Non-life insurance pricing with gradient tree-boosted mixture models.

Guangyuan Gao   Jiahong Li

February 18, 2022

Abstract

Insurance loss data often cannot be well modeled by a single distribution. Mixture of models are often applied in insurance loss modeling. The Expectation-Maximization (EM) algorithm is used for parameter estimation in mixture of models. Feature engineering and variable selection are challenging for mixture of models due to several component models involving. Overfitting is also a concern when predicting future loss. To address those issues, we propose an Expectation-Boosting (EB) algorithm, which replaces the maximization step in the EM algorithm by a gradient boosting with decision tree. The boosting is overfitting-sensitive. Moreover, it performs automated feature engineering, model fitting and variable selection simultaneously. The EB algorithm fully explores the predictive power of covariate space. We illustrate those advantages in two simulated data and a real insurance loss data.

1 Introduction

Insurance loss data usually cannot be well modelled by a single distribution. For claims counts data, there may be an excess of zero claims, so a Poisson distribution is not the best option. For claims amount data, there may be a heavy tail, so a gamma distribution is not enough to describe the entire data. One solution to such issues is to apply mixture models. For claims counts data, we can utilize zero-modified Poisson distribution. For claims amount data, we can utilize mixture of gamma distribution and heavy-tailed distribution such as Pareto distribution. When the individual risk factors are available, the mixture of distributions can be extended to mixture of regressions to address the risk heterogeneity in the portfolio.

The Expectation-Maximization (EM) algorithm is an iterative method to estimate component parameters and (hidden) component indicator variable. Parameter estimation in mixture models is challenging since component distribution/regression parameters are related to each other. Variable selection is also challenging since we need to perform variable selection for each component regression.

literature review.

In this paper, we propose an Expectation-Boosting (EB) algorithm, which replaces the maximization step by an overfitting-sensitive boosting step. There are several advantages of EB algorithm over the EM algorithm. First, boosting algorithm is a flexible non-parametric regression facilitating both non-linear effects and interaction. Second, boosting algorithm is overfitting-sensitive, if we add suitable normalization to boosting step (e.g. using decision tree as weak learner) we can perform variable selection simultaneously. Third, by investigating the final

boosted component, we can select component by removing those components containing very few weak learners.
Commonly used boosting algorithms are listed below

1. Binary classification: AdaBoost, LogitBoost (real, discrete, gentle AdaBoost), AdaBoost.M1
2. Multinomial classification: Stagewise Additive Modeling using a Multi-class Exponential loss (SAMME), SAMME.R (multi-class real AdaBoost),
3. Gradient based: gradient boosting machine/model (GBM), Newton boosting, gradient boosting decision tree (GBDT), eXtreme Gradient Boosting (XGBoost), light gradient boosting machine (LightGBM)

The last type is based on gradient of loss function, and our EB algorithm uses this version of boosting.

2 Review: mixture of models and the EM algorithm

2.1 Mixture of models

In this paper, we focus on the exponential distribution family with expectation parameter μ and dispersion parameter ϕ. The exponential distribution family is normally sufficient to fit to most insurance loss data. Suppose a random variable y follows the k-th *component distribution* from

$$\{f_1(y;\mu_1,\phi_1),\dots,f_K(y;\mu_K,\phi_K)\}$$

with *mixing probability* $p_k, k=1,\dots,K$, then the probability density function for y is

$$f(y)=\sum_{k=1}^{K}p_kf_k(y;\mu_k,\phi_k).$$

If *individual features* $\boldsymbol{x}_i$ are available and they have systematic effects on the distribution of y_i, then we establish a mixture of regressions:

$$f(y|\boldsymbol{x})=\sum_{k=1}^{K}p_k(\boldsymbol{x})f_k(y;\mu_k(\boldsymbol{x}),\phi_k(\boldsymbol{x})).$$

Above is the most general form of mixing. In applications, we always put some constraints on the *mixing structure*:

- Mixing probabilities are not related to $\boldsymbol{x}$

$$f(y)=\sum_{k=1}^{K}p_kf_k(y;\mu_k(\boldsymbol{x}),\phi_k(\boldsymbol{x})).$$

- Both mixing probabilities and dispersions are not related to $\boldsymbol{x}$

$$f(y)=\sum_{k=1}^{K}p_kf_k(y;\mu_k(\boldsymbol{x}),\phi_k),$$

- If component distributions are from the same distribution family, we might assume different component distributions have the same dispersion ϕ

$$f(y) = \sum_{k=1}^{K} p_k f_k(y; \mu_k(\boldsymbol{x}), \phi).$$

- Covariates $\boldsymbol{x}$ are only related to the mixing probabilities:

$$f(y) = \sum_{k=1}^{K} p_k(\boldsymbol{x}) f_k(y; \mu_k, \phi_k).$$

We need to determine the mixture structure according to the data and our aim. Imposing suitable constraints on mixture, we can accelerate model fitting without compromising predictive performance.

2.2 EM algorithm

Suppose that we know which component distribution f_k each sample y is from. That is we know the *full information* $(Y, \boldsymbol{Z}, \boldsymbol{x})$ where

$$\boldsymbol{Z} = (Z_1, \ldots, Z_K)^\top = (\mathbb{1}_1(k), \ldots, \mathbb{1}_K(k))^\top$$

is the one-hot encoding of *component indicator variable.* The joint distribution function for full information (one sample) is given by

$$f(y, \boldsymbol{z}|\boldsymbol{x}) = \prod_{k=1}^{K} \left[p_k(\boldsymbol{x}) f_k(y; \mu_k(\boldsymbol{x}), \phi_k(\boldsymbol{x}))\right]^{z_k}$$

The log-likelihood function is given by

$$l(p, \mu, \phi|y, \boldsymbol{z}, \boldsymbol{x}) = \sum_{k=1}^{K} z_k \left[\log p_k(\boldsymbol{x}) + \log f_k(y; \mu_k(\boldsymbol{x}), \phi_k(\boldsymbol{x}))\right] \tag{2.1}$$

Given a data set $(y_i, \boldsymbol{z}_i, \boldsymbol{x}_i)_{i=1:n}$, the regression coefficients $\theta_p, \theta_\mu, \theta_\phi$ in p, μ, ϕ can be estimated by the following $K+1$ independent optimizations

$$\hat{\theta}_p = \underset{\theta_p}{\arg\max} \sum_{i=1}^{n} \sum_{k=1}^{K} z_{i,k} \log p_k(\boldsymbol{x}_i; \theta_p) \tag{2.2}$$

$$\left(\hat{\theta}_\mu^{(k)}, \hat{\theta}_\phi^{(k)}\right) = \underset{\theta_\mu^{(k)}, \theta_\phi^{(k)}}{\arg\max} \sum_{i=1}^{n} \sum_{k=1}^{K} z_{i,k} \log f_k\left(y_i; \mu_k\left(\boldsymbol{x}_i; \theta_\mu^{(k)}\right), \phi_k\left(\boldsymbol{x}_i; \theta_\phi^{(k)}\right)\right), \text{ for } k = 1, \ldots, K. \tag{2.3}$$

Those optimizations are corresponding to *a multinomial logistic classification* and K *regressions.* Note that the multinomial logistic classification (2.2) are fitted to all samples, while K regressions (2.3) are fitted to partial samples with $\{i : z_{i,k} = 1\}$. In practice we do not have full information $(Y, \boldsymbol{Z}, \boldsymbol{x})$ but only the incomplete information $(Y, \boldsymbol{x})$. The following EM algorithm is inspired by the above discussion.

Expectation step. With iterated $\hat{p}, \hat{\mu}, \hat{\phi}$, calculate the *conditional expectation* of $\boldsymbol{z}$:

$$\hat{z}_{i,k} = \hat{z}_k(\boldsymbol{x}_i) = \frac{\hat{p}_{i,k} f_k(y_i; \hat{\mu}_{i,k}, \hat{\phi}_{i,k})}{\sum_{l=1}^K \hat{p}_{i,l} f_l(y_i; \hat{\mu}_{i,l}, \hat{\phi}_{i,l})},$$

where $\hat{p}_{i,k} = \hat{p}_k(\boldsymbol{x}_i) = p_k(\boldsymbol{x}_i; \hat{\theta}_p), \hat{\mu}_{i,k} = \hat{\mu}_k(\boldsymbol{x}_i) = \mu_k\left(\boldsymbol{x}_i; \hat{\theta}_\mu^{(k)}\right), \hat{\phi}_{i,k} = \hat{\phi}_k(\boldsymbol{x}_i) = \phi_k\left(\boldsymbol{x}_i; \hat{\theta}_\phi^{(k)}\right)$.

Maximization step. Based on the following likelihood function for full information, calculate the MLE of regression coefficients $\theta_p, \theta_\mu, \theta_\phi$.

$$\begin{aligned}
& l(p, \mu, \phi | y, \hat{z}, \boldsymbol{x}) \\
=& \sum_{i=1}^n \sum_{k=1}^K \hat{z}_{i,k} \left[\log p_k(\boldsymbol{x}_i; \theta_p) + \log f_k\left(y_i; \mu_k\left(\boldsymbol{x}_i; \theta_\mu^{(k)}\right), \phi_k\left(\boldsymbol{x}_i; \theta_\phi^{(k)}\right)\right)\right] \\
=& \sum_{i=1}^n \sum_{k=1}^K \hat{z}_{i,k} \log p_k(\boldsymbol{x}_i; \theta_p) + \sum_{i=1}^n \sum_{k=1}^K \hat{z}_{i,k} \log f_k\left(y_i; \mu_k\left(\boldsymbol{x}_i; \theta_\mu^{(k)}\right), \phi_k\left(\boldsymbol{x}_i; \theta_\phi^{(k)}\right)\right)
\end{aligned} \tag{2.4}$$

This step is similar to (2.2) and (2.3). However, now we have $\hat{z}_{i,k} \in (0, 1)$, so we need to consider a multinomial logistic classification with fractional response and K weighted-regressions fitted to all samples.

3 Expectation-Boosting algorithm

We replace the maximization step in the EM algorithm by a boosting step. The boosting step increases the likelihood in each iteration, but overfitting-sensitively. The boosting step follows a generic functional gradient descent algorithm. We first review generic functional gradient descent algorithm, then describe our proposed EB algorithm.

3.1 Generic functional gradient descent algorithm

Suppose a non-parametric regression function as $F : \mathbb{R}^P \to \mathbb{R}$. Our purpose is to estimate F to minimize the expected loss

$$\hat{F} = \arg\min_F \mathbb{E}\left[C(Y, F(\boldsymbol{x}))\right],$$

where $C : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ is the *loss function.* We always replace the expected loss by sample average loss:

$$\hat{F} = \arg\min_F \frac{1}{n} \sum_{i=1}^n C(y_i, F(\boldsymbol{x}_i))$$

We choose the *negative log-likelihood* function as the loss function. Hence, minimizing the loss function is equivalent to maximizing the likelihood function. The link function is used to link the regression function F with the parameter interested such as mean, probability odds, etc. Commonly used link function and loss functions in different boosting algorithms are listed below:

- AdaBoost:

$$C(y, F) = \exp(yF), \quad y \in \{-1, 1\}$$

$$F(\boldsymbol{x}) = \frac{1}{2} \log\left(\frac{\Pr[Y = 1|\boldsymbol{x}]}{\Pr[Y = -1|\boldsymbol{x}]}\right)$$

- LogitBoost:

$$C(y,F) = \log_2(1+\exp(-2yF)), \quad y \in \{-1,1\}$$
$$F(\boldsymbol{x}) = \frac{1}{2}\log\left(\frac{\Pr[Y=1|\boldsymbol{x}]}{\Pr[Y=-1|\boldsymbol{x}]}\right)$$

- L_2Boost:

$$C(y,F) = (y-F)^2/2, \quad y \in \mathbb{R} \tag{3.1}$$
$$F(\boldsymbol{x}) = \mathbb{E}(Y|\boldsymbol{x})$$

We follow Bühlmann and Yu (2003) to review the generic functional gradient decent algorithm:

1. Initialization: $\hat{F}^{[0]}(\boldsymbol{x};\hat{\theta}^{[0]})$, where $\hat{\theta}^{[0]}$ are determined by $(y_i,\boldsymbol{x}_i)_{i=1:n}$. Let $m=0$.
2. Projection of gradient to weak learner: Calculate *negative gradient*

   $$u_i = -\frac{\partial C(y_i,F)}{\partial F}\bigg|_{F=\hat{F}^{[m]}(\boldsymbol{x}_i)}, i=1,\dots,n.$$

   Data $(u_i,\boldsymbol{x}_i)_{i=1:n}$ is used to calibrate a weak learner $\hat{f}^{[m+1]}(\boldsymbol{x};\hat{\theta}^{[m+1]})$ with *loss function* L_2 (3.1).
3. One-dimensional optimization: Solve an one-dimensional optimization to find *expansion coefficient* $\hat{\omega}^{[m+1]}$:

   $$\hat{\omega}^{[m+1]} = \arg\min_{\omega}\sum_{i=1}^{n} C(y_i,\hat{F}^{[m]}(\boldsymbol{x}_i)+\omega\hat{f}^{[m+1]}(\boldsymbol{x}_i))$$

   Update

   $$\hat{F}^{[m+1]} = \hat{F}^{[m]} + s\hat{\omega}^{[m+1]}\hat{f}^{[m+1]},$$

   where s is *shrinkage factor* (learning rate).
4. Iteration: Let m increase by 1, and repeat steps 2-3.

In the algorithm, weak learners are fitted to negative gradient U rather than Y. Loss function in weak learners is always L_2, independently with model loss function C. If weak learners are trees, the algorithm is called *gradient boosting decision tree (GBDT)*, and calibration and variable selection are performed simultaneously. Bühlmann and Hothorn (2007) argue that step 3 of one-dimensional optimization seems unnecessary given learning rate s is sufficiently small according to some empirical experiments.

3.2 Expectation-Boosting algorithm

We denote the regression functions for the interested parameters p,μ,ϕ by F,G,H. In the EB algorithm, we need to specify link functions and cost/loss functions, which determines negative gradient function. For mixing probabilities p, we choose multiple logistic link function for mixing probabilities.

$$p_k(F(\boldsymbol{x})) = \Pr(Z_k=1|\boldsymbol{x}) = \frac{\exp F_k(\boldsymbol{x})}{\sum_{l=1}^{K}\exp F_l(\boldsymbol{x})}, \tag{3.2}$$

or equivalently

$$F_k(\boldsymbol{x}) = \log p_k(\boldsymbol{x}) - \frac{1}{K}\sum_{l=1}^{K} \log p_l(\boldsymbol{x}), \quad k = 1, \dots, K. \tag{3.3}$$

We use the negative log-likelihood function as the cost function

$$C_Z(\boldsymbol{Z}, F(\boldsymbol{x})) = -\sum_{k=1}^{K} Z_k \log p_k(\boldsymbol{x}). \tag{3.4}$$

The negative gradient of the cost function w.r.t. F_k is given by

$$U_k(\boldsymbol{Z}, F(\boldsymbol{x})) = -\frac{\partial C_Z(\boldsymbol{Z}, F(\boldsymbol{x}))}{\partial F_k(\boldsymbol{x})} = Z_k - p_k(\boldsymbol{x}), \quad k = 1, \dots, K \tag{3.5}$$

For component parameters μ and ϕ, the link function and cost function (distribution) depend on the loss data. Commonly used link functions in component models are indicated as follows

- Component Gaussian model:
$$\mu_k(G_k(\boldsymbol{x})) = G_k(\boldsymbol{x})$$
$$\phi_k(H_k(\boldsymbol{x})) = \exp H_k(\boldsymbol{x})$$
- Component Poisson model:
$$\mu_k(G_k(\boldsymbol{x})) = \exp G_k(\boldsymbol{x})$$
- Component gamma model:
$$\mu_k(G_k(\boldsymbol{x})) = \exp G_k(\boldsymbol{x})$$
$$\phi_k(H_k(\boldsymbol{x})) = \exp H_k(\boldsymbol{x})$$

Note that we slightly abuse function notations p_k, μ_k, ϕ_k (i.e. the inverse link function); in Section (2.1) they define different functions. The negative log-likelihood function of each component model is used as its cost function:

$$C_k(Y, \boldsymbol{Z}, G_k(\boldsymbol{x})) = -\, Z_k \log f_k(Y; \mu_k(G_k(\boldsymbol{x})), \phi_k), \;\; k = 1 : K. \tag{3.6}$$

The negative gradient of C_k w.r.t. G_k is given by

$$V_k(Y, \boldsymbol{Z}, G_k(\boldsymbol{x})) = -\frac{\partial C_k(Y, \boldsymbol{Z}, G_k(\boldsymbol{x}))}{\partial G_k(\boldsymbol{x})}.$$

Note that we have assumed that dispersion ϕ_k is fixed among samples (not related to $\boldsymbol{x}$) to avoid lengthy notations. The extension to dispersion modelling is straightforward; please refer to the real data analysis in Section ??.

Upon the above defined notations, we are ready to introduce the proposed EB algorithm. We first summarize the proposed EB algorithm as follows, then we detail each step.

1 Initialization of EB algorithm $\hat{p}^{[0]}, \hat{\mu}^{[0]}, \hat{\phi}^{[0]}$. Set $t = 0$.

2 Calculating conditional expectation of latent variable $\hat{z}^{[t]}$ given $\hat{p}^{[t]}, \hat{\mu}^{[t]}, \hat{\phi}^{[t]}$.

3.1 Gradient boosting mixing probabilities $\hat{p}^{[t+1]}$ given latent variable $\hat{z}^{[t]}$.

3.2 Gradient boosting component model parameters $\hat{\mu}^{[t+1]}$ given latent variable $\hat{z}^{[t]}$.

4 Calculate the MLE $\hat{\phi}^{[t+1]}$. Increase t by 1. Repeat steps 2-3 until t reaches to T.

Note that we can perform steps 3.1 and 3.2 simultaneously by parallel computing which dramatically save running time. Also note that in the algorithm we have assumed that dispersion ϕ_k is fixed among samples (not related to $\boldsymbol{x}$). The extension to dispersion modelling is straightforward by adding another gradient boosting step 3.3 for component model parameter ϕ and removing the MLE calculation in step 4.
In step 1 of initialization of EB algorithm, we neeed to initialize the following quantities.

1.1 Initialize $\hat{p}_1^{[0]},\dots,\hat{p}_K^{[0]}$ and $\hat{F}_1^{[0]},\dots,\hat{F}_K^{[0]}$:

- $\hat{p}_k^{[0]}=\frac{1}{K}$
- $\hat{F}_1^{[0]},\dots,\hat{F}_K^{[0]}$ are obtained by (3.3).

1.2 Initialize $\hat{\mu}_1^{[0]},\dots,\hat{\mu}_K^{[0]}$ and $\hat{G}_1^{[0]},\dots,\hat{G}_K^{[0]}$:

- $\hat{\mu}_k^{[0]}=\frac{\sum_{i=1}^n Y_i}{n}$
- $\hat{G}_k^{[0]}=\mu_k^{-1}(\hat{\mu}_k^{[0]})$

1.3 Initialize $\hat{\phi}_1^{[0]},\dots,\hat{\phi}_K^{[0]}$ as the sample variance. (Assume that dispersion is independent with covariates.)

1.4 Set $t=0$.

In step 2 of conditional expectation of latent variable, we set $\hat{z}_{i,k}^{[t]}$ as

$$\hat{z}_{i,k}^{[t]}=\frac{\hat{p}_k^{[t]}(\boldsymbol{x}_i)f_k\left(y_i;\hat{\mu}_k^{[t]}(\boldsymbol{x}_i),\phi_k^{[t]}\right)}{\sum_{l=1}^K\hat{p}_l^{[t]}(\boldsymbol{x}_i)f_l\left(y_i;\hat{\mu}_l^{[t]}(\boldsymbol{x}_i),\phi_l^{[t]}\right)},\quad k=1:K.$$

In step 3.1 of gradient boosting mixing probabilities, we implement a multinomial logistic classification boosting with fractional response:

3.1.1 Initialization. Set $\hat{p}_1^{[t,0]},\dots,\hat{p}_K^{[t,0]}$ and $\hat{F}_1^{[t,0]},\dots,\hat{F}_K^{[t,0]}$ as

$$\hat{p}_k^{[t,0]}=\frac{1}{K},\quad \hat{F}_k(\boldsymbol{x})^{[t,0]}=\log\hat{p}_k^{[t,0]}-\frac{1}{K}\sum_{l=1}^K\log\hat{p}_l^{[t,0]} \tag{3.7}$$

Set $m=0$.

3.1.2 Projection of gradient to learner. Compute the negative gradient sample $u_{1,k}^{[t,m]},\dots,u_{n,k}^{[t,m]}$, in which

$$u_{i,k}^{[t,m]}=U_k(\hat{\boldsymbol{z}}_i^{[t]},\hat{F}^{[t,m]}(\boldsymbol{x}_i))=\hat{z}_{i,k}^{[t]}-\hat{p}_{i,k}^{[t,m]}.$$

For each k, the data $(u_{i,k}^{[t,m]},\boldsymbol{x}_i)_{i=1:n}$ is used to calibrate a L-terminal node regression trees $\hat{f}_k^{[t,m+1]}\left(\boldsymbol{x};R_{l=1:L}^{[t,m+1]},\bar{u}_{l=1:L}^{[t,m+1]}\right)$ with L_2 loss, where $R_{l=1:L}^{[t,m+1]}$ is the partition of covariate space and $\bar{u}_{l=1:L}^{[t,m+1]}$ contains the average gradient in each terminal node.

3.1.3 The one-dimensional optimization for expansion coefficient leads to the following update (Friedman, 2001):

$$\hat{F}_k^{[t,m+1]}(\boldsymbol{x}_i) = \hat{F}_k^{[t,m]}(\boldsymbol{x}_i) + s\sum_{l=1}^{L} \gamma_l^{[t,m+1]} \mathbb{1}(\boldsymbol{x} \in R_l^{[t,m+1]}),\ k = 1, \dots, K$$

where

$$\gamma_l^{[t,m+1]} = \frac{K-1}{K} \frac{\sum_{\boldsymbol{x}_i \in R_l^{[t,m+1]}} u_{i,k}^{[t,m]}}{\sum_{\boldsymbol{x}_i \in R_l^{[t,m+1]}} |u_{i,k}^{[t,m]}|(1 - |u_{i,k}^{[t,m]}|)}$$

We then derive the updated mixing probability $\hat{p}_{k=1:K}^{[t,m+1]}$ using (4.2).

3.1.4 Increase m by 1, and repeat steps 3.1.2-3.1.3 until m reaches to the pre-determined M_p. Set

$$\hat{F}_k^{[t+1]} = \hat{F}_k^{[t,M_p]},\quad \hat{p}_k^{[t+1]} = \hat{p}_k^{[t,M_p]},$$

Note that K regression trees are fitted independently in step 3.1.2. We can accelerate this step by parallel computing.

In step 3.2 of gradient boosting component models, we implement K weighted boosting:

3.2.1 Initialization. Set $\hat{\mu}_1^{[t,0]}, \dots, \hat{\mu}_K^{[t,0]}$ and $\hat{G}_1^{[t,0]}, \dots, \hat{G}_K^{[t,0]}$:

$$\hat{\mu}_k^{[t,0]} = \frac{\sum_{i=1}^n Y_i}{n}, \hat{G}_k^{[t,0]} = \mu_k^{-1}(\hat{\mu}_k^{[t,0]}). \tag{3.8}$$

Set $m = 0$.

3.2.2 Projection of gradient to learner. Compute the negative gradient samples $v_{1,k}^{[t,m]}, \dots, v_{n,k}^{[t,m]}$, in which

$$v_{i,k}^{[t,m]} = V_k(y_i, \hat{\boldsymbol{z}}_i^{[t]}, \hat{G}^{[t,m]}(\boldsymbol{x}_i)),$$

For each k, the data $(v_{i,k}^{[t,m]}, \boldsymbol{x}_i)_{i=1:n}$ is used to calibrate a L-terminal node regression trees $\hat{g}_k^{[t,m+1]}\left(\boldsymbol{x}; S_{l=1:L}^{[t,m+1]}, \bar{v}_{l=1:L}^{[t,m+1]}\right)$ with L_2 loss, where $S_{l=1:L}^{[t,m+1]}$ is the partition of covariate space and $\bar{v}_{l=1:L}^{[t,m+1]}$ contains the average gradient in each terminal node.

3.2.3 Conduct the following K independent one-dimensional optimizations to find the best expansion coefficients.

$$\hat{w}_k^{[t,m+1]} = \arg\min_w \sum_{i=1}^n C_k(y_i, \hat{\boldsymbol{z}}_i^{[t]}, \hat{G}_k^{[t,m]}(\boldsymbol{x}_i) + w\hat{g}_k^{[t,m+1]}(\boldsymbol{x}_i)).$$

3.2.4 For each k, compute the updates

$$\hat{G}_k^{[t,m+1]}(\boldsymbol{x}_i) = \hat{G}_k^{[t,m]}(\boldsymbol{x}_i) + s\hat{w}_k^{[t,m+1]}\hat{g}_k^{[t,m+1]}(\boldsymbol{x}_i),$$

$$\hat{\mu}_k^{[t,m+1]}(\boldsymbol{x}_i) = \mu_k(\hat{G}_k^{[t,m+1]}(\boldsymbol{x}_i)).$$

3.2.5 Increase m by 1, and repeat steps 3.2.2-3.2.4 until m reaches to the pre-determined M_μ. Set

$$\hat{G}_k^{[t+1]} = \hat{G}_k^{[t,M_\mu]},\quad \hat{\mu}_k^{[t+1]} = \hat{\mu}_k^{[t,M_\mu]}.$$

Note that K independent boosting are implemented in step 3.2, we can accelerate this step by parallel computing.
In step 4, compute the MLE $\hat{\phi}_k^{[t+1]}$ given all the other parameters $\hat{p}_k^{[t+1]}, \hat{\mu}_k^{[t+1]}$, then increase t by 1 and repeat steps 2-3 until t reaches to the pre-determined T.
In the EB algorithm, we have the following tuning parameters and the corresponding tuning strategies are listed below:

- Number of EB iterations T (iterations in outer loop). This can be determined by screening the trace plot of loss.
- Number of iterations in boosting M_p, M_μ (iterations in inner loops). We can specify a sufficiently large M_p, M_μ and use early stop according to validation loss.
- Learning rate s. Smaller learning rate tends to lead to a better fitting and predictive performance but requiring more inner iterations.
- Tuning parameters in base learner tree: complexity parameter, maximum depth, number of terminal nodes. Those parameters can be tuned as in typical decision trees.

Remarks. In initialization of boosting steps 3.1.1 and 3.2.1, we initialize parameters independently with the previously boosting estimates $\hat{p}_k^{[t-1]}, \hat{\mu}_k^{[t-1]}$; see equations (3.7) and (3.8). We call this as independent boosting. In contrast, we might initialize parameters as the previously boosted estimates

$$\hat{p}_k^{[t,0]} = \hat{p}_k^{[t-1]}, \hat{\mu}_k^{[t,0]} = \hat{\mu}_k^{[t-1]}.$$

This would lead a smaller required inner boosting iterations M (or a earlier stop on validation loss). We call this as forward boosting. However, with forward boosting, it is difficult to predict for new data due to the iterative initialization. One solution is to apply an additional boosting with default initialization (3.7) and (3.8) given the last expected hidden variables $\hat{\boldsymbol{z}}_i^{[T]}$. In the following applications, we only implement the independent boosting.

4 Applications

We conduct two simulated data analysis and a real data analysis. The first simulated data follows a mixture of Gaussian models. We use this example to illustrate the advantages of the EB algorithm over the EM algorithm. We also discuss mixture structure and paralleling computing in this example. The second simulated data follows a zero-inflated Poisson model, which mimics claims counts data with excess of zero. The real data contains the claims severity of an insurance company. This example demonstrates how we choose component models and mixing structure in practice.

4.1 First simulated example: mixture of Gaussians

The underlying model is a mixture of three Gaussians. The mixing probabilities are related to covariates, while the parameters of Gaussians are homogeneous among all samples. The probability density function is given by

$$f(Y|\boldsymbol{x};\mu,\sigma,p) = p_1(\boldsymbol{x})f_N(Y;\mu_1,\sigma_1) + p_2(\boldsymbol{x})f_N(Y;\mu_2,\sigma_2) + p_3(\boldsymbol{x})f_N(Y;\mu_3,\sigma_3), \tag{4.1}$$

where $f_N(;\mu,\sigma)$ is the probability density function of a Gaussian distribution with mean μ and standard deviation σ. We assume that the mixing probabilities depend on the covariates $\boldsymbol{x} = (x_1, x_2, x_3, x_4)^\top$ via the following equations

$$p_k(\boldsymbol{x}) = \frac{\exp F_k(\boldsymbol{x})}{\sum_{l=1}^3 \exp F_l(\boldsymbol{x})}, \text{ for } k = 1, 2, 3. \tag{4.2}$$

and

$$\begin{aligned} F_1(\boldsymbol{x}) &= x_1 + \log x_2, \\ F_2(\boldsymbol{x}) &= 1 - 0.5x_1 - x_1x_2 + 2x_3, \\ F_3(\boldsymbol{x}) &= 2\sin x_1 + \log x_2 + x_1x_3. \end{aligned}$$

Note that x_4 is a redundant variable. We specify $\mu_1 = -5, \mu_2 = 0, \mu_3 = 5, \sigma_1 = \sigma_2 = \sigma_3 = 1$. We generate the four covariates from the following distribution independently:

$$\begin{aligned} x_1 &\sim N(2,1), \\ x_2 &\sim Exp(2), \\ x_3 &\sim Bernulli(0.5), \\ x_4 &\sim \Gamma(0.5, 0.5). \end{aligned}$$

Note that we use shape-rate parameters for gamma distribution. We generate $n = 12000$ samples, among which $10,000$ samples are learning data and $2,000$ samples are test data. We fit the following 7 models:

- Mixture of distributions (homogeneous model without any covariates):

$$f(Y|\boldsymbol{x};\mu,\sigma,p) = p_1 f_N(Y;\mu_1,\sigma_1) + p_2 f_N(Y;\mu_2,\sigma_2) + p_3 f_N(Y;\mu_3,\sigma_3). \tag{4.3}$$

- Mixture of distributions with heterogeneous mixing probabilities modeled by multiclass logistic regression:

$$f(Y|\boldsymbol{x};p,\mu,\sigma) = p_1(\boldsymbol{x}) f_N(Y;\mu_1,\sigma_1) + p_2(\boldsymbol{x}) f_N(y;\mu_2,\sigma_2) + p_3(\boldsymbol{x}) f_N(Y;\mu_3,\sigma_3) \tag{4.4}$$

  where

$$p_k(\boldsymbol{x}) = \frac{\exp F_k(\boldsymbol{x})}{\sum_{l=1}^3 \exp F_l(\boldsymbol{x})}, \text{ for } k = 1, 2, 3, \tag{4.5}$$

  and

$$F_1(\boldsymbol{x}) = \langle \boldsymbol{a}, \boldsymbol{x} \rangle, F_2(\boldsymbol{x}) = \langle \boldsymbol{b}, \boldsymbol{x} \rangle, F_3(\boldsymbol{x}) = \langle \boldsymbol{c}, \boldsymbol{x} \rangle. \tag{4.6}$$

  Note that this model follows the underlying mixture structure. However, this model cannot address non-linear effects of covariates on the mixing probabilities.

- Mixture of linear regressions with heterogeneous mean and homogeneous mixing probabilities:

$$f(Y|\boldsymbol{x};p,\mu,\sigma) = p_1 f_N(Y;\mu_1(\boldsymbol{x}),\sigma_1) + p_2 f_N(y;\mu_2(\boldsymbol{x}),\sigma_2) + p_3 f_N(Y;\mu_3(\boldsymbol{x}),\sigma_3), \tag{4.7}$$

  where

$$\mu_1(\boldsymbol{x}) = \langle \alpha, \boldsymbol{x} \rangle, \mu_2(\boldsymbol{x}) = \langle \beta, \boldsymbol{x} \rangle, \mu_3(\boldsymbol{x}) = \langle \gamma, \boldsymbol{x} \rangle. \tag{4.8}$$

- Mixture of linear regressions with both heterogeneous mean and mixing probabilities:

$$f(Y|\boldsymbol{x};p,\mu,\sigma) = p_1(\boldsymbol{x})f_N(Y;\mu_1(\boldsymbol{x}),\sigma_1) + p_2(\boldsymbol{x})f_N(y;\mu_2(\boldsymbol{x}),\sigma_2) + p_3(\boldsymbol{x})f_N(Y;\mu_3(\boldsymbol{x}),\sigma_3) \tag{4.9}$$

  where the mean and mixing probabilities follow equations (4.5), (4.6) and (4.8).

- Mixture of distributions with heterogeneous mixing probabilities modeled by boosting:

$$f(Y|\boldsymbol{x};p,\mu,\sigma) = p_1(\boldsymbol{x})f_N(Y;\mu_1,\sigma_1) + p_2(\boldsymbol{x})f_N(y;\mu_2,\sigma_2) + p_3(\boldsymbol{x})f_N(Y;\mu_3,\sigma_3) \tag{4.10}$$

  where

$$p_k(\boldsymbol{x}) = \frac{\exp F_k(\boldsymbol{x})}{\sum_{l=1}^{3}\exp F_l(\boldsymbol{x})}, \text{ for } k=1,2,3. \tag{4.11}$$

  Here F_1, F_2, F_3 are estimated non-parametrically via boosting algorithm.

- Mixture of boosting with heterogeneous mean and homogeneous mixing probabilities:

$$f(Y|\boldsymbol{x};p,\mu,\sigma) = p_1 f_N(Y;\mu_1(\boldsymbol{x}),\sigma_1) + p_2 f_N(y;\mu_2(\boldsymbol{x}),\sigma_2) + p_3 f_N(Y;\mu_3(\boldsymbol{x}),\sigma_3), \tag{4.12}$$

  where μ_1, μ_2, μ_3 are estimated non-parametrically via boosting algorithm.

- Mixture of boosting with both heterogeneous mean and mixing probabilities:

$$f(Y|\boldsymbol{x};p,\mu,\sigma) = p_1(\boldsymbol{x})f_N(Y;\mu_1(\boldsymbol{x}),\sigma_1) + p_2(\boldsymbol{x})f_N(y;\mu_2(\boldsymbol{x}),\sigma_2) + p_3(\boldsymbol{x})f_N(Y;\mu_3(\boldsymbol{x}),\sigma_3) \tag{4.13}$$

  where the mean and mixing probabilities are estimated non-parametrically via boosting algorithm.

We summarize the test losses in Table **??**. Models with linear predictor have large test loss since they cannot capture non-linear effects of covariates. Models with heterogeneous mean have no obvious better out-of-sample performance compared with the counterpart models with homogeneous mean. Models with heterogeneous mixing probabilities have a better out-of-sample performance compared with the counterpart models with homogeneous mixing probabilities. Hence, an appropriate mixture structure is heterogeneous mixing probabilities with homogeneous mean.
Hence, we see the mixture of boosting with varying mixing probabilities has the outstanding out-of-sample predictive performance.
Remarks. early stop for boosting and EB outer iteration.
Accelerating by paralleling computing

4.2 Second simulated example: zero-inflated Poisson model

We consider another simulated data which is generated from a zero-inflated Poisson (ZIP) model. This simulated example mimics number of claims which usually involves excess of zero claims compared with a fitted Poisson distribution. The underlying model is given as follows:

$$f_{\text{ZIP}}(N;\lambda,\pi_0) = \begin{cases} \pi_0 + (1-\pi_0)e^{-\lambda} & \text{for} \quad N=0 \\ (1-\pi_0)\frac{e^{-\lambda}\lambda^N}{N!} & \text{for} \quad N\in\mathbb{N}_+. \end{cases} \tag{4.14}$$

The ZIP model is a mixture of a probability mass of 1 at 0 and a Poisson distribution. The mixing probability is π_0 and $1-\pi_0$. The probability density function can be written as

$$f_{\mathrm{ZIP}}(N;\lambda,\pi_0)=\pi_0\mathbb{1}_{\{N=0\}}+(1-\pi_0)\frac{e^{-\lambda}\lambda^N}{N!}.$$

Five covariates $\boldsymbol{x}=(x_1,\dots,x_5)^\top$ have systematic effects on both π_0 and λ as follows:

$$\pi_0(\boldsymbol{x})=\frac{\exp F(\boldsymbol{x})}{1+\exp F(\boldsymbol{x})},\quad \lambda(\boldsymbol{x})=\exp G(\boldsymbol{x}),\tag{4.15}$$

where

$$F(\boldsymbol{x})=0.3-2x_2^2+x_2+0.2x_5,\quad G(\boldsymbol{x})=\log 0.5+x_1^2+0.2\log x_3-0.2x_1x_4.\tag{4.16}$$

The covariates are generated from the following distributions:

$$x_1\ \sim\ N(0,0.5^2),\quad x_2\ \sim\ U(0,1),\quad x_3\ \sim\ \Gamma(2,0.5),$$

$$x_4\ \sim\ \mathit{Bernulli}(0.5),\quad x_5\ \sim\ \mathit{Bernulli}(0.2).$$

Note that we use shape-rate parameters for gamma distribution. We generate for $n=10,000$ samples $(N_i,\boldsymbol{x}_i)_{i=1:n}$, which are partitioned into a learning data $i\in\mathcal{I}_L$ and a test data $i\in\mathcal{I}_T$. The empirical proportion of zero claims in the test data is 83.10%. If a Poisson distribution is fitted to the learning data, the MLE of Poisson mean is $1/|\mathcal{I}_L|\sum_{i\in\mathcal{I}_L}N_i=0.2136$ implying the estimated proportion of zero claims is $\exp(-0.2136)=80.77\%$ less than the empirical proportion. We compare different models in terms of test loss defined as the average negative log-likelihood on the test data:

$$-\frac{1}{|\mathcal{I}_{\text{test}}|}\sum_{i\in\mathcal{I}_{\text{test}}}\log\left(\hat{\pi}_0(\boldsymbol{x}_i)\mathbb{1}_{\{N_i=0\}}+(1-\hat{\pi}_0(\boldsymbol{x}_i))\frac{e^{-\hat{\lambda}(\boldsymbol{x}_i)}\hat{\lambda}(\boldsymbol{x}_i)^{N_i}}{N_i!}\right),\tag{4.17}$$

where $\hat{\pi}_0$ and $\hat{\lambda}$ are estimated on the learning data. Since we know the underlying $\pi_0(\boldsymbol{x})$ and $\lambda(\boldsymbol{x})$, we define another two test metrics based on F and G:

$$e_\pi=\frac{1}{|\mathcal{I}_{\text{test}}|}\sum_{i\in\mathcal{I}_{\text{test}}}(\hat{F}(\boldsymbol{x}_i)-F(\boldsymbol{x}_i))^2,\tag{4.18}$$

$$e_\lambda=\frac{1}{|\mathcal{I}_{\text{test}}|}\sum_{i\in\mathcal{I}_{\text{test}}}(\hat{G}(\boldsymbol{x}_i)-G(\boldsymbol{x}_i))^2.\tag{4.19}$$

We start with a null ZIP model without any covariates:

$$F(\boldsymbol{x})=\alpha_0,\quad G(\boldsymbol{x})=\beta_0\tag{4.20}$$

The MLE can be obtained as $\hat{\pi}_0=0.5604,\hat{\lambda}=0.4860$ via the EM algorithm on the learning data. The test loss is calculated as 0.5299. The estimated proportion of zero claims is calculated as

$$\frac{1}{|\mathcal{I}_T|}\sum_{i\in\mathcal{I}_T}\left(\bar{\pi}_0+(1-\bar{\pi}_0)e^{-\bar{\lambda}}\right)=\bar{\pi}_0+(1-\bar{\pi}_0)e^{-\bar{\lambda}}=0.8308,$$

which is quite close the empirical proportion of 83.10%. Since we know the underlying $\pi_0(\boldsymbol{x})$ and $\lambda(\boldsymbol{x})$, we can calculate the true test loss as

$$-\frac{1}{|\mathcal{I}_{\text{test}}|}\sum_{i\in\mathcal{I}_{\text{test}}}\log\left(\pi_0(\boldsymbol{x}_i)\mathbb{1}_{\{N_i=0\}}+(1-\pi_0(\boldsymbol{x}_i))\frac{e^{-\lambda(\boldsymbol{x}_i)}\lambda(\boldsymbol{x}_i)^{N_i}}{N_i!}\right)=0.5150, \tag{4.21}$$

which is (surely) smaller than 0.5299 from the null model. The test metrics (4.18) and (4.19) are calculated as $e_\pi = 0.1151, e_\lambda = 0.1795$.

Next we regress N on the covariate space via the linear predictors $F(x)$ and $G(x)$. Note that one component model is degenerated into a probability mass 1 at $N = 0$, so there is no coefficient in this component model. The MLE of regression coefficients α and β can be obtained by the EM algorithm. We consider three different mixture structure, homogeneous mixing probability (4.22), homogeneous Poisson parameter (4.23) and both mixing probability and Poisson parameter are related to covariates (4.24):

$$F(\boldsymbol{x}) = \alpha_0, \quad G(\boldsymbol{x}) = \langle\boldsymbol{x},\beta\rangle, \tag{4.22}$$

$$F(\boldsymbol{x}) = \langle\boldsymbol{x},\alpha\rangle, \quad G(\boldsymbol{x}) = \beta_0, \tag{4.23}$$

$$F(\boldsymbol{x}) = \langle\boldsymbol{x},\alpha\rangle, \quad G(\boldsymbol{x}) = \langle\boldsymbol{x},\beta\rangle. \tag{4.24}$$

We have summarized the results in Table 1. Model (4.24) has the best out-of-sample performance followed by Model (4.22), while Model (4.23) has the worst out-of-sample performance. All the three models are better than the null model (4.20). Note that those models ignore the potential non-linear effects of covariates.

Finally we estimate F and G non-parametrically via the proposed EB algorithm. Similarly we consider three different mixture structure, homogeneous mixing probability (4.25), homogeneous Poisson parameter (4.26) and both mixing probability and Poisson parameter are related to covariates (4.27):

$$F(\boldsymbol{x}) = \alpha_0, \quad G(\boldsymbol{x}) = \sum_{m=0}^{M_\lambda} s\omega^{[m]}g^{[m]}(\boldsymbol{x}), \tag{4.25}$$

$$F(\boldsymbol{x}) = \sum_{m=0}^{M_\pi} s\omega^{[m]}f^{[m]}(\boldsymbol{x}), \quad G(\boldsymbol{x}) = \beta_0, \tag{4.26}$$

$$F(\boldsymbol{x}) = \sum_{m=0}^{M_\pi} s\omega^{[m]}f^{[m]}(\boldsymbol{x}), \quad G(\boldsymbol{x}) = \sum_{m=0}^{M_\lambda} s\omega^{[m]}g^{[m]}(\boldsymbol{x}). \tag{4.27}$$

The results are listed in Table 1. Although the test loss (negL) changes slightly, we can clearly distinguish the considered models according to the two metrics e_π and e_λ. We observe that Model (4.27) has the best out-of-sample performance in terms of all the metrics considered. Furthermore, all the models can address the excess of zero claims since they are all based on the ZIP distribution.

4.3 A real data example: claims severity modelling

Distribution of average claim amount Claims amount data `freMTPL2sev` from R package **CASdatasets**. Sample size $n = 24,938$. Three peaks and heavy tail.

Table 1: Comparison of different ZIP models on the test data. Note that Models (4.20), (4.22), (4.23) and (4.24) are estimated via the EM algorithm, while Models (4.25), (4.26) and (4.27) are estimated via the EB algorithm.

model	negL	e_π	e_λ	proportion of zero claims
Null (4.20)	0.5299	0.1151	0.1795	0.8308
Constant π (4.22)	0.5258	0.1112	0.1654	0.8305
Constant λ (4.23)	0.5265	0.0959	0.182	0.8304
Varying π and λ (4.24)	0.5257	0.1159	0.1776	0.8304
Constant π (4.25)	0.5234	0.1114	0.0928	0.8316
Constant λ (4.26)	0.5256	0.0579	0.1738	0.8293
Varying π and λ (4.27)	0.5199	0.0687	0.0588	0.8336
True	0.5150	0.0000	0.0000	0.8310

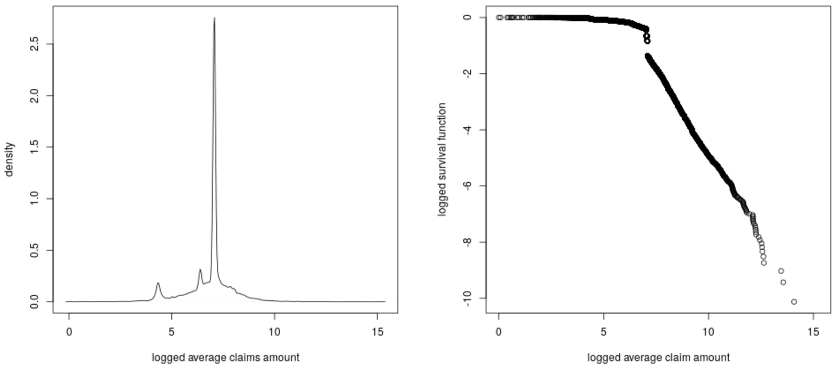

Figure 1: Histogram and logged survival function of logged average claims.

Mixture of distributions Three gamma distributions for three peaks. One gamma distribution for resting non-tail part. One Pareto distribution for tail.

$$f(y) = \sum_{k=1}^{4} p_k f_{gamma}(y; \mu_k, \phi_k) + p_5 f_{pareto}(y; \alpha, M)$$

where μ, ϕ are mean and dispersion parameter, and α, M are tail index and threshold. The threshold is pre-determined as 8158.13 according to Hill plot.

Initialization of hidden variable

Figure 1 indicates a way to initialize the hidden variable:

$$\begin{aligned}\hat{z}_i^{[0]} &= (\hat{z}_{i,1}^{[0]}, \hat{z}_{i,2}^{[0]}, \hat{z}_{i,3}^{[0]}, \hat{z}_{i,4}^{[0]}, \hat{z}_{i,5}^{[0]})^\top \\ &= (\mathbb{1}_{(0,500]} y_i, \mathbb{1}_{(500,1000]} y_i, \mathbb{1}_{(1000,1200]} y_i, \mathbb{1}_{(1200,8158.13]} y_i, \mathbb{1}_{(8158.13,\infty)} y_i)^\top\end{aligned} \tag{4.28}$$

Other parameters can be initialized as the MLE based on the full likelihood function.

EM algorithm

Estimated parameters

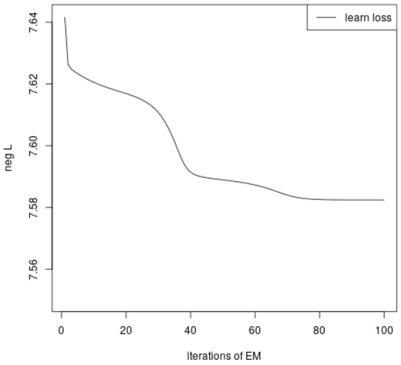

Figure 2: Learning loss of mixture of distributions.

Table 2: MLE of four component gamma distributions. Tail index is estimated as $\hat{\alpha} = 1.0773$

component k	μ_k	shape $(1/\phi_k)$	scale	rate
1	76.8727	105.556	0.7283	1.3731
2	592.5909	653.539	0.9067	1.1029
3	1171.3811	999.9999	1.1714	0.8537
4	1534.5143	1.0377	1478.7768	7e-04

Those large shape parameters (small dispersion) implies the difficulties with gamma mean modeling.

Boosting mixing probabilities

$$f(y|\boldsymbol{x}) = \sum_{k=1}^{4} p_k(\boldsymbol{x}) f_{gamma}(y; \mu_k, \phi_k) + p_5(\boldsymbol{x}) f_{pareto}(y; \alpha, M)$$

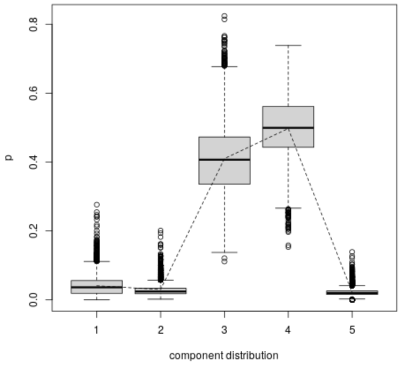

Figure 3: Boxplot of estimated mixing probabilities.

Boosting the forth component mean

$$f(y|\boldsymbol{x}) = \sum_{k=1}^{3} p_k(\boldsymbol{x}) f_{gamma}(y;\mu_k,\phi_k) + \\ p_4(\boldsymbol{x}) f_{gamma}(y;\mu_4(\boldsymbol{x}),\phi_4) + p_5(\boldsymbol{x}) f_{pareto}(y;\alpha,M)$$

Test loss (negative log-likelihood)

1. Mixture of distributions: 7.5815
2. Boosting mixing probabilities: **7.5588**
3. Boosting mixing probabilities and the forth component mean: 7.5573.

5 Conclusions

Our proposal: Expectation-Boosting algorithm Expectation-Boosting (EB) algorithm:

- Replaces the maximization step by an overfitting-sensitive boosting step.
- The boosting step follows a generic functional gradient descent algorithm.

Advantages Several advantages of EB algorithm over the EM algorithm.

- No need for specifying the form of component regression functions and performing covariate transformation.
- Only need for component loss functions.
- Boosting algorithm is a flexible non-parametric regression facilitating both non-linear effects and interaction.
- Boosting algorithm is overfitting-sensitive, we can perform variable selection simultaneously during the EB algorithm.

References

Bühlmann, P. and Hothorn, T. (2007). Boosting algorithms: Regularization, prediction and model fitting, *Statistical science* **22**(4): 477–505.

Bühlmann, P. and Yu, B. (2003). Boosting with the l 2 loss: regression and classification, *Journal of the American Statistical Association* **98**(462): 324–339.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine, *Annals of statistics* pp. 1189–1232.

Mayr, A., Fenske, N., Hofner, B., Kneib, T. and Schmid, M. (2012). Generalized additive models for location, scale and shape for high dimensional data—a flexible approach based on boosting, *Journal of the Royal Statistical Society: Series C (Applied Statistics)* **61**(3): 403–427.

Thomas, J., Mayr, A., Bischl, B., Schmid, M., Smith, A. and Hofner, B. (2018). Gradient boosting for distributional regression: faster tuning and improved variable selection via noncyclical updates, *Statistics and Computing* **28**(3): 673–687.