Gradient Guide Decision Trees: Distilling the Knowledge in a Decision Tree As Functional Fixed Point Iteration

Zhang Jinxiong

November 9, 2019

Abstract

Compared with gradient descent, we find that the result of gradient boost decision trees is the sum of successive decision trees rather than the final fitted decision tree. The parameter scale of gradient boost decision trees is much larger than the base weak learner - decision tree, which makes it different from gradient descent methods. We try to make a direct extension of gradient decsent methods to the function space, which is called gradient guide decision tree as a counterpart of gradient boost decision tree. It is easy to generalize to any continuous optimization methods such as ADMM. A general paradigm is developed to connnect optimization methods and 'boosting' algorithms. It is also explained in the framework of knowledge distillization.

1 Introduction

Boosting as Entropy Projection, A Duality View of Boosting Algorithms study some boosting algorithms from a entropy minimization perspective. The gradient boost machine algorithms seems far from continuous optimization methods. Gradient boost machine is usually considered as functional gradient descent. However,

Boosted decision tree or multiple additive regression tree is the sum of successive decision tree

$$F(x) = \sum_{n=1}^{T} f_t(x)$$

where f_n relies on the outputs of its 'parent tree' f_{n-1} for $n = 2, 3, \cdots, N$ when training.

The following algorithm describe the gradient boost decision as boosted decision trees

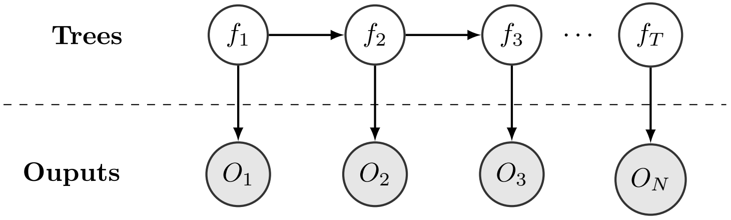
At each stage, a new tree $f_t(x)$ is learnt to fit the gradients $\{r_{i,t} \mid i = 1, 2, \cdots, N\}$. It is obvious that there are more parameters of the final learner $F_T(x)$ than the last base learner f_T.

Algorithm 1 Gradient Boost Decision Trees

```
Input training data set $\{(x_n, y_n) \mid x_n \in \mathrm{X} \subset \mathbb{R}^p, y_n \in \mathbb{R}, n = 1, 2, \cdots, N\}$
Initialize $f_0 = \arg\min_\gamma \sum_{i=1}^{N} L(x_i, \gamma)$
for $t = 1, 2, \ldots, T$ do
    for $i = 1, 2, \ldots, N$ do
        compute $r_{i,t} = -[\frac{\partial L(\mathrm{y}_i, f(x_i))}{\partial f(x_i)} \mid_{f=F^{(t-1)}}]$.
    end for
    Fit a regression tree to the targets $r_{i,t}$ giving terminal regions
                    $R_{j,m}, j = 1, 2, \ldots, J_m.$
    for $j = 1, 2, \ldots, J_m$ do
        compute $\gamma_{j,t} = \arg\min_\gamma \sum_{x_i \in R_{j,m}} L(\mathrm{y}_i, F^{(t-1)}(x_i) + \gamma)$.
    end for
    $f_t = \sum_{j=1}^{J_m} \gamma_{j,t} \mathbb{I}(x \in R_{j,m})$
    Update $F^{(t)} = F^{(t-1)} + \nu f_t, \nu \in (0, 1)$
end for
Output $F^{(T)}(x)$.
```

The model size of gradient boost decision trees is proportional to the number of base learners T while the size of machine learning models is identical to the dimension of the parameter space after updated by gradient descent. The final model of gradient boost decision trees is cumbersome as combination of many decision trees. The question comes how can we do to compress the ensemble tree-based models?

The gradient descent is a fixed point iteration in parameter space. However, gradient boost decision tree is not in the fixed point iteration form. It is more similar to Taylor expansion of differentiable functions where more terms mean higher precision. It sends the gradients to next tree when training gradient boost decision trees. After training, we can compute the results of each tree $f_1(x), f_2(x), \cdots, f_T(x)$ parallelly and add up them with some weights as the result of gradient boost decision trees when inputting x. Such sequential training and parallel inference does not happen in gradient descent.



2 Gradient Guide Decision Trees

In gradient boost decision tree, it takes additive training:

$$\gamma_{j,t} = \arg\min_{\gamma} \sum_{x_i \in R_{j,m}} L(\mathrm{y}_i, \underbrace{F^{(t-1)}(x_i) + \gamma}_{\text{additive training}})$$

which results as weighted sum of decision trees. Here $\gamma_{j,t}$ is expected to be parallel of $-[\frac{\partial L(\mathrm{y}_i, f(x_i))}{\partial f(x_i)} \mid_{f=F^{(t-1)}(x_i)}] = -g_{t-1,i}$ thus $F^{(t)}(x_i) \approx F^{t-1}(x_i) - \alpha g_{t-1,i}$.

If let $F^{(t-1)}(x_i) = \tilde{y}_i$, the following inequality holds for some $\alpha \in (0, 1)$ if the loss function L is smooth

$$\sum_{i=1}^{N} L(\mathrm{y}_i, \tilde{y}_i) \geq \sum_{i=1}^{N} L(\mathrm{y}_i, \tilde{y}_i - \alpha[\frac{\partial L(\mathrm{y}_i, \tilde{y}_i)}{\partial \tilde{y}_i}]).$$

This is basic idea of gradient descent. In gradient boost decision trees, a tree f_t is used to fit the negative gradient $-\frac{\partial L(\mathrm{y}_i, \tilde{y}_i)}{\partial \tilde{y}_i}$ so that it brings some noise or error in this step. If we use a tree $f_t(x_i)$ to fit $\tilde{y}_i - \alpha[\frac{\partial L(\mathrm{y}_i, \tilde{y}_i)}{\partial \tilde{y}_i}]$, it shares the same iterative schemes of gradient descent. Ideally if $f_t(x_i) = \tilde{y}_i - \alpha[\frac{\partial L(\mathrm{y}_i, \tilde{y}_i)}{\partial \tilde{y}_i}]$, it is exactly the gradient descent. It is a parallel to gradient descent in function space so it is a direct extension of gradient descent in function space. However, more usual case is that $f_t(x_i) \approx \tilde{y}_i - \alpha[\frac{\partial L(\mathrm{y}_i, \tilde{y}_i)}{\partial \tilde{y}_i}]$. In another word, $f_t(x_i) = \tilde{y}_i - \alpha[\frac{\partial L(\mathrm{y}_i, \tilde{y}_i)}{\partial \tilde{y}_i}] + \epsilon_{i,t}$ where $\epsilon_{i,t}$ is the error.

Based on gradient boost decision tree, we define a novel way to update decision tree called as gradient guide decision tree.

In gradient guide decision trees, the targets $r_{i,t}$ are updated in the fixed point iteration and a new tree f_t is used to fit such targets where the data flows as following. And the output of gradient guide decision tree is the last trained decision tree. It is the first connection of continuous optimization methods and boosting (ensemble) methods in iterative scheme as known.

3 Optimization and Boosting

We will revisit the connection of optimization and boosting based on the AnyBoost and BOOSTING ALGORITHMS: REGULARIZATION, PREDICTION AND MODEL FITTING.

First we start with the gradient descent

$$\theta^{(t+1)} = \theta^{(t)} - \alpha_t \nabla_{\theta} L(\theta^{(t)})$$

for $t = 0, 1, 2, \cdots, T-1$. The final result of gradient descent is

$$\theta^{(T)} = \theta^{(0)} - \sum_{t=1}^{T-1} \alpha_t \nabla_{\theta} L(\theta^{(t)}).$$

Algorithm 2 Gradient Guide Decision Trees

```
Input training data set $\{(x_n, y_n) \mid x_n \in \mathrm{X} \subset \mathbb{R}^p, y_n \in \mathbb{R}, n = 1, 2, \cdots, N\}$
Initialize $f_0 = \arg\min_\gamma \sum_{i=1}^{N} L(x_i, \gamma)$
for $t = 1, 2, \dots, T$ do
    for $i = 1, 2, \dots, N$ do
        compute $\alpha_{i,t} = \arg\min_\alpha L(\mathrm{y}_i, f_{t-1}(x_i) - \alpha[\frac{\partial L(\mathrm{y}_i, f(x_i))}{\partial f(x_i)} \mid_{f=f_{t-1}}]);$
        compute $r_{i,t} = f_{t-1}(x_i) - \alpha_{i,t}[\frac{\partial L(\mathrm{y}_i, f(x_i))}{\partial f(x_i)} \mid_{f=f_{t-1}}].$
    end for
    Fit a regression tree to the targets $r_{i,t}$ giving terminal regions

                    $R_{j,m}, j = 1, 2, \dots, J_m.$

    for $j = 1, 2, \dots, J_m$ do
        compute $\gamma_{j,t} = \arg\min_\gamma \sum_{x_i \in R_{j,m}} L(\mathrm{y}_i, \gamma).$
    end for
    Update $f_t = \sum_{j=1}^{J_m} \gamma_{j,t} \mathbb{I}(x \in R_{j,m})$
end for
Output $f_T(x)$.
```

In some sense, a decision tree is to fit $-\alpha_t \nabla_\theta L(\theta^{(t)})$ in gradient boost decision trees. So we say that it is analogous to gradient descent rather than parallel to gradient descent like gradient guide decision tree.

In optimization, we call the point γ at point $\tilde{y}_i$ as profitable direction if

$$\sum_{i=1}^{N} L(\mathrm{y}_i, \tilde{y}_i) \geq \sum_{i=1}^{N} L(\mathrm{y}_i, \tilde{y}_i + \gamma).$$

There are many *gradient-like boosting machines* to fit the profitable directions different from negative gradients such as Functional Frank-Wolfe Boosting, Historical Gradient Boost Machine, XGBoost, NGBoost.

It is also the basic idea behind AnyBoost. All these gradient-like boosting machines can be transformed to gradient-like guide machines. Any techniques in gradient-based optimization methods can be applied to gradient guide decision tree such as the distributed optimization techniques, acceleration techniques, variance reduction techniques. It is really a bridge between continuous optimization methods and booted decision tree algorithms by replacing the targets with update formula in optimization methods.

Second we take alternating direction method of multipliers (ADMM) into consideration. The cost function is $\sum_{i=1}^{N} \ell(y_i, f(x_i)) + r \sum_{i=1}^{N} |f(x_i)|$ subject to $z_i = f(x_i)$.

Algorithm 3 ADMM Guide Decision Trees

```
Input training data set $\{(x_n, y_n) \mid x_n \in \mathrm{X} \subset \mathbb{R}^p, y_n \in \mathbb{R}, n = 1, 2, \cdots, N\}$
Initialize $f_0 = \arg\min_\gamma \sum_{i=1}^N L(x_i, \gamma)$
for $t = 1, 2, \ldots, T$ do
    for $i = 1, 2, \ldots, N$ do
        compute $a_{i,t} = \arg\min_{\gamma_i} L_\beta^{(t)}(\gamma_i, z, \lambda)$
        compute new targets $r_{i,t} = a_{i,t} + f_t(x_i)$
    end for
    Fit a regression tree to the targets $r_{i,t}$ giving terminal regions
                        $R_{j,m}, j = 1, 2, \ldots, J_m.$
    for $j = 1, 2, \ldots, J_m$ do
        compute $\gamma_{j,t} = \arg\min_\gamma \sum_{x_i \in R_{j,m}} L(\mathrm{y}_i, \gamma)$.
    end for
    $f_t = \sum_{j=1}^{J_m} \gamma_{j,t} \mathbb{I}(x \in R_{j,m})$
end for
compute $z^{t+1} = \arg\min_z \sum_{i=1}^N L_\beta^{(t)}(r_{i,t}, z_i, \lambda_i^t)$
compute $\lambda^{t+1} = \arg\min_\lambda \sum_{i=1}^N L_\beta^{(t)}(r_{i,t}, z_i^{t+1}, \lambda_i)$
Output $f_T(x)$.
```

Here the augmented Lagragian is given by

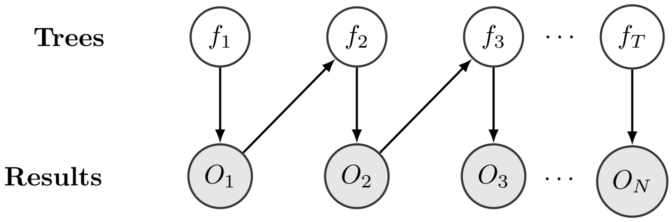
$$
\begin{aligned}
L_\beta^{(t)}(\gamma_i, z_i, \lambda_i) = {} & \ell(y_i, f_{t-1}(x_i) + \gamma_i) \\
& + r|z_i|_1 - \langle z_i - (f_{t-1}(x_i) + \gamma_i), \lambda_i \rangle \\
& + \frac{\beta}{2} |z_i - (f_{t-1}(x_i) + \gamma_i)|.
\end{aligned}
$$

And $f(\cdot)$ is a decision tree; $\ell(\cdot, \cdot)$ is the loss function. This algorithm cannot be derived from the first case.

4 Knowledge Distillization

In this section, we will explain the gradient guide decision trees as special knowledge distillization.

Knowledge Distillization is originally designed to compress the knowledge in a cumbersome model into a single model. The basic idea of knowledge distillization is to learn the soft targets of a cumbersome precise model. In gradient guide decision trees, the soft targets are defined as the gradient descent formula. Even the ancestors of the final decision tree do not perform well, they point out a better direction with the help of gradients.



The knowledge is distilled at each tree and sent to its children trees as follows:

$$r_{i,t} = f_{t-1}(x_i) - \alpha_{i,t} g_{i,t}, i = 1, 2, \cdots, N$$

where $g_{i,t} = [\frac{\partial L(\mathrm{y}_i, f(x_i))}{\partial f(x_i)} |_{f=f_{t-1}}]$ and the child tree is to fit these targets.

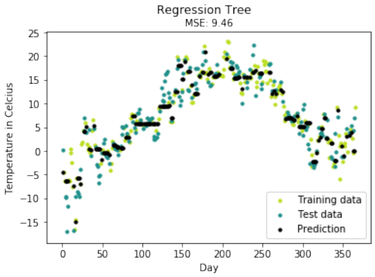
Note that the following relation always holds for most loss function:

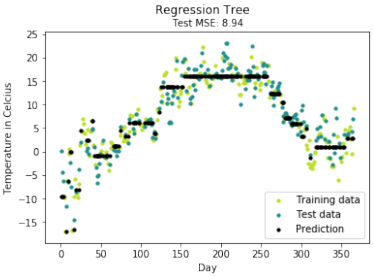
$$L(\mathrm{y}_i, f(x_i)) = 0 \iff \mathrm{y}_i = f(x_i) \iff [\frac{\partial L(\mathrm{y}_i, f(x_i))}{\partial f(x_i)} |_{f=f_{t-1}}] = 0$$

which means if $f(x_i) = \mathrm{y}_i$ the lebel will be learnt in next decision tree. In another words, the gradient is always to be 0 when the loss is 0. And the outputs is a constant in vanilla decision tree for different input in the same leaf so that the new targets $r_{i,t}$ and $r_{j,t}$ are equal if x_i, x_j are output in the same leaf and $y_i = y_j$. As a result, such pair x_i, x_j tends to be in the same terminal region in the next tree. The samples in the same terminal region with different targets tend to be seperated in the next tree. The gradients guide the decision tree to the better one. It is why we call it as gradient guide decision trees.

"If I have seen further, it is by standing on the shoulders of giants." The new targets $\{f_{t-1}(x_i) - \alpha_{i,t} g_{i,t} \mid i = 1, 2, \cdots, N, t = 1, 2, \cdots, T.\}$ are the shoulders of giants in decision tree. The the soft targets can be generated by any surrogate of loss function. We generalize gradient guide decision trees to surrogate guide decision trees.

5 Experiments





Algorithm 4 Surrogate Guide Decision Trees

```
Input training data set $\{(x_n, y_n) \mid x_n \in \mathrm{X} \subset \mathbb{R}^p, y_n \in \mathbb{R}, n = 1, 2, \cdots, N\}$
Initialize $f_0 = \arg\min_\gamma \sum_{i=1}^{N} L(x_i, \gamma)$
for $t = 1, 2, \ldots, T$ do
    for $i = 1, 2, \ldots, N$ do
        compute $a_{i,t} = \arg\min_x Q(f_t(x_i), f_t(x_i) + x)$.
        compute $r_{i,t} = f_t(x_i) + a_{i,t}$.
    end for
    Fit a regression tree to the targets $r_{i,t}$ giving terminal regions
        $$R_{j,m}, j = 1, 2, \ldots, J_m.$$
    for $j = 1, 2, \ldots, J_m$ do
        compute $\gamma_{j,t} = \arg\min_\gamma \sum_{x_i \in R_{j,m}} L(\mathrm{y}_i, \gamma)$.
    end for
    $f_t = \sum_{j=1}^{J_m} \gamma_{j,t} \mathbb{I}(x \in R_{j,m})$
end for
Output $f_T(x)$.
```

6 Conclusion

Our contribution concentrates on (1) the connection between boosting algorithms and continuous optimization method and (2) the explaintion of gradient guide decision trees in the framework of knowledge distillization. The numerical experiments show gradient guide decision trees(GGDT) are competitive with gradient boost decision trees. The generalization of GGDT such as ADMM guide decision trees is worthy of investigating. There are much further investigation on the connection of fixed point iteration optimizations and function approximation.