Boosting methods

Emanuel Sommer

2021-04-07

Contents

1 Prerequisites **5**

2 Introduction **7**

3 Theory **9**

3.1 The powerful idea of gradient boosting 9

3.2 General gradient tree boosting 10

3.3 XGBoost a highly efficient implementation 10

4 Expore the data **11**

4.1 Train-test split 11

4.2 Visualize the data 11

4.3 Create recipe 12

5 Let's boost the models **13**

6 Conclusion **17**

7 References **19**

Chapter 1

Prerequisites

Here some Notation/ Prerequisites

List: - CART - General Machine learining set up (test train split/CV)

Chapter 2

Introduction

Introduction to the whole setting (seminar topic/ intro to topic maybe mention kaggle wins)

> "Boosting is one of the most powerful learning ideas introduced in the last twenty years." (Hastie et al., 2009)

Chapter 3

Theory

3.1 The powerful idea of gradient boosting

As roughly mentioned in the introduction section 2 the main idea of boosting is to sequentially build 'weak' learners that form a powerful ensemble model. It is not totally clear from which field these methods emerged but some claim that the work of Freund and Schapire with respect to PAC learning in the 1990s were instrumental for their growth. (Hastie et al., 2009) PAC learning can be considered one field within the field of the broader learning theory that tries to find generalization bounds for algorithms that are probably approximately correct (PAC). (Wolf, 2020)

3.1.1 Forward Stagewise Additive Modeling

In the setting of the dataset $\mathcal{D} = \{(y_i, x_i) \mid i \in [N]\}$ with $x_i \in \mathbb{R}^m$ and $y_i \in \mathbb{R}$ boosting is fitting the following additive, still quite general, model.

$$\hat{y_i} = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F} \tag{3.1}$$

Where $\mathcal{F}$ is the space of learning algorithms that will be narrowed down later on. Additive expansions like this are at the core of many other powerful machine learning algorithms like Neural Networks or Wavelets. (Hastie et al., 2009)

This leads to the algorithm of Forward Stagewise Additive Modeling. (Hastie et al., 2009) tbdtbdtbd

3.1.2 Robust loss functions for regression

3.1.3 Off shelf performance

Although arguably one of the most famous algorithms in boosting is the Adaboost algorithm for classification, here the focus will be on a regression task and thus Adaboost will not be covered.

3.2 General gradient tree boosting

first some stuff from elements (more general)

3.2.1 numerical optimization

3.2.2 gradient tree boosting algorithm from elements

3.2.3 problem of right sized trees

3.2.4 regularisation

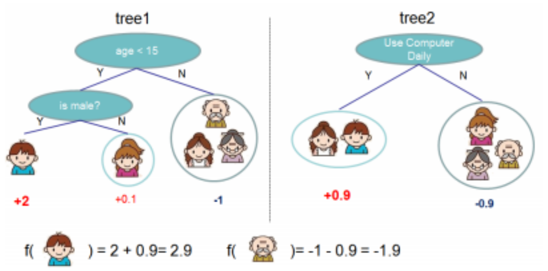

Figure 3.1: Example of an additive tree ensamble(Chen and Guestrin, 2016)

3.3 XGBoost a highly efficient implementation

use the whole paper

We will cite the great paper!(Chen and Guestrin, 2016)

Chapter 4

Expore the data

We use the following packages.(Wickham et al., 2019; Kuhn and Wickham, 2020)

```
library(tidyverse)
library(tidymodels)
```

4.1 Train-test split

```
set.seed(2)
# ames_split <- initial_split(ames, prop = 0.80, strata = Sale_Price)
# ames_train <- training(ames_split)
# ames_test  <-  testing(ames_split)
```

4.2 Visualize the data

- pair plots
  - indiviual main effects
  - correlations
- correlation plot
- individual distributions
- identify categorical features
- feature engineering

4.3 Create recipe

one for lm (dummify categorical features) and one for the tree based approaches

```r
# ames_rec <-
#   recipe(Sale_Price ~ Neighborhood + Gr_Liv_Area + Year_Built + Bldg_Type +
#            Latitude + Longitude, data = ames_train) %>%
#   step_log(Gr_Liv_Area, base = 10) %>%
#   step_other(Neighborhood, threshold = 0.01) %>%
#   step_dummy(all_nominal()) %>%
#   step_interact( ~ Gr_Liv_Area:starts_with("Bldg_Type_") ) %>%
#   step_ns(Latitude, Longitude, deg_free = 20)
```

Chapter 5

Let's boost the models

boooooooooooooooooooooooost them (gif here?)

Cite additional model packages.(Chen et al., 2021, Wright and Ziegler (2017))

```
library(xgboost)
library(ranger)
```

To do

1. register parallel backend
2. create models and workflows
3. create resampling objects
4. choose tuning method (racing/iterative or grid search)
5. tune models and select best ones
6. compare final models to test set
7. save final model

cite (Corporation and Weston, 2019)

```
library(doParallel)

# Create a cluster object and then register:
cl <- makePSOCKcluster(2)
registerDoParallel(cl)

# Put at the end:
stopCluster(cl)
```

for 2. set engine specific arguments in the set_engine() function use tune() for parameters that should be tuned (optional tune("id_name"))

```
# Models:
lm_model <- linear_reg() %>%
  set_engine("lm")

rf_model <- rand_forest(trees = 1000) %>%
  set_engine("ranger") %>%
  set_mode("regression")

boost_model <- boost_tree() %>%
  set_engine("xgboost") %>%
  set_mode("regression")
```

```
# Workflows:

# lm_wflow <-
#   workflow() %>%
#   add_model(lm_model) %>%
#   add_recipe()
```

for 3.

```
# Create Resampling objects
# set.seed(2)
# ames_folds <- vfold_cv(ames_train, v = 10)
```

for fitting resampling objects without tuning:

```
# set.seed(2)
# rf_res <- rf_wflow %>%
#   fit_resamples(resamples = ames_folds)
# # collect the metrics during resampling with
# collect_metrics()
```

for 4. and following

```
# show the parameters to be tuned with the range
# dials::parameters()
# get and modify the parameters to be tuned with:
# name()
# wflow_param %>% pull_dials_object("threshold")
# parameters(ames_rec) %>%
#  update(threshold = threshold(c(0.8, 1.0)))

# finalize for data dependent params
# updated_param <-
#   workflow() %>%
#   add_model(rf_spec) %>%
#   add_recipe(pca_rec) %>%
#   parameters() %>%
#   finalize(ames_train)
```

grids: grid_regular(levels = c(hidden_units = 3, penalty = 2, epochs = 2)) space filling: grid_latin_hypercube(size = 15, original = FALSE) then tune_grid() function instead of fit_resamples

```
roc_res <- metric_set(roc_auc)
set.seed(99)
mlp_reg_tune <-
  mlp_wflow %>%
  tune_grid(
    cell_folds,
    grid = mlp_param %>% grid_regular(levels = 3),
    metrics = roc_res
  )

autoplot(mlp_reg_tune) + theme(legend.position = "top")

show_best(mlp_reg_tune)

# for spacefilling
mlp_sfd_tune <-
  mlp_wflow %>%
  tune_grid(
    cell_folds,
    grid = 20,
    # Pass in the parameter object to use the appropriate range:
    param_info = mlp_param,
    metrics = roc_res
  )

select_best()
```

```
logistic_param <-
  tibble(
    num_comp = 0,
    epochs = 125,
    hidden_units = 1,
    penalty = 1
  )

final_mlp_wflow <-
  mlp_wflow %>%
  finalize_workflow(logistic_param)

final_mlp_fit <-
  final_mlp_wflow %>%
  fit(cells)
```

Racing:

```
library(finetune) # if used to be cited

set.seed(99)
mlp_sfd_race <-
  mlp_wflow %>%
  tune_race_anova(
    cell_folds,
    grid = 20,
    param_info = mlp_param,
    metrics = roc_res,
    control = control_race(verbose_elim = TRUE)
  )
```

Here no iterative search as big parameter space, could be done after grid search or for single parameters. (If then Simulated Annealing)

Chapter 6

Conclusion

Let's wrap it up!

Chapter 7

References

Bibliography

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., Li, M., Xie, J., Lin, M., Geng, Y., and Li, Y. (2021). *xgboost: Extreme Gradient Boosting*. R package version 1.3.2.1.

Corporation, M. and Weston, S. (2019). *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*. R package version 1.0.15.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning (12th printing)*. Springer New York.

Kuhn, M. and Wickham, H. (2020). *Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles.*

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K., and Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686.

Wolf, M. M. (2020). Lecture notes in mathematical foundations of supervised learning.

Wright, M. N. and Ziegler, A. (2017). ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17.