CCS 2022: Federated Boosted Decision Trees with Differential Privacy

Samuel Maddock[1], Graham Cormode[2], Tianhao Wang[3], Carsten Maple[1], Somesh Jha[4]

[1] University of Warwick, [2] Meta AI, [3] University of Virginia, [4] University of Wisconsin-Madison

WARWICK

Our Work

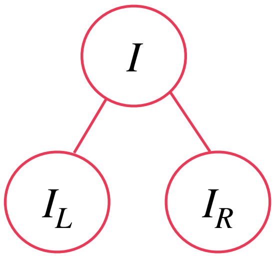
WARWICK

- Gradient Boosted Decision Trees (GBDTs) widely used for Tabular data (i.e, XGBoost, LightGBM, CatBoost)
- Modern ML moving to large-scale federated settings, data is highly distributed, interested in formal privacy guarantees
- **Goal:** Can we develop accurate, lightweight GBDT methods for the federated setting, under Differential Privacy (DP)?
- **Motivation:**
  - “Simple” baseline models that could be used in the federated setting
  - Existing private tree-based methods focus on central DP and only on decision trees (DTs) or Random Forest (RFs)
  - Existing federated GBDT methods lack DP
    - Focus on replicating GBDT exactly under Homomorphic Encryption (HE) or Secure Multi-party Computation (MPC)
- **Setting:**
  - Horizontal Federated Learning (HFL), each client holds some data over all features
  - Honest-but-curious clients and central aggregator (server), threat model orthogonal to our framework
  - Assume each client holds a single data item, can be easily extended

General approach for GBDTs

- Compute a set of split candidates - i.e thresholds to split on
    - ***Example:*** XGBoost computes quantiles of each feature and use these as thresholds
    - Splitting feature j with threshold t, $I = I_L \cup I_R, I_L = \{i : x_{ij} \leq t\}, I_R = \{i : x_{ij} > t\}$
- Build a tree greedily
    - Compute a "split score" for each (feature, split candidate pair), choose the largest

$$SS(I_L, I_R) = \frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda}$$



- If no good splits exist then the node becomes a leaf with a weight which is used for prediction

$$w_l = - \frac{\sum_{i \in I} g_i}{\sum_{i \in I} h_i + \lambda}$$

- **Observation:** Only depends on aggregated gradients and hessians

Private GBDT Framework: Components

WARWICK

- Break the GBDT algorithm into 3 main components:
  - **(C1)** Split Method
    - "Choosing the split"
  - **(C2)** Weight Update Method
    - "Computing the leaf weights"
  - **(C3)** Split Candidate Method
    - "Computing candidate thresholds"
- All three require querying the data
  - Need to add DP noise
- Two additional components:
  - (Maximum) Feature Interactions
  - Batched Updates

Algorithm 1 General GBDT

Input: Number of trees T, maximum depth d, number of split candidates Q, privacy parameters ϵ, δ

1: For each feature $j = 1, \dots, m$ generate Q split candidates $S_j := \{s_1^j, \dots, s_Q^j\}$ **(C3)**
2: Initialise the forest $\mathcal{T} \leftarrow \emptyset$
3: **for** $t = 1, \dots, T$ **do**
4:   For each $(x_i, y_i) \in D$ compute the required gradient information (g_i, h_i) based on $\hat{y}_i^{(t-1)}$ **(C2)**
5:   Choose a subset of features $F^{(t)} \subseteq \{1, \dots, m\}$ with $|F^{(t)}| = k$ for the current tree f_t **(A1)**
6:   **while** depth of the current node $\leq d$ **do**
7:     Choose a feature split candidate pair (j, s_q^j) from $F^{(t)}$ **(C1)**
8:     Split the current node with observations I into two child nodes with index sets $I_{\leq} = \{i : x_{ij} \leq s_q^j\}$ and $I_{>} = I \setminus I_{\leq}$
9:     Repeat (6)-(9) recursing separately on the child nodes
10:   For each leaf l calculate a weight $w_l^{(t)}$ from the examples in the leaf according to the chosen update method **(C2)**
11:   Update predictions $\hat{y}_i^{(t)}$ or batch updates **(A2)**
12:   Add the tree to the ensemble $\mathcal{T} = \mathcal{T} \cup \{f_t\}$
13: **return** the trained forest $\mathcal{T}$

Private GBDT Framework: Accounting

- All quantities require aggregated gradients/Hessians
- Consider queries of the form

$$\tilde{q}(I) = \left(\sum_{i \in I} g_i^{(t)}, \sum_{i \in I} h_i^{(t)}\right) + N(0, \sigma^2 I_2)$$

  - Just need bounded gradient information
- **Setting:** express popular GBDT methods under this query - why?
  - Can utilise ***secure-aggregation***, easy to federate, utilise RDP
  - Assume honest-but-curious setting with central aggregator
- To guarantee formal privacy
  - Need to count # of queries $\tilde{q}(I)$ made by each component
  - Explore data-intensive methods (high query count) vs data-independent (no or little query count)

Algorithm 1 General GBDT

Input: Number of trees T, maximum depth d, number of split candidates Q, privacy parameters ϵ, δ

1: For each feature $j = 1, \dots, m$ generate Q split candidates $S_j := \{s_1^j, \dots, s_Q^j\}$ **(C3)**
2: Initialise the forest $\mathcal{T} \leftarrow \emptyset$
3: **for** $t = 1, \dots, T$ **do**
4:   For each $(x_i, y_i) \in D$ compute the required gradient information (g_i, h_i) based on $\hat{y}_i^{(t-1)}$ **(C2)**
5:   Choose a subset of features $F^{(t)} \subseteq \{1, \dots, m\}$ with $|F^{(t)}| = k$ for the current tree f_t **(A1)**
6:   **while** depth of the current node $\leq d$ **do**
7:     Choose a feature split candidate pair (j, s_q^j) from $F^{(t)}$ **(C1)**
8:     Split the current node with observations I into two child nodes with index sets $I_{\leq} = \{i : x_{ij} \leq s_q^j\}$ and $I_{>} = I \setminus I_{\leq}$
9:     Repeat (6)-(9) recursing separately on the child nodes
10:   For each leaf l calculate a weight $w_l^{(t)}$ from the examples in the leaf according to the chosen update method **(C2)**
11:   Update predictions $\hat{y}_i^{(t)}$ or batch updates **(A2)**
12:   Add the tree to the ensemble $\mathcal{T} = \mathcal{T} \cup \{f_t\}$
13: **return** the trained forest $\mathcal{T}$

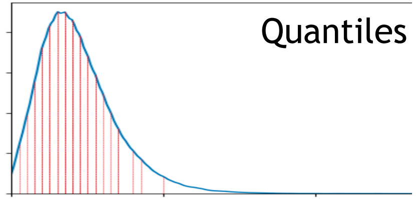
C1: Split Methods

- **Histogram-based (Hist):**
  - Compute a (private) histogram over split-candidates and use this to compute required split-scores at each level
  - Can be computed easily with $\tilde{q}(I)$ under secure-agg + DP
  - Queries proportional to num of trees * features * max depth
- **Partially Random (PR):**
  - Pick a random split-candidate for each feature and compute split scores
  - Same queries as Hist but doesn't require a histogram
- **Totally Random (TR):**
  - Pick a feature, split-candidate pair completely at random
  - Almost fully data-independent, requires only perturbing leaf weights

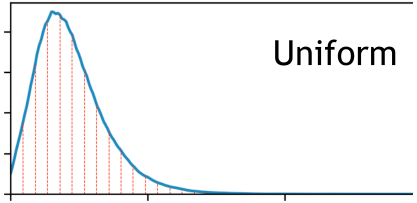
Features
Age
Sex
Salary
…

C3: Split Candidates

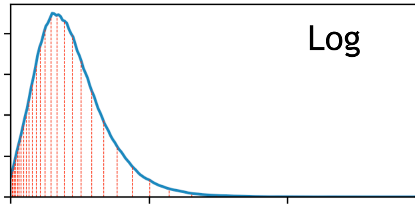
- **Quantiles (non-private):** Standard method used for GBDTs



- **Uniform:** Uniformly divide up features from minimum and maximum
    - Data independent so no privacy-cost

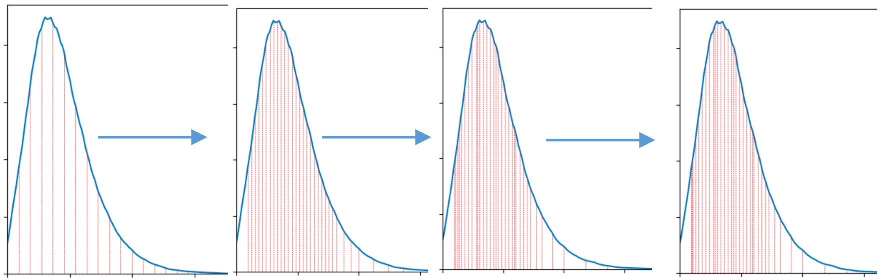


- **Log:** Divide uniformly over the log of a feature (depending on the skew direction)
    - Data independent if you know the skew direction of features

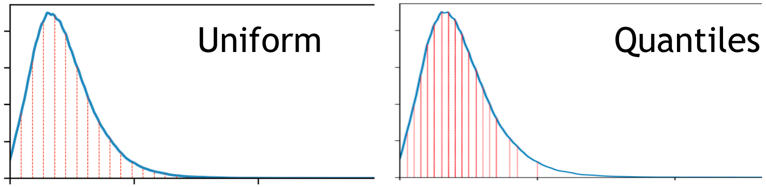


C3: Split Candidates

- **Iterative Hessian (IH)**
  - Mimics XGBoost that form quantile sketches where Hessians are used as weights
  - Form a private Hessian histogram over the current split-candidates
    - **Merge** bins with small total Hessian
    - **Split** bins with large total Hessian (i.e by taking their midpoint)
  - Refine over a number of rounds, helps deal with skew
  - For most splitting methods this information is already gathered so incurs no cost
    - Only time you pay “extra” is with TR splits



Round 1 (uniform) | Round 2 (T=2) | Round 3 (T=3) | Round 4 (T=4)



GBDT Framework

WARWICK

Component	Methods
(C1) Split Method	• Histogram-based (Hist) (§4.3.1) • Partially Random (PR) (§4.3.2) • Totally Random (TR) (§4.3.2)
(C2) Weight Update	• Averaging (§4.4.1) • Gradient (§4.4.2) • Newton (§4.4.3)
(C3) Split Candidate	• Uniform, Log (§4.5.1) • Quantiles (non-private) (§4.5.1) • Iterative Hessian (IH) (§4.5.2)
(A1) Feature Interactions	• Cyclical k-way (§5.1) • Random k-way (§5.1)
(A2) Batched Updates	• $B = 1$ (Boosting) (§5.2) • $B = T$ (RF-type predictions) (§5.2) • $B = p \cdot T$ for some $p \in (0, 1)$ (§5.2)

GBDT Framework: Related Work / Baselines

WARWICK

Table 2: Related works under our framework

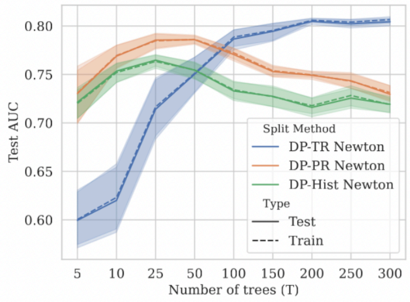
	DP-EBM [51]	FEVERLESS [62]	DP-RF [25]
C1: Split Method	TR	Hist	TR
C2: Weight Update	Gradient	Newton	Averaging
C3: Split Candidate	Uniform (DP Hist)	Quantile Sketch	N/A
A1: Feature Interactions	Cyclical ($k = 1$)	m-way	m-way
A2: Batched Updates	$B = 1$	$B = 1$	$B = T$

- **DP-EBM (Nori et al, ICML21):** Focus on private and explainable GBDT model
  - Uses TR splits with gradient updates under GDP
  - Each tree only considers a single feature
- **FEVERLESS (Wang et al 2022):** Originally vertical FL, faithfully translating XGBoost into a DP-FL setting
- **DP-RF (Fletcher et al 2015):** Central DP method that builds an RF via TR splits
  - DP-RF corresponds to using TR splits, averaging weight update and uniform split candidates

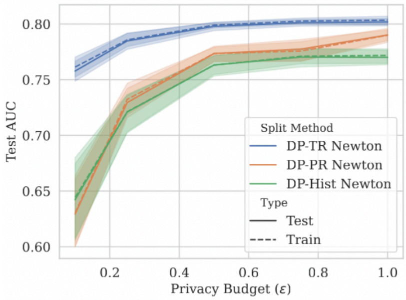
C1 & C2: Split Methods and Weight Updates

WARWICK

- Varying split-methods on Credit 1
  - Histogram, TR, PR
- **Conclusions:**
  - TR competitive but typically requires large T to get better results than histogram
  - PR helps but still performs worse than TR
  - Newton updates perform well for larger privacy budget ($\varepsilon > 0.5$)
  - For higher privacy ($\varepsilon = 0.1 - 0.5$) averaging updates (i.e, RFs) sometimes perform better



(a) **Varying T with $d = 4, \epsilon = 1$**



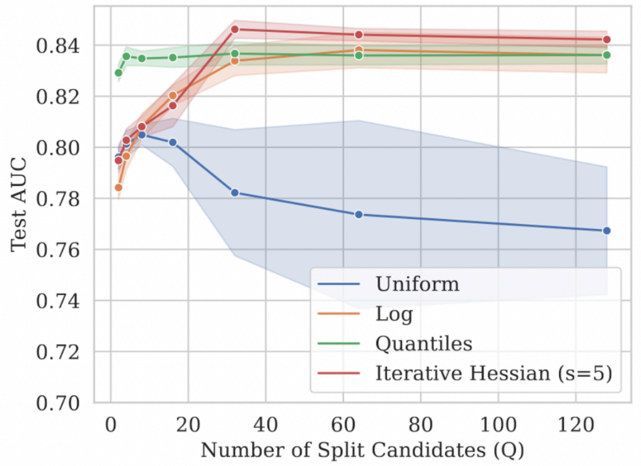
(c) **Varying ϵ**

WARWICK

C3: Split Candidates

- Varying the number of split candidates Q
- **Methods:** Uniform, Log, Quantiles (non-private), IH
- With skewed features
    - (Our) IH method performs the best
    - As Q increases, uniform splits variably degrade performance
- Without skewed features
    - Split candidate methods often perform similarly

- **Conclusion:**
    - Refining split-candidates over rounds can help
    - Only a small amount of budget is needed to give good improvements

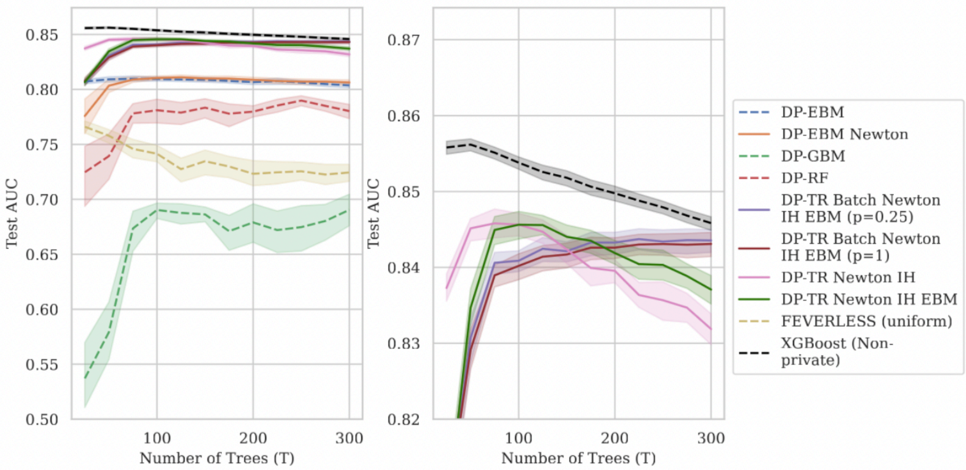


(c) **Varying Q with $T = 100, d = 4, \epsilon = 1$**

End-to-End Comparison

WARWICK

- **Bottom 3:**
  - Methods that faithfully replicate the centralised algorithm under DP perform worst
    - *DP-GBM, FEVERLESS, DP-RF*
  - Too high a privacy cost
- **Top 3:**
  - Combining Newton updates, totally random splits, IH split candidates, large batches
  - Essentially a private, random XGBoost model
- **Conclusions:**
  - The best individual components also work the best when combined together
  - Batching is surprisingly effective
  - RDP accounting + GBDT achieves results close to that of non-private baselines



$$\varepsilon = 1, d = 4$$

Key Takeaways

Core message: *Faithfully replicating the GBDT algorithm under privacy in the Federated setting is not the ideal solution for high-utility*

- Can achieve good performance with few rounds of communication by batching random trees
- Proposing split candidates over multiple rounds can often lead to better utility
- Boosting doesn't always have a clear advantage over RF in high privacy settings
- ***Overall:*** Use less data-intensive (or even data-independent) methods in areas where we can afford to lose performance