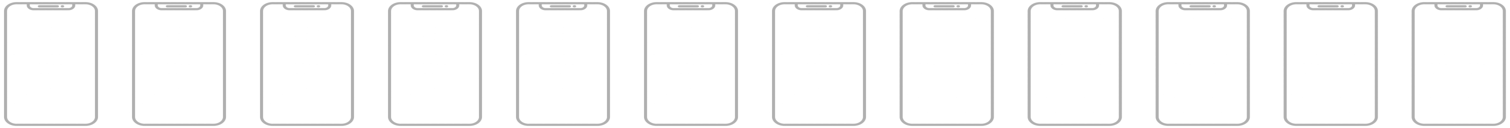
Federated Calibration and Evaluation of Binary Classifiers

Graham Cormode
Igor Markov
Meta

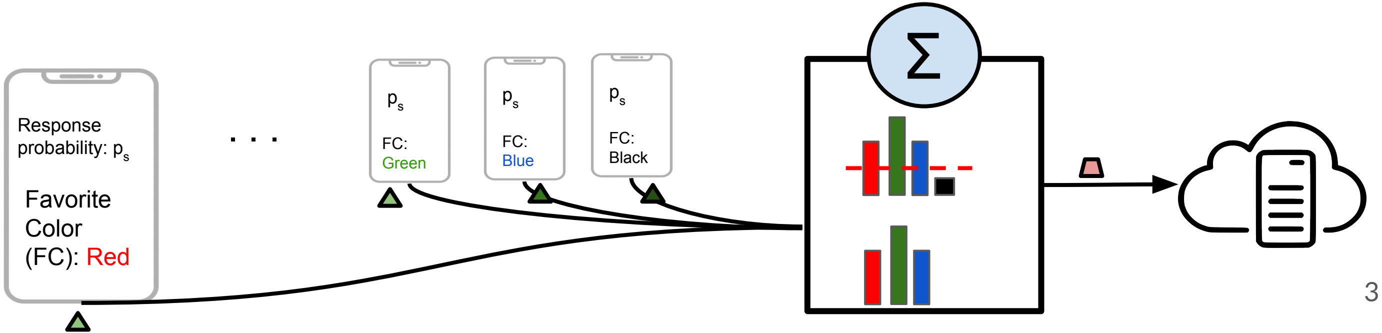
The Federated Computation model

- Privacy-preserving computations distributed over collections of users at web scale
    - Can be thought of as “A privacy-preserving MapReduce”
- Data stays on client devices, only sufficient statistics are shared: data minimization, purpose limitation
- Additional privacy comes from (local or central) differential privacy and secure multiparty computation
- Federated Computation has been widely adopted in practice (Google, Apple, Meta, etc.)



Federated Learning and Federated Analytics

- Federated Learning is the most well-known case of federated computation
  - Many users work together to train an ML model privately
- Federated Analytics captures a broader range of other computations
  - Gathering statistics and metrics, informing decisions
- Collectively, Federated Computation is built on a growing set of primitives
  - Performing fundamental tasks (sums, counts and more) with various privacy guarantees



Federated Analytics to support Federated Learning

Most focus on Federated Learning (FL) has been on the core training procedure

- Typically, via collection of gradients from batches of clients over multiple epochs

A complete end-to-end learning solution has many additional steps:

- Feature selection
- Feature normalization
- Model calibration and evaluation
- Model maintenance / checking

These steps share the same privacy concerns as the core FL training

Post-training statistics

Given a (binary) classifier that has been trained, we want to evaluate:

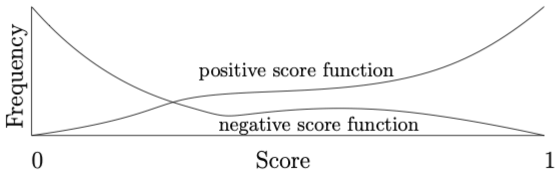
- ROC AUC (Area Under Curve): a measure of quality of the classifier
- Calibration curves: function to accurately measure the confidence of a prediction
- Other metrics: precision, recall, accuracy, normalized entropy (NE) …

In the federated setting, each client has an example with a ground truth label (positive or negative)

We want to calculate these measures under appropriate privacy guarantees

From score functions to score histograms

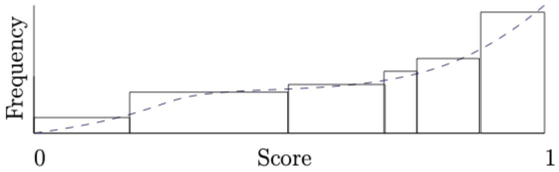
A classifier is described by a *score function* s(x) that maps examples x to [0, 1]



Many classifier metrics are based on example scores and ground truth labels

Score histograms approximate this: how many (positive, negative) examples are there for a range of scores?

We can build score histograms in different federated computing models:



- Secure Aggregation: server only sees the sum of the inputs
- Local DP: each client message achieves differential privacy locally
- Distributed DP: clients each add small noise to produce an overall DP guarantee

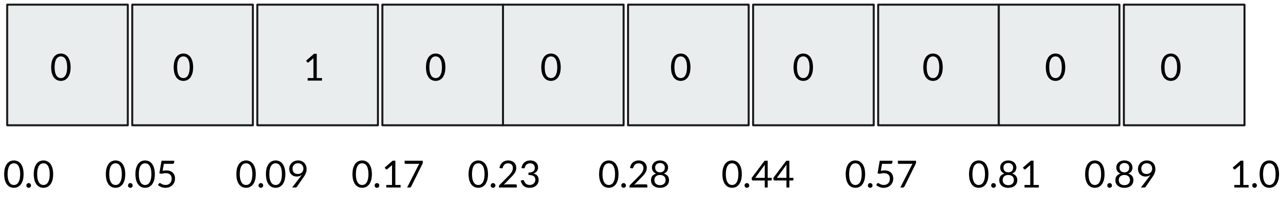
Federated Score Histogram construction

Divide scores into B equal sized bins:

- Round 1: Compute (approximate) **quantiles** on the client score functions to define bin boundaries

Compute (empirical) frequency in each bin

- Round 2: Share bin boundaries, and collect **histogram** of positive and negative examples



Area Under Curve

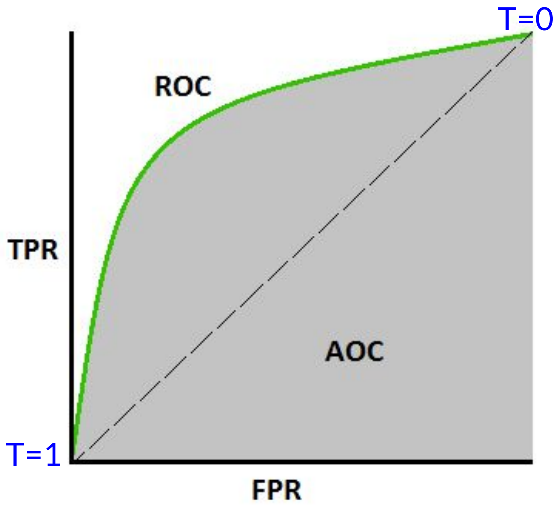
Given the score function, we predict x is positive if s(x) > T, else negative

Different choices of T give false positive (FP) / false negative (FN) tradeoffs

Receiver Operator Characteristic curve: plot FPR against TPR as T varies;
Area Under Curve (AUC) measures the tradeoff, between 0.5 and 1.0

Basic calculation: sort examples by score, numerically integrate (quadrature)

Alternate interpretation of AUC:
AUC is the probability a positive example is ranked above a negative one



- Compute the number of correctly ordered pairs, divided by the number of all pairs

Federated Area Under Curve

Computing statistics about pairs of examples is hard in federated model when examples are distributed

We can use score histograms to approximate the AUC:

- Multiply the number of negative examples in a bin by the number of positive examples in lower bins
- The uncertainty is bounded by the number of positive and negative examples in the same bin
- We can bound the error based on the size of each bin, and the privacy noise added to each bin

We prove bounds on these errors in the full paper

Experimental Results on AUC

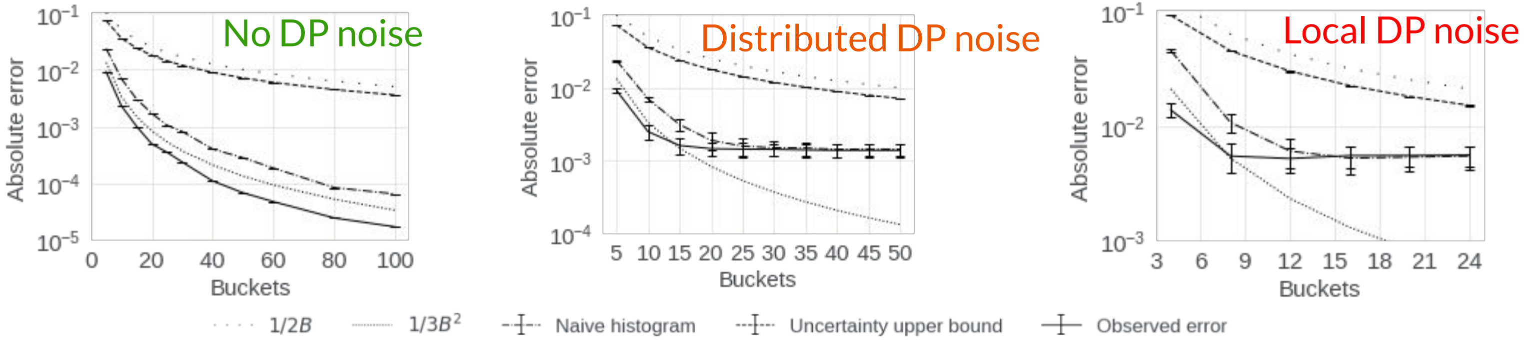
We evaluated our methods on synthetic data from recent Kaggle “tabular” challenges

We compared the accuracy of AUC estimation against the exact value, via:

- Naive score histogram (divide score range into uniform buckets)
- Quantile score histogram (use sum of approximate ranks)
- Compare against pessimistic upper bound on uncertainty

Compare secure aggregation (no noise), distributed DP (moderate noise) and local DP (significant noise)

Secure Aggregation AUC Results



- Error quickly becomes negligible (10^{-3} with 20 buckets, 10^{-4} with 60 buckets) for no noise (left)
- For distributed DP noise (centre), error plateaus at around 0.002
- 10-20 buckets achieves < 0.005 error for LDP noise (right)

Concluding Remarks

Histograms are a powerful tool in federated computation

- Well-studied by the privacy community in different models
- Provide accurate solutions for a wide range of private data analysis problems

Federated computation still has much potential for more work in the data management community

- Federated learning has mostly concentrated on the training step, what about other tasks?
- Federated analytics combines data summarization with privacy/security: often a good match!