Insurance loss modelling with gradient tree-boosted mixture models

Yanxi Hou* Jiahong Li† Guangyuan Gao‡

March 7, 2022

Abstract

In actuarial practice, mixture models [mixture models or mixture of models? both appear in the paper] is one widely applied statistical method to model the insurance loss data. Although the Expectation-Maximization (EM) algorithm usually plays an important tool for the parameter estimation of mixture models, it suffers from other issues which cause unstable predictions. For example, the feature engineering and variable selection are two important modelling issues, which are challenging for mixture models due to several component models involving. Moreover, to avoid overfitting is another technical concern of the modelling method for 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 decision trees. Our proposed EB algorithm can estimate the component regression functions non-paramerically and overfitting-sensitively, as well as perform automated feature engineering, model fitting and variable selection simultaneously, which fully explores the predictive power of feature space. We also 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 computing; Finite mixture of regressions; Zero-inflated Poisson model.

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 claim count data, a Poisson distribution may not be the best choice to model an excess of zero claims. For claim amount data, it usually exhibits multimodal or heavy-tailed phenomenons, so a gamma distribution is not sufficient to address these data features. In the literature, 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 used instead to model the risk heterogeneity, which is an extension to the mixture of distributions. In the literature, mixture models are proposed by Goldfeld and Quandt (1973). We refer to Lindsay (1995) and MacLahlan and Peel (2000) for a detailed review of mixture

*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.

models. In the field of machine learning research, mixture models are also called as mixture of experts models (Jacobs et al., 1991; Jiang and Tanner, 1999).

We summarize some studies on mixture models for insurance loss modelling. 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 mixture of Erlangs. Fung et al. (2019a), Fung et al. (2019b) 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 (GLM) framework. In this paper, we will relax this constraint by estimating regression functions non-parametrically.

Several aspects of modelling are very noteworthy in research. 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 latent component indicator variables. In addition, the variable selection and the feature engineering in mixture models are also challenging since the dependence between the covariate 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, the selection of 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 and variables. Kasahara and Shimotsu (2015) proposed a likelihood-based test to contrast the 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 for mixture models, which replaces the maximization step by a boosting step. It is well known that the boosting algorithm is an iterative method to estimate the 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 Modelling 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 and Friedman (2001) has studied this type of 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 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. Thus, the variable selection and feature engineering are

performed automatically during the model fitting. Second, it is well known that the boosting algorithm is overfitting-sensitive (Bühlmann and Yu, 2003). This property is very useful since the ultimate goal of insurance loss modelling is to predict the future loss. Therefore, the out-of-sample prediction is more important than the goodness-of-fitting in our applications.

The rest of paper is structured as follows. Section 2 reviews mixture models and the EM algorithm. Section 3 presents the EB algorithm. We also discuss initialization, tuning parameters and parallel computing in detail. Section 4 studies two simulated data and a 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 Review: mixture models and the EM algorithm

In this section, we first review mixture models, particularly finite mixture of regressions (MacLahlan and Peel, 2000), and discuss different mixture structure. 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 are fitting for 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 distributions

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

and *mixing probability* $p_k, k=1,\ldots,K(K\geqslant 1)$. 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 *individual features* $\boldsymbol{x}_i$ are available and have a systematic effect on the response variable y_i, then we can establish a finite mixture of regressions:

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

Note that (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 modelling, 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, we might assume that different component distributions have the same dispersion ϕ

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

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

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

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. It actually provides practical flexibility according to different data structures and the goals of study.

2.2 The EM algorithm

Suppose that for a single sample y, 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 given by

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

Equivalently, the complete log-likelihood function is

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

Given a data set $\{(y_i, \boldsymbol{z}_i, \boldsymbol{x}_i)\}_{i=1:n}$, we assume the parametric models of $p = (p_1, \ldots, p_K), \mu_k, \phi_k$ such that $p(\boldsymbol{x}) = (p_1(\boldsymbol{x}; \theta_p), \ldots, p_K(\boldsymbol{x}; \theta_p))$, $\mu_k(\boldsymbol{x}) = \mu_k(\boldsymbol{x}; \theta_\mu^{(k)})$ and $\phi_k(\boldsymbol{x}) = \phi_k(\boldsymbol{x}; \theta_\phi^{(k)})$, where $\theta_p, (\theta_\mu^{(k)})_{k=1:K}, (\theta_\phi^{(k)})_{k=1:K}$ are the unknown parameters. Note that θ_p does not have a superscript since it corresponds to coefficients in a multinomial logistic classification while $\theta_\mu^{(k)}, \theta_\phi^{(k)}$ can be considered in separate component models; see below.

The regression coefficients can be estimated by the following $K+1$ independent optimizations:

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

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 the k-th regression in (2.4) 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 $\hat{p}, \hat{\mu}, \hat{\phi}$, 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 log-likelihood function for full information

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

calculate the MLE of regression coefficients by the following $K + 1$ independent optimizations:

$$\hat{\theta}_p = \underset{\theta_p}{\arg\max} \sum_{i=1}^{n} \sum_{k=1}^{K} \hat{z}_{i,k} \log p_k(\boldsymbol{x}_i; \theta_p) \tag{2.6}$$

and

$$\left(\hat{\theta}_{\mu}^{(k)}, \hat{\theta}_{\phi}^{(k)}\right) = \underset{\theta_{\mu}^{(k)}, \theta_{\phi}^{(k)}}{\arg\max} \sum_{i=1}^{n} \sum_{k=1}^{K} \hat{z}_{i,k} \log f_k \left(y_i; \mu_k \left(\boldsymbol{x}_i; \theta_{\mu}^{(k)}\right), \phi_k \left(\boldsymbol{x}_i; \theta_{\phi}^{(k)}\right)\right), \text{ for } k = 1, \ldots, K. \tag{2.7}$$

[why using bold $\boldsymbol{p}, \boldsymbol{\mu}, \boldsymbol{\phi}, \boldsymbol{y}$? Need to define some new notions above?] These optimizations are similar to those in (2.3) and (2.4). The only difference is that $z_{i,k} \in \{0, 1\}$ in (2.3) and (2.4) while $\hat{z}_{i,k} \in (0, 1)$ in (2.6) and (2.7). Therefore, in the maximization step we perform a multinomial logistic classification with *fractional* response $\hat{z}_{i,k}$ [No illustration for a fractional response] and K weighted-regressions [No illustration as well] 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 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 at each iteration, but overfitting-sensitively. The boosting step follows a generic functional gradient descent algorithm. We first describe generic functional gradient descent algorithm, then propose the EB algorithm. [why not put the review of generic functional gradient descent algorithm in Section 2.]

3.1 Generic functional gradient descent algorithm

Suppose a to-be-boosted non-parametric regression function as $F : \mathbb{R}^P \to \mathbb{R}$. 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 normal 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 connects the linear predictor with the parameters of interest such as mean, probability odds, etc. Here, we apply the same terminology where the link function connects the non-parametric regression function F with the parameters of interest. 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 $g : \mathbb{R} \to \mathbb{R}$

$$g(\text{mean}) = \text{mean} = F(\boldsymbol{x}).$$

Remark that in our setting of statistical modeling, we first specify the likelihood function and the link function, then we derive the cost function. [The link function is not defined and not pointed to. It is better to give some descriptions for both loss and link functions].

We follow Bühlmann and Yu (2003) to briefly describe the generic functional gradient decent algorithm in Algorithm 1, which will be used 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. [Consider using `algorithm` package here]

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).[The weaker learner is not defined or described, and appear in the algorithm at the first time. It is better to put this sentence in the article and give this step a straightforward description.]

4: One-dimensional optimization: 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** $\hat{F}(\boldsymbol{x}) = \hat{F}^{[M]}$.

In the algorithm, weak learners are fitted to negative gradient U rather than Y. Loss function in weak learners is always L_2, independently with model loss function C. If 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. Please refer to Hastie et al. (2009) for decision trees and the recursive partitioning algorithm. Bühlmann and Hothorn (2007) argue that step 4 of one-dimensional optimization seems unnecessary given learning rate s is sufficiently small according to some empirical experiments.

3.2 Expectation-Boosting algorithm

In the EB algorithm, we follow the exact same expectation step as in the EM algorithm and replace the maximization step by boosting. Therefore, the boosting algorithm is applied to estimate mixing probabilities and component parameters given the conditional expectation of

latent component indicator variables $\hat{z}_{i,k}$. To apply a boosting algorithm, we need to specify the likelihood function and the link function. In mixture models, the log-likelihood function is already given by (2.5). Therefore, we only need to specify the link functions for mixing probabilities, component means and component dispersions, respectively.

For mixing probabilities, we connect the mixing probabilities $p_1, \ldots, p_K$ with the to-be-boosted regression functions $F_1, \ldots, F_K$ by the multiple logistic link function $g_k : (0,1)^K \to \mathbb{R}$: [Using $p(\boldsymbol{x}; F(\boldsymbol{x}))$ or $p(F(\boldsymbol{x}))$? They are different. Similarly for other functions.]

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

or equivalently

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

[It is not clear the difference between p or p_k, F and F_k. Similarly for other functions. Need definition of notions?] [$p_k(\boldsymbol{x})$ appears. The notions are not consistent. Similarly for other functions] With the negative log-likelihood function (2.6) of multinomial distribution, we derive the loss function as

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

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

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

For component parameters μ and ϕ, we denote the to-be-boosted regression functions by G and H, respectively. The link functions depend on the component models. For example, in component Gaussian model we assume $G_k = \mu_k, H_k = \log \phi_k$; in component Poisson model $G_k = \log \mu_k$; in component gamma model $G_k = \log \mu_k, H_k = \log \phi_k$. With the negative log-likelihood function (2.7), we derive the loss function as

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

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:[Consider using `algorithm` package here. It is better to write it into two loops as described in the following]

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

2 Calculate conditional expectation of latent variable $\hat{z}^{[t+1]}$ given $\hat{p}^{[t]}, \hat{\mu}^{[t]}, \hat{\phi}^{[t]}$.

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

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

4 Calculate the MLE $\hat{\phi}^{[t+1]}$ given $\hat{\mu}^{[t+1]}$ and $\hat{z}^{[t+1]}$. 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 2. 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 2. [Would it imply a parallel computation approach?] 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 modelling of both mean and dispersion is not straightforward since they are not independent; please refer to Jorgensen (1997) for dispersion modelling. Joint modelling both mean and dispersion is not considered in this paper.

We now detail each step of the EB algorithm.

Step 1: 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.

Step 2: Conditional expectation of latent variable. We set $\hat{z}_{i,k}^{[t+1]}$ as

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

Step 3.1: 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

$$\hat{p}_k^{[t,0]} = \frac{1}{K}, \quad \hat{F}_k^{[t,0]} = \log \hat{p}_k^{[t,0]} - \frac{1}{K} \sum_{l=1}^{K} \log \hat{p}_l^{[t,0]}. \tag{3.8}$$

Set $m = 0$. In contrast to step 1, this initialization will be repeated for each outer EB iteration t. 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 for each data point $i = 1, \ldots, n$ as

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

For each $k = 1, \ldots, K$, the data $(u_{i,k}^{[t,m+1]}, \boldsymbol{x}_i)_{i=1:n}$ 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

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

We then derive the updated mixing probability $\hat{p}_{i,k=1:K}^{[t,m+1]}$ using (4.16).

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 $F_k, k = 1, \ldots, K$. 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.

Step 3.2: Gradient boosting component models. We implement another K independent *weighted* boosting with weights $\hat{z}_{i,k}^{[t+1]}$:

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:

$$\hat{\mu}_k^{[t,0]} = \frac{\sum_{i=1}^n Y_i}{n}, \ \hat{G}_k^{[t,0]} = \mu_k^{-1}(\hat{\mu}_k^{[t,0]}), \tag{3.9}$$

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 for each data point $i = 1, \ldots, n$ as

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

For each $k = 1, \ldots, K$, the data $(v_{i,k}^{[t,m+1]}, \boldsymbol{x}_i)_{i=1:n}$ 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.

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

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.

Step 4: Update the dispersion parameters. 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. [It is better to provide the updating formulas in the algorithm]

In summary, $(M_p + M_\mu) \times K \times T$ weak learners are calibrated in the EB algorithm. In each EB iteration t[it is not clear what each EB iteration is as the algorithm has two loops], $2K$ independent boosting algorithms are performed. Thus, the EB algorithm can be accelerated by conducting those boosting algorithms simultaneously in each EB iteration t. [The reason seems not convincing.]

In the EB algorithm, we have the following tuning parameters and the corresponding tuning strategies are listed below:[For tuning parameters, shall we write it into a subsection solely with descriptions of validation? The list below is descriptive not very clear to readers I think. For example, using some minimization formulas to describe the way. Moreover, for tuning parameters in different inner and outer loops, the order of doing them is important. Currently cannot see the order of tuning parameters in the algorithm.]

- Number of EB iterations T (iterations in outer loop). This can be determined by screening the trace plot of loss.
- Number of iterations M_p, M_μ in boosting algorithm (iterations in inner loops). We can specify a sufficiently large M_p, M_μ and use early stop on the 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 decision trees: complexity parameter, maximum depth, number of terminal nodes, etc. Those parameters can be tuned as in typical decision trees. Please refer to Hastie et al. (2009) for more details.

In steps 3.1.1 and 3.2.1 for the initialization of boosting, we initialize parameters in iteration $t+1$ independently with the previously boosted estimates $\hat{p}_k^{[t]}, \hat{\mu}_k^{[t]}$; see equations (3.8) and (3.9). We call this as uncorrelated boosting. However, please note that boosting in iteration $t+1$ is not independent with boosting in iteration t since boosting in iteration $t+1$ is based on $\hat{\boldsymbol{z}}^{[t+1]}$ which depends on boosting in iteration t. The gains from boosting are passed to the boosting in the next EB iteration via the conditional expectation $\hat{\boldsymbol{z}}$.

We might initialize parameters in the iteration $t+1$ as the previously boosted estimates

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

This would lead a smaller required inner boosting iterations M (or a earlier stop on validation loss) since the boosting starts from better initial values. 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 step with default initialization (3.8) and (3.9) given the last conditional expected hidden variables $\hat{\boldsymbol{z}}^{[T]}$. In the following applications, we only implement the uncorrelated boosting.

[As mentioned in the introduction, boosting algorithms possess good properties to deal with some issues like variable selection, feature engineering and overfitting problems. Currently there is no discussion about these topics in this section and no illustration in the next simulation or real data analysis. It is better to address these aspects of EB algorithm and add some results in the next section. For example, variable importance plots and partial dependence plots. These are very useful for the interpretation of real data analyses.]

4 Applications

We conduct two simulated data analysis and a real data analysis. The first simulated data follows a zero-inflated Poisson model, which mimics claim count data with excess of zero. 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 discuss constraints on mixing structure and parallel computing in this example. The real data contains the claim amount data of an insurance company. This example demonstrates how to choose component models and mixing structure in practice.

4.1 First simulated example: zero-inflated Poisson model

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

$$f_{\text{ZIP}}(N;\lambda,\pi_0) = \begin{cases} \pi_0 + (1-\pi_0)e^{-\lambda} & \text{for} \quad N=0 \\ (1-\pi_0)\frac{e^{-\lambda}\lambda^N}{N!} & \text{for} \quad N \in \mathbb{N}_+. \end{cases} \tag{4.1}$$

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

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

Suppose that five covariates $\boldsymbol{x} = (x_1,\ldots,x_5)$ have systematic effects on both π_0 and λ: [It is better to give brackets for the exp function. Similarly for those in the rest of paper]

$$\pi_0(\boldsymbol{x}) = \frac{\exp F(\boldsymbol{x})}{1+\exp F(\boldsymbol{x})}, \quad \lambda(\boldsymbol{x}) = \exp G(\boldsymbol{x}), \tag{4.2}$$

where

$$F(\boldsymbol{x}) = 0.3 - 2x_2^2 + x_2 + 0.2x_5, \quad G(\boldsymbol{x}) = \log 0.5 + x_1^2 + 0.2\log x_3 - 0.2x_1x_4. \tag{4.3}$$

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 data $i \in \mathcal{I}_L$ of sample size $8,000$ and a test data $i \in \mathcal{I}_T$ of sample size $2,000$. 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 as $\exp(-0.2136) = 80.77\%$ less than the empirical proportion 83.10% for the test data.

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

$$-\frac{1}{|\mathcal{I}_{\text{test}}|}\sum_{i\in\mathcal{I}_{\text{test}}} \log\left(\hat{\pi}_0(\boldsymbol{x}_i)\mathbb{1}_0(N_i) + (1-\hat{\pi}_0(\boldsymbol{x}_i))\frac{e^{-\hat{\lambda}(\boldsymbol{x}_i)}\hat{\lambda}(\boldsymbol{x}_i)^{N_i}}{N_i!}\right), \tag{4.4}$$

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:

$$e_\pi = \frac{1}{|\mathcal{I}_{\text{test}}|} \sum_{i \in \mathcal{I}_{\text{test}}} (\hat{F}(\boldsymbol{x}_i) - F(\boldsymbol{x}_i))^2, \tag{4.5}$$

$$e_\lambda = \frac{1}{|\mathcal{I}_{\text{test}}|} \sum_{i \in \mathcal{I}_{\text{test}}} (\hat{G}(\boldsymbol{x}_i) - G(\boldsymbol{x}_i))^2. \tag{4.6}$$

We start with a null ZIP model without any covariates:

$$F(\boldsymbol{x}) = \alpha_0, \quad G(\boldsymbol{x}) = \beta_0 \tag{4.7}$$

The MLE can be 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 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

$$-\frac{1}{|\mathcal{I}_{\text{test}}|} \sum_{i \in \mathcal{I}_{\text{test}}} \log \left(\pi_0(\boldsymbol{x}_i) \mathbb{1}_0(N_i) + (1 - \pi_0(\boldsymbol{x}_i)) \frac{e^{-\lambda(\boldsymbol{x}_i)} \lambda(\boldsymbol{x}_i)^{N_i}}{N_i!} \right) = 0.5150, \tag{4.8}$$

which is (surely) smaller than 0.5299 from the null model. The test metrics (4.5) and (4.6) 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 function of $\boldsymbol{x}$. One component model is degenerated into a probability mass 1 at $N = 0$, so there is no coefficient in this component model. We consider three different mixture of models, heterogeneous Poisson parameter (4.9), heterogeneous mixing probability (4.10) and both heterogeneous mixing probability and Poisson parameters (4.11):

$$F(\boldsymbol{x}) = \alpha_0, \quad G(\boldsymbol{x}) = \langle \boldsymbol{\beta}, \boldsymbol{x} \rangle, \tag{4.9}$$

$$F(\boldsymbol{x}) = \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle, \quad G(\boldsymbol{x}) = \beta_0, \tag{4.10}$$

$$F(\boldsymbol{x}) = \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle, \quad G(\boldsymbol{x}) = \langle \boldsymbol{\beta}, \boldsymbol{x} \rangle. \tag{4.11}$$

[Define inner product?]The MLE of regression coefficients $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ can be obtained by the EM algorithm. We have summarized the results in Table 1. Model (4.11) has the best out-of-sample performance followed by Model (4.9), while Model (4.10) has the worst out-of-sample performance. All the three models are better than the null model (4.7). Note that those models ignore the potential non-linear effects and interactions of covariates.

Finally we estimate F and G non-parametrically via the proposed EB algorithm. We also consider three different mixing structures, heterogeneous Poisson parameter (4.12), heterogeneous mixing probability (4.13) and both heterogeneous mixing probability and Poisson parameters

(4.14):

$$F(\boldsymbol{x}) = \alpha_0, \quad G(\boldsymbol{x}) = \sum_{m=0}^{M_\lambda} s_\lambda \omega_\lambda^{[m]} g^{[m]}(\boldsymbol{x}), \tag{4.12}$$

$$F(\boldsymbol{x}) = \sum_{m=0}^{M_\pi} s_\pi \omega_\pi^{[m]} f^{[m]}(\boldsymbol{x}), \quad G(\boldsymbol{x}) = \beta_0, \tag{4.13}$$

$$F(\boldsymbol{x}) = \sum_{m=0}^{M_\pi} s_\pi \omega_\pi^{[m]} f^{[m]}(\boldsymbol{x}), \quad G(\boldsymbol{x}) = \sum_{m=0}^{M_\lambda} s_\lambda \omega_\lambda^{[m]} g^{[m]}(\boldsymbol{x}). \tag{4.14}$$

The results are listed in Table 1. Although the test loss changes slightly among different models, we can clearly order the considered models according to the two metrics e_π and e_λ. We observe that Model (4.14) 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 covariate 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 (4.7), (4.9), (4.10) and (4.11) are estimated via the EM algorithm, while Models (4.12), (4.13) and (4.14) are estimated via the EB algorithm.

model	test loss (4.4)	e_π (4.5)	e_λ (4.6)	proportion of zero claims
Null (4.7)	0.5299	0.1151	0.1795	0.8308
Varying λ (4.9)	0.5258	0.1112	0.1654	0.8305
Varying π (4.10)	0.5265	0.0959	0.182	0.8304
Varying π and λ (4.11)	0.5257	0.1159	0.1776	0.8304
Varying λ (4.12)	0.5234	0.1114	0.0928	0.8316
Varying π (4.13)	0.5256	0.0579	0.1738	0.8293
Varying π and λ (4.14)	0.5199	0.0687	0.0588	0.8336
True	0.5150	0.0000	0.0000	0.8310

Remarks. The running time in this example takes only several seconds since there are only two regression functions F and G to be estimated. When the mixture of models contains $K > 2$ components, the boosting of mixing probabilities takes much more time since K boosting algorithms are conducted in each EB iteration. This will be discussed in the next example.

4.2 Second 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 component Gaussians are homogeneous among all samples. The probability density function is given by

$$f(y|\boldsymbol{x}; \mu, \sigma, p) = p_1(\boldsymbol{x}) f_N(y; \mu_1, \sigma_1) + p_2(\boldsymbol{x}) f_N(y; \mu_2, \sigma_2) + p_3(\boldsymbol{x}) f_N(y; \mu_3, \sigma_3), \tag{4.15}$$

where $f_N(\cdot; \mu, \sigma)$ 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 vector

$\boldsymbol{x} = (x_1, x_2, x_3, x_4)$ via the following equations

$$p_k(\boldsymbol{x}) = \frac{\exp F_k(\boldsymbol{x})}{\sum_{l=1}^{3} \exp F_l(\boldsymbol{x})}, \text{ for } k = 1, 2, 3, \tag{4.16}$$

and

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

Note that covariate x_4 is a redundant variable and the relationship between mixing probabilities and covariates is non-linear with interaction. 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 Γ. We generate $n = 12,000$ samples, among which $10,000$ samples are learning data and $2,000$ samples are test data. We fit the following four mixtures of models based on generalized linear model:

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

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

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

$$f(y|\boldsymbol{x}; p, \mu, \sigma) = p_1 f_N(y; \mu_1(\boldsymbol{x}), \sigma_1) + p_2 f_N(y; \mu_2(\boldsymbol{x}), \sigma_2) + p_3 f_N(y; \mu_3(\boldsymbol{x}), \sigma_3), \tag{4.18}$$

  where

$$\mu_1(\boldsymbol{x}) = \langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle, \mu_2(\boldsymbol{x}) = \langle \boldsymbol{\beta}, \boldsymbol{x} \rangle, \mu_3(\boldsymbol{x}) = \langle \boldsymbol{\gamma}, \boldsymbol{x} \rangle. \tag{4.19}$$

- Mixture models with homogeneous mean and heterogeneous mixing probabilities modeled by a multiclass logistic regression:

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

  where

$$p_k(\boldsymbol{x}) = \frac{\exp F_k(\boldsymbol{x})}{\sum_{l=1}^{3} \exp F_l(\boldsymbol{x})}, \text{ for } k = 1, 2, 3, \tag{4.21}$$

  and

$$F_1(\boldsymbol{x}) = \langle \boldsymbol{a}, \boldsymbol{x} \rangle, F_2(\boldsymbol{x}) = \langle \boldsymbol{b}, \boldsymbol{x} \rangle, F_3(\boldsymbol{x}) = \langle \boldsymbol{c}, \boldsymbol{x} \rangle. \tag{4.22}$$

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

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

$$f(y|\boldsymbol{x};p,\mu,\sigma)=p_1(\boldsymbol{x})f_N(y;\mu_1(\boldsymbol{x}),\sigma_1)+p_2(\boldsymbol{x})f_N(y;\mu_2(\boldsymbol{x}),\sigma_2)+p_3(\boldsymbol{x})f_N(y;\mu_3(\boldsymbol{x}),\sigma_3) \tag{4.23}$$

  where the means and mixing probabilities follow equations (4.19), (4.21) and (4.22).

The parameters in above mixture models can be estimated by the EM algorithm. Next, we fit the following three mixture of boosting models by the proposed EB algorithm:

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

$$f(y|\boldsymbol{x};p,\mu,\sigma)=p_1f_N(y;\mu_1(\boldsymbol{x}),\sigma_1)+p_2f_N(y;\mu_2(\boldsymbol{x}),\sigma_2)+p_3f_N(y;\mu_3(\boldsymbol{x}),\sigma_3), \tag{4.24}$$

  where μ_1,μ_2,μ_3 are estimated non-parametrically via the boosting algorithm. This model is comparable to model (4.18).

- Mixture models with heterogeneous mixing probabilities modeled by boosting:

$$f(y|\boldsymbol{x};p,\mu,\sigma)=p_1(\boldsymbol{x})f_N(y;\mu_1,\sigma_1)+p_2(\boldsymbol{x})f_N(y;\mu_2,\sigma_2)+p_3(\boldsymbol{x})f_N(y;\mu_3,\sigma_3) \tag{4.25}$$

  where

$$p_k(\boldsymbol{x})=\frac{\exp F_k(\boldsymbol{x})}{\sum_{l=1}^3\exp F_l(\boldsymbol{x})},\ \text{for } k=1,2,3. \tag{4.26}$$

  Here F_1,F_2,F_3 are estimated non-parametrically via boosting algorithm. This model is comparable to model (4.20).

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

$$f(y|\boldsymbol{x};p,\mu,\sigma)=p_1(\boldsymbol{x})f_N(y;\mu_1(\boldsymbol{x}),\sigma_1)+p_2(\boldsymbol{x})f_N(y;\mu_2(\boldsymbol{x}),\sigma_2)+p_3(\boldsymbol{x})f_N(y;\mu_3(\boldsymbol{x}),\sigma_3) \tag{4.27}$$

  where the mean and mixing probabilities are estimated non-parametrically via boosting algorithm. This model is comparable to model (4.23).

We summarize the results of the seven models in Table 2 including average negative log-likelihood on the test data, and the running time. Note that the true negative log-likelihood is calculated based on the simulated parameters $p(\boldsymbol{x}),\mu,\sigma$. All the three models with linear predictors (4.18), (4.20) and (4.23) have a larger test loss than their counterpart boosted models (4.24), (4.25) and (4.27) since they cannot capture non-linear effects and interactions of covariates. Thus, when there are complicated effects of covariates on the parameters, the EB algorithm is better than the EM algorithm. The two models with heterogeneous mean and mixing probabilities (4.23) and (4.27) have no obvious better out-of-sample performance than their counterpart models with homogeneous mean (4.20) and (4.25), respectively. The two models with heterogeneous mixing probabilities (4.20) and (4.25) have a better out-of-sample performance than their counterpart models with homogeneous mixing probabilities (4.17). Thus, an appropriate mixing structure is heterogeneous mixing probabilities but with homogeneous mean.

Comparing the running time without parallel computing of models (4.24) and (4.25), it implies that the boosting of mixing probabilities takes the majority of running time in the EB algorithm (rather than the boosting of component means). We accelerate the EB algorithm

Table 2: Comparison of different models in terms of test loss and running time.[Not easy to distinguish the results between EM and EB]

model	test loss	running time without parallel computing	running time with parallel computing
Null (4.17)	2.4821	$<$ 10 seconds	NA
Varying μ (4.18)	2.4779	$<$ 10 seconds	NA
Varying p (4.20)	2.2016	$<$ 10 seconds	NA
Varying μ and p (4.23)	2.2011	$<$ 10 seconds	NA
Varying μ (4.24)	2.4139	$<$ 10 seconds	NA
Varying p (4.25)	2.1330	650 seconds	381 seconds
Varying μ and p (4.27)	2.1320	1054 seconds	597 seconds
True	2.1231	NA	NA

by applying parallel computing for the boosting of mixing probabilities in step 3.1 of the EB algorithm, i.e., we fit K trees simultaneously in each inner boosting iteration m. It shows that parallel computing accelerates the EB algorithm significantly.

In the EB algorithm, we need to determine when to stop inner boosting loop and when to stop outer EB loop. We further partition the learning data into training data and validation data. We early stop the inner boosting iteration according to validation loss and stop the outer EB iteration according to training loss. Early stop of the inner boosting iteration can avoid overfitting in each EB iteration, while the outer EB iteration should stop at the convergence of the training loss.

4.3 Third example: claims severity modelling

[Is it necessary to separate the simulation and real data analysis into two subsections? Instead of saying the real data analysis the third example] 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). We plot the histogram and the cumulative distribution 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 modeled 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 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 of gamma distribution, and α, M are tail index and threshold of Pareto distribution. The logged survival function for logged average claims as shown in Figure 2 indicates a regularly varying tail at infinity (Embrechts et al., 1997). The threshold of Pareto distribution is selected as $M = 8158.13$ according to the Hill plot (Resnick, 1997) as shown in Figure 2. Please refer to Wüthrich and Merz (2022) for more discussions on this data.

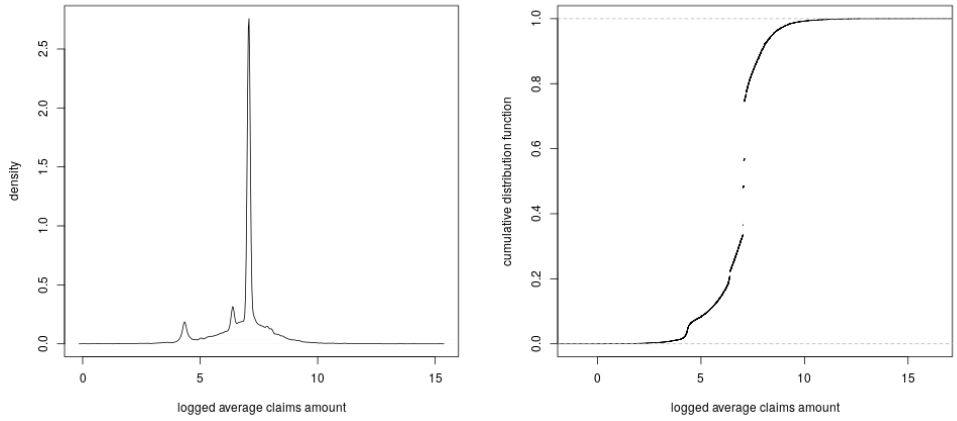

Figure 1: Histogram and the cumulative distribution function of logged average claims.

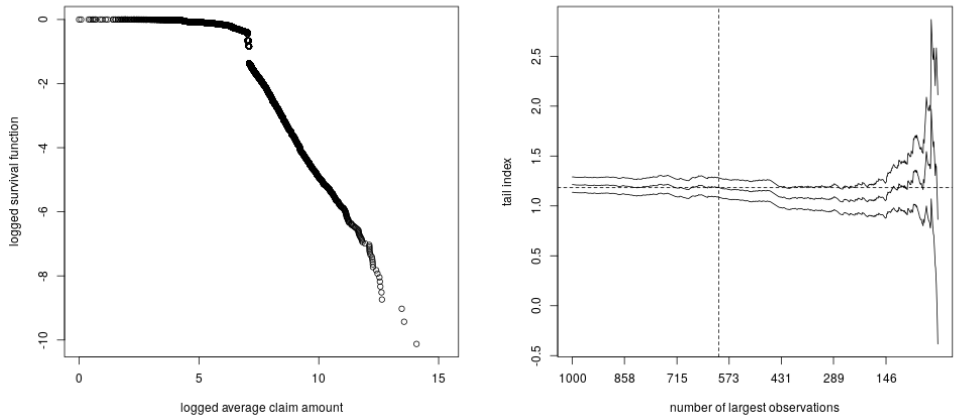

Figure 2: The logged survival function and the Hill plot for logged average claims.

The three peaks as shown in Figure 1 imply a way of initialization through the hidden component indicator variable:

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

Component parameters μ, ϕ, α can be initialized as the MLEs in a homogeneous model for the full information $(y_i, \hat{\boldsymbol{z}}_i^{[0]}, \boldsymbol{x}_i)_{i=1:n}$, i.e., the MLEs of parameters in gamma/Pareto distribution fitted to the partial samples in five intervals $(0, 500]$, $(500, 1000]$, $(1000, 1200]$, $(1200, 8158.13]$, $(8158.13, \infty)$, respectively.

We first fit a mixture of distributions (4.28) to the data via the EM algorithm. The trace of learning loss is shown in Figure 3, implying a convergence after 80 iterations. 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 dispersion) imply the difficulties with mean modelling in the first three component models. The

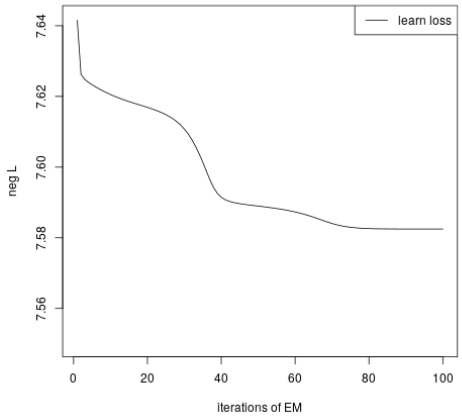

Figure 3: The trace plot the learning loss for the homogeneous model (4.28) during the EM iterations.

Table 3: The MLEs of component parameters in the homogeneous model (4.28). 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

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 $\hat{p}_5 = 0.0218$.

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

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

We draw the boxplot of the mixing probabilities in Figure 4. We observe that the mixing probabilities for the 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.

Finally, 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 mixture model:

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

The test loss is calculated as 7.5573 slightly smaller than that for model (4.30).

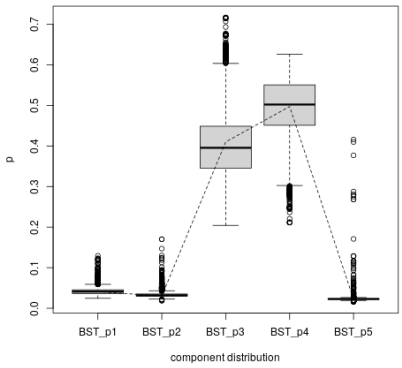

Figure 4: The boxplot of the estimated mixing probabilities in Model (4.30).

5 Conclusions

Insurance loss data sometimes cannot be sufficiently modeled by a single distribution, so a mixture of models is usually more preferred. Since there are multiple component models in a mixture of models, the parameter estimation problems 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 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 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. Automated feature engineering is performed during the EB algorithm. Second, boosting algorithm is overfitting-sensitive, and variable selection is conducted simultaneously during the EB algorithm. The cost of more flexibility is more computing time than the EM algorithm since there are several inner boosting loops in each EB iteration.[I don't understand this sentence] We can accelerate the EB algorithm by conducting all the inner boosting loops simultaneously since they are independent given the component indicator variable z.

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) 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. (1997). *Modelling Extremal Events for Insurance and Finance*, Springer.

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. and Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction 2nd Edition*, 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.

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

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.