**Faster Boosting with Smaller Memory**

Juliai Alafate and Yoav Freund

Department of Computer Science and Engineering, University of California, San Diego

---

### Introduction

- Boosting algorithms, in particular gradient boosted trees, are some of the most popular learning algorithms used in practice.
- One significant limitation of these methods is that they require all of the training data be stored in main memory, otherwise the training time will suffer a significant penalty.
- We propose **Sparrow** that can run efficiently on machines with smaller memory than the training set, suffer no penalty in accuracy, and train 10-100x faster than other popular boosting implementations.

### Setting

- **Input:** joint distribution \( D \) over \((X,Y)\), where \( X \) are feature vectors, \( Y \in \{-1, +1\} \) are labels
- **Objective:** find a classifier \( c: X \to Y \) with small error: \( P_{(X,Y) \sim D} (c(X) \neq Y) \)
- We are given a set \( H \) of base classifiers \( h: X \to [-1, +1] \)
- The final score function is a weighted sum of \( T \) rules from \( H \)

\[
S_T(f) = \sum_{t=1}^{T} \alpha_t h_t(f).
\]

The strong classifier is the sign of the score function: \( H_T = \text{sign}(S_T) \)

**Boosting algorithms** coordinates-wise gradient descent on the average potential, where each coordinate corresponds to one base rule \( h_k \).

**Example:** AdaBoost iteratively finds the direction (base rule) that maximizes the decrease of the average potential function,

\[
\Phi(u) = \frac{1}{n} \sum_{i=1}^{n} \exp(-y_i f(x_i))
\]

### Weighted Sampling: Disk-to-Memory Efficiency

- The estimation accuracy of \( \hat{\gamma}(h) \) can be quantified by the variance of the estimator, \( \text{Var}(\hat{\gamma}) \).
- We define the effective sample size \( n_e \) to be \( 1/\text{Var}(\hat{\gamma}) \):

\[
n_e = \frac{\sum_{i=1}^{n} w_i^2}{\sum_{i=1}^{n} y_i}
\]

Example: Suppose the memory can hold \( n \) data points, whose weights are \( w_1 = \cdots = w_k = 1/k \) and \( w_{k+1} = \cdots = w_n = 0 \). In this case \( n_e = k \).

- If \( n_e \ll n \) then we are wasting valuable memory space on examples with small weights, which can significantly increase over-fitting.
- We use weighted sampling to repopulate memory with \( n \) equally weighted examples to improve memory efficiency and continue learning without over-fitting.

- **Minimal-variance weighted sampling** reads examples \( i \) from disk, and accepts it with probability proportional to its updated weight \( w_i \).
- **Accepted examples** are stored in memory, and uniformly weighted.
- **Resampling happens as Sparrow** is making progress and the weights are becoming increasingly skewed.

### Edge of Weak Rules

- **Sparrow uses an estimator** to identify rules which reduce the true potential:

\[
\Phi(S_t) = E_{(x,y) \sim D} [e^{-\gamma(h)}], \quad \text{where } S_t = S_{t-1} + \alpha_t h_t
\]

- Taking the partial derivative of the potential with respect to \( \alpha_t \) at the iteration \( t \), we get true edge \( \hat{\gamma}(h) \):

\[
\hat{\gamma}(h) = E_{(x,y) \sim D_t} [h/b(x)y],
\]

where \( D_{t-1} = \frac{1}{Z} \exp(-S_{t-1}(x)y) \) and \( Z \) is the normalization term.

- **Most boosting implementations** (e.g., XGBoost and LightGBM) minimize the potential function by finding the base rule with the largest empirical gradient:

\[
\hat{\gamma}(h) = \sum_{i=1}^{n} \frac{w_i}{Z} h_i(x_i)y_i, \quad \text{where } y_i = e^{-S_{t-1}(x_i)y}.
\]

**Sparrow** finds a base rule which, with high probability, has a large true edge.

### Sequential Analysis

- **Sparrow** reads the minimal number of examples that is necessary to find a weak rule with a significant edge \( \gamma \), as opposed to reading all examples for every model update.
- **Optimal analysis needs to read** \( n > (|M|/\gamma)^2 \) examples to obtain an estimation on a loss in the range of \([-M, M]\) with the standard deviation smaller than some \( \epsilon > 0 \).
- **Sparrow** decides the sample size \( n \) on the fly. It calculates loss on one example at a time, and uses a stopping rule to decide if the deviation of the estimation is not likely to be larger than \( \epsilon \), based on the variance of the sequence of losses seen so far.

Define \( M_t = \sum_{i=1}^{n} w_i(y_i(h(x_i)y - \gamma)) \). Stopping rule triggers when it identifies a base rule \( h \) whose correlation is larger than \( \gamma \).

- The stopping rule depends on \( 1/n_e \), when the weights diverge, \( n_e \) becomes smaller than \( n \) and the stopping rule requires proportionally more examples before stopping.

### Stratified Sampling

- **Rejection rate** is high when sampling from a highly skewed distribution.
- We want the sampler to avoid reading examples that it will likely to reject.
- **Sparrow** organizes examples on disk into stratas; the stratum \( k \) contains examples whose weights are in \([2^k, 2^{k+1})\).
- The reject rate within each stratum is at most 1/2, which greatly improves the speed of sampling.

### Experiments

- **Trained on two instances with different memory sizes, one smaller than the data size \( S \), the other larger than the data size \( L \).**
- **Sparrow** enjoys 3-20x speed-up, especially when the memory size is small.  

---

**References**

1. Department of Computer Science and Engineering, University of California, San Diego
2. Faster Boosting with Smaller Memory
3. Bathymetry dataset
4. Splice Site dataset
5. Experiments
6. Stratified Sampling