LightGBM: A Highly Efficient Gradient Boosting Decision Tree

Guolin Ke[1], **Qi Meng**[2], **Thomas Finley**[3], **Taifeng Wang**[1],
Wei Chen[1], **Weidong Ma**[1], **Qiwei Ye**[1], **Tie-Yan Liu**[1]
[1]Microsoft Research [2]Peking University [3] Microsoft Redmond
[1]{guolin.ke, taifengw, wche, weima, qiwye, tie-yan.liu}@microsoft.com;
[2]qimeng13@pku.edu.cn; [3]tfinely@microsoft.com;

Abstract

Gradient Boosting Decision Tree (GBDT) is a popular machine learning algorithm, and has quite a few effective implementations such as XGBoost and pGBRT. Although many engineering optimizations have been adopted in these implementations, the efficiency and scalability are still unsatisfactory when the feature dimension is high and data size is large. A major reason is that for each feature, they need to scan all the data instances to estimate the information gain of all possible split points, which is very time consuming. To tackle this problem, we propose two novel techniques: *Gradient-based One-Side Sampling* (GOSS) and *Exclusive Feature Bundling* (EFB). With GOSS, we exclude a significant proportion of data instances with small gradients, and only use the rest to estimate the information gain. We prove that, since the data instances with larger gradients play a more important role in the computation of information gain, GOSS can obtain quite accurate estimation of the information gain with a much smaller data size. With EFB, we bundle mutually exclusive features (i.e., they rarely take nonzero values simultaneously), to reduce the number of features. We prove that finding the optimal bundling of exclusive features is NP-hard, but a greedy algorithm can achieve quite good approximation ratio (and thus can effectively reduce the number of features without hurting the accuracy of split point determination by much). We call our new GBDT implementation with GOSS and EFB *LightGBM*. Our experiments on multiple public datasets show that, LightGBM speeds up the training process of conventional GBDT by up to over 20 times while achieving almost the same accuracy.

1 Introduction

Gradient boosting decision tree (GBDT) [1] is a widely-used machine learning algorithm, due to its efficiency, accuracy, and interpretability. GBDT achieves state-of-the-art performances in many machine learning tasks, such as multi-class classification [2], click prediction [3], and learning to rank [4]. In recent years, with the emergence of big data (in terms of both the number of features and the number of instances), GBDT is facing new challenges, especially in the tradeoff between accuracy and efficiency. Conventional implementations of GBDT need to, for every feature, scan all the data instances to estimate the information gain of all the possible split points. Therefore, their computational complexities will be proportional to both the number of features and the number of instances. This makes these implementations very time consuming when handling big data.

To tackle this challenge, a straightforward idea is to reduce the number of data instances and the number of features. However, this turns out to be highly non-trivial. For example, it is unclear how to perform data sampling for GBDT. While there are some works that sample data according to their weights to speed up the training process of boosting [5, 6, 7], they cannot be directly applied to GBDT

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

since there is no sample weight in GBDT at all. In this paper, we propose two novel techniques towards this goal, as elaborated below.

Gradient-based One-Side Sampling (GOSS). While there is no native weight for data instance in GBDT, we notice that data instances with different gradients play different roles in the computation of information gain. In particular, according to the definition of information gain, those instances with larger gradients[1] (i.e., under-trained instances) will contribute more to the information gain. Therefore, when down sampling the data instances, in order to retain the accuracy of information gain estimation, we should better keep those instances with large gradients (e.g., larger than a pre-defined threshold, or among the top percentiles), and only randomly drop those instances with small gradients. We prove that such a treatment can lead to a more accurate gain estimation than uniformly random sampling, with the same target sampling rate, especially when the value of information gain has a large range.

Exclusive Feature Bundling (EFB). Usually in real applications, although there are a large number of features, the feature space is quite sparse, which provides us a possibility of designing a nearly lossless approach to reduce the number of effective features. Specifically, in a sparse feature space, many features are (almost) exclusive, i.e., they rarely take nonzero values simultaneously. Examples include the one-hot features (e.g., one-hot word representation in text mining). We can safely bundle such exclusive features. To this end, we design an efficient algorithm by reducing the optimal bundling problem to a graph coloring problem (by taking features as vertices and adding edges for every two features if they are not mutually exclusive), and solving it by a greedy algorithm with a constant approximation ratio.

We call the new GBDT algorithm with GOSS and EFB *LightGBM*[2]. Our experiments on multiple public datasets show that LightGBM can accelerate the training process by up to over 20 times while achieving almost the same accuracy.

The remaining of this paper is organized as follows. At first, we review GBDT algorithms and related work in Sec. 2. Then, we introduce the details of GOSS in Sec. 3 and EFB in Sec. 4. Our experiments for LightGBM on public datasets are presented in Sec. 5. Finally, we conclude the paper in Sec. 6.

2 Preliminaries

2.1 GBDT and Its Complexity Analysis

GBDT is an ensemble model of decision trees, which are trained in sequence [1]. In each iteration, GBDT learns the decision trees by fitting the negative gradients (also known as residual errors).

The main cost in GBDT lies in learning the decision trees, and the most time-consuming part in learning a decision tree is to find the best split points. One of the most popular algorithms to find split points is the pre-sorted algorithm [8, 9], which enumerates all possible split points on the pre-sorted feature values. This algorithm is simple and can find the optimal split points, however, it is inefficient in both training speed and memory consumption. Another popular algorithm is the histogram-based algorithm [10, 11, 12], as shown in Alg. 1[3]. Instead of finding the split points on the sorted feature values, histogram-based algorithm buckets continuous feature values into discrete bins and uses these bins to construct feature histograms during training. Since the histogram-based algorithm is more efficient in both memory consumption and training speed, we will develop our work on its basis.

As shown in Alg. 1, the histogram-based algorithm finds the best split points based on the feature histograms. It costs $O(\#data \times \#feature)$ for histogram building and $O(\#bin \times \#feature)$ for split point finding. Since #bin is usually much smaller than #data, histogram building will dominate the computational complexity. If we can reduce #data or #feature, we will be able to substantially speed up the training of GBDT.

2.2 Related Work

There have been quite a few implementations of GBDT in the literature, including XGBoost [13], pGBRT [14], scikit-learn [15], and gbm in R [16] [4]. Scikit-learn and gbm in R implements the pre-sorted algorithm, and pGBRT implements the histogram-based algorithm. XGBoost supports both

[1] When we say larger or smaller gradients in this paper, we refer to their absolute values.

[2] The code is available at GitHub: `https://github.com/Microsoft/LightGBM`.

[3] Due to space restriction, high level pseudo code is used. The details could be found in our open-source code.

[4] There are some other works speed up GBDT training via GPU [17, 18], or parallel training [19]. However, they are out of the scope of this paper.

the pre-sorted algorithm and histogram-based algorithm. As shown in [13], XGBoost outperforms the other tools. So, we use XGBoost as our baseline in the experiment section.

To reduce the size of the training data, a common approach is to down sample the data instances. For example, in [5], data instances are filtered if their weights are smaller than a fixed threshold. SGB [20] uses a random subset to train the weak learners in every iteration. In [6], the sampling ratio are dynamically adjusted in the training progress. However, all these works except SGB [20] are based on AdaBoost [21], and cannot be directly applied to GBDT since there are no native weights for data instances in GBDT. Though SGB can be applied to GBDT, it usually hurts accuracy and thus it is not a desirable choice.

Similarly, to reduce the number of features, it is natural to filter weak features [22, 23, 7, 24]. This is usually done by principle component analysis or projection pursuit. However, these approaches highly rely on the assumption that features contain significant redundancy, which might not always be true in practice (features are usually designed with their unique contributions and removing any of them may affect the training accuracy to some degree).

The large-scale datasets used in real applications are usually quite sparse. GBDT with the pre-sorted algorithm can reduce the training cost by ignoring the features with zero values [13]. However, GBDT with the histogram-based algorithm does not have efficient sparse optimization solutions. The reason is that the histogram-based algorithm needs to retrieve feature bin values (refer to Alg. 1) for each data instance no matter the feature value is zero or not. It is highly preferred that GBDT with the histogram-based algorithm can effectively leverage such sparse property.

To address the limitations of previous works, we propose two new novel techniques called Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). More details will be introduced in the next sections.

Algorithm 1: Histogram-based Algorithm

Input: I: training data, d: max depth
Input: m: feature dimension
$nodeSet \leftarrow \{0\}$ ▷ tree nodes in current level
$rowSet \leftarrow \{\{0, 1, 2, ...\}\}$ ▷ data indices in tree nodes
for $i = 1$ ***to*** d **do**
  for $node$ ***in*** $nodeSet$ **do**
    usedRows $\leftarrow rowSet[node]$
    for $k = 1$ ***to*** m **do**
      $H \leftarrow$ new Histogram()
      ▷ Build histogram
      for j ***in*** *usedRows* **do**
        bin $\leftarrow I$.f[k][j].bin
        H[bin].y $\leftarrow H$[bin].y + I.y[j]
        H[bin].n $\leftarrow H$[bin].n + 1
      Find the best split on histogram H.
      ...
  Update $rowSet$ and $nodeSet$ according to the best split points.
  ...

Algorithm 2: Gradient-based One-Side Sampling

Input: I: training data, d: iterations
Input: a: sampling ratio of large gradient data
Input: b: sampling ratio of small gradient data
Input: $loss$: loss function, L: weak learner
models $\leftarrow$ {}, fact $\leftarrow \frac{1-a}{b}$
topN $\leftarrow$ a $\times$ len(I) , randN $\leftarrow$ b $\times$ len(I)
for $i = 1$ ***to*** d **do**
  preds $\leftarrow$ models.predict(I)
  g $\leftarrow loss(I$, preds), w $\leftarrow$ {1,1,...}
  sorted $\leftarrow$ GetSortedIndices(abs(g))
  topSet $\leftarrow$ sorted[1:topN]
  randSet $\leftarrow$ RandomPick(sorted[topN:len(I)], randN)
  usedSet $\leftarrow$ topSet + randSet
  w[randSet] $\times =$ fact ▷ Assign weight $fact$ to the small gradient data.
  newModel $\leftarrow$ L(I[usedSet], $-$ g[usedSet], w[usedSet])
  models.append(newModel)

3 Gradient-based One-Side Sampling

In this section, we propose a novel sampling method for GBDT that can achieve a good balance between reducing the number of data instances and keeping the accuracy for learned decision trees.

3.1 Algorithm Description

In AdaBoost, the sample weight serves as a good indicator for the importance of data instances. However, in GBDT, there are no native sample weights, and thus the sampling methods proposed for AdaBoost cannot be directly applied. Fortunately, we notice that the gradient for each data instance in GBDT provides us with useful information for data sampling. That is, if an instance is associated with a small gradient, the training error for this instance is small and it is already well-trained. A straightforward idea is to discard those data instances with small gradients. However, the data distribution will be changed by doing so, which will hurt the accuracy of the learned model. To avoid this problem, we propose a new method called Gradient-based One-Side Sampling (GOSS).