# Compboost

## Modular framework for component-wise boosting

Daniel Schalk

June 22, 2018

LMU Munich
Working Group Computational Statistics

# Table of contents

1. What is Component-Wise Boosting
2. About Compboost
3. Small Usecase
4. Next Steps

# What is Component-Wise Boosting

# Component-Wise Boosting: Terminology

- Loss Function:

$$L : \mathcal{Y} \times \mathcal{X} \to \mathbb{R}$$

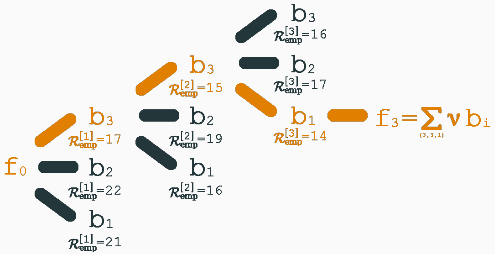
- Empirical Risk:

$$\mathcal{R}_{\text{emp}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} L\left(y^{(i)}, f(x^{(i)})\right)$$

- Estimated model/parameter at iteration $m$:

$$\hat{f}^{[m]}, \theta^{[m]}$$

# Component-Wise Boosting: The Idea




Iteration 1: $\hat{f}^{[1]}(x) = \beta b_3(x_3, \theta^{[1]})$

Iteration 2: $\hat{f}^{[2]}(x) = \beta b_3(x_3, \theta^{[1]}) + \beta b_3(x_3, \theta^{[2]})$

Iteration 2: $\hat{f}^{[3]}(x) = \beta b_3(x_3, \theta^{[1]}) + \beta b_3(x_3, \theta^{[2]}) + \beta b_1(x_1, \theta^{[3]})$

$$\Rightarrow \hat{f}^{[3]}(x) = \beta \left( b_3(x_3, \theta^{[1]} + \theta^{[2]}) + b_1(x_1, \theta^{[3]}) \right)$$

# Component-Wise Boosting: The Algorithm

**Result:** Component-wise boosting model $\hat{f}(x)$

Initialize $\hat{f}^{[0]}(x) = \arg\min_{c \in \mathbb{R}} \mathcal{R}_{\text{emp}}(c)$ ;

**for** $m \in \{1, \dots, M\}$ **do**

// Update pseudo residuals:

$$r^{[m](i)} = -\left[\frac{\delta}{\delta f(x^{(i)})} L\left(y^{(i)}, f(x^{(i)})\right)\right]_{f = f^{[m-1]}}, \ \forall i \in \{1, \dots, n\} ;$$

// Get index $j^*$ of $m$-th base-learner from optimizer:

**for** $j \in \{1, \dots, J\}$ **do**

// Fit each base-learner $b_j^{[m]}$ to the pseudo residuals:

$$\hat{\theta}_j^{[m]} = \arg\min_{\theta_j} \sum_{i=1}^{n} \left(r^{[m](i)} - b_j^{[m]}(x^{(i)}, \theta_j)\right)^2 ;$$

// Calculate the SSE of the fitted base-learner:

$$\text{SSE}_j = \sum_{i=1}^{n} \left(r^{[m](i)} - b_j^{[m]}(x^{(i)}, \hat{\theta}_j)\right)^2 ;$$

**end**

// Add selected component to model:

$$\hat{f}^{[m]}(x) = \hat{f}^{[m-1]}(x) + \beta b_{j^*}^{[m]}\left(x, \theta_{j^*}^{[m]}\right)$$

**end**

**Returns:** $\hat{f}(x) = \hat{f}^{[m]}(x)$;

# Available R Packages

- Tree-based implementations:
  - `xgboost`
  - `catboost`
  - `gbm`
- Model-based implementations:
  - `mboost`

So, why another boosting implementation?

# About Compboost

# About Compboost

- Installation

```r
devtools::install_github("schalkdaniel/compboost")
library(compboost)
```

# Compboost Members and Member Functions

- **Member Functions**:
  - `addBaselearner()`
  - `addLogger()`
  - `train()`
  - `coef()`
  - `predict()`
  - `risk()`
  - `selected()`
  - `plot()`
  - `...`
- **Public Members**:
  - `model`
  - `bl.factory.list`
  - `loss`
  - `optimizer`
  - `...`

# Small Usecase

# Initializng Model

```
mtcars$mpg_cat = ifelse(mtcars$mpg > 15, "A", "B")
cboost = Compboost$new(mtcars, "mpg", loss = QuadraticLoss$new())

cboost$addBaselearner("wt", "spline", PSplineBlearnerFactory,
       degree = 3, knots = 10, penalty = 2, differences = 2)
cboost$addBaselearner("mpg_cat", "linear", PolynomialBlearnerFactory,
       degree = 1, intercept = FALSE)

cboost$train(2000, trace=FALSE)
cboost
```

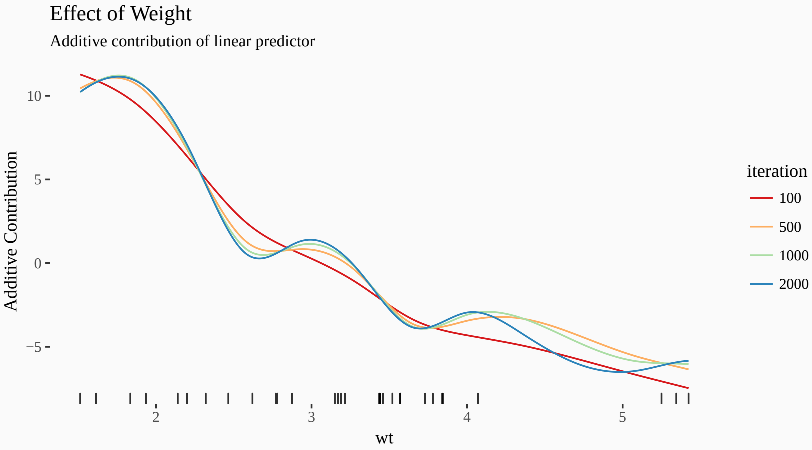
```
## Componentwise Gradient Boosting
##
## Trained on mtcars with target mpg
## Number of base-learners: 3
## Learning rate: 0.05
## Iterations: 2000
## Offset:20.090625
##
## QuadraticLoss Loss:
##
##   Loss function: y = (y - f(x))^2
##
##
```

# Plot Results

```r
library(ggplot2)
library(ggthemes)

cboost$plot("wt_spline", iters = c(100, 500, 1000, 2000)) +
        labs(title = "Effect of Weight",
                subtitle = "Additive contribution of linear predictor") +
        theme_tufte() +
        scale_color_brewer(palette = "Spectral")
```

# Plot Results



# Next Steps

Questions?