Insurance loss modeling with gradient tree-boosted mixture models.

Guangyuan Gao Jiahong Li

February 24, 2022

Abstract

Mixture of models are often applied in insurance loss modeling. The Expectation-Maximization (EM) algorithm is usually applied 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 trees. The boosting algorithm estimates the regression functions in component models non-paramerically and overfitting-sensitively. The EB algorithm performs automated feature engineering, model fitting and variable selection simultaneously, fully exploring the predictive power of covariate space. Simulations show that the proposed method performs very well. An empirical analysis of the claims amounts data is illustrated for the proposed methodology.

Keywords: Mixture models; EM algorithm; Gradient boosting.

1 Introduction

Insurance loss data always cannot be well modelled by a single distribution. For claim count data, it may have an excess of zero claims, so a Poisson distribution may not be the best option. For claim amount data, its probability density function may be multimodal or heavy tailed, so a gamma distribution is not enough to describe the entire data. One solution to such issues is to apply mixture of distributions. 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. Mixture model is proposed by Goldfeld and Quandt (1973). We refer to Lindsay (1995) and Peel and MacLahlan (2000) for detailed review of mixture models. In machine learning applications, mixture models are also called as mixture of experts models (Jacobs et al., 1991; Jiang and Tanner, 1999).

Mixture models have been used for insurance loss modeling frequently. Zhang et al. (2020, 2022) developed a multivariate zero-inflated hurdle model to describe multivariate count data with extra zeros. Delong et al. (2021) propose a mixture of neural networks with gamma loss to model insurance claim amounts. Verbelen et al. (2015) and Lee and Lin (2010) used mixture of Erlangs to model insurance claim amounts including censored and truncated data. Lee and Lin (2012) developed the multivariate version of mixture Erlangs. Fung et al. (2019b), Fung et al. (2019a) and Tseung et al. (2021) proposed a so called logit-weighted reduced mixture of experts models for multivariate claim frequencies or severities distributions. To the best of our knowledge, the existing studies always impose a linearity constraint on the regression function, i.e., under a generalized linear model framework. In this paper, we will relax this constraint by estimating regression function non-parametrically.

Parameter estimation in mixture models is challenging since component parameters are related to each other. The Expectation-Maximization (EM) algorithm proposed by Dempster et al. (1977) is an iterative method to estimate component parameters and (hidden) component indicator variable. Variable selection and feature engineering in mixture models is also challenging since the relationship between covariates and response vary from one component to another. Khalili and Chen (2007) introduced a penalized likelihood approach for variable selection. Huang et al. (2013) and Huang and Yao (2012) proposed non-parametric mixture of regression to relax the linearity assumption on the regression function. Selecting the number of components is another challenging problem. Naik et al. (2007) derived a new information criterion for selection of component numbers and variables. Kasahara and Shimotsu (2015) proposed a likelihood-based test to contrast mixture models with different numbers of components. In this paper, we assume that the number of components is known, and we will address variable selection and feature engineering in the proposed method.

We propose an Expectation-Boosting (EB) algorithm, which replaces the maximization step by a boosting step. Boosting algorithm is an iterative method to estimate regression function non-parametrically, where weak learners are calibrated step-wisely and added together. We roughly categorize some commonly used boosting algorithms into three groups: (a) Binary classification: AdaBoost, LogitBoost (real, discrete, gentle AdaBoost), AdaBoost.M1; (b) Multinomial classification: Stagewise Additive Modeling using a Multi-class Exponential loss (SAMME), SAMME.R (multi-class real AdaBoost); (c) 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. Friedman (2001) has studied this type of boosting from a statistical perspective. Our EB algorithm uses GBDT in the boosting step. With the negative log-likelihood as the loss function for boosting, the boosting step increases the likelihood at each iteration of the EB algorithm. 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. Variable selection and feature engineering are performed automatically during model fitting. Second, it is well known that boosting algorithm is overfitting-sensitive (Bühlmann and Yu, 2003). This property is very valuable since the ultimate goal of insurance loss modeling is to predict the future loss. The out-of-sample prediction is more important than the goodness-of-fitting in our applications.

The resting of paper is structured as follows. Section 2 reviews mixture models and the EM algorithm. Section 3 proposes the EB algorithm. Initialization, tuning parameters and parallel computing are discussed. Section 4 studies two simulated data and a real data to illustrate the proposed methodology. Section 5 concludes the paper with several important findings.

2 Review: mixture of models and the EM algorithm

In this section, we first review mixture models, particularly finite mixture of regressions (Peel and MacLahlan, 2000), and discuss different mixture structure. Then we review the EM algorithm which is the foundation of the proposed EB 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 a collection of distributions

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

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

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

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

$$f(y|\boldsymbol{x})=\sum_{k=1}^{K}p_k(\boldsymbol{x})f_k(y|\boldsymbol{x};\mu_k(\boldsymbol{x}),\phi_k(\boldsymbol{x})). \tag{2.1}$$

Above equation (2.1) is the most general form for a finite mixture of regressions, i.e., both mixing probabilities and component parameters depend on $\boldsymbol{x}$. In applications, based on shape of data and aspects of interest, we always put some constraints on this general form. For example,

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

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

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

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

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

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

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

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

We select among those above alternatives according to the data and our aim. Imposing suitable constraints, we can accelerate and stabilize model fitting without compromising predictive performance.

2.2 The EM algorithm

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

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

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

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

Thus, the log-likelihood function is derived as

$$l(p, \mu, \phi|y, \boldsymbol{z}, \boldsymbol{x}) = \sum_{k=1}^{K} z_k \left[\log p_k(\boldsymbol{x}) + \log f_k(y|\boldsymbol{x}; \mu_k(\boldsymbol{x}), \phi_k(\boldsymbol{x}))\right]. \tag{2.2}$$

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:

$$\hat{\theta}_p = \underset{\theta_p}{\arg\max} \sum_{i=1}^{n} \sum_{k=1}^{K} z_{i,k} \log p_k(\boldsymbol{x}_i; \theta_p) \tag{2.3}$$

and

$$\left(\hat{\theta}_\mu^{(k)}, \hat{\theta}_\phi^{(k)}\right) = \underset{\theta_\mu^{(k)}, \theta_\phi^{(k)}}{\arg\max} \sum_{i=1}^{n} \sum_{k=1}^{K} z_{i,k} \log f_k\left(y_i; \mu_k\left(\boldsymbol{x}_i; \theta_\mu^{(k)}\right), \phi_k\left(\boldsymbol{x}_i; \theta_\phi^{(k)}\right)\right), \text{ for } k = 1, \ldots, K. \tag{2.4}$$

Those optimizations are corresponding to *a multinomial logistic classification* and K *regressions.* The multinomial logistic classification (2.3) are fitted to all samples, while K regressions (2.4) 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 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 log-likelihood function for full information, calculate the MLE of regression coefficients $\theta_p, \theta_\mu, \theta_\phi$.

$$\begin{aligned}
&l(p, \mu, \phi|y, \hat{z}, \boldsymbol{x}) \\
=&\sum_{i=1}^{n} \sum_{k=1}^{K} \hat{z}_{i,k} \left[\log p_k(\boldsymbol{x}_i; \theta_p) + \log f_k\left(y_i; \mu_k\left(\boldsymbol{x}_i; \theta_\mu^{(k)}\right), \phi_k\left(\boldsymbol{x}_i; \theta_\phi^{(k)}\right)\right)\right] \\
=&\sum_{i=1}^{n} \sum_{k=1}^{K} \hat{z}_{i,k} \log p_k(\boldsymbol{x}_i; \theta_p) + \sum_{i=1}^{n} \sum_{k=1}^{K} \hat{z}_{i,k} \log f_k\left(y_i; \mu_k\left(\boldsymbol{x}_i; \theta_\mu^{(k)}\right), \phi_k\left(\boldsymbol{x}_i; \theta_\phi^{(k)}\right)\right).
\end{aligned} \tag{2.5}$$

This step is similar to (2.3) and (2.4). 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

In the EM algorithm, the regression functions $p(\boldsymbol{x};\theta_p), \mu(\boldsymbol{x};\theta_\mu), \phi(\boldsymbol{x};\theta_\phi)$ are always assumed to be parametric functions. To make component models more flexible, we replace the maximization step in the EM algorithm by a boosting step. Boosting algorithm estimates the regression functions non-parametrically and facilitates more flexible shapes of regression function. With negative log-likelihood function as the loss function in boosting algorithm, the boosting step increases the likelihood in each iteration, but overfitting-sensitively. The boosting step follows a generic functional gradient descent algorithm. So 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 connects the regression function F with the parameters of interest such as mean, probability odds, etc. Commonly used link functions 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: Set a suitable initial value $\hat{F}^{[0]}(\boldsymbol{x})$. Let $m = 0$.

2. Projection of gradient to weak learner: Calculate *negative gradient*

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

   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 a 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 this 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 decision trees (Breiman et al., 1983), the algorithm is called *gradient boosting decision tree (GBDT)*, where calibration of decision trees and variable selection are performed simultaneously. Please refer to Hastie et al. (2009) for decision trees and the recursive partitioning algorithm. 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 parameters of interest p, μ, ϕ by F, G, H. In the EB algorithm, we need to specify link functions and loss functions, which determines negative gradient function. For mixing probabilities p, we choose multiple logistic link function:

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

or equivalently

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

We use the negative log-likelihood function of multinomial distribution as the loss 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 loss function w.r.t. F_k is given by

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

For component parameters μ and ϕ, the link function and loss function depend on the component models. Commonly used link functions in component models are listed as follows

- Component Gaussian model: $G_k = \mu_k, H_k = \log \phi_k$;
- Component Poisson model: $G_k = \log \mu_k$;
- Component gamma model: $G_k = \log \mu_k, H_k = \log \phi_k$.

The negative log-likelihood function of each component model is used as its loss function:

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

where Z_k is treated as a weight multiplied to the usual negative log-likelihood function $-\log f_k$. 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 $\boldsymbol{x}$). Upon the above defined notations, we are ready to introduce the proposed EB algorithm. We first summarize the proposed EB algorithm as follows:

1 Initialize $\hat{p}^{[0]}, \hat{\mu}^{[0]}, \hat{\phi}^{[0]}$. Set $t = 0$.

2 Calculate 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 parameters $\hat{\mu}^{[t+1]}$ given latent variable $\hat{z}^{[t]}$.

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

In this algorithm we have two loops: the outer EB iteration and the inner boosting iteration. The loop of inner boosting iteration is nested into the loop of outer EB iteration. Hence, the EB algorithm takes much more time than the EM algorithm which contains only one loop; see Table 1. In step 3.1, K independent boosting algorithms are conducted for $F_k, k = 1, \ldots, K$. In step 3.2, another K independent boosting algorithms are conducted for $G_k, k = 1, \ldots, K$. Therefore, the EB algorithm can be accelerated significantly by performing those $2K$ boosting algorithms simultaneously; see Table 1. Note that in the algorithm we have assumed that dispersion ϕ_k is fixed among samples (not related to $\boldsymbol{x}$). The extension to joint modeling of both mean and dispersion is not straightforward since they are not independent; please refer to Jorgensen (1997) for dispersion modeling. Joint modeling both mean and dispersion is not considered in this paper.
We now detail each step of the EB algorithm. In step 1 of initialization of EB algorithm, we initialize the following quantities.

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

- $\hat{p}_k^{[0]} = \frac{1}{K}$,

- $\hat{F}_1^{[0]}, \ldots, \hat{F}_K^{[0]}$ are obtained by (3.3).

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

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

where μ_k^{-1} is the link function in the component model k.

1.3 Initialize $\hat{\phi}_1^{[0]}, \ldots, \hat{\phi}_K^{[0]}$ as the sample variance. We assume that dispersion is not related to covariates.

1.4 Set $t = 0$.

Note that step 1 will not be repeated in the algorithm. 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 $\hat{p}_1^{[t,0]}, \ldots, \hat{p}_K^{[t,0]}$ and $\hat{F}_1^{[t,0]}, \ldots, \hat{F}_K^{[t,0]}$ as

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

Set $m = 0$. In contrast to step 1, this initialization will be repeated for each outer EB iteration. In the inner loop of boosting step 3.1, this initialization will not be repeated; see step 3.1.4.

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

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

For each $k = 1, \ldots, K$, the data $(u_{i,k}^{[t,m]}, \boldsymbol{x}_i)_{i=1:n}$ is used to calibrate a L-terminal node regression tree with L_2 loss

$$\hat{f}_k^{[t,m+1]}\left(\boldsymbol{x}; R_{k,l=1:L}^{[t,m+1]}, \bar{u}_{k,l=1:L}^{[t,m+1]}\right),$$

where $R_{k,l=1:L}^{[t,m+1]}$ is the partition of covariate space and $\bar{u}_{k,l=1:L}^{[t,m+1]}$ contains the average gradient in each terminal node.

3.1.3 One dimensional optimization and update. It turns out that the one-dimensional optimization for expansion coefficient leads to the following update (see equation (32) in Friedman (2001)):

$$\hat{F}_k^{[t,m+1]}(\boldsymbol{x}_i) = \hat{F}_k^{[t,m]}(\boldsymbol{x}_i) + s\sum_{l=1}^{L}\gamma_{k,l}^{[t,m+1]}\mathbb{1}_{R_{k,l}^{[t,m+1]}}(\boldsymbol{x}_i),\ k = 1,\ldots,K,$$

where

$$\gamma_{k,l}^{[t,m+1]} = \frac{K-1}{K}\frac{\sum_{\boldsymbol{x}_i\in R_{k,l}^{[t,m+1]}} u_{i,k}^{[t,m]}}{\sum_{\boldsymbol{x}_i\in R_{k,l}^{[t,m+1]}}\left|u_{i,k}^{[t,m]}\right|\left(1-\left|u_{i,k}^{[t,m]}\right|\right)},\ k = 1,\ldots,K, l = 1,\ldots,L.$$

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

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

In step 3.1, K independent boostings are performed for $F_k, k = 1:K$. Hence, we can accelerate step 3.1 by parallel computing. To the best of our knowledge, there is no available R package supporting a multinomial logistic boosting with fractional response. So we code step 3.1 from bottom. Only the R package **rpart** is called to apply the recursive partition algorithm to calibrate the weak learner of decision trees.

In step 3.2 of gradient boosting component models, we implement another K independent *weighted* boosting with weights $\hat{z}_{i,k}^{[t]}$:

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

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

where μ_k^{-1} is the link function in the component model k. Set $m = 0$. In contrast to step 1, this initialization will be repeated for each outer EB iteration. In the inner loop of boosting step 3.2, this initialization will not be repeated; see step 3.2.5.

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

$$v_{i,k}^{[t,m]} = V_k(y_i, \hat{\boldsymbol{z}}_i^{[t]}, \hat{G}^{[t,m]}(\boldsymbol{x}_i)),\ i = 1,\ldots,n,\ k = 1,\ldots,K.$$

For each $k = 1,\ldots,K$, the data $(v_{i,k}^{[t,m]}, \boldsymbol{x}_i)_{i=1:n}$ is used to calibrate a L-terminal node regression trees with L_2 loss

$$\hat{g}_k^{[t,m+1]}\left(\boldsymbol{x}; S_{k,l=1:L}^{[t,m+1]}, \bar{v}_{k,l=1:L}^{[t,m+1]}\right),$$

where $S_{k,l=1:L}^{[t,m+1]}$ is the partition of covariate space and $\bar{v}_{k,l=1:L}^{[t,m+1]}$ 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.

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

3.2.4 For each $k = 1, \ldots, K$, 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 weighted boostings are implemented in step 3.2, we can accelerate this step by parallel computing as in step 3.1. We rely on the R packages **gbm** and **mboost** in this step. Those two packages can facilitate boosting with additional weights.

In step 4, we calculate 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. Hence, we have an inner loop in step

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 count 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):

$$f(Y|\boldsymbol{x};\mu,\sigma,p)=p_1f_N(Y;\mu_1,\sigma_1)+p_2f_N(Y;\mu_2,\sigma_2)+p_3f_N(Y;\mu_3,\sigma_3). \tag{4.3}$$

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

$$f(Y|\boldsymbol{x};p,\mu,\sigma)=p_1f_N(Y;\mu_1(\boldsymbol{x}),\sigma_1)+p_2f_N(y;\mu_2(\boldsymbol{x}),\sigma_2)+p_3f_N(Y;\mu_3(\boldsymbol{x}),\sigma_3), \tag{4.4}$$

  where

$$\mu_1(\boldsymbol{x})=\langle\alpha,\boldsymbol{x}\rangle,\mu_2(\boldsymbol{x})=\langle\beta,\boldsymbol{x}\rangle,\mu_3(\boldsymbol{x})=\langle\gamma,\boldsymbol{x}\rangle. \tag{4.5}$$

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

$$f(Y|\boldsymbol{x};p,\mu,\sigma) = p_1(\boldsymbol{x})f_N(Y;\mu_1,\sigma_1) + p_2(\boldsymbol{x})f_N(y;\mu_2,\sigma_2) + p_3(\boldsymbol{x})f_N(Y;\mu_3,\sigma_3) \tag{4.6}$$

  where

$$p_k(\boldsymbol{x}) = \frac{\exp F_k(\boldsymbol{x})}{\sum_{l=1}^{3}\exp F_l(\boldsymbol{x})}, \text{ for } k = 1,2,3, \tag{4.7}$$

  and

$$F_1(\boldsymbol{x}) = \langle \boldsymbol{a}, \boldsymbol{x}\rangle, F_2(\boldsymbol{x}) = \langle \boldsymbol{b}, \boldsymbol{x}\rangle, F_3(\boldsymbol{x}) = \langle \boldsymbol{c}, \boldsymbol{x}\rangle. \tag{4.8}$$

  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 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 (4.7), (4.8) and (4.5).

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

$$f(Y|\boldsymbol{x};p,\mu,\sigma) = p_1f_N(Y;\mu_1(\boldsymbol{x}),\sigma_1) + p_2f_N(y;\mu_2(\boldsymbol{x}),\sigma_2) + p_3f_N(Y;\mu_3(\boldsymbol{x}),\sigma_3), \tag{4.10}$$

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

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

$$f(Y|\boldsymbol{x};p,\mu,\sigma) = p_1(\boldsymbol{x})f_N(Y;\mu_1,\sigma_1) + p_2(\boldsymbol{x})f_N(y;\mu_2,\sigma_2) + p_3(\boldsymbol{x})f_N(Y;\mu_3,\sigma_3) \tag{4.11}$$

  where

$$p_k(\boldsymbol{x}) = \frac{\exp F_k(\boldsymbol{x})}{\sum_{l=1}^{3}\exp F_l(\boldsymbol{x})}, \text{ for } k = 1,2,3. \tag{4.12}$$

  Here F_1,F_2,F_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 1. Note that the true negative log-likelihood is calculated based on the simulated parameters $p(\boldsymbol{x}),\mu,\sigma$. Models with linear predictor (4.4), (4.6) and (4.9) have larger test loss than Models (4.10), (4.11) and (4.13) since they cannot capture non-linear effects of covariates. Models with heterogeneous mean and mixing probabilities (4.9) and (4.13) have no obvious better out-of-sample performance compared with the counterpart models with homogeneous mean (4.6) and (4.11). Models with heterogeneous mixing probabilities (4.6) and (4.11) have a better out-of-sample performance compared with the counterpart models with homogeneous mixing probabilities (4.3). Hence, an appropriate mixture structure is heterogeneous mixing probabilities with homogeneous mean.

Table 1: Comparison of different models in terms of test loss and running time.

model	negative log-likelihood	running time without parallel computing	running time with parallel computing
Null (4.3)	2.4821	<10 seconds	NA
Constant μ (4.4)	2.4779	<10 seconds	NA
Constant p (4.6)	2.2016	<10 seconds	NA
Varying μ and p (4.9)	2.2011	<10 seconds	NA
Constant μ (4.10)	2.4139	<10 seconds	NA
Constant p (4.11)	2.1330	650 seconds	381 seconds
Varying μ and p (4.13)	2.1320	1054 seconds	597 seconds
True	2.1231	NA	NA

The boosting of mixing probabilities takes the majority of running in the EB algorithm. We accelerate the EB algorithm by applying parallel computing for the boosting of mixing probabilities in step 3.1 of the EB algorithm, i.e., we fit K trees for $F_k, k = 1 : K$ simultaneously. The running time with and without parallel computing are listed in Table 1. Parallel computing accelerates the EB algorithm significantly.

Remarks. In the EB algorithm, we need to determine when to stop inner boosting loop and when to stop outer EB loop. We partition the learning data into training data and validation data. We use the validation loss to early stop the inner boosting iteration and the training loss to stop the outer EB iteration.

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:

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

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 $\boldsymbol{x} = (x_1, \ldots, x_5)^\top$ 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:

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

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

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 2. 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 2. 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.

Table 2: 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

4.3 Third example: claims severity modelling

In this example, we study the claims amount data `freMTPL2sev` from R package **CASdatasets**, which contains $n = 24,938$ samples. We plot the histogram and the logged survival function of logged average claim amount in Figure 1, which shows three peaks and a heavy tail. An intuitive choice of component distributions is that three peaks are modelled by three gamma distributions, respectively, the resting non-tail part by a forth gamma distribution, and the tail part by a Pareto distribution. Therefore, the probability density function (of the null model) is

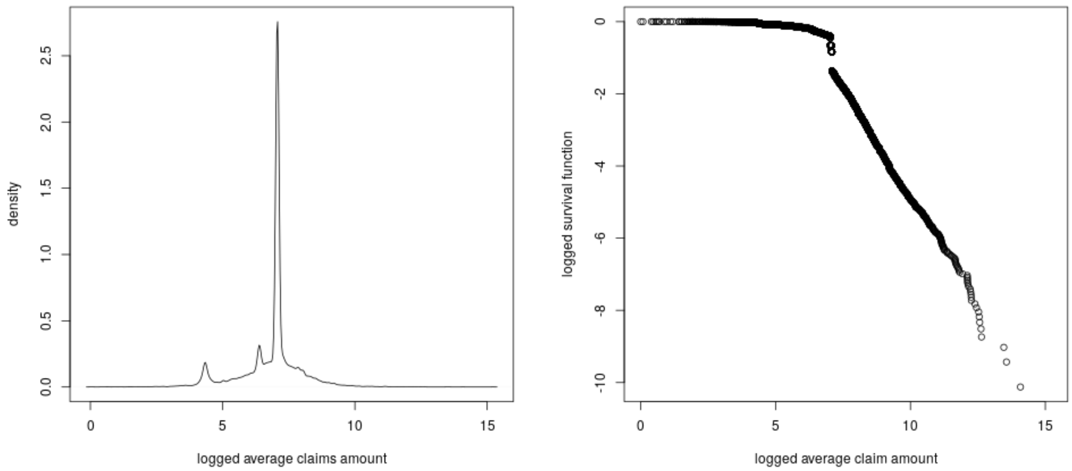

Figure 1: Histogram and logged survival function of logged average claims.

given by

$$f(Y|\boldsymbol{x}; p, \mu, \phi, \alpha) = \sum_{k=1}^{4} p_k f_G(Y; \mu_k, \phi_k) + p_5 f_P(Y; \alpha, M) \tag{4.28}$$

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.

Figure 1 indicates a way to initialize the hidden variable:

$$\begin{aligned} \hat{z}_i^{[0]} &= (\hat{z}_{i,1}^{[0]}, \hat{z}_{i,2}^{[0]}, \hat{z}_{i,3}^{[0]}, \hat{z}_{i,4}^{[0]}, \hat{z}_{i,5}^{[0]})^\top \\ &= (\mathbb{1}_{(0,500]} y_i, \mathbb{1}_{(500,1000]} y_i, \mathbb{1}_{(1000,1200]} y_i, \mathbb{1}_{(1200,8158.13]} y_i, \mathbb{1}_{(8158.13,\infty)} y_i)^\top \end{aligned} \tag{4.29}$$

Other parameters can be initialized as the MLE based on the full likelihood function. We first fit a null model (4.28) to the data via the EM algorithm. The trace of learning loss is shown in Figure 2. The estimated parameters are listed in Table 3. We observe that three peaks are captured by the first three component gamma distributions. Those large shape parameters

Table 3: 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

(small dispersion) implies the difficulties with gamma mean modeling. The test loss is calculated as 7.5815.

Next we model the mixing probabilities by boosting algorithm:

$$f(Y|\boldsymbol{x}; p, \mu, \phi, \alpha) = \sum_{k=1}^{4} p_k(\boldsymbol{x}) f_G(Y; \mu_k, \phi_k) + p_5(\boldsymbol{x}) f_P(Y; \alpha, M). \tag{4.30}$$

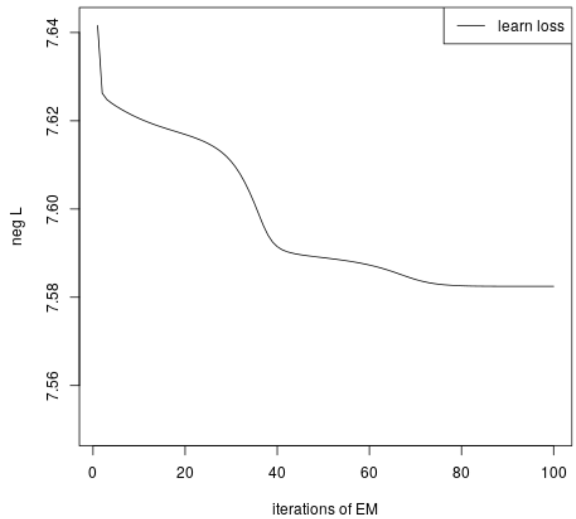

Figure 2: Learning loss of mixture of distributions.

We draw the boxplot of the mixing probabilities in Figure 3, which implies that the mixing probabilities of third and forth components p_3 and p_4 are more related with the covariates. The test loss is calculated as 7.5588 smaller than the null model.

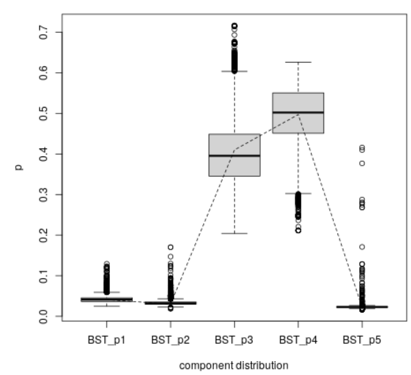

Figure 3: Boxplot of estimated mixing probabilities.

The small shape parameter of the forth component as shown in Table 3 indicates a possible improvement by boosting the forth component mean. Therefore, we fit the following model:

$$
\begin{aligned}
f(Y|\boldsymbol{x};p,\mu,\phi,\alpha) = &\sum_{k=1}^{3} p_k(\boldsymbol{x}) f_G(Y;\mu_k,\phi_k) + \\
& p_4(\boldsymbol{x}) f_G(Y|\boldsymbol{x};\mu_4(\boldsymbol{x}),\phi_4) + p_5(\boldsymbol{x}) f_P(Y;\alpha,M).
\end{aligned}
$$

Test loss is calculated as 7.5573 smaller than that of model (4.30).

5 Conclusions

Insurance loss data sometimes cannot be sufficiently modelled by a single distribution. Those data is usually modelled by mixture of models. The traditional estimation method for mixture of models is the EM algorithm. However, thee EM algorithm is not quite useful when the component model is supposed to be non-parametric. In this paper, we propose an Expectation-Boosting (EB) algorithm for mixture of models, where both the mixing probabilities and the component models are supposed to be modelled non-parametrically. The proposed algorithm replaces the maximization step of the EM algorithm by a generic functional gradient descent algorithm. 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. There is no need for specifying the form of component regression functions and performing covariate transformation, which is difficult in the mixture of models. Second, boosting algorithm is overfitting-sensitive, we can perform variable selection simultaneously during the EB algorithm. In this paper, we do not discuss the model interpretation, which is quite similar to those in normal boosting algorithm, such as variable importance, ice curve, etc. We discuss number of components selection ad hoc rather than systematically. One may refer to ?? for more details.

References

Breiman, L., Friedman, J. H., Olshen, R. A. and Stone, C. J. (1983). *Classification and regression trees*, Wadsworth.

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 L2 loss: regression and classification, *Journal of the American Statistical Association* **98**(462): 324–339.

Delong, Ł., Lindholm, M. and Wüthrich, M. V. (2021). Gamma mixture density networks and their application to modelling insurance claim amounts, *Insurance: Mathematics and Economics* **101**: 240–261.

Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm, *Journal of the Royal Statistical Society: Series B (Methodological)* **39**(1): 1–22.

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

Fung, T. C., Badescu, A. L. and Lin, X. S. (2019a). A class of mixture of experts models for general insurance: Application to correlated claim frequencies, *ASTIN Bulletin: The Journal of the IAA* **49**(3): 647–688.

Fung, T. C., Badescu, A. L. and Lin, X. S. (2019b). A class of mixture of experts models for general insurance: Theoretical developments, *Insurance: Mathematics and Economics* **89**: 111–127.

Goldfeld, S. M. and Quandt, R. E. (1973). A markov model for switching regressions, *Journal of econometrics* **1**(1): 3–15.

Hastie, T., Tibshirani, R., Friedman, J. H. and Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*, Vol. 2, Springer.

Huang, M., Li, R. and Wang, S. (2013). Nonparametric mixture of regression models, *Journal of the American Statistical Association* **108**(503): 929–941.

Huang, M. and Yao, W. (2012). Mixture of regression models with varying mixing proportions: a semiparametric approach, *Journal of the American Statistical Association* **107**(498): 711–724.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J. and Hinton, G. E. (1991). Adaptive mixtures of local experts, *Neural computation* **3**(1): 79–87.

Jiang, W. and Tanner, M. A. (1999). Hierarchical mixtures-of-experts for exponential family regression models: approximation and maximum likelihood estimation, *The Annals of Statistics* **27**(3): 987–1011.

Jorgensen, B. (1997). *The theory of dispersion models*, CRC Press.

Kasahara, H. and Shimotsu, K. (2015). Testing the number of components in normal mixture regression models, *Journal of the American Statistical Association* **110**(512): 1632–1645.

Khalili, A. and Chen, J. (2007). Variable selection in finite mixture of regression models, *Journal of the american Statistical association* **102**(479): 1025–1038.

Lee, S. C. and Lin, X. S. (2010). Modeling and evaluating insurance losses via mixtures of erlang distributions, *North American Actuarial Journal* **14**(1): 107–130.

Lee, S. C. and Lin, X. S. (2012). Modeling dependent risks with multivariate erlang mixtures, *ASTIN Bulletin: The Journal of the IAA* **42**(1): 153–180.

Lindsay, B. G. (1995). Mixture models: theory, geometry, and applications, Ims.

Naik, P. A., Shi, P. and Tsai, C.-L. (2007). Extending the akaike information criterion to mixture regression models, *Journal of the American Statistical Association* **102**(477): 244–254.

Peel, D. and MacLahlan, G. (2000). Finite mixture models, *John & Sons* .

Tseung, S. C., Badescu, A. L., Fung, T. C. and Lin, X. S. (2021). Lrmoe. jl: a software package for insurance loss modelling using mixture of experts regression model, *Annals of Actuarial Science* **15**(2): 419–440.

Verbelen, R., Gong, L., Antonio, K., Badescu, A. and Lin, S. (2015). Fitting mixtures of erlangs to censored and truncated data using the em algorithm, *ASTIN Bulletin: The Journal of the IAA* **45**(3): 729–758.

Zhang, P., Calderin, E., Li, S. and Wu, X. (2020). On the type i multivariate zero-truncated hurdle model with applications in health insurance, *Insurance: Mathematics and Economics* **90**: 35–45.

Zhang, P., Pitt, D. and Wu, X. (2022). A new multivariate zero-inflated hurdle model with applications in automobile insurance, *ASTIN Bulletin: The Journal of the IAA* pp. 1–24.