Insurance Loss modeling with Gradient Tree-Boosted Mixture Models

Yanxi Hou* Jiahong Li† Guangyuan Gao‡

March 18, 2022

Abstract

In actuarial practice, the mixture model is one widely applied statistical method to model the insurance loss data. Although the Expectation-Maximization (EM) algorithm usually plays an essential tool for the parameter estimation of mixture models, it suffers from other issues which cause unstable predictions. For example, feature engineering and variable selection are two crucial modeling issues that are challenging for mixture models as they involve several component models. Moreover, avoiding overfitting is another technical concern of the modeling method for the prediction of future losses. To address those issues, we propose an Expectation-Boosting (EB) algorithm, which replaces the maximization step in the EM algorithm by gradient boosting machines with regression trees. Our proposed EB algorithm can estimate the mixing probabilities and the component regression functions non-parametrically and overfitting-sensitively and perform automated feature engineering, model fitting, and variable selection simultaneously, which fully explores the predictive power of feature space. Moreover, the proposed algorithm can be combined with parallel computation methods to improve computation efficiency. Finally, we conduct two simulation studies to show the good performance of the proposed algorithm and an empirical analysis of the claims amounts data for illustration.

Keywords: Insurance loss; Mixture models; EM algorithm; Gradient boosting; Parallel computation; Finite mixture of regressions; Zero-inflated Poisson model.

*School of Data Science, Fudan University, 200433 Shanghai, China.

†School of Mathematical Sciences, Peking University, 100871 Beijing, China.

‡Center for Applied Statistics and School of Statistics, Renmin University of China, 100872 Beijing, China.

1 Introduction

It is well known that insurance loss data sometimes cannot be well modeled by a single distribution due to its complex data structures. For instance, Poisson distribution may not fit the claim count data well because of an excess of zero claims. Claim amount data usually exhibit multimodal or heavy-tailed phenomenons, so a gamma distribution is insufficient to address these data features. In statistical analysis, one alternative way is to apply a mixture of distributions for univariate loss data. When the individual risk factors are available, a mixture of regressions is then a natural extension to modeling the risk heterogeneity, which is captured by the individual risk factors. In the literature, the mixture model is proposed by Goldfeld and Quandt (1973) which serves as the basis of a Markov-switching regression model. We refer to Lindsay (1995) and McLahlan and Peel (2000) for a detailed review of mixture models. In the field of machine learning research, a mixture model is often called a mixture of experts model, which is an ensemble learning technique that implements the idea of training experts on subtasks of a predictive modeling problem (Jacobs et al., 1991; Jiang and Tanner, 1999).

Some recent studies in modeling insurance loss also focus on applying mixture models to deal with complicated data structures. Zhang et al. (2020, 2022) developed a multivariate zero-inflated hurdle model to describe multivariate count data with extra zeros. Delong et al. (2021) proposed a mixture of neural networks with gamma loss to model insurance claim amounts. Lee and Lin (2010) and Verbelen et al. (2015) used a mixture of Erlangs to model insurance claim amounts including censored and truncated data. Lee and Lin (2012) developed the multivariate version of a mixture of Erlangs. Fung et al. (2019a), Fung et al. (2019b) and Tseung et al. (2021) proposed a class of so called logit-weighted reduced mixture of experts (LRMoE) 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 (GLM) framework. In this paper, we will relax this constraint by estimating regression functions non-parametrically.

Several aspects of modeling are very noteworthy in research. First, it is challenging to estimate parameters in mixture models since the parameters in component models are related to each other. To overcome it, Dempster et al. (1977) proposed the Expectation-Maximization (EM) algorithm, which is an iterative method to estimate the component parameters and the conditional expectation of latent component indicator variables. Although the EM algorithm becomes a standard approach for estimation of mixture models, it is hard to address other practical issues, for example, the feature engineering and the variable selection, both of which are also challenging since the dependence between the covariates and response variables varies from one component to another. Khalili and Chen (2007) introduced a penalized likelihood approach for the variable selection. Huang and Yao (2012) and Huang et al. (2013) proposed a non-parametric mixture of regressions to relax the linearity assumption on the regression function. Moreover, selecting the number of the components is another challenging problem. Naik et al. (2007) derived a new information criterion for the selection of the component number. 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 feature engineering and variable selection in the proposed method.

We propose an Expectation-Boosting (EB) algorithm for mixture models, which replaces the maximization step in the EM algorithm with a boosting step. It is well known that the boost-

ing algorithm is an iterative method to estimate the regression function non-parametrically, where weak learners are calibrated step-wisely and added together. Commonly used boosting algorithms include AdaBoost, LogitBoost (real AdaBoost), stagewise additive modeling using a multi-class exponential loss (SAMME), multi-class real AdaBoost, gradient boosting machine (GBM), etc. A class of boosting algorithms is based on gradient of loss functions including gradient boosting machine (GBM), Newton boosting, gradient boosting decision tree (GBDT), eXtreme Gradient Boosting (XGBoost), light gradient boosting machine (LightGBM), etc. Friedman (2001) has studied the gradient based boosting from a statistical perspective as an additive model.

Our EB algorithm follows the GBDT algorithm in the boosting step. With the negative log-likelihood function as the loss function for boosting, the boosting step increases the likelihood at each iteration of the EB algorithm. There are several advantages of the EB algorithm over the EM algorithm. First, the boosting algorithm is a flexible non-parametric regression method, which facilitates both non-linear effects and interaction of covariates. The feature engineering and variable selection are performed automatically during the model fitting. Second, the boosting algorithm is overfitting-sensitive given suitable regularization parameters (Bühlmann and Yu, 2003). This property is beneficial since the ultimate goal of insurance loss modeling is to predict future losses. Therefore, the out-of-sample prediction is more important than the goodness-of-fitting in our applications.

The rest of the paper is structured as follows. Section 2 reviews mixture models and the EM algorithm. Section 3 presents the EB algorithm. We also discuss interpretation, hyperparameters tuning, initialization and parallel computation in this section. Section 4 studies two simulated data and one real data to illustrate the proposed methodology. Section 5 concludes the paper with several important findings. The R code of our work is publicly available on the github (`https://github.com/sxpyggy/boosting-mixture-code`).

2 Mixture models and the EM algorithm revisited

In this section, we first review the mixture model, particularly a finite mixture of regressions (McLahlan and Peel, 2000), and discuss different mixture structures. Then we review the EM algorithm, which is the foundation of the proposed EB algorithm. In this paper, we focus on the exponential distribution family with an expectation parameter μ and a dispersion parameter ϕ. The exponential distributions form a general family of parametric distributions, which fit well different types of insurance loss data.

2.1 Mixture models

Suppose a random variable Y follows a finite mixture of distributions with the k-th *component distribution* from a finite collection of $K \geq 2$ distributions

$$\{f_1(Y;\mu_1,\phi_1),\ldots,f_K(Y;\mu_K,\phi_K)\}.$$

The mixing probabilities $\boldsymbol{p}=(p_1,\ldots,p_K)$ lie in the $(K-1)$-unit simplex $\mathcal{P}\subset(0,1)^K$, i.e. have normalization $\sum_{k=1}^K p_k=1$. The probability density function for Y is given by a weighted sum

of component distributions such that

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

If the J-dimensional *individual feature* vector $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^J$ is available and has a systematic effect on the response variable Y, then we can establish a finite mixture of regressions:

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

where we assume the parametric models of $p : \mathcal{X} \to \mathcal{P}, \mu_k : \mathcal{X} \to \mathbb{R}, \phi_k : \mathcal{X} \to \mathbb{R}_+$ such that

$$\begin{aligned} p(\boldsymbol{x}) &= (p_1(\boldsymbol{x}), \ldots, p_K(\boldsymbol{x})) = (p_1(\boldsymbol{x}; \theta_p), \ldots, p_K(\boldsymbol{x}; \theta_p)) \in \mathcal{P}, \\ \mu_k(\boldsymbol{x}) &= \mu_k(\boldsymbol{x}; \theta_\mu^{(k)}) \in \mathbb{R}, \quad \text{for} \quad k = 1, \ldots, K, \\ \phi_k(\boldsymbol{x}) &= \phi_k(\boldsymbol{x}; \theta_\phi^{(k)}) \in \mathbb{R}_+, \quad \text{for} \quad k = 1, \ldots, K. \end{aligned}$$

Here $\theta_p, (\theta_\mu^{(k)})_{k=1:K}, (\theta_\phi^{(k)})_{k=1:K}$ are the unknown parameters. Note that θ_p does not have a superscript since it corresponds to the coefficients in a multinomial logistic classification while $\theta_\mu^{(k)}, \theta_\phi^{(k)}$ can be considered in separate component models; see Section 2.2.

Equation (2.1) is a general representation for a finite mixture of regressions, i.e., both mixing probabilities and component parameters depend on $\boldsymbol{x}$. In real applications, based on the features of data and the goal of modeling, we can make specific assumptions on (2.1). For example,

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

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

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

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

- If the component distributions are from the same distribution family f, we might assume a single dispersion ϕ

$$f(Y|\boldsymbol{x}) = \sum_{k=1}^{K} p_k(\boldsymbol{x}) f(Y|\boldsymbol{x}; \mu_k(\boldsymbol{x}), \phi); \tag{2.2}$$

- The feature vector $\boldsymbol{x}$ is only related to the mixing probabilities

$$f(Y|\boldsymbol{x}) = \sum_{k=1}^{K} p_k(\boldsymbol{x}) f_k(Y; \mu_k, \phi_k). \tag{2.3}$$

Selection among the above model alternatives can be viewed as imposing suitable constraints on the general form (2.1), which accelerates and stabilizes model fitting without compromising predictive performance. The LRMoE proposed by Fung et al. (2019b) uses (2.3). In Section 4, the first simulated data follows (2.1) with $K = 2$. The second simulated data follows (2.2) with $K = 3$. The real data example uses (2.3) with $K = 5$.

2.2 The EM algorithm

Suppose that for a single sample, we know exactly which component distribution it 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. Then the joint distribution function of $(Y, \boldsymbol{Z})$ given $\boldsymbol{x}$ is

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

or equivalently, the complete log-likelihood function is

$$\sum_{k=1}^{K} Z_k \left[\log p_k(\boldsymbol{x}) + \log f_k(Y|\boldsymbol{x}; \mu_k(\boldsymbol{x}), \phi_k(\boldsymbol{x}))\right]. \tag{2.4}$$

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

$$\hat{\theta}_p = \arg\max_{\theta_p} \sum_{i=1}^{n} \sum_{k=1}^{K} z_{i,k} \log p_k(\boldsymbol{x}_i; \theta_p) \tag{2.5}$$

and

$$\left(\hat{\theta}_\mu^{(k)}, \hat{\theta}_\phi^{(k)}\right) = \arg\max_{\theta_\mu^{(k)}, \theta_\phi^{(k)}} \sum_{i=1}^{n} \sum_{k=1}^{K} z_{i,k} \log f_k\left(y_i; \mu_k\left(\boldsymbol{x}_i; \theta_\mu^{(k)}\right), \phi_k\left(\boldsymbol{x}_i; \theta_\phi^{(k)}\right)\right), \text{ for } k = 1, \ldots, K. \tag{2.6}$$

The optimizations (2.5) and (2.6) are corresponding to *a multinomial logistic classification* and K *regressions*, respectively. The multinomial logistic classification (2.5) is fitted to all samples, while the k-th regression in (2.6) is fitted to partial samples with $\{i : z_{i,k} = 1\}$. In practice we do not have the 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 parameters $\hat{\theta}_p, (\hat{\theta}_\mu^{(k)})_{k=1:K}, (\hat{\theta}_\phi^{(k)})_{k=1:K}$, 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 log-likelihood function for full information $(y_i, \hat{\boldsymbol{z}}_i, \boldsymbol{x}_i)_{i=1:n}$

$$\begin{aligned}
&l\left(\theta_p, (\theta_\mu^{(k)})_{k=1:K}, (\theta_\phi^{(k)})_{k=1:K}\right) \\
=&\sum_{i=1}^{n} \sum_{k=1}^{K} \hat{z}_{i,k} \left[\log p_{i,k} + \log f_k\left(y_i; \mu_{i,k}, \phi_{i,k}\right)\right] \\
=&\sum_{i=1}^{n} \sum_{k=1}^{K} \hat{z}_{i,k} \log p_k(\boldsymbol{x}_i; \theta_p) + \sum_{i=1}^{n} \sum_{k=1}^{K} \hat{z}_{i,k} \log f_k\left(y_i; \mu_k\left(\boldsymbol{x}_i; \theta_\mu^{(k)}\right), \phi_k\left(\boldsymbol{x}_i; \theta_\phi^{(k)}\right)\right),
\end{aligned} \tag{2.7}$$

the MLEs of parameters $\theta_p, (\theta_\mu^{(k)})_{k=1:K}, (\theta_\phi^{(k)})_{k=1:K}$ are given by the following $K+1$ independent optimizations

$$\hat{\theta}_p = \arg\max_{\theta_p} \sum_{i=1}^{n} \sum_{k=1}^{K} \hat{z}_{i,k} \log p_k(\boldsymbol{x}_i; \theta_p), \tag{2.8}$$

and

$$\left(\hat{\theta}_\mu^{(k)}, \hat{\theta}_\phi^{(k)}\right) = \arg\max_{\theta_\mu^{(k)}, \theta_\phi^{(k)}} \sum_{i=1}^{n} \sum_{k=1}^{K} \hat{z}_{i,k} \log f_k\left(y_i; \mu_k\left(\boldsymbol{x}_i; \theta_\mu^{(k)}\right), \phi_k\left(\boldsymbol{x}_i; \theta_\phi^{(k)}\right)\right), \text{ for } k = 1, \ldots, K. \tag{2.9}$$

These optimizations are similar to those in (2.5) and (2.6). The only difference is that $z_{i,k} \in \{0, 1\}$ in (2.5) and (2.6) while $\hat{z}_{i,k} \in (0, 1)$ in (2.8) and (2.9). Therefore, in the maximization step we perform a multinomial logistic classification with *fractional* response $\hat{z}_{i,k}$ and K weighted-regressions fitted to *all* samples with weights $\hat{z}_{i,k}$.

3 Expectation-Boosting algorithm

In the EM algorithm, the functions $p(\boldsymbol{x}), \mu_k(\boldsymbol{x}), \phi_k(\boldsymbol{x})$ are always assumed to be parametric functions. To make it possible for non-parametric modeling, we replace the maximization step in the EM algorithm with a boosting step. The boosting algorithm estimates the regression functions non-parametrically, and thus facilitates more flexible shapes. With negative log-likelihood function as the loss function in the boosting algorithm, the boosting step increases the likelihood at each iteration, but overfitting-sensitively. The boosting step follows a generic functional gradient descent algorithm. In this section, we first describe the generic functional gradient descent algorithm, then propose the EB algorithm, finally discuss interpretation, hyperparameter tuning and initialization in the EB algorithm.

3.1 Generic functional gradient descent algorithm

Suppose a to-be-boosted non-parametric regression function as $F : \mathcal{X} \to \mathbb{R}$. In boosting algorithm, the goal 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*. The expected loss is always replaced by sample average loss:

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

In our setting of statistical modeling, the loss function depends on the corresponding likelihood function. We choose the *negative log-likelihood* function for the loss function. Hence, minimizing the loss function is equivalent to maximizing the likelihood. For example, in LogitBoost the loss function is negative binomial log-likelihood

$$C(Y, F) = \log(1 + \exp(-2YF)), \quad Y \in \{-1, 1\},$$

where

$$F(\boldsymbol{x}) = \frac{1}{2} \log\left(\frac{\Pr[Y = 1|\boldsymbol{x}]}{\Pr[Y = -1|\boldsymbol{x}]}\right). \tag{3.1}$$

In L_2Boost, the loss function is negative Gaussian log-likelihood

$$C(Y, F) = (Y - F)^2/2, \quad Y \in \mathbb{R}, \tag{3.2}$$

where $F(\boldsymbol{x}) = \mathbb{E}(Y|\boldsymbol{x})$.

In the GLM framework, the link function $g : \mathcal{Y} \to \mathbb{R}$ is a mapping from the parameters of interest (e.g. mean or probability odds) to the linear predictor. Here, we apply the same terminology where the link function is a mapping from the parameters of interest to the to-be-boosted non-parametric regression function F. In LogitBoost, equation (3.1) implies a half logged probability odds link function $g : \mathbb{R}_+ \to \mathbb{R}$

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

In L_2Boost, we have an identify link function $g : \mathbb{R} \to \mathbb{R}$

$$g(\text{mean}) = \text{mean} = \mathbb{E}(Y|\boldsymbol{x}) = F(\boldsymbol{x}).$$

In our setting of statistical modeling, we first specify the likelihood function and the link function, then derive the loss function.

We follow Bühlmann and Yu (2003) to briefly describe the generic functional gradient descent algorithm in Algorithm 1, which will serve as building blocks in the proposed EB algorithm. The algorithm estimates F in an additive form

$$\hat{F}(\boldsymbol{x}) = \hat{F}^{[0]}(\boldsymbol{x}) + \sum_{m=1}^{M} \hat{f}^{[m]}(\boldsymbol{x}),$$

where $\hat{F}^{[0]}$ is the initial value and $\hat{f}^{[m]}$ is called the m-th weak learner which involves only a small number of covariates.

In the algorithm, weak learners are fitted to a negative gradient U rather than Y. The loss function in weak learners is always L_2, independently with model loss function C. If decision trees (Breiman et al., 1984) are used as weak learners, the algorithm is called *gradient boosting decision tree* (GBDT), where calibration of decision trees and variable selection are performed simultaneously. One may refer to Hastie et al. (2009) for decision trees and the recursive partitioning algorithm. Note that Bühlmann and Hothorn (2007) argues that step 4 of one-dimensional optimization seems unnecessary given the learning rate s is sufficiently small according to some empirical experiments.

3.2 Expectation-Boosting algorithm

In the EB algorithm, we follow the same expectation step as in the EM algorithm and replace the maximization step by boosting. The boosting algorithm is applied to estimate mixing probabilities and component parameters given the conditional expectation of latent component indicator variables $\hat{z}_{i,k}$. Before applying a boosting algorithm, we need to specify the likelihood function and the link function, which determine the loss function. In mixture models, the log-likelihood function is already given by (2.7). Therefore, we only need to specify the link functions for mixing probabilities, component means and component dispersions, respectively.

Algorithm 1 The generic functional gradient decent algorithm.

1: Initialization. Set a suitable initial value $\hat{F}^{[0]}(\boldsymbol{x})$.

2: **for** $m = 1$ to M **do**

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

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

Data $(u_i, \boldsymbol{x}_i)_{i=1:n}$ is used to calibrate a weak learner $\hat{f}^{[m]}(\boldsymbol{x}; \hat{\theta}^{[m]})$ with *loss function* L_2 (3.2).

4: One-dimensional optimization and update. Solve a one-dimensional optimization to find *expansion coefficient* $\hat{\omega}^{[m]}$:

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

Update

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

where s is *shrinkage factor* (learning rate).

5: **end for**

6: **return** $\hat{F}(\boldsymbol{x}) = \hat{F}^{[M]}$.

For mixing probabilities $p_1(\boldsymbol{x}), \ldots, p_K(\boldsymbol{x})$, they are mapped to the to-be-boosted regression functions $F_1(\boldsymbol{x}), \ldots, F_K(\boldsymbol{x})$ by the multiple logistic link function $g_k : \mathcal{P} \to \mathbb{R}$ such that

$$g_k(p_1(\boldsymbol{x}), \ldots, p_K(\boldsymbol{x})) = \log p_k(\boldsymbol{x}) - \frac{1}{K} \sum_{l=1}^{K} \log p_l(\boldsymbol{x}) = F_k(\boldsymbol{x}), \quad k = 1, \ldots, K. \tag{3.3}$$

or equivalently

$$g_k^{-1}(F_1(\boldsymbol{x}), \ldots, F_K(\boldsymbol{x})) = \frac{\exp(F_k(\boldsymbol{x}))}{\sum_{l=1}^{K} \exp(F_l(\boldsymbol{x}))} = p_k(\boldsymbol{x}), \quad k = 1, \ldots, K. \tag{3.4}$$

Given the negative log-likelihood function (2.8) of multinomial distribution, we derive the loss function as

$$C_0((Z_k, F_k(\boldsymbol{x}))_{k=1:K}) = -\sum_{k=1}^{K} Z_k \log p_k(\boldsymbol{x}). \tag{3.5}$$

The negative gradient of the loss function C_0 w.r.t. F_k is given by

$$U_k((Z_k, F_k(\boldsymbol{x}))_{k=1:K}) = -\frac{\partial C_0((Z_k, F_k(\boldsymbol{x}))_{k=1:K})}{\partial F_k(\boldsymbol{x})} = Z_k - p_k(\boldsymbol{x}), \quad k = 1, \ldots, K. \tag{3.6}$$

For the component parameters μ and ϕ, we denote the to-be-boosted regression functions by $G(\boldsymbol{x})$ and $H(\boldsymbol{x})$, respectively. The link functions depend on the component models. For example, in a component Gaussian model we assume $\mu_k(\boldsymbol{x}) = G_k(\boldsymbol{x}), \log \phi_k(\boldsymbol{x}) = H_k(\boldsymbol{x})$; in a component Poisson model $\log \mu_k(\boldsymbol{x}) = G_k(\boldsymbol{x})$; in a component gamma model $\log \mu_k(\boldsymbol{x}) =$

$G_k(\boldsymbol{x}), \log \phi_k(\boldsymbol{x}) = H_k(\boldsymbol{x})$. With the negative log-likelihood function (2.9), we derive the loss function as

$$C_k(Y, Z_k, G_k(\boldsymbol{x})) = -Z_k \log f_k(Y; \mu_k(\boldsymbol{x}), \phi_k),\ k = 1, \ldots, K, \tag{3.7}$$

where Z_k is treated as a weight multiplying to the usual negative log-likelihood function $-\log f_k$. The negative gradient of loss function C_k w.r.t. G_k is given by

$$V_k(Y, Z_k, G_k(\boldsymbol{x})) = -\frac{\partial C_k(Y, Z_k, G_k(\boldsymbol{x}))}{\partial G_k(\boldsymbol{x})}. \tag{3.8}$$

Note that we have assumed that the dispersion ϕ_k is fixed among samples (not related to $\boldsymbol{x}$). We will discuss dispersion modeling later. Upon the above defined notations, we are ready to introduce the proposed EB algorithm. We first summarize the proposed EB algorithm in Algorithm 2.

Algorithm 2 The Expectation-Boosting algorithm.

1: Initialization. Set $\hat{p}_k^{[0]}, \hat{\mu}_k^{[0]}, \hat{\phi}_k^{[0]}$, $k = 1, \ldots, K$.
2: **for** $t = 1$ to T **do**
3: Expectation. Calculate $\hat{z}_{i,k}^{[t]}$ given $\hat{p}_k^{[t-1]}, \hat{\mu}_k^{[t-1]}, \hat{\phi}_k^{[t-1]}$, $k = 1, \ldots, K$.
4: Initialization of mixing probabilities. Set $\hat{p}_k^{[t,0]}, \hat{F}_k^{[t,0]}$, $k = 1, \ldots, K$.
5: **for** $m = 1$ to M_0 **do**
6: **for** $k = 1$ to K **do**
7: Projection of gradient to weak learner. Calibrate a weak learner $\hat{f}_k^{[t,m]}$.
8: One dimensional optimization and update. Update $\hat{F}_k^{[t,m]}$ using $\hat{F}_k^{[t,m-1]}, \hat{f}_k^{[t,m]}$.
9: **end for**
10: **return** $\hat{p}_k^{[t,m]}$ according to (3.4), $k = 1. \ldots, K$.
11: **end for**
12: **return** $\hat{F}_k^{[t]} = \hat{F}_k^{[t,M_0]}, \hat{p}_k^{[t]} = \hat{p}_k^{[t,M_0]}$, $k = 1, \ldots, K$.
13: Initialization of component means. Set $\hat{\mu}_k^{[t,0]}, \hat{G}_k^{[t,0]}$, $k = 1, \ldots, K$.
14: **for** $k = 1$ to K **do**
15: **for** $m = 1$ to M_k **do**
16: Projection of gradient to weak learner. Calibrate a weak learner $\hat{g}_k^{[t,m]}$.
17: One dimensional optimization and update. Update $\hat{G}_k^{[t,m]}, \hat{\mu}_k^{[t,m]}$ using $\hat{G}_k^{[t,m-1]}, \hat{g}_k^{[t,m]}$.
18: **end for**
19: **return** $\hat{G}_k^{[t]} = \hat{G}_k^{[t,M_k]}, \hat{\mu}_k^{[t]} = \hat{\mu}_k^{[t,M_k]}$.
20: **end for**
21: **return** $\hat{G}_k^{[t]}, \hat{\mu}_k^{[t]}, k = 1, \ldots, K$.
22: Estimation of dispersion. Calculate the MLE $\hat{\phi}_k^{[t]}$ given $\hat{\mu}_k^{[t]}$ and $\hat{z}_k^{[t]}$, $k = 1, \ldots, K$.
23: **end for**
24: **return** $\hat{p}_k^{[T]}, \hat{\mu}_k^{[T]}, \hat{\phi}_k^{[T]}$, $k = 1, \ldots, K$.

In Algorithm 2, we have a hierarchical loop: the outer EB iteration lines 2-23 and the inner boosting iteration lines 5-11 and 14-20, respectively. The loop of inner boosting iteration follows a similar course of Algorithm 1, which is nested inside the loop of outer EB iteration. Apparently, the EB algorithm takes more computation time than the EM algorithm. However, the EB algorithm can be accelerated by parallel computing. In lines 6-9, the functions $(F_k)_{k=1:K}$

are boosted independently. In lines 14-20, the component means are boosted independently. Thus, those independent steps can be conducted simultaneously to reduce running time; see Table 2. Notice that the two loops in lines 14-20 and 15-18 are exchangeable, since $\hat{\mu}_k^{[t,m]}$ only depends on $\hat{G}_k^{[t,m]}$. However, the two loops in lines 5-11 and 6-9 are not exchangeable since $\hat{p}_k^{[t,m]}$ depends on all $(\hat{F}_k^{[t,m]})_{k=1:K}$ via equation (3.4). In the EB algorithm, we have assumed that dispersion ϕ_k is fixed among samples (not related to $\boldsymbol{x}$). The extension to joint modeling of mean and dispersion is not straightforward since they are not independent; please refer to Jorgensen (1997) for dispersion modeling. Joint modeling of both mean and dispersion is not considered in this paper.

In the following, we present more details for each step of the EB algorithm in Algorithm 2.

1. Initialization of EB algorithm (line 1). Set $\hat{p}_k^{[0]}, \hat{F}_k^{[0]}, k = 1, \ldots, K$ as:

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

Set $\hat{\mu}_k^{[0]}, \hat{G}_k^{[0]}, k = 1, \ldots, K$, as:

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

where μ_k^{-1} denotes the link function in the component model k. Set $\hat{\phi}_k^{[0]}, k = 1, \ldots, K$ as the sample variance.

2. Expectation (line 3). We can calculate the conditional expectation of latent component indicator variable given $\hat{p}_k^{[t-1]}, \hat{\mu}_k^{[t-1]}, \hat{\phi}_k^{[t-1]}$, $k = 1, \ldots, K$:

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

Note that the conditional expectation $\hat{z}_{i,k}^{[t]}$ carries the information obtained from the last EB iteration $t-1$, which then interacts with the boosting via the negative gradients (3.6) and (3.8) and the loss functions (3.5) and (3.7) at this EB iteration t.

3. Initialization of mixing probabilities (line 4). Set $\hat{p}_k^{[t,0]}, \hat{F}_k^{[t,0]}$, $k = 1, \ldots, K$ as

$$\begin{aligned} \hat{p}_k^{[t,0]} =& \frac{1}{K}, \ k = 1, \ldots, K, \\ \hat{F}_k^{[t,0]} =& \log \hat{p}_k^{[t,0]} - \frac{1}{K} \sum_{l=1}^{K} \log \hat{p}_l^{[t,0]}, \ k = 1, \ldots, K. \end{aligned} \tag{3.9}$$

In contrast to the initialization in line 1, the current one will be repeated at each outer EB iteration t. Later we will discuss another initialization strategy which considers the boosted values at the previous EB iteration.

4. Projection of gradient to weak learner (line 7). We compute the negative gradient for each data point $i = 1, \ldots, n$ as

$$u_{i,k}^{[t,m]} = U_k \left(\left(\hat{\boldsymbol{z}}_{i,k}^{[t]}, \hat{F}_k^{[t,m-1]}(\boldsymbol{x}_i) \right)_{k=1:K} \right) = \hat{z}_{i,k}^{[t]} - \hat{p}_{i,k}^{[t,m-1]}.$$

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

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

where $R_{k,l=1:L}^{[t,m]}$ is a partition of covariate space $\mathcal{X} \subset \mathbb{R}^J$ and $\bar{u}_{k,l=1:L}^{[t,m]}$ contains the average gradient in each terminal node. Note that for different k, the regression trees can have different numbers of terminal nodes. Here we assume L-terminal node to simplify the notation.

5. One dimensional optimization and update (line 8). It turns out that the one-dimensional optimization for expansion coefficient leads to the following update:

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

where s_p is the learning rate and

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

Please refer to equation (32) in Friedman (2001). Note that for different k, the learning rate can be different.

6. Return $\hat{p}_k^{[t,m]}$ according to (3.4)**, $k = 1, \ldots, K$ (line 10)**. The mixing probabilities are updated as

$$\hat{p}_k^{[t,m]}(\boldsymbol{x}) = \frac{\exp\left(\hat{F}_k^{[t,m]}(\boldsymbol{x})\right)}{\sum_{l=1}^{K} \exp\left(\hat{F}_l^{[t,m]}(\boldsymbol{x})\right)}, \quad k = 1, \ldots, K.$$

Lines 4-12 are a typical K-class logistic gradient boosting. In lines 6-9, K independent boostings are performed for $(F_k)_{k=1:K}$. Hence, we can accelerate lines 6-9 by parallel computing. To the best of our knowledge, there is no available R package supporting a multinomial logistic boosting with fractional response $\hat{\boldsymbol{z}}_i \in \mathcal{P} \subset (0,1)^K$. So we code lines 4-12 from bottom. Only the R package `rpart` is called to apply the recursive partition algorithm to calibrate the weak learner $\hat{f}_k^{[t,m]}$ in line 7.

7. Initialization of component means (line 13). Set $\hat{\mu}_k^{[t,0]}, \hat{G}_k^{[t,0]}$, $k = 1, \ldots, K$ as:

$$\hat{\mu}_k^{[t,0]} = \frac{\sum_{i=1}^{n} y_i}{n}, \; \hat{G}_k^{[t,0]} = \mu_k^{-1}(\hat{\mu}_k^{[t,0]}), k = 1, \ldots, K, \tag{3.10}$$

where μ_k^{-1} denotes the link function in the component model k. Similar to line 4, this initialization will be repeated at each outer EB iteration t. Later we will discuss another initialization strategy which considers the boosted values at the previous EB iteration.

8. Projection of gradient to learner (line 16). We compute the negative gradient for each data point $i = 1, \ldots, n$ as

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

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

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

where $S_{k,l=1:L}^{[t,m]}$ is the partition of covariate space $\mathcal{X} \subset \mathbb{R}^J$ and $\bar{v}_{k,l=1:L}^{[t,m]}$ contains the average gradient in each terminal node. Note that the number of terminal nodes L can be different for different component models.

9. One dimensional optimization and update (line 17). We conduct the following one-dimensional optimizations to find the best expansion coefficients:

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

And then we update

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

where s_μ is the learning rate. Note that for different component models the learning rates can be different. Update the component mean as

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

where μ_k denotes the inverse link function in component model k.

Lines 13-21 implement K independent weighted gradient boostings with weights $\hat{z}_{i,k}^{[t]}$ for each component mean. We can accelerate this loop by conducting the K boostings simultaneously. We rely on the R packages `gbm` or `mboost` to perform the boosting in this loop. Those two packages can facilitate boosting with additional weights.

10. Estimation of dispersion (line 22). The MLE $\hat{\phi}_k^{[t]}$ is derived given the conditional expectation $\hat{z}_{i,k}^{[t]}$ and the component mean $\hat{\mu}_k^{[t]}$:

$$\hat{\phi}_k^{[t]} = \arg\max_{\phi_k} \sum_{i=1}^n \hat{z}_{i,k}^{[t]} \log f_k(y_i; \hat{\mu}_k^{[t]}(\boldsymbol{x}_i), \phi_k), \ k = 1, \ldots, K. \tag{3.11}$$

In summary, a total of $(KM_0 + M_1 + \ldots, +M_K) \times T$ weak learners are calibrated in the EB algorithm. In lines 6-9, $(F_k)_{k=1:K}$ can be boosted simultaneously at each m. Furthermore, in lines 14-20, the K boosting algorithms can be conducted simultaneously at each t. Such parallel computing will accelerate the EB algorithm significantly; see Table 2.

Gradient boosting with regression trees inherits the desirable features of trees. One of the advantages is automated feature engineering with regard to non-linearity, variable transformation, and interaction. First, a tree produces a piecewise constant function facilitating non-linear effects of a covariate. Second, trees are invariant under monotone transformations of the individual covariate. For example, using $x_j, \log x_j, \exp x_j$ as the j-th covariate leads to the same result. So there is no need to consider individual variable transformation. Third, a tree with L terminal nodes produces a function with interaction order at most $L - 1$. Another advantage of trees is internal variable selection. Trees tend to be robust against irrelevant covariates. Moreover, trees are overfitting-sensitive given a suitable complexity parameter, i.e., any split not sufficiently improving the fit will not be attempted.

The boosting also mitigates many undesirable features of the single tree model. Small trees are inaccurate due to the coarse nature of the piecewise constant functions. In contrast, large trees are not stable due to the high-order interaction involved. All of these are mitigated by boosting, which produces much finer piecewise constant functions; it enhances stability by using small trees and averaging over many of them. One remarkable feature of boosting is the

resistance to overfitting, which owes to overfitting-sensitive split in trees and two regularization hyperparameters of the learning rate s and the number of iterations M. We usually specify a large M and early stop iterating by minimizing the validation loss before reaching M. For a sufficiently small learning rate, we usually do not observe a typical tick shape of validation loss but a leveling-off pattern; see Figure 1 in Friedman (2001).

3.3 Interpretation, hyperparameter tuning and initialization in the EB algorithm

A particularly useful interpretative tool for boosting is relative importance proposed by Breiman et al. (1984). In mixture models, we consider the relative importance of covariates in each component and the overall relative importance. Considering the weak learner of tree $\hat{f}_k^{[T,m]}$ at the last EB iteration T, the relative importance of covariate x_j is

$$I_j(\hat{f}_k^{[T,m]}) = \sum_{l=1}^{L-1} e_l \mathbb{1}_{d_l}(j), \quad j = 1, \ldots, J, \tag{3.12}$$

where the summation is over the non-terminal nodes l of the L-terminal node tree $\hat{f}_k^{[T,m]}$, d_l is the splitting variable associated with node l, and e_l is the corresponding empirical improvement in squared error as a result of the split given by

$$e_l = \frac{w_{\text{left}} w_{\text{right}}}{w_{\text{left}} + w_{\text{right}}} \times (m_{\text{left}} - m_{\text{right}})^2.$$

Here, m_{left} and m_{right} are the gradient means in the left and right splits, respectively, and w_{left} and w_{right} are the corresponding sums of the weights. Considering a collection of trees $(\hat{f}_k^{[T,m]})_{m=1:M_0}$ in the k-th mixing probability $\hat{p}_k^{[T]}$, (3.12) can be generalized by its average over all of the trees

$$IP_{j,k} = \frac{1}{M_0} \sum_{m=1}^{M_0} I_j(\hat{f}_k^{[T,m]}), \quad j = 1, \ldots, J, \tag{3.13}$$

which is interpreted as the relative importance of x_j in separating component k from the other components. The overall relative importance of x_j is obtained by averaging over all components:

$$IP_j = \frac{1}{K} \sum_{k=1}^{K} IP_{j,k}, \quad j = 1, \ldots, J.$$

The above formulae are about the relative importance of covariates in mixing probabilities. Similarly, we can obtain the relative importance of covariates in each component mean as

$$IM_{j,k} = \frac{1}{M_k} \sum_{m=1}^{M_k} I_j(\hat{g}_k^{[T,m]}), \quad j = 1, \ldots, J, \tag{3.14}$$

and the overall relative importance of covariates in means as

$$IM_j = \frac{1}{K} \sum_{k=1}^{K} IM_{j,k}, \quad j = 1 \ldots, J.$$

The EB algorithm has four types of hyperparameters: the number of EB iterations T, the numbers of boosting iterations M_0 and $(M_k)_{k=1:K}$, the learning rates s_p and s_μ, and hyperparameters in weak learners. Although there are methods to systematically tune hyperparameters, such as grid research and Bayesian optimization, tuning hyperparameters in many practical algorithms is more art than a science. In predictive modelling with hyperparameters, we usually split the data into three mutually exclusive sets, training set, validation set and test set, whose indices are denoted by $\mathcal{I}_{\text{train}}$, $\mathcal{I}_{\text{val}}$ and $\mathcal{I}_{\text{test}}$, respectively.

The number of EB iterations T can be tuned by screening the trace plot of the log-likelihood on the learning data $\mathcal{I}_{\text{learn}} = \mathcal{I}_{\text{train}} \cup \mathcal{I}_{\text{val}}$ at each iteration t:

$$\sum_{i \in \mathcal{I}_{\text{learn}}} \log \left(\sum_{k=1}^{K} \hat{p}_k^{[t]}(\boldsymbol{x}_i) f_k \left(y_i; \hat{\mu}_k^{[t]}(\boldsymbol{x}_i), \phi_k^{[t]} \right) \right).$$

If there is no noticeable improvement, the EB algorithm converges.

For the numbers of boosting iterations $M_0, (M_k)_{k=1:K}$, we specify a sufficiently large maximal iteration number and calculate the following $K+1$ validation losses on $\mathcal{I}_{\text{val}}$ at each iteration m:

$$-\sum_{i \in \mathcal{I}_{\text{val}}} \sum_{k=1}^{K} \hat{z}_{i,k}^{[t]} \log \hat{p}_k^{[t,m]}(\boldsymbol{x}_i), \tag{3.15}$$

$$-\sum_{i \in \mathcal{I}_{\text{val}}} \hat{z}_{i,k}^{[t]} \log f_k \left(y_i; \hat{\mu}_k^{[t,m]}(\boldsymbol{x}_i), \hat{\phi}_k^{[t]} \right), \ k = 1, \ldots, K. \tag{3.16}$$

If there is no noticeable improvement of the validation loss, the corresponding boosting can be early stopped before reaching the maximal iteration number.

For the learning rates s_p and s_μ, lower learning rates tend to lead to a better fitting and predictive performance but require more boosting iterations. We adjust the learning rates by monitoring the validation loss (3.15) and (3.16), respectively. Note that we can specify different learning rates s_μ for different component means.

The hyperparameters in weak learners of regression trees include complexity parameters, maximum depth, number of terminal nodes, etc. Those hyperparameters can be tuned as in typical regression trees. One may refer to Hastie et al. (2009) for more details.

In lines 4 and 13 of the initialization of boosting, we initialize parameters at iteration t independently with the previously boosted estimates $\hat{p}_k^{[t-1]}, \hat{\mu}_k^{[t-1]}$; see equations (3.9) and (3.10). We call this *uncorrelated boosting*. However, one may spot that boosting at iteration t is not independent with boosting at iteration $t-1$ since boosting at iteration t is based on $\hat{\boldsymbol{z}}_i^{[t]}$ which depends on the boosted results at iteration $t-1$. The gains from the boosting at iteration $t-1$ are passed to the boosting at iteration t via the conditional expectation $\hat{\boldsymbol{z}}_i^{[t]}$.

We might initialize parameters at iteration t as the previously boosted estimates

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

It would lead to fewer inner boosting iterations $M_0, (M_k)_{k=1:K}$ (i.e., an earlier stop on the validation loss) since the boosting starts from better initial values. We call this *forward boosting*. However, with forward boosting, it is impossible to predict for new data using the fitted model at the last iteration $(\hat{p}_k^{[T]}, \hat{\mu}_k^{[T]}, \hat{\phi}_k^{[T]})_{k=1:K}$ due to the iterative initialization. One solution is to apply an additional boosting step with default initialization (3.9) and (3.10) given the last conditional expectations of the latent variables $\hat{\boldsymbol{z}}_i^{[T]}$. In the following applications, we only implement uncorrelated boosting.

4 Applications

In this section, we conduct two simulation studies and one real data analysis. The first simulated data follows a zero-inflated Poisson model, which mimics claim count data with an excess of zeros. We use this example to illustrate the advantages of the EB algorithm over the EM algorithm. The second simulated data follows a mixture of Gaussian models. We compare different mixing structures and investigate parallel computing in this example. The real data contains the claims amount data of an insurance company. This example demonstrates how to choose component models and mixing structures in practice.

4.1 First simulated example: zero-inflated Poisson model

We consider a simulated data set generated from a zero-inflated Poisson (ZIP) model. This simulated data mimics the numbers of claims with an excess of zeros. The underlying model is given by:

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

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

$$f_{\text{ZIP}}(N;\lambda,\pi_0) = \pi_0 \mathbb{1}_0(N) + (1-\pi_0)\frac{e^{-\lambda}\lambda^N}{N!}.$$

Suppose that five covariates $\boldsymbol{x} = (x_1,\ldots,x_5)$ have systematic effects on both π_0 and λ:

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

where

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

The covariates are generated from the following distributions:

$$\begin{aligned} x_1 &\sim N(0, 0.5^2), \\ x_2 &\sim U(0,1), \\ x_3 &\sim \Gamma(2, 0.5), \\ x_4 &\sim Bernulli(0.5), \\ x_5 &\sim Bernulli(0.2). \end{aligned}$$

Note that we use shape-rate parameters for gamma distribution Γ. We generate for $n = 10,000$ samples $(N_i, \boldsymbol{x}_i)_{i=1:n}$, which are partitioned into a learning set $i \in \mathcal{I}_{\text{learn}}$ of sample size $8,000$ and a test set $i \in \mathcal{I}_{test}$ of sample size $2,000$. If a Poisson distribution is fitted to the learning data, the MLE of Poisson mean is $1/|\mathcal{I}_{\text{learn}}| \sum_{i\in\mathcal{I}_{\text{learn}}} N_i = 0.2136$, implying the estimated proportion of zero claims as $\exp(-0.2136) = 80.77\%$ less than the empirical proportion 83.10% in the test data.

We will compare different models in terms of test loss defined as the average negative log-likelihood on the test data:

$$-\frac{1}{|\mathcal{I}_{\text{test}}|}\sum_{i\in\mathcal{I}_{\text{test}}}\log\left(\hat{\pi}_0(\boldsymbol{x}_i)\mathbb{1}_0(N_i)+(1-\hat{\pi}_0(\boldsymbol{x}_i))\frac{e^{-\hat{\lambda}(\boldsymbol{x}_i)}\hat{\lambda}(\boldsymbol{x}_i)^{N_i}}{N_i!}\right), \tag{4.1}$$

where $\hat{\pi}_0$ and $\hat{\lambda}$ are estimated on the learning data. Since we know the true underlying model, we define another two test metrics for π and λ in terms of F and G, respectively:

$$e_\pi = \frac{1}{|\mathcal{I}_{\text{test}}|}\sum_{i\in\mathcal{I}_{\text{test}}}(\hat{F}(\boldsymbol{x}_i)-F(\boldsymbol{x}_i))^2, \tag{4.2}$$

$$e_\lambda = \frac{1}{|\mathcal{I}_{\text{test}}|}\sum_{i\in\mathcal{I}_{\text{test}}}(\hat{G}(\boldsymbol{x}_i)-G(\boldsymbol{x}_i))^2. \tag{4.3}$$

We start with a null ZIP model without any covariates:

$$F(\boldsymbol{x})=\alpha_0,\quad G(\boldsymbol{x})=\beta_0. \tag{4.4}$$

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

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

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

$$-\frac{1}{|\mathcal{I}_{\text{test}}|}\sum_{i\in\mathcal{I}_{\text{test}}}\log\left(\pi_0(\boldsymbol{x}_i)\mathbb{1}_0(N_i)+(1-\pi_0(\boldsymbol{x}_i))\frac{e^{-\lambda(\boldsymbol{x}_i)}\lambda(\boldsymbol{x}_i)^{N_i}}{N_i!}\right)=0.5150, \tag{4.5}$$

which is (surely) smaller than 0.5299 from the null model. The test metrics (4.2) and (4.3) are calculated as $e_\pi = 0.1151, e_\lambda = 0.1795$.

Next ZIP models with linear predictors are fitted, where $F(x)$ and $G(x)$ are assumed to be linear functions of $\boldsymbol{x}$. One component model degenerates into a probability mass 1 at $N = 0$, so there is no coefficient in this component model. We consider three different mixture models, heterogeneous Poisson means only (4.6), heterogeneous mixing probabilities only (4.7) and both heterogeneous mixing probabilities and Poisson means (4.8):

$$F(\boldsymbol{x})=\alpha_0,\quad G(\boldsymbol{x})=\langle\boldsymbol{\beta},\boldsymbol{x}\rangle, \tag{4.6}$$

$$F(\boldsymbol{x})=\langle\boldsymbol{\alpha},\boldsymbol{x}\rangle,\quad G(\boldsymbol{x})=\beta_0, \tag{4.7}$$

$$F(\boldsymbol{x})=\langle\boldsymbol{\alpha},\boldsymbol{x}\rangle,\quad G(\boldsymbol{x})=\langle\boldsymbol{\beta},\boldsymbol{x}\rangle, \tag{4.8}$$

where $\langle\boldsymbol{\beta},\boldsymbol{x}\rangle$ denotes the dot product of two vectors $\boldsymbol{\beta},\boldsymbol{x}$. The MLE of the regression coefficients $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ can be obtained by the EM algorithm. We have summarized the results in Table 1. Model (4.8) has the best out-of-sample performance followed by Model (4.6), while Model (4.7) has the worst out-of-sample performance. All three models are better than the null model (4.4). Note that those models ignore the potential non-linearity and interactions in covariates.

Finally, we estimate F and G non-parametrically via the proposed EB algorithm. We also consider three different mixture models, heterogeneous Poisson means only (4.9), heterogeneous mixing probabilities only (4.10) and both heterogeneous mixing probabilities and Poisson means (4.11):

$$F(\boldsymbol{x}) = \alpha_0, \quad G(\boldsymbol{x}) = G^{[0]}(\boldsymbol{x}) + \sum_{m=1}^{M_\lambda} s_\lambda \omega_\lambda^{[m]} g^{[m]}(\boldsymbol{x}), \tag{4.9}$$

$$F(\boldsymbol{x}) = F^{[0]}(\boldsymbol{x}) + \sum_{m=1}^{M_\pi} s_\pi \omega_\pi^{[m]} f^{[m]}(\boldsymbol{x}), \quad G(\boldsymbol{x}) = \beta_0, \tag{4.10}$$

$$F(\boldsymbol{x}) = F^{[0]}(\boldsymbol{x}) + \sum_{m=1}^{M_\pi} s_\pi \omega_\pi^{[m]} f^{[m]}(\boldsymbol{x}), \quad G(\boldsymbol{x}) = G_0(\boldsymbol{x}) + \sum_{m=1}^{M_\lambda} s_\lambda \omega_\lambda^{[m]} g^{[m]}(\boldsymbol{x}). \tag{4.11}$$

The results are listed in Table 1. Although the test losses (4.1) change slightly among different models, we can clearly order the considered models according to the other two metrics e_π (4.2) and e_λ (4.3). We observe that Model (4.11) has the best out-of-sample performance in terms of all the metrics. Furthermore, all the models can address the excess of zeros since they are all based on the ZIP distribution. We conclude that when there is a complicated effect of covariates on parameters, the EB algorithm is more appropriate than the EM algorithm since more flexible regression functions are allowed in the EB algorithm.

Table 1: Comparison of different ZIP models on the test data. Note that Models (4.4), (4.6), (4.7) and (4.8) are estimated via the EM algorithm, while Models (4.9), (4.10) and (4.11) are estimated via the EB algorithm.

model	test loss (4.1)	e_π (4.2)	e_λ (4.3)	proportion of zero claims
Null (4.4)	0.5299	0.1151	0.1795	0.8308
Varying λ (4.6)	0.5258	0.1112	0.1654	0.8305
Varying π (4.7)	0.5265	0.0959	0.182	0.8304
Varying π and λ (4.8)	0.5257	0.1159	0.1776	0.8304
Varying λ (4.9)	0.5234	0.1114	0.0928	0.8316
Varying π (4.10)	0.5256	0.0579	0.1738	0.8293
Varying π and λ (4.11)	0.5199	0.0687	0.0588	0.8336
True	0.5150	0.0000	0.0000	0.8310

Remarks. In the case of $K = 2$, there is only one to-be-boosted function F in mixing probabilities. Here, we replace the boosting loop (lines 5-12) in Algorithm 2 by a binary logistic gradient boosting, which boosts only one function F. Indeed, we might still apply the boosting loop (lines 5-12), which boosts two functions F_1 and F_2. The difference between F_1 and F_2 is F, i.e., $F_1 - F_2 = F$. In this example, the EB algorithm takes only several seconds since there are only two regression functions F and G to be estimated. When mixture models contain $K > 2$ components, the boosting of mixing probabilities takes much more time since K functions are to be boosted. This will be discussed in the next example.

4.2 Second simulated example: mixture of Gaussians

We consider a mixture of three Gaussians. Suppose that the mixing probabilities are related to covariates, while the parameters of component models are homogeneous among all samples. The probability density function is given by

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

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

$$p_k(\boldsymbol{x}) = \frac{\exp\left(F_k(\boldsymbol{x})\right)}{\sum_{l=1}^{3}\exp\left(F_l(\boldsymbol{x})\right)}, \text{ for } k = 1, 2, 3, \tag{4.13}$$

where

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

Note that covariate x_4 is a redundant variable. We specify other parameters as $\mu_1 = -5, \mu_2 = 0, \mu_3 = 5, \sigma_1 = \sigma_2 = \sigma_3 = 1$. We generate the covariates from the following distributions:

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

Note that we use shape-rate parameters for gamma distribution Γ. We generate $n = 12,000$ samples, among which $10,000$ samples are learning data and $2,000$ samples are test data.

First, we consider the parametric mixture models calibrated by the EM algorithm. The following mixtures of three linear regressions (or distributions) are fitted:

- Mixture of three distributions with homogeneous mixing probabilities (null model without any covariates):

$$f(Y|\boldsymbol{x}) = p_1f_N(Y;\mu_1,\sigma_1) + p_2f_N(Y;\mu_2,\sigma_2) + p_3f_N(Y;\mu_3,\sigma_3). \tag{4.14}$$

- Mixture of three linear regressions with homogeneous mixing probabilities:

$$f(Y|\boldsymbol{x}) = p_1f_N(Y;\mu_1(\boldsymbol{x}),\sigma_1) + p_2f_N(Y;\mu_2(\boldsymbol{x}),\sigma_2) + p_3f_N(Y;\mu_3(\boldsymbol{x}),\sigma_3), \tag{4.15}$$

  where

$$\mu_1(\boldsymbol{x}) = \langle\boldsymbol{\alpha},\boldsymbol{x}\rangle, \mu_2(\boldsymbol{x}) = \langle\boldsymbol{\beta},\boldsymbol{x}\rangle, \mu_3(\boldsymbol{x}) = \langle\boldsymbol{\gamma},\boldsymbol{x}\rangle. \tag{4.16}$$

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

$$f(Y|\boldsymbol{x}) = p_1(\boldsymbol{x})f_N(Y;\mu_1,\sigma_1) + p_2(\boldsymbol{x})f_N(Y;\mu_2,\sigma_2) + p_3(\boldsymbol{x})f_N(Y;\mu_3,\sigma_3) \tag{4.17}$$

where

$$p_k(\boldsymbol{x}) = \frac{\exp\left(F_k(\boldsymbol{x})\right)}{\sum_{l=1}^{3} \exp\left(F_l(\boldsymbol{x})\right)}, \text{ for } k = 1, 2, 3, \tag{4.18}$$

and

$$F_1(\boldsymbol{x}) = \langle \boldsymbol{a}, \boldsymbol{x} \rangle, F_2(\boldsymbol{x}) = \langle \boldsymbol{b}, \boldsymbol{x} \rangle, F_3(\boldsymbol{x}) = \langle \boldsymbol{c}, \boldsymbol{x} \rangle. \tag{4.19}$$

Note that this model follows the underlying mixture model. However, this model cannot address the non-linearity and interaction of covariates on the mixing probabilities.

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

$$f(Y|\boldsymbol{x}) = p_1(\boldsymbol{x}) f_N(Y; \mu_1(\boldsymbol{x}), \sigma_1) + p_2(\boldsymbol{x}) f_N(Y; \mu_2(\boldsymbol{x}), \sigma_2) + p_3(\boldsymbol{x}) f_N(Y; \mu_3(\boldsymbol{x}), \sigma_3) \tag{4.20}$$

  where the means and the mixing probabilities follow equations (4.16), (4.18) and (4.19).

Next, we consider non-parametric mixture models calibrated by the proposed EB algorithm. The following mixture models are fitted:

- Mixture model with heterogeneous means and homogeneous mixing probabilities:

$$f(Y|\boldsymbol{x}) = p_1 f_N(Y; \mu_1(\boldsymbol{x}), \sigma_1) + p_2 f_N(Y; \mu_2(\boldsymbol{x}), \sigma_2) + p_3 f_N(Y; \mu_3(\boldsymbol{x}), \sigma_3), \tag{4.21}$$

  where μ_1, μ_2, μ_3 are estimated non-parametrically via the boosting algorithm. This model is comparable to model (4.15).

- Mixture model with heterogeneous mixing probabilities:

$$f(Y|\boldsymbol{x}) = p_1(\boldsymbol{x}) f_N(Y; \mu_1, \sigma_1) + p_2(\boldsymbol{x}) f_N(Y; \mu_2, \sigma_2) + p_3(\boldsymbol{x}) f_N(Y; \mu_3, \sigma_3) \tag{4.22}$$

  where

$$p_k(\boldsymbol{x}) = \frac{\exp\left(F_k(\boldsymbol{x})\right)}{\sum_{l=1}^{3} \exp\left(F_l(\boldsymbol{x})\right)}, \text{ for } k = 1, 2, 3. \tag{4.23}$$

  Here, F_1, F_2, F_3 are estimated non-parametrically via the boosting algorithm. This model is comparable to model (4.17).

- Mixture model with both heterogeneous means and heterogeneous mixing probabilities:

$$f(Y|\boldsymbol{x}) = p_1(\boldsymbol{x}) f_N(Y; \mu_1(\boldsymbol{x}), \sigma_1) + p_2(\boldsymbol{x}) f_N(Y; \mu_2(\boldsymbol{x}), \sigma_2) + p_3(\boldsymbol{x}) f_N(Y; \mu_3(\boldsymbol{x}), \sigma_3) \tag{4.24}$$

  where both means and mixing probabilities are estimated non-parametrically via the boosting algorithm. This model is comparable to model (4.20).

We summarize the results of the seven models in Table 2 including average negative log-likelihood on the test data and the running time of the EM algorithm or the EB algorithm. Note that the true negative log-likelihood is calculated based on the true underlying model (i.e., without estimation error). All three models with linear predictors (4.15), (4.17) and (4.20) have a larger test loss than their counterpart non-parametric models (4.21), (4.22) and (4.24), respectively, since they cannot capture non-linear effects and interactions of covariates. The two models with both heterogeneous means and mixing probabilities (4.20) and (4.24) have no evident better out-of-sample performance than the two models with only heterogeneous

Table 2: Comparison of different models in terms of test loss and running time. Note that Models (4.14), (4.15), (4.17) and (4.20) are estimated via the EM algorithm, while Models (4.21), (4.22) and (4.24) are estimated via the EB algorithm.

model	test loss	running time without parallel computing	running time with parallel computing
Null (4.14)	2.4821	$<$ 10 seconds	NA
Varying μ (4.15)	2.4779	$<$ 10 seconds	NA
Varying p (4.17)	2.2016	$<$ 10 seconds	NA
Varying μ and p (4.20)	2.2011	$<$ 10 seconds	NA
Varying μ (4.21)	2.4139	$<$ 10 seconds	NA
Varying p (4.22)	2.1330	650 seconds	381 seconds
Varying μ and p (4.24)	2.1320	1054 seconds	597 seconds
True	2.1231	NA	NA

probabilities (4.17) and (4.22), respectively, indicating that the component means are not entirely related to the covariates.

For Model (4.24), we draw the trace plot of the loss during the EB iterations $t = 1, \ldots, T$ in Figure 1. It shows that the EB algorithm converges after around 10 iterations. At the last EB iteration T, we draw the trace plot of the loss during the boosting iterations $m = 1, \ldots, M_0$ for mixing probabilities in Figure 1. It shows that the boosting is robust against overfitting. At the last EB iteration T, we also draw the trace plot of the loss during the boosting iterations $m = 1, \ldots, M_k$ for each component mean in Figure 2. It shows that the boosting stops early at the very beginning without finding heterogeneity in component means. We calculate the overall relative importance (scaled to sum 100) of x_1, x_2, x_3, x_4 in the mixing probabilities $(IP_j)_{j=1:4}$ as $48.23, 34.51, 17.00, 0.25$, which aligns with the underlying model.

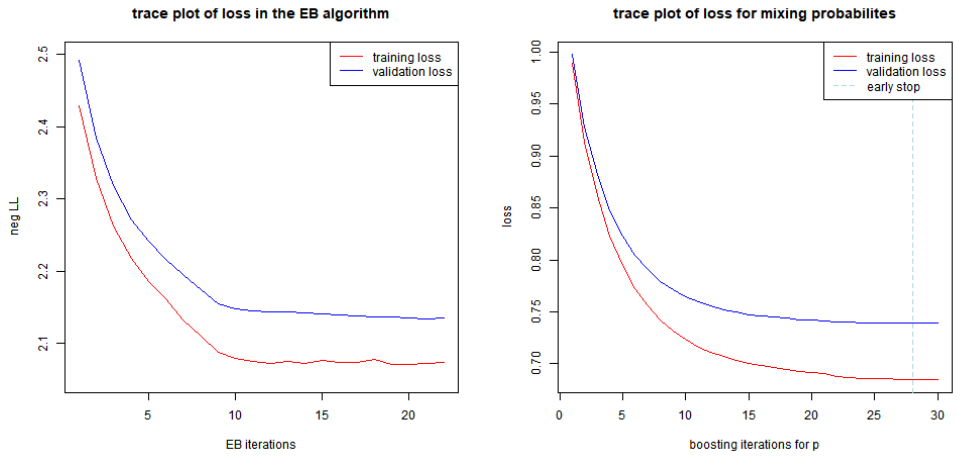

Figure 1: Left: The trace plot of loss during the EB iterations. Right: The trace plot of loss during the boosting iterations for mixing probabilities at last EB iteration T.

Comparison of the running times for models (4.21) and (4.22) without parallel computing implies that the boosting of the mixing probabilities takes much more time than the boosting of

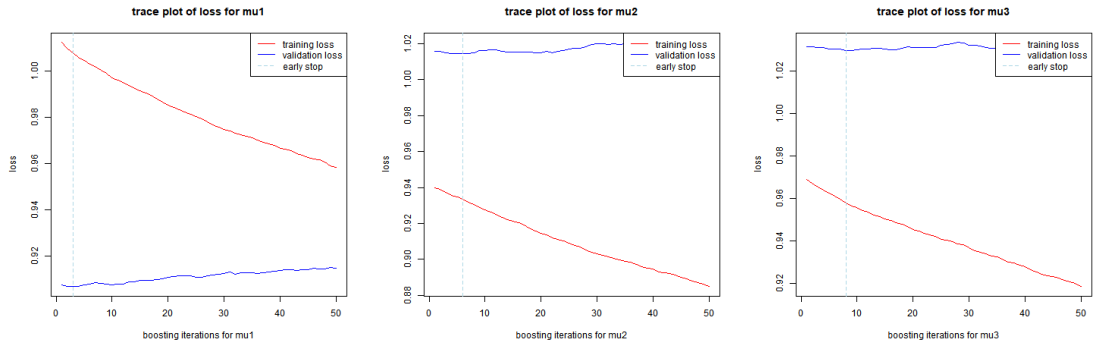

Figure 2: The trace plots of loss during the boosting iterations for component means at last EB iteration T.

the component means. One reason is that the boosting of component means stops early while the boosting of mixing probabilities takes more time to discover the heterogeneity; see Figures 1 and 2. We accelerate the EB algorithm by applying parallel computation in the boosting of mixing probabilities, i.e., in lines 6-9 of Algorithm 2 we fit K trees simultaneously. The last few lines in Table 2 show that parallel computing accelerates the EB algorithm significantly.

4.3 Real data example: claims severity modeling

In this example, we study the claims amount data `freMTPL2sev` from R package *CASdatasets*, which contains risk features and claim amounts from $n = 24,938$ motor third-part liabilities policies (after data cleaning). The claims severities are calculated as the ratio of the totoal claims amount to the number of claims for each policy.

There are 9 risk features available. The location-related features include the region in France `Reg`, the density of the city community `Area` and the density of inhabitants `Dens`. The vehicle-related features include the power of the car `VehP`, the vehicle age `VehA`, the vehicle brand `VehB` and the car gas `VehG`. The driver-related feature is the driver age `DrivA`. The claims-history-related feature is the bonus-malus score `BM`. In the boosting algorithm, we replace the two categorical variables `Reg` and `VehB` by means of claims severities in each categorical level. There is a strong linearity between `Area` and `Dens` and a heavy tail in `Dens`. In the GLM, we typically need to remove one of `Area` and `Dens` and take the logarithm of `Dens`. However, there is no need for such feature engineering in the boosting.

We plot the histogram and the cumulative distribution function of logged claims severities in Figure 3, which shows three peaks and a heavy tail. An intuitive choice of component distributions is that three peaks are modeled by three gamma distributions, the resting non-tail part by a fourth gamma distribution, and the tail part by a Pareto distribution. Therefore, the probability density function (of the null model) is given by

$$f(Y|\boldsymbol{x}) = \sum_{k=1}^{4} p_k f_G(Y; \mu_k, \phi_k) + p_5 f_P(Y; \alpha, M), \tag{4.25}$$

where μ, ϕ are the mean and dispersion parameters of gamma distribution, and α, M are the tail index and threshold of Pareto distribution.

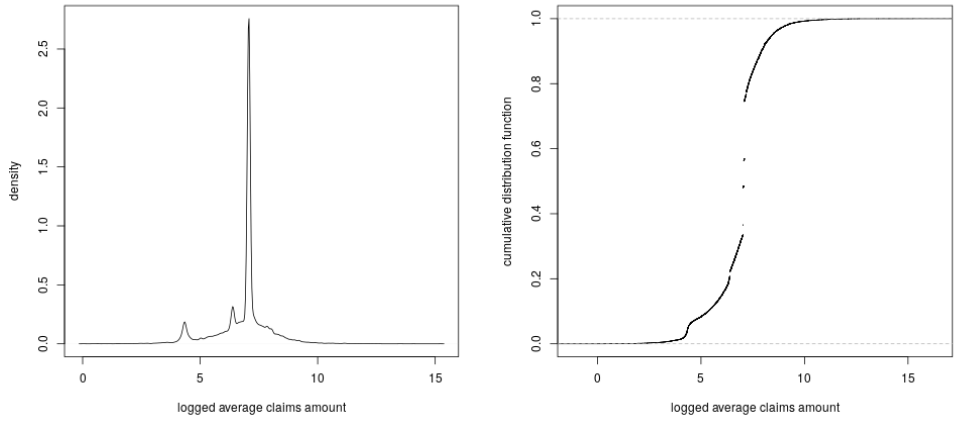

Figure 3: The histogram and the cumulative distribution function of logged claims severities.

The logged survival function for logged claims severities as shown in Figure 4 indicates a regularly varying tail at infinity (Embrechts et al., 2013). The threshold of Pareto distribution is selected as $M = 8158.13$ according to the Hill plot (Resnick, 1997) as shown in Figure 4. One may refer to Wüthrich and Merz (2022) for more discussions on this data.

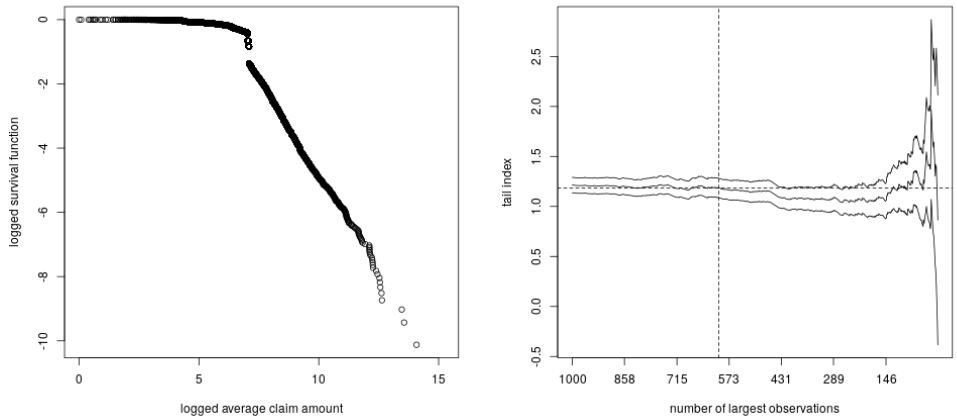

Figure 4: The logged survival function and the Hill plot for logged claims severities.

The three peaks as shown in Figure 3 imply a way of initialization through the latent component indicator variable:

$$\begin{aligned}\hat{\boldsymbol{z}}_i^{[0]} &= (\hat{z}_{i,1}^{[0]}, \hat{z}_{i,2}^{[0]}, \hat{z}_{i,3}^{[0]}, \hat{z}_{i,4}^{[0]}, \hat{z}_{i,5}^{[0]}) \\ &= (\mathbb{1}_{(0,500]} y_i, \mathbb{1}_{(500,1000]} y_i, \mathbb{1}_{(1000,1200]} y_i, \mathbb{1}_{(1200,8158.13]} y_i, \mathbb{1}_{(8158.13,\infty)} y_i).\end{aligned} \tag{4.26}$$

Given the latent component indicator variable $\hat{\boldsymbol{z}}_{i=1:n}^{[0]}$, the parameters can be initialized as the MLEs in the null model on the full information $(y_i, \hat{\boldsymbol{z}}_i^{[0]}, \boldsymbol{x}_i)_{i=1:n}$. We split the data into a training data set, a validation data set and a test data set by a ratio of 3:1:1.

We first fit a mixture of distributions (4.25) to the data via the EM algorithm. The estimated component parameters are listed in Table 3. We observe that the three peaks are captured by

the first three component gamma distributions. The first three large shape parameters (small dispersions) imply the difficulties with mean modeling in the first three component models. The test loss is calculated as 7.5815. The mixing probabilities are estimated as $\hat{p}_1 = 0.0409$, $\hat{p}_2 = 0.0300$, $\hat{p}_3 = 0.4100$, $\hat{p}_4 = 0.4973$ and $\hat{p}_5 = 0.0218$.

Table 3: The MLEs of component parameters in the homogeneous model (4.25). 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

Next we fit a mixture of distributions with heterogeneous mixing probabilities to the data via the proposed EB algorithm:

$$f(Y|\boldsymbol{x}) = \sum_{k=1}^{4} p_k(\boldsymbol{x}) f_G(Y; \mu_k, \phi_k) + p_5(\boldsymbol{x}) f_P(Y; \alpha, M). \tag{4.27}$$

For the last EB iteration, we draw the trace plot of the multiclass logistic loss during the boosting of mixing probabilities in Figure 5, which shows that the boosting is quite robust against overfitting. We draw the boxplot of the estimated mixing probabilities in Figure 5. We observe that the mixing probabilities for the third and forth components p_3 and p_4 are more related with the covariates than p_1, p_2, p_5. The test loss is 7.5580, smaller than the null model.

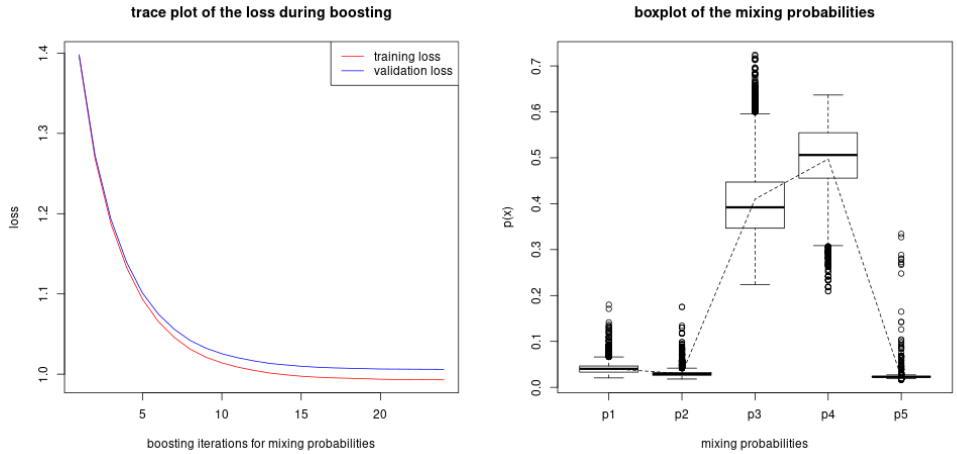

Figure 5: Left: The trace plot of the loss during the boosting of mixing probabilities. Right: The boxplot of the estimated mixing probabilities in Model (4.27).

We calculate the relative importance of risk features in each mixing probability, respectively. The unscaled relative importance $(IP_{j,k})_{j=1:9,k=1:5}$ shown in Figure 6 indicates that the risk features have much stronger systematic effects on p_3, p_4 than p_1, p_2, p_5, which is consistent with the observation in Figure 5. The overall relative importance $(IP_j)_{j=1:9}$ is shown in the last plot of Figure 6. The most important risk features in predicting claims severity are the bonus-malus

score and the driver age. The vehicle age and brand are two important vehicle-related features, while the vehicle power and gas are not important in predicting claims severity. All region-related risk features are important. Since `Area` has a strong linearity with `Dens`, we do not need the first risk feature if the second one is in the model.

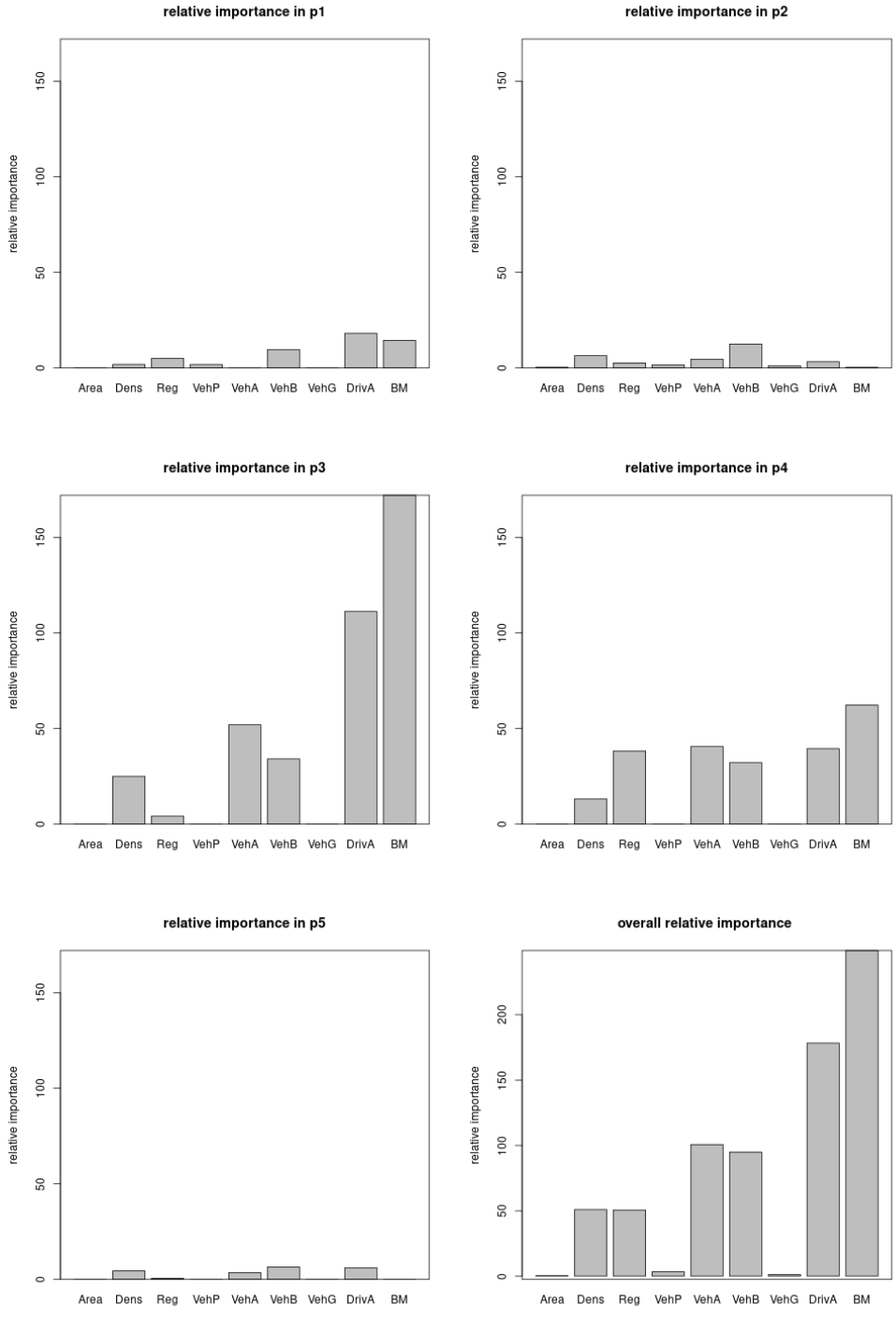

Figure 6: The relative importance of risk features in the mixing probabilities.

5 Conclusions

Insurance loss data sometimes cannot be sufficiently modeled by a single distribution, so a mixture model is usually more preferred. Since there are multiple component models in a mixture model, the parameter estimation problem is one key issue for data analysis. Thus, the EM algorithm is one critical estimation method for mixture models in the literature. However, the EM algorithm usually requires a parametric form of the component models, which limits the practical flexibility of mixture models. In this paper, we propose an Expectation-Boosting (EB) algorithm for mixture models, where both the mixing probabilities and the component models can be modeled non-parametrically.

The core of the proposed algorithm is to replace the maximization step of the EM algorithm with a generic functional gradient descent algorithm. There are several advantages of the EB algorithm over the EM algorithm. First, boosting is a flexible non-parametric regression facilitating both non-linearity and interaction among the covariates. Automated feature engineering in mixing probabilities and all component models is performed in the EB algorithm. Second, boosting is overfitting-sensitive, and variable selection in mixing probabilities and all component models is conducted simultaneously during model fitting. The proposed algorithm requires more computing time for modeling flexibility than the EM algorithm since the boosting is nested inside each EB iteration. To accelerate the EB algorithm, one approach is to apply parallel computation since there are some independent steps or loops at each EB iteration given the component indicator variable. Another approach is to use iterative initialization in the boosting, i.e., the boosting starts with an iteratively better initialization and stops earlier.

Acknowledgment

Guangyuan Gao gratefully acknowledges the financial support from the National Natural Science Foundation of China (71901207). Yanxi Hou gratefully acknowledges the financial support from the National Natural Science Foundation of China (72171055, 71991471) and Natural Science Foundation of Shanghai (20ZR1403900).

References

Breiman, Leo, F., H, J., Olshen, R. A. and Stone, C. J. (1984). *Classification and Regression Trees*, Wadsworth Statistics/Probability Series.

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.

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.

Embrechts, P., Klüppelberg, C. and Mikosch, T. (2013). *Modelling Extremal Events: for Insurance and Finance*, Springer Science & Business Media.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine, *The Annals of statistics* **29**: 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. and Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 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.

McLahlan, G. and Peel, D. (2000). *Finite Mixture Models*, Wiley-Interscience.

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.

Resnick, S. I. (1997). Heavy tail modeling and teletraffic data: special invited paper, *The Annals of Statistics* **25**(5): 1805–1869.

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.

Wüthrich, M. V. and Merz, M. (2022). Statistical foundations of actuarial learning and its applications, *Available at SSRN 3822407* .

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.