of data, which is why LocalRNM does not satisfy the utility requirements of PrivateSplit. Nevertheless, LocalRNM performs noticeably better than NoisyCounts on four out of seven of our chosen datasets. This suggests that at least one of the locally-best splitting functions tend to be good enough in practice.

4 Experimental evaluation

We evaluated DP-TopDown on seven datasets, described below, that exhibit a wide spectrum of use cases where privacy could be of concern in the distributed setting. We preprocessed categorical features with one-hot encoding. The splitting class for each dataset comprised of evenly spaced thresholds for each feature, with just one threshold at 0.5 for the one-hot encoded features.

Adult: US Census Data to predict if one's income exceeds $\$50K$ (Dua and Graff 2017). It has 32,561 entries in total.

Bank: Predict subscription behavior based on client info from marketing campaigns (Moro, Cortez, and Rita 2014). It has 45,211 entries in total.

KDDCup 1999: Classify network connections as malicious or normal (Cup 1999). It has 494,021 entries in total.

MNIST: Greyscale 28×28 images of handwritten digits (LeCun 1998). The training set consists of 60,000 images, and the test set consists of 10,000 images.

Creditcard: Client payment information to predict default outcome (Yeh and Lien 2009). It has 30,000 entries in total.

Avazu CTR: Based on anonymized mobile ad information, predict whether the ad is clicked (Avazu 2015). It has 1,100,000 entries in total.

Skin: Classify RGB pixels as skin or non-skin (Bhatt and Dhall 2010). It has 245,057 entries in total.

With the exception of MNIST, which had 60,000 training images and 10,000 test images, each dataset was split into fixed training and testing sets with size ratio $9:1$. Adult, Bank, Creditcard and KDDCup all used 10 evenly spaced thresholds for each continuous feature, giving splitting class sizes of 159, 115, 210, 427 respectively. Skin used 32 evenly spaced thresholds, giving a splitting class of size 96. MNIST and CTR had splitting class sizes of 147 and 263 respectively, and we defer their descriptions to Appendix A.7.

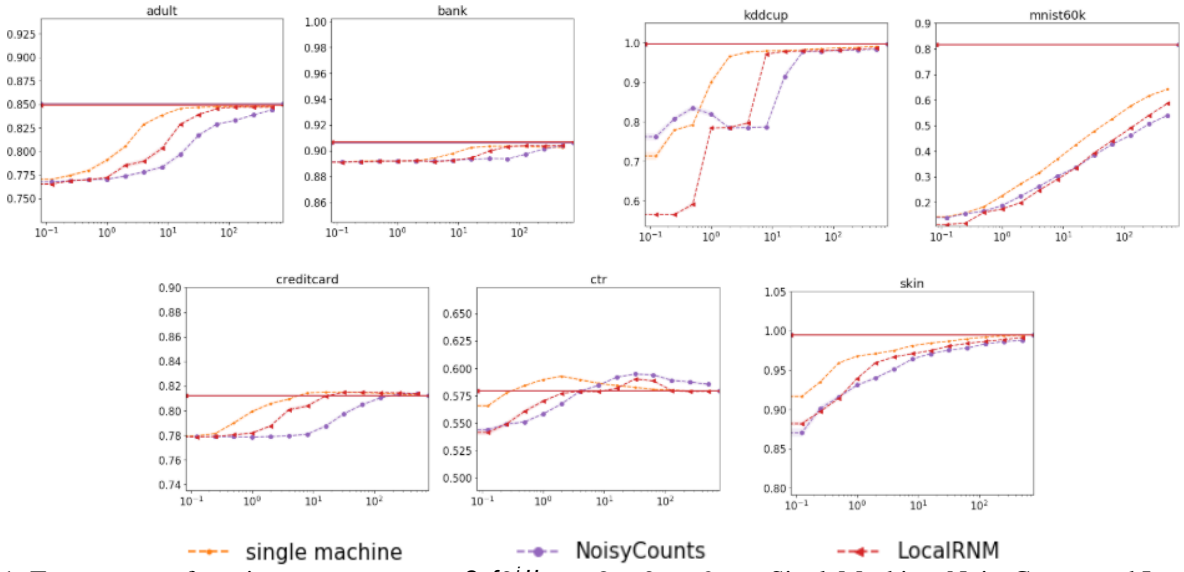

Figure 1: Test accuracy for privacy parameters $\varepsilon \in \{2^i : i = -3, -2, \ldots, 9\}$ on SingleMachine, NoisyCounts and LocalRNM. Baseline runs without noise are shown as solid lines in same color as corresponding dotted lines, representing noised runs.

Our experiments used the entropy splitting criterion, the decay budgeting strategy, $M = 512$, and $\gamma = 0.1$. Data was divided amongst four entities uniformly randomly. When $\hat{J} < 0.01$, we did not push the leaf into Q, since this suggested that it was mostly homogeneous or the entropy estimate was too noisy. Our results are averaged over 100 independent runs. Shaded error bars denote one standard deviation in the mean, which is tiny in most plots.

Figure 1 shows the trade-off between privacy and testing accuracy (i.e. privacy curves) for the three private DP-TopDown algorithms presented in this paper. The (dotted) accuracy of the private algorithms increases as a function of the privacy parameter ε and eventually converges to the (solid) non-private baseline with no noise. As expected, single machine RNM performs the best. In the distributed setting, LocalRNM, while lacking utility guarantees, performs consistently better than NoisyCounts on Adult, Bank, Creditcard and Skin. Both trends agree with our intuition that injecting noise should result in worse performance.

For CTR, the test accuracy is noticeably higher for the private algorithms than their non-private counterparts for ε values. This counter-intuitive result shows the benefit of injecting noise into models that are prone to overfitting, such as decision trees. We found that the non-private decision trees had higher error due to some unlucky greedy choices, which were avoided by adding small amounts of noise through our private mechanisms. This is similar in spirit to adding noise in non-convex optimization problems to escape local optima (Zhang, Liang, and Charikar 2017). Of course, when ε is too small, the decision tree inevitably has low accuracy since the noise is so large that the learning algorithm cannot correctly label the leaves. This observation suggests that when designing private greedy algorithms, there is trade-off not just between privacy and accuracy, but also generalization.

Figure 2 shows the effects of training dataset size on the testing accuracy privacy curves. The curve shapes are regular and larger training sizes consistently result in better test accuracy. We also observe that smaller dataset sizes generally lead to higher variance in the error bars, which is expected since the injected noise has greater relative affect to the data queries. For the privacy curves of small datasets, the benefit of injecting noise for generalization is now very clear. For example, NoisyCount's Bank plot shows that (dotted) private curves of 0.01 and 0.05 consistently outperform the (solid) non-private baselines trained on the same subset of data. These results confirm the intuition we gain from Theorem 3 and its corollaries, that DP-TopDown enjoys the same utility guarantees as TopDown when trained on large datasets.

We present more interesting results in Appendix A.8. Figure 4 shows that training on larger datasets also leads to higher training accuracy, which is counter-intuitive since smaller datasets are easier to overfit. Figure 3 plots the depths and sizes of the learned trees of Figure 1.

5 Conclusion

In this paper, we unify the theory and practice of differentially private top-down decision tree learning in the distributed setting. We provide the first utility guarantees for such algorithms as well as a comprehensive experimental analysis on seven relevant datasets. Our design of DP-TopDown reduces the problem of private top-down learning to the key challenge of approximating optimal splits with PrivateSplit. Using Theorem 3, it is easy to derive utility guarantees for other implementations of PrivateSplit. With the code that will be made available, this is a great starting point for applying scalable and provably accurate differentially private decision trees in the distributed setting.

Acknowledgements: This work was supported in part by NSF grants CCF-1535967, IIS-1618714, CCF-1910321, SES-1919453, and an Amazon Research Award.

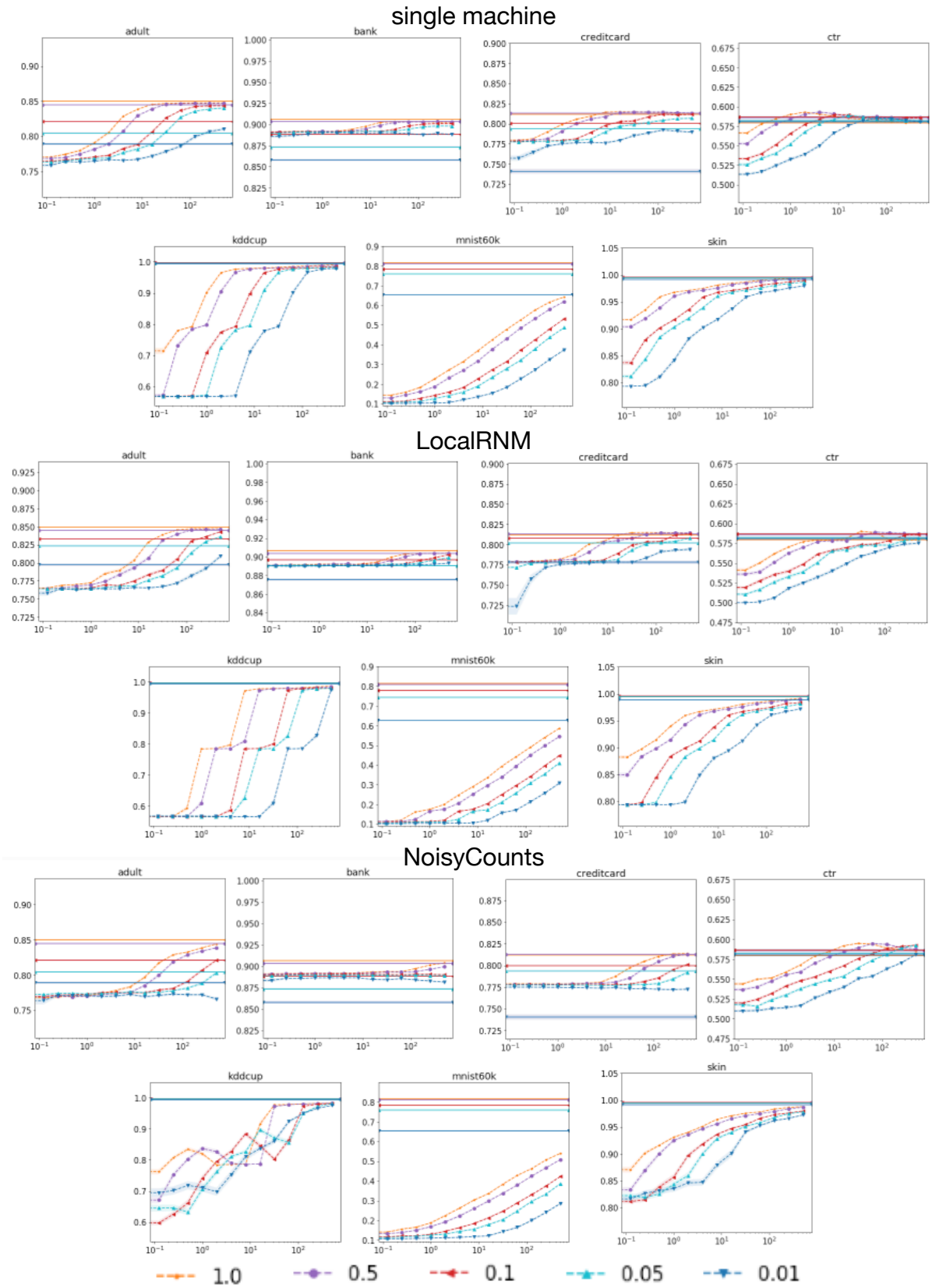

Figure 2: Each plot compares how test accuracy varies with the privacy parameter for a fixed algorithm trained on differently sized training sets. For example, a 0.01 curve was trained on 1% of the total dataset.

References

Avazu. 2015. Avazu click-through rate prediction dataset. Kaggle competition at https://www.kaggle.com/c/avazu-ctr-prediction/.

Balcan, M. F.; Blum, A.; Fine, S.; and Mansour, Y. 2012. Distributed learning, communication complexity and privacy. In *Conference on Learning Theory*, 26–1.

Bhatt, R., and Dhall, A. 2010. Skin segmentation dataset. *UCI Machine Learning Repository*.

Blum, A.; Dwork, C.; McSherry, F.; and Nissim, K. 2005. Practical privacy: the sulq framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 128–138. ACM.

Bojarski, M.; Choromanska, A.; Choromanski, K.; and LeCun, Y. 2014. Differentially-and non-differentially-private random decision trees. *arXiv preprint arXiv:1410.6973*.

Cormode, G.; Jha, S.; Kulkarni, T.; Li, N.; Srivastava, D.; and Wang, T. 2018. Privacy at scale: Local differential privacy in practice. In *Proceedings of the 2018 International Conference on Management of Data*, 1655–1658. ACM.

Cup, K. 1999. Dataset. *available at the following website http://kdd. ics. uci. edu/databases/kddcup99/kddcup99. html* 72.

Dua, D., and Graff, C. 2017. UCI machine learning repository.

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

Dwork, C.; McSherry, F.; Nissim, K.; and Smith, A. 2006. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Theory of Cryptography Conference (TCC)*, 265–284. Springer.

Dwork, C.; Rothblum, G. N.; and Vadhan, S. 2010. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, 51–60. IEEE.

Evans, D.; Kolesnikov, V.; Rosulek, M.; et al. 2018. A pragmatic introduction to secure multi-party computation. *Foundations and Trends R in Privacy and Security* 2(2-3):70–246.

Friedman, A., and Schuster, A. 2010. Data mining with differential privacy. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 493–502. ACM.

Jagannathan, G.; Pillaipakkamnatt, K.; and Wright, R. N. 2009. A practical differentially private random decision tree classifier. In *2009 IEEE International Conference on Data Mining Workshops*, 114–121. IEEE.

Kearns, M., and Mansour, Y. 1999. On the boosting ability of top–down decision tree learning algorithms. *Journal of Computer and System Sciences* 58(1):109–128.

Laurent, H., and Rivest, R. L. 1976. Constructing optimal binary decision trees is np-complete. *Information processing letters* 5(1):15–17.

LeCun, Y. 1998. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

McSherry, F. D. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, 19–30. ACM.

Mitchell, T. M. 1997. Machine learning.

Mohammed, N.; Chen, R.; Fung, B.; and Yu, P. S. 2011. Differentially private data release for data mining. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 493–501. ACM.

Moro, S.; Cortez, P.; and Rita, P. 2014. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems* 62:22–31.

Oneto, L.; Ridella, S.; and Anguita, D. 2017. Differential privacy and generalization: Sharper bounds with applications. *Pattern Recognition Letters* 89:31–38.

Satyanarayanan, M. 2017. The emergence of edge computing. *Computer* 50(1):30–39.

Shalev-Shwartz, S., and Ben-David, S. 2014. *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Warner, S. L. 1965. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association* 60(309):63–69.

Yeh, I.-C., and Lien, C.-h. 2009. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications* 36(2):2473–2480.

Zhang, Y.; Liang, P.; and Charikar, M. 2017. A hitting time analysis of stochastic gradient langevin dynamics. *arXiv preprint arXiv:1702.05575*.

A Appendix

A.1 Composition theorems for differential privacy

Theorem 8 (Sequential Composition)**.** *Let $A_1;:::;A_k$ be k mechanisms such that A_i satisfies $\ _i$-differential privacy. Then the mechanism $A(S) = (A_1(S);:::;A_k(S))$ satisfies $(\ \ _i\ \ _i)$-differential privacy. This holds even if A_i is chosen based on the outputs of $A_1(S);:::;A_{i\ \ 1}(S)$.*

Theorem 9 (Parallel Composition)**.** *Let $D_1;:::;D_k$ be a partition of the input domain and suppose $A_1;:::;A_k$ are mechanisms such that A_i satisfies $\ _i$-differential privacy. Then the mechanism $A(S) = (A_1(S \setminus D_1);:::;A_k(S \setminus D_k))$ satisfies $(\max_i\ \ _i)$-differential privacy.*

A.2 Sensitivity of entropy

In this section, we analyze the sensitivity of entropy. We will use the following standard result.

Lemma 10. *For any $x;y \ 2\ (0;1)$, if $jx\ \ \ yj\ \ \ \frac{1}{2}$, then $jx\lg(x)\ \ \ y\lg(y)j\ \ \ jx\ \ \ yj\lg(jx\ \ \ yj)$.*

We now deduce the global sensitivity of J, used in our implementation of PrivateSplit with RNM. In the proof below, we use the notation $J(S;h)$ where S is a dataset, which is more general than $J(`;h)$ (equivalently $J(S_{\cdot};h)$) for leaf $`$.

Lemma 11. *Suppose $G(q) = \ \ q\lg(q)\ \ \ (1\ \ \ q)\lg(1\ \ \ q)$, the entropy function. When applied to a dataset of size m, then for any $h\ 2\ H$, the sensitivity of $J^0(S) = J(S;h)$ is $\Delta J^0\ \ \ \frac{2}{m}(3\lg(m)+1)$.*

Proof. For a dataset S of size m and splitting function h, denote $r_{y=i;h=j} = \frac{jfx2Sjf(x)=i;h(x)=jgj}{m}; r_{y=i} = \sum_{j\,2\,f0;1g} r_{y=i;h=j}$ and similarly for $r_{h=j}$. Since the numerator is a count that changes by at most 1 for neighboring datasets S and S^0, we have that $jr\ \ \ r^0j\ \ \ \frac{1}{m}$ (here, $\cdot$ denotes wildcard). Consider an arbitrary $h\ 2\ H$ and fix h. Then,

$$
\begin{aligned}
&jJ(S;h)\ \ \ J(S^0;h)j \\
&= j\ \ [\ \ \sum_{a\,2\,f0;1g} r_{y=a}\lg(r_{y=a})\ \ \ r^0_{y=a}\lg(r^0_{y=a})] \\
&+ [r_{h=0} \sum_{a\,2\,f0;1g} r_{y=a;h=0}\lg(r_{y=a;h=0}) \\
&\quad r^0_{h=0} \sum_{a\,2\,f0;1g} r^0_{y=a;h=0}\lg(r^0_{y=l;h=0})] \\
&+ [r_{h=1} \sum_{a\,2\,f0;1g} r_{y=a;h=1}\lg(r_{y=l;h=1}) \\
&\quad r^0_{h=1} \sum_{a\,2\,f0;1g} r^0_{y=a;h=1}\lg(r^0_{y=a;h=1})]j \\
&\quad \frac{2}{m}\lg(m) + 2(\frac{1}{m} + \frac{2}{m}\lg(m))
\end{aligned}
$$

by triangle inequality, the above lemma, and $x\log(x)$ is increasing when $x\ 2\ (0;e^{\ \ 1})$ (assuming $\frac{1}{m}\ \ \ e^{\ \ 1}$). □

A.3 RNM implements single machine PrivateSplit

We first recall the utility guarantee for Laplace mechanism (Dwork and Roth 2014).

Lemma 12. *If $Y\ \ \ \mathrm{Lap}(b)$ then $\Pr(jYj\ \ \ \ln(\frac{1}{})\ \ b) = $. In general, if $Y_1;Y_2;:::;Y_k$ are i.i.d. samples from $\mathrm{Lap}(b)$, then $\Pr(\max_i jY_ij\ \ \ \ln(\frac{k}{})\ \ b)$.*

We first restate the RNM algorithm for $(\hat{h};\hat{J}) = \mathrm{PrivateSplit}(`;\ ;\ )$. For every $i = 1;2;:::;jHj$, define $f_i(S_{\cdot}) = J(`;h_i)$. By Lemma 11, $\Delta f_i\ \ \ 10\frac{\lg(m)}{m}$. Thus, RNM samples $X_i\ \ \ \mathrm{Lap}(\frac{20\lg(m)}{m})$, calculates $\hat{f}_i = f_i(S_{\cdot}) + X_i$ and computes $i\ \ = \arg\max_{i\,2\,[jHj]} \hat{f}_i$, while preserving -differential privacy. Now we show that this procedure finds nearly optimal splits w.h.p.

Lemma 13. *Let $b > 0$. If $x\ \ \ 2b\log(b)$, then $x\ \ \ b\log(x)$ (Shalev-Shwartz and Ben-David 2014).*

Theorem 14. *The single machine RNM-based procedure described above satisfies: $8\ \ > 0;9N = \Theta(\frac{1}{})$ so that if $jS_{\cdot}j\ \ \ N$, then with probability at least $1\ \ \ $, we have $J(`;\hat{h})\ \ \ \max_h J(`;h)\ \ \ $ and $jJ(`;\hat{h})\ \ \ \hat{J}j\ \ \ $.*

Proof. We show that a stronger claim holds with high probability. By Lemma 12, $jX_ij = jf_i\ \ \ \hat{f}_ij\ \ \ _{\overline{2}};8i\ 2\ [jHj]$ holds with probability $1\ \ \ $ whenever $\ln(\frac{jHj}{})\frac{20\lg(m)}{m}\ \ \ _{\overline{2}}$ for large enough m. By Lemma 13, it suffices to have $m\ \ \ 2b\log(b)$ where $b = \ln(\frac{jHj}{})\ \ \ \frac{40}{}$.

We now show that $jf_i\ \ \ \hat{f}_ij\ \ \ _{\overline{2}};8i$ is sufficient. This is equivalent to $jJ(`;h_i)\ \ \ \hat{J}(`;h_i)j\ \ \ _{\overline{2}};8i$. In particular, $jJ(l;\hat{h})\ \ \ \hat{J}j\ \ \ _{\overline{2}}$.

Let $j\ \ = \arg\max_{i\,2\,[s]} J(`;h_i)$. Then, $f_i\ \ \ f_j$ , which is equivalent to $J(`;\hat{h})\ \ \ \max_h J(`;h)$. This is because $f_j\ \ \ f_i\ \ \ (f_j\ \ \ \hat{f}_j\ )+(\hat{f}_j\ \ \ f_i\ )\ \ \ _{\overline{2}}+(\hat{f}_i\ \ \ f_i\ )$.

Therefore $N(\ ;\ ;\ ) = \Theta(\frac{1}{})$. □

A.4 NoisyCounts implements distributed PrivateSplit

In NoisyCounts, each entity j estimates counts $\hat{c}^j_{y=a;h_i=b};\hat{c}_{y=a};\hat{c}_{h_i=b}$ for every splitting function h_i using the LM with $\frac{3}{jHj}$ privacy budget for each type of count. Since counts have sensitivity of 1, the noise added for LM is sampled from $\mathrm{Lap}(\frac{jHj}{3})$. By Theorem 8, this entire operation preserves -differential privacy.

The coordinator aggregates the counts by summing over the entities $\hat{c}_{y=a;h_i=b} = \sum_{j\,2\,[k]} \hat{c}^j_{y=a;h_i=b}$. Then, the aggregated noisy counts are used to compute each $\hat{J}(`;h_i)$ and locate the maximum.

Theorem 15. *The NoisyCounts procedure described above satisfies: $8\ \ > 0,\ 9N = \Theta(\frac{kjHj}{})$ so that if $jS_{\cdot}j\ \ \ N$ then with probability at least $1\ \ \ $, we have $J(`;\hat{h})\ \ \ \max_h J(`;h)\ \ \ $ and $jJ(`;\hat{h})\ \ \ \hat{J}j\ \ \ $.*

Proof. The idea is that the estimated counts are accurate w.h.p. Then, applying analysis similar to the sensitivity of J, we know that inputting these accurately estimated counts produces accurate estimates of J.

Any estimated count $\hat{c}_{y=a;h_i=b}$ is the true count $c_{y=a;h_i=b}$ noised by $X_1 + X_2 + \cdots + X_k$ where each $X_i \sim \mathrm{Lap}(\frac{3|H|}{\gamma})$. Since w.p. $1 - \frac{\gamma}{3k|H|}$, we have $|X_i| \le \ln(\frac{3k|H|}{\gamma})\frac{3|H|}{\gamma}$, applying union bound yields that w.p. $1 - \frac{\gamma}{3}$, we have $|\hat{c}_{y=a;h_i=b} - c_{y=a;h_i=b}| \le \ln(\frac{3k|H|}{\gamma})\frac{3k|H|}{\gamma}$ for every splitting function h_i. This is also true for $c_{y=a}$ and $c_{h_i=b}$. In the notation of the proof for Lemma 11, we have $|r' - r| \le \ln(\frac{3k|H|}{\gamma})\frac{3k|H|}{m}$; the same analysis leads to $|\hat{J}(\ell; h_i) - J(\ell; h_i)| \le 10\ln(\frac{3k|H|}{\gamma})\frac{3k|H|}{m}\ln(\frac{m}{\ln(\frac{3k|H|}{\gamma})3k|H|}) \le 30\ln(\frac{3k|H|}{\gamma})\lg(m)\frac{k|H|}{m}$, assuming $\ln(\frac{3k|H|}{\gamma})\frac{3k|H|}{m} \le (0, \frac{1}{e})$. Thus, with probability at least $1 - \gamma$, $|\hat{J}(\ell; h_i) - J(\ell; h_i)| \le \frac{\alpha''}{2}; \forall i$ when $m \ge 2b\log(b)$ with $b = 60\ln(\frac{3k|H|}{\gamma})\frac{k|H|}{\gamma}$. This is sufficient, as shown in the proof of Theorem 14.

Therefore, $N(\alpha, \gamma, \varepsilon) = \tilde{\Theta}(\frac{k|H|}{\alpha\varepsilon})$. □

A.5 Proof of Main Theorem

Theorem 3 is our main boosting-based utility guarantee. We first define some notation and recount key arguments used to prove boosting-based utility guarantees for TopDown (Kearns and Mansour 1999).

Define $\varepsilon_t = \varepsilon(T_t)$ and $G_t = G(T_t)$ where T_t is the decision tree at the t^{th} iteration for DP-TopDown. Let $t \in [M]$ be arbitrary but fixed step in the algorithm. After t splits, there exists leaf ℓ such that $w(\ell)\min(q(\ell); 1 - q(\ell)) \ge \frac{\varepsilon_t}{t}$, since the tree T_t has t leaves. By using the weak learning assumption, Kearns and Mansour (Kearns and Mansour 1999) show that splitting the leaf ℓ with the optimal splitting function (i.e., the split $h \in H$ that maximizes $|J(\ell; h)|$) reduces G_t significantly: $G_{t+1} \le G_t - \frac{\gamma^2 G_t}{4t\log(2=G_t)}$. A solution to this recurrence is given by $G_t \le e^{\frac{c}{\log(t)=c}}$ for constant c, so it suffices to make $(1=\varepsilon')^{O(\log(1=\varepsilon')=\gamma^2)}$ splits.

Now we proceed with the proof of our main Theorem 3. Then, we deduce the distributed version in 16.

Theorem 3. *Let $G(q)$ be entropy. Let $M, \gamma, \varepsilon', \alpha$ be inputs to DP-TopDown, as defined in Algorithm 1. Under the Weak Learning Assumption with advantage γ, DP-TopDown suffices to make $(1=\varepsilon')^{O(\log(1=\varepsilon')=\gamma^2)}$ splits to achieve training error ε' with probability $1 - \delta$ provided the dataset S has size at least*

$$|S| \ge \max\left(\tilde{\Theta}\left(\frac{M}{\varepsilon'\alpha\varepsilon}\right), \frac{2M}{\varepsilon'}N\left(\tilde{\Theta}\left(\frac{\varepsilon'}{M}\right), \frac{\delta}{4}, O\left(\frac{1}{M}\right)\right), \right.$$

where $b = \min_{1 \le d \le M} B(d)$ and $N(\alpha, \gamma, \varepsilon)$ is the data requirement of PrivateSplit.

Proof. Let $\alpha \in (0, \frac{\varepsilon'}{2M})$ be a target accuracy for the guarantees of PrivateSplit. We will set a value for α later in the proof. Recall that $\varepsilon_\ell = \frac{\varepsilon}{2}B(\mathrm{depth}(\ell))$ is the privacy budget for ℓ. We now case on each event:

There are at most $2M + 1$ calls to PrivateSplit. By the properties of PrivateSplit, each call has low error when $|S_\ell| \ge N(\alpha, \frac{\gamma}{2}, \frac{\varepsilon}{2(2M+1)})$ with probability at least $1 - \frac{\gamma}{2(2M+1)}$; that is $|J(\ell; \hat{h}) - \max_h J(\ell; h)| \le \alpha$ and $|J(\ell; \hat{h}) - \hat{J}| \le \alpha$. Thus, all the calls have low error w.p. at least $1 - \frac{\gamma}{2}$.

There are also $2M$ weight estimates on line 10, and we want all of them to have error $\le \alpha$ w.p. at least $1 - \frac{\gamma}{4}$. So we want the weight estimate has low error $|\hat{w}(\ell) - w(\ell)| \le \alpha$ w.p. $1 - \frac{\gamma}{8M}$. By Lemma 12, we need $|S| \ge \ln(\frac{8M}{\gamma})\frac{2}{\alpha^2}$. There are also at most $M + 1$ leaves. We want to bound the total error of leaf labeling by $\frac{\varepsilon'}{2}$ w.p. at least $1 - \frac{\gamma}{4}$, so bound each leaf's error by $\frac{\varepsilon'}{4(M+1)}$ w.p. at least $1 - \frac{\gamma}{4(M+1)}$. Thus, even if we mistakenly picked a label other than the most common label, the label we did pick give error at most $\frac{\varepsilon'}{2(M+1)}$ away from the most common label. RNM with budget $\frac{\varepsilon}{2}$ requires $|S| \ge \ln(\frac{4(M+1)}{\gamma})\frac{8(M+1)}{\varepsilon'}$.

By union bound, all of the above happens with probability at least $1 - \delta$, so assume this high probability event for the rest of the proof. Note that since $\alpha < \frac{\varepsilon'}{2M}$ and $\gamma < \gamma$, the sample complexity of weight estimates is asymptotically lower bounded by the sample complexity of leaf labeling.

Now we show that our algorithm chooses leaves with enough data to satisfy the sample complexity of PrivateSplit. We know after t splits, there exists a leaf ℓ_t s.t. $w(\ell_t)\min(q(\ell_t); 1 - q(\ell_t)) \ge \frac{\varepsilon_t}{t}$. Since $\min(q(\ell_t); 1 - q(\ell_t)) \le \frac{1}{2}$, we have $w(\ell_t) \ge \frac{2\varepsilon_t}{t} \ge \frac{2\varepsilon'}{M}$. Restricting $\alpha < \frac{\varepsilon'}{2M}$, then $\hat{w}(\ell_t) \ge w(\ell_t) - \frac{\varepsilon'}{M}$; so ℓ_t is pushed onto Q, which implies that DP-TopDown will consider splitting this leaf. Furthermore, any leaf ℓ in Q satisfies $w(\ell) \ge \frac{\varepsilon'}{2M}$ since $\alpha < \frac{\varepsilon'}{2M}$. In other words, there is at least one leaf in the queue Q with weight large relative to ε', and no leaves that have very small weight. Therefore, to satisfy the sufficient conditions for PrivateSplit to achieve low error when run on a leaf ℓ, we require that $|S_\ell| \ge N(\alpha, \frac{\gamma}{2}, \frac{\varepsilon}{2(2M+1)})$, which is guaranteed to be satisfied for any leaf in the queue Q when $|S| \ge \frac{2M}{\varepsilon'}N(\alpha, \frac{\gamma}{2}, \frac{\varepsilon}{2(2M+1)})$, since $|S_\ell| \ge \frac{\varepsilon'}{2M}|S|$.

Next, we argue that every iteration of DP-TopDown reduces the potential $G(T)$ by at least half of the non-private algorithm. This is due to a combination of the weak learning assumption, together with the fact that for every leaf $\ell \in Q$ we are guaranteed that PrivateSplit returned a α-optimal split for that leaf. Let $J = \max_h J(\ell; h)$ and $(\hat{h}; \hat{J})$ be the output of PrivateSplit run on leaf ℓ, then

$$\begin{aligned}
&|w(\ell)J - \hat{w}(\ell)\hat{J}| \\
&\quad \le |w(\ell)J - \hat{J}| + |\hat{J}|w(\ell) - \hat{w}(\ell)| \\
&\quad \le |J - J(\ell; \hat{h})| + |J(\ell; \hat{h}) - \hat{J}| + |w(\ell) - \hat{w}(\ell)| \\
&\quad \le 3
\end{aligned}$$

where we used that $w(\ell); \hat{J}$ are bounded by 1. Note that the above is true for any estimated $\hat{J}$ from PrivateSplit; in particular it's true for the leafs in Q. Therefore, after t splits, the leaf l we pop from Q has $\hat{w}\hat{J}$ nearly as good as the optimal split for leaf ℓ. Hence, applying the Kearns

and Mansour weak-learning analysis, we are guaranteed that $G_{t+1} \quad G_t \quad \frac{^2 G_t}{4t \log(2=G_t)} + 6$. Choose such that $6 = \frac{^2 "}{8M \log(2="})}$. Then we have $G_{t+1} \quad G_t \quad \frac{^2 G_t}{8t \log(2=G_t)}$. Then, the solution to the recurrence inequality differs only by a constant. Thus, to achieve error at most $\frac{"}{2}$, it suffices to make $(2=")^{O(\log(2="))= \ ^2)}$ splits, which simplifies to $(\frac{1}{"}="))^{O(\log(1="))= \ ^2)}$. As we bounded the leaf labeling error by $\frac{"}{2}$, the total error of DP-TopDown is at most $"$ as desired.

For the above claim to hold, the size of the dataset needs to be at least

$$jSj \quad \max_{`} f \ln(\frac{8M}{`}) \frac{2}{`}; \frac{2M}{"} N( \ ; \frac{`}{2}; \frac{}{2(2M+1)}) g$$

where the maximum is taken over all leaves $`$ that are encountered by DP-TopDown (i.e., the leaves of any tree $T_t$). The terms in the above maximum only depend on the leaves $`$ through the privacy budget $\ `$ assigned to each leaf. Since the learned decision tree has depth at most $M$, we are guaranteed that $\ ` = \frac{}{2} B(\text{depth } `) \quad \frac{b}{2}$, where $b = \min_{1 \ d \ M} B(d)$. Substituting this into the above bound on jSj completes the proof. □

Corollary 16. *Suppose the distributed setting. Given a distributed implementation of* PrivateSplit*, DP-TopDown has the guarantees of Theorem 3 with sample complexity*

$$jSj \quad \max \ \mathcal{O} \ \frac{kM}{^2 " \ b} \ ; \frac{2M}{"} N \ \mathcal{O}(\frac{^2 "}{M}); \frac{b}{4}; O(\frac{}{M}) \quad :$$

Proof. We are only concerned with formally quantifying the effects of weight estimation and leaf labeling.

Weight estimation can be done distributedly by having each entity publish an estimate of the number of data points at a particular leaf with LM. The budget remains the same as single machine from parallel composition of the entities.

Leaf labeling can be done distributedly by having each entity publish an estimate of the number of data points for each label, with LM. Since we are no longer using RNM, the noise parameter increases by factor of 2 (since there are two possible labels).

In both cases, instead of bounding the magnitude of one Laplace distributed random variable, we bound the sum of k independent Laplace distributed random variables to be less than some error bound with failure probability . We can do so by bounding each error $\frac{}{k}$ with failure probability $\frac{}{k}$, which Lemma 12 provides the sample complexity for. We see in both cases, we accrue a k factor inside and outside the $\ln$ term. In particular, weight estimation requires $jSj \quad \ln(\frac{8kM}{}) \frac{2k}{`}$ and leaf labeling requires $jSj \quad \ln(\frac{4k(M+1)}{}) \frac{8k(M+1)}{"}$. Thus, to account for the distributed setting in Theorem 3, the first term in max accrues an extra k factor, becoming $\mathcal{O}(\frac{kM}{^2 " \ b})$. □

We note that the same style of analysis also applies for the splitting criterion $G(q) = 4q(1 \quad q)$ or $G(q) = 2 \sqrt{q(1 \quad q)}$ as in (Kearns and Mansour 1999).

A.6 Future work

In our current work, we give half the budget for internal nodes, specifically weight estimation and PrivateSplit, and the other half for labeling leaves. In general, spending half the budget for labeling leaves can be too excessive, especially if the budget is large enough that the noise is much less than 0.5 w.h.p. In any case, having noise $< 0{:}5$ will guarantee selecting the correct label. Therefore, exploring how to balance the budget between splits and labeling leaves is an interesting direction for future work.

We also think it's possible to extend our algorithm for private weak learners; in particular, PrivateSplit only needs to privately return a splitting function that beats random guessing for the leaf w.h.p. We leave the details for future work.

A.7 Splitting classes for MNIST and Avazu CTR

In this section we describe the splitting class we used for MNIST and Avazu CTR.

MNIST: The splitting class for MNIST consists of three thresholds on the average pixel intensity of the 49 $4 \quad 4$ blocks of each $28 \quad 28$ image. For example, the smallest threshold on the "first" block for a flattened MNIST image x would return 1 if $(x[0] + x[1] + x[28] + x[29])=4 \quad 42{:}5$ and 0 otherwise.

Avazu CTR: The splitting class for Avazu CTR were chosen without a fixed number of thresholds for each feature since the range for each feature fluctuates more than that of the other datasets. The splitting class consists of 7 thresholds on $C1$, 100 thresholds on $C14$, 4 thresholds on $C15$ and $C16$, 40 thresholds on $C17$, 10 thresholds on $C19$ and $C21$, and 15 thresholds on $C20$.

A.8 More experimental results

Figure 3 shows the training accuracy privacy curves as well as plots of the depths and sizes of the learned decision trees in Figure 1. In general, the depth and size of the tree decreases as the privacy budget decreases. This suggests that DP-TopDown is inclined to terminate earlier if it determines that it is no longer productive to make the deeper splits. We also observe an interesting phenomenon that NoisyCounts consistently learns the deepest trees, while LocalRNM consistently learns the shallowest trees.

Figure 4 shows the effects of training dataset size on the training accuracy privacy curves. It is essentially the training accuracy analog of Figure 2. In these plots, we see the same trend that greater training size leads to better training accuracy, which is what we observed for testing accuracy in Figure 2. This is especially interesting for training accuracy since smaller datasets are easy to overfit and thus have higher training accuracies, which we indeed observe in the baselines. However, the training privacy curves show that the training accuracy for private decision trees trained on smaller datasets remain lower than that of private decision trees trained on larger datasets. Therefore, the plots in Figure 4 suggest that the effect of dataset size on the performance of private algorithms is strong enough to even overcome the effect of overfitting on small datasets.

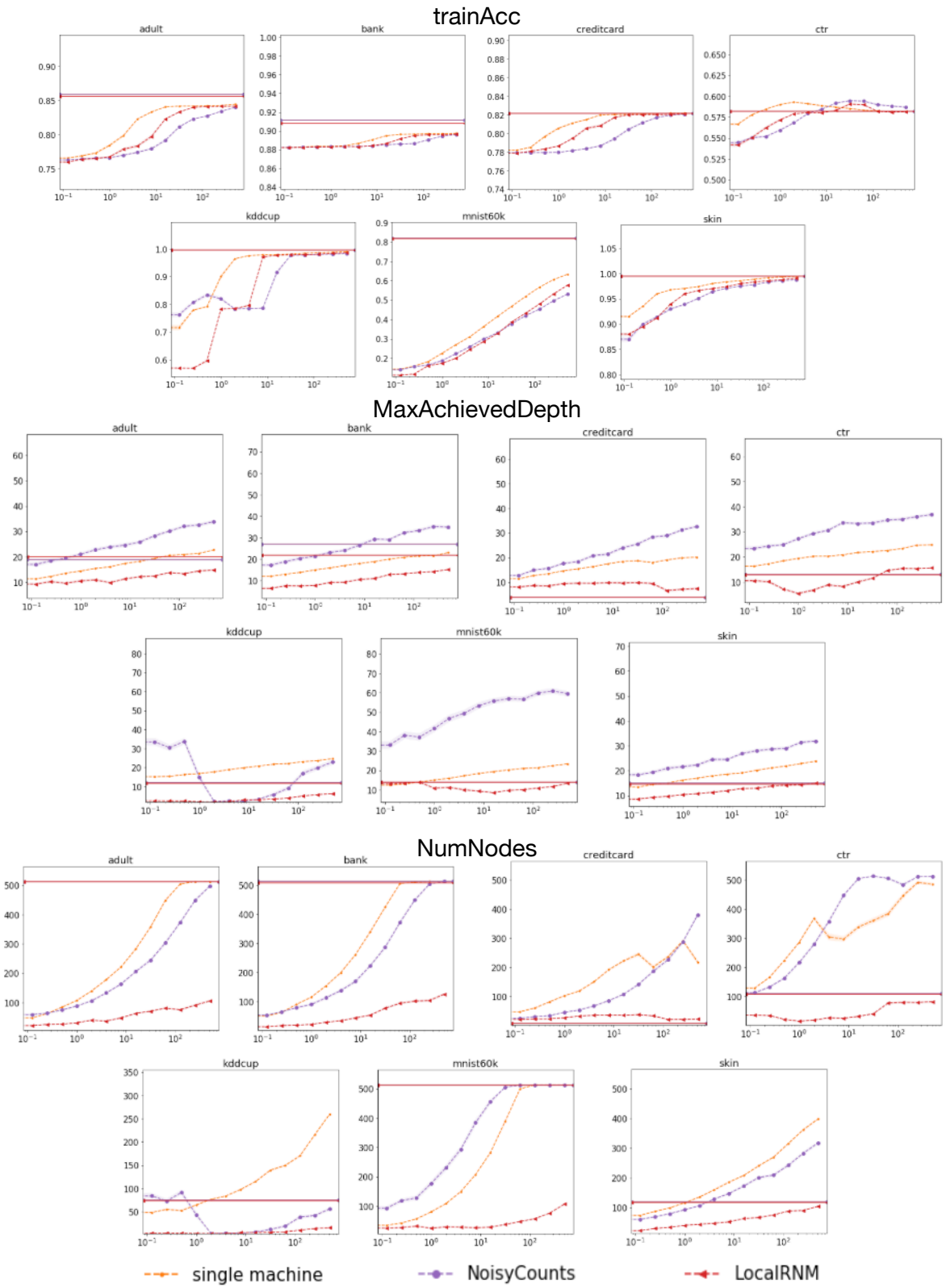

Figure 3: Training accuracy, number of nodes and maximum achieved depth of decision trees under same setup as Figure 1.

Figure 4: Training accuracy under same setup as Figure 2.