Final Project Presentation

Gradient Boosting Method

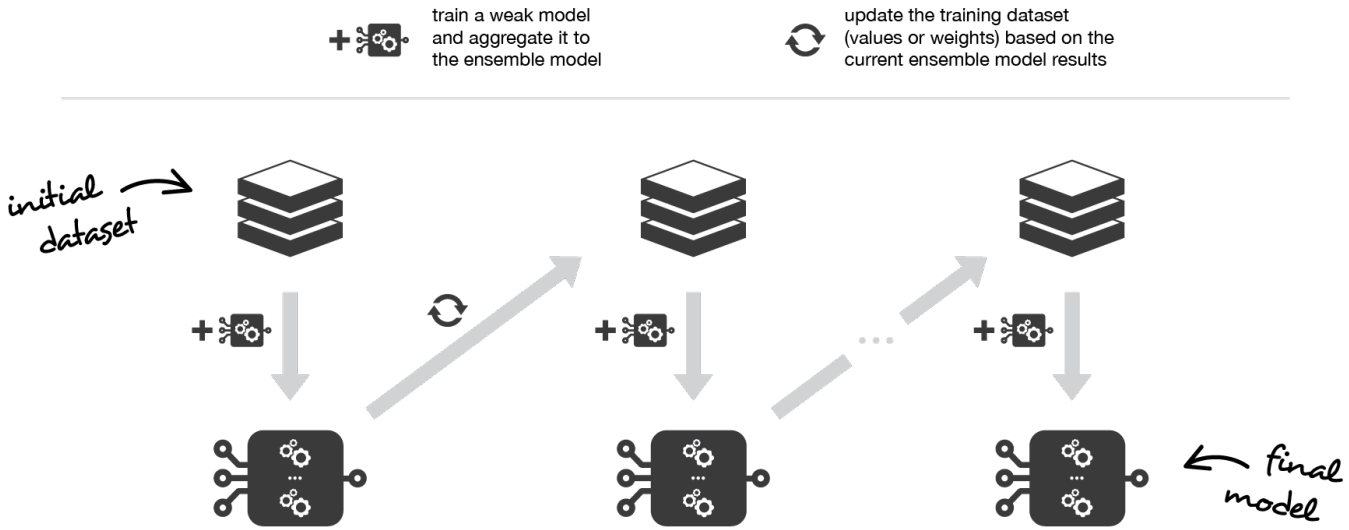
Overview

1. Introduction of Ensemble Methods
2. Gradient Boosting Introduction
3. Procedures of GB and GBDT algorithm
4. Dataset and Implementation
5. Result Comparison
6. Conclusion and Future works

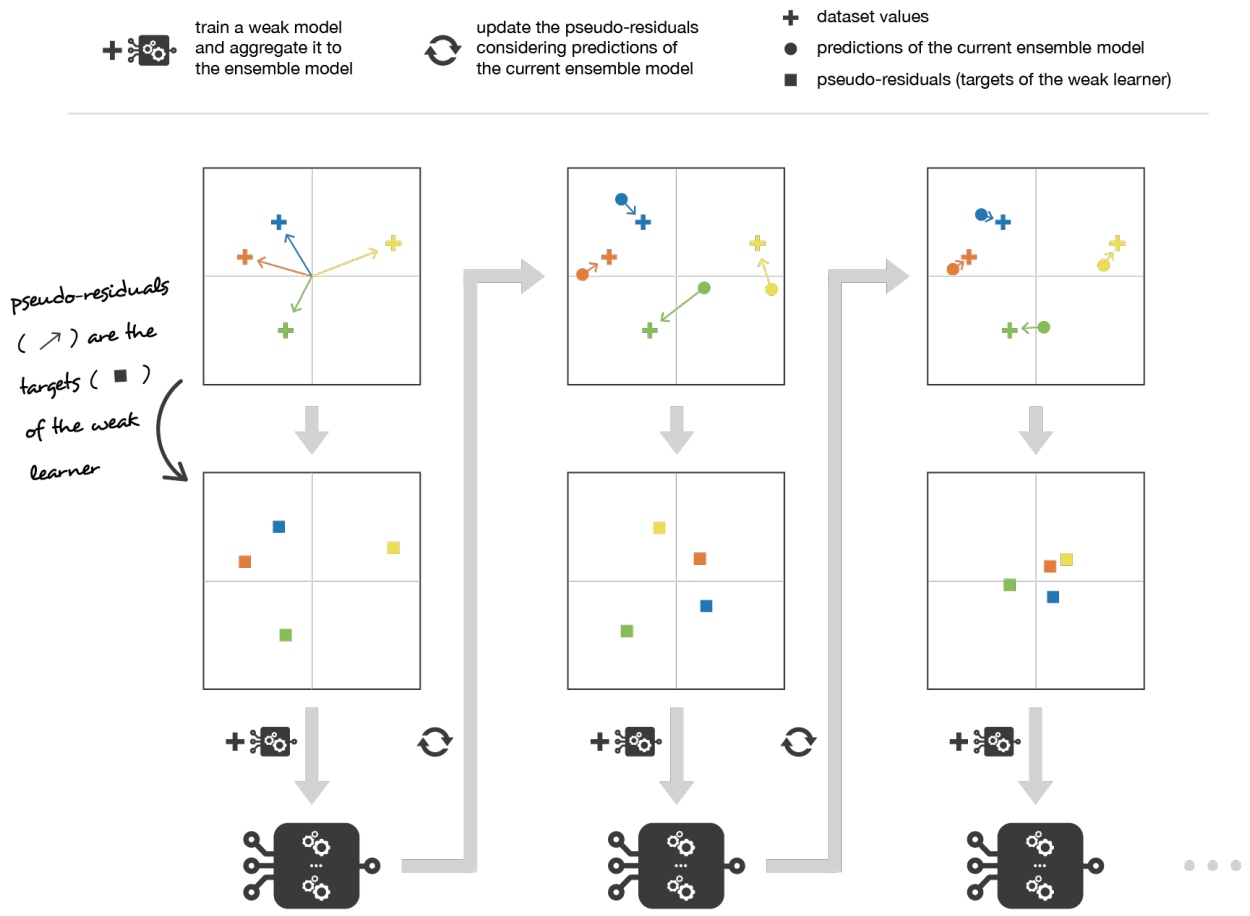
Introduction of Ensemble Methods

Basic idea：Ensemble learning is to use certain means to learn multiple classifiers, and then combine multiple classifiers for public prediction.

General method: The core idea is to combine many weak models to get a strong one and find how to train multiple weak classifiers and how to combine them. The combined models in ensemble methods can be models of the same class or models of different types.



Introduction of Gradient Boosting



In gradient boosting, each iteration generates **a weak learner**, which fits the **residuals** with respect to the previous cumulative model.

Then, **adding** the new fitted weak learner to the cumulative model to gradually reduce the loss of the cumulative model.

To formulate this:

Our approximation $\hat{F}(\mathrm{x})$ is an additive model that minimizes the loss function $L(y, F(x))$:

$$\hat{F}(x) = \sum_{i=1}^{M} \gamma_i h_i(x) + const$$

Procedure of Gradient Boosting

1. The main goal of the Gradient Boosting

$$\hat{F}(x) = argmin_F \, L(\mathrm{y}, F(\mathrm{x}))$$

2. The formula of $\hat{F}(\mathrm{x})$:

$$\hat{F}(x) = \sum_{i=1}^{M} \gamma_i h_i(x) + const$$

3. **Review the Gradient descent**

$$\omega_{i+1} = \omega_i - \lambda \frac{\partial L(\omega)}{\partial \omega}$$

In the Gradient boosting, we consider the $F(x)$ as the parameter, then we can update the our model as:

$$F_m(x) = F_{m-1}(x) - \gamma_m \frac{\partial L(y, F_{m-1}(x))}{\partial F_{m-1}(x)}$$

Procedure of Gradient Boosting

4 We use a base learner $h(x)$ to fit the **negative gradient (pseudo-residuals)** of each step, then we train the base model as follow:

$$h_m(x) \approx -\frac{\partial L(y,F_{m-1}(x))}{\partial F_{m-1}(x)}, F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

where the γ_m is the step length.

5 A proper step length γ_m should minimize the loss function $L(y,F(x))$

$$\gamma_m = argmin_\gamma \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$

6 Last formula is a one-dimension optimization problem, for complex loss function we usually can not get the exact result. To optimize it, we can use line search or simplify the loss fuction to get an approximate result.

Regularization of Gradient Boosting

1. **Shrinkage**

A simple regularization strategy that scale the contribution of each weak learner use a learning rate parameter $\boldsymbol{\nu}$ between $(0,1]$

$$F_m(x) = F_{m-1}(x) + \boldsymbol{\nu}\,\gamma_m h_m(x_i)$$

2. **Subsampling**

Combining gradient boosting with bootstrap averaging (bagging). The subsample is drawn without replacement.

With subsampling, we are then able to calculate out-of-bag error using the unused samples.

3. **Early stopping**

Score the model using the validation set, if the improvement of score is less than a threshold then the model is considered to be converged, thus stop training.

Gradient Boosting Decision Tree

1. Base learner： Decision Tree

   NOTE：For both GBDT classification and regression we use the regression tree.

2. Loss Function:

   - For Classification:
     - Logistic loss : $L(y, f(x)) = -\sum_{k=1}^{K} y_k \log p_k(x)$
   - For Regression:
     - Mean Square loss : $L(y, f(x)) = (y - f(x))^2$

3. Some noticeable points

   1. The GBDT update the step length γ_m in **each leaf region**.

$$F_m(x) = F_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} \mathbf{1}_{R_{jm}}(x), \quad \gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma).$$

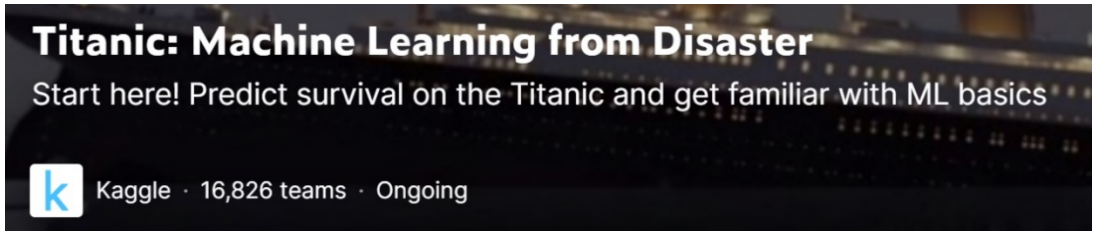
   2. The optimal of γ_m is simplify by **Newton's method** as

$$\gamma_{jkm} = \frac{K-1}{K} \frac{\sum_{x_i \in R_{jkm}} \tilde{y}_{ik}}{\sum_{x_i \in R_{jkm}} |\tilde{y}_{ik}|(1-|\tilde{y}_{ik}|)}$$

Dataset / Implementation

Our dataset

~800 instances

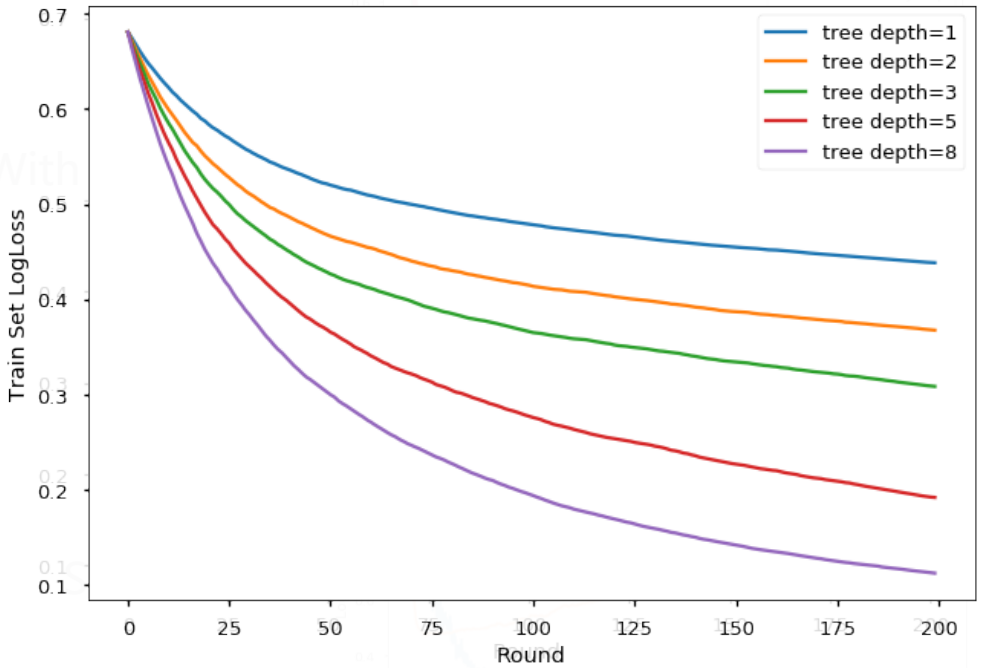


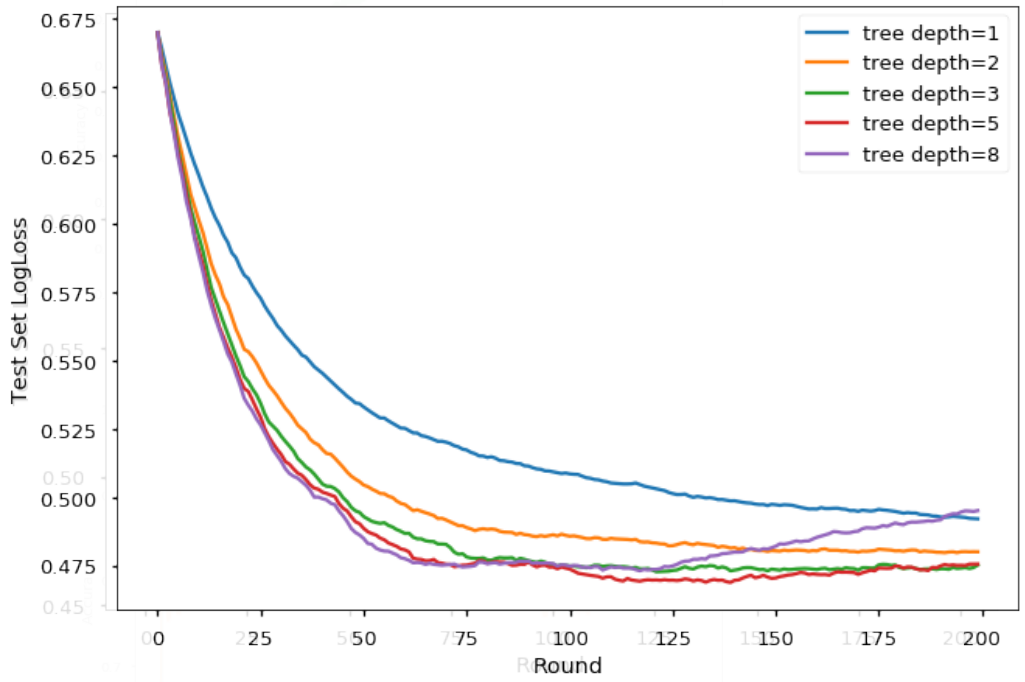
- Predictors include “Ticket class”, “Sex”, “Age”, “Fare”, etc.
- Prediction goal is **whether someone survives**.

Our two set of implementation

- First one is based on DecisionTreeClassifier from sklearn, which implements Gradient Boosting for both regression and classification, it also includes regularization methods such as shrinkage, subsampling, out-of-bag error, early stop…
- Second one is completely written from scratch, using custom decision trees. One major difference is that it calculates **a seperate γ_{jm} for each leaf**, which is the latter algorithm named “Gradient Boosting Tree”.

Result Comparasion





Conclusion

- We implemented the Gradient Boosting algorithm and the GBDT algorithm. The performance of these two algorithms are similar, and they both yield good results.

- We have learned:
  1. The theory behind Gradient Boosting and GBDT.
  2. Coding a custom estimator (with or without sklearn).

- Future works:
  1. GBDT only does second order Taylor expansion to the loss function, maybe we can try to do higher-order Taylor expansion to see whether the result can be improved.
  2. We can try to add regularization term in the loss function to avoid the base model become too complex.
  3. Use subsampling both to the data and the features, which might improve training speed and further reduce overfitting.

THANKS