Response versus gradient boosting trees, GLMs and neural networks under Tweedie loss and log-link

Donatien Hainaut
Institute of Statistics, Biostatistics and Actuarial Science
UCLouvain
Louvain-la-Neuve, Belgium

Julien Trufin
Department of Mathematics
Université Libre de Bruxelles (ULB)
Brussels, Belgium

Michel Denuit
Institute of Statistics, Biostatistics and Actuarial Science
UCLouvain

January 27, 2022

Abstract

Thanks to its outstanding performances, boosting has rapidly gained wide acceptance among actuaries. To speed up calculations, boosting is often applied to gradients of the loss function, not to responses (hence the name gradient boosting). When the model is trained by minimizing Poisson deviance, this amounts to apply the least-squares principle to raw residuals. This exposes gradient boosting to the same problems that lead to replace least-squares with Poisson Generalized Linear Models (GLM) to analyze low counts (typically, the number of reported claims at policy level in personal lines). This paper shows that boosting can be conducted directly on the response under Tweedie loss function and log-link, by adapting the weights at each step. Numerical illustrations demonstrate similar or better performances compared to gradient boosting when trees are used as weak learners, with a higher level of transparency since responses are used instead of gradients.

Keywords: Risk classification, Boosting, Gradient Boosting, Regression Trees, GLM, Neural Networks.

1 Introduction and motivation

Boosting emerged from the field of machine learning and became rapidly popular among insurance analysts. Broadly speaking, boosting is an iterative fitting procedure using weak, or base learners. In a regression context, weak learners have rigid parametric forms and cannot accurately adapt to the data under consideration. In each iteration, boosting fits a weak learner that improves the fit of the overall model such that the ensemble arrives at an accurate prediction. Thus, the score is not specified by the actuary and estimated at once, as in generalized linear models (GLMs) or generalized additive models (GAMs) but it is built sequentially, adding terms that weakly improve on the estimations obtained from the preceding step.

Boosting requires that the analyst first decides which metric should be optimized for estimating the score. In the boosting terminology, this metric is usually referred to as the "loss function". In insurance applications, loss functions generally correspond to negative log-likelihood, or deviance functions. L1, L2 or Huber loss functions are popular choices in other disciplines. Boosting works by adding in each iteration, the newly fitted base learner to the previously estimated score, as shown in Algorithm 1. The analyst thus has to decide which base learner to use. Boosting is particularly effective when base learners are trees of limited depth. But the base learner can also be linear, or a penalized spline with a relatively small number of degrees of freedom. Each feature can even possess its own base learners. For instance, one could model the effect of the first feature on the score using a tree, and the one of the second feature using a linear model. In every boosting iteration, only the best-performing base learner and hence the best-performing feature is included in the final model. We refer the readers to Denuit et al. (2019b, 2020) for an extensive treatment of boosting in the context of insurance, with applications to tree-based methods and neural networks, and to the review papers by Mayr et al. (2014a,b) for a general introduction to the topic.

To ease numerical aspects, boosting is often applied on gradients of the loss function, that is, on the gradients of the deviance function in insurance applications. Instead of maximizing the log-likelihood associated with the response, gradient boosting applies a least-squares principle on its gradients, as described in Algorithm 2. In this form, boosting has become very popular among data analysts and several packages available in open-source software implement highly effective boosting algorithms such as the Extreme Gradient boosting (XGBoost) algorithm, for instance.

There have been numerous applications of gradient boosting to insurance pricing in the last decade. We refer the interested reader to Lee and Lin (2018) for an extensive review of the different boosting algorithms that have been proposed so far. Let us just mention a few of them, to demonstrate the wide applicability of this approach in this context. Guelman (2012) applied gradient boosted trees to predict motor insurance losses. Liu et al. (2014) considered claim frequencies, using multi-class AdaBoost trees. Yang et al. (2018) adapted gradient boosted tree algorithm to Tweedie models. Pesantez-Narvaez et al. (2019) employed XGBoost to predict the occurrence of claims using telematics data. Henckaerts et al. (2021) worked with random forests and boosted trees to develop full tariff plans built from both the frequency and severity of claims.

However, boosting is not limited to gradients and can also be regarded as an iteratively re-

weighted, or re-offseted procedure applied to the original data, as shown in Proposition 3.1. This alternative view to boosting makes the approach very intuitive and better adapted to the shape of data commonly encountered in insurance applications (low counts for claim numbers in personal lines and highly skewed severity data). In a sense, this revives the early examples of boosting procedures which were not based on gradients such as the popular Adaboost algorithm. The central idea behind AdaBoost, an early form of boosting for classification is to iteratively re-weight the observations so that observations that were misclassified in the preceding step get higher weights in the next step. This forces the algorithm to concentrate on the cases which are hard to classify, boosting its accuracy. As shown in Friedman et al. (2000), Adaboost turns out to be equivalent to forward stagewise additive modeling based on an exponential loss function that yields the same solution at population level as Binomial likelihood.

Interpreting boosting as a functional gradient descent algorithm opened the door to responses obeying distributions in the Exponential Dispersion family (Poisson, Gamma or Tweedie, for instance), commonly used in insurance applications since the adoption of GLMs for insurance pricing and reserving in the 1990s. Building on the wide acceptance of GLMs in the actuarial community, gradient boosting rapidly reached a dominant position among the machine learning algorithms applied in insurance applications since its introduction in Friedman (2001). In this approach, estimation is performed by a forward stagewise procedure, fitting the base learner not to re-weighted observations, as in AdaBoost, but to the negative gradient vector of the loss function evaluated at the previous iteration.

For Normal responses, gradient boosting consists in including the prediction from the last iteration in the offset. This is because gradients reduce to the raw residuals with the squared error function (that is, the negative Normal log-likelihood). In the Normal case (see Section 2.4), the boosting algorithm thus re-fits the residuals from the preceding step, adding the effect of the feature that best improves model performances. Equivalently, the prediction obtained in the previous iteration is put in offset when including a new term in the score. Gradient boosting thus reduces to an iteratively re-offseting procedure in the Normal case, still working with the responses under consideration by modifying offsets at each step.

However, gradient boosting beyond the Normal case typically faces the same problem that lead to the adoption of GLMs for insurance applications. Instead of maximizing the log-likelihood, gradient boosting applies a least-squares principle on its gradients. In Poisson regression for instance, gradient boosting consists in fitting raw residuals by least squares. However, we know that least-squares is outperformed by Poisson regression for low counts, as those encountered in pricing personal lines. Indeed, for low expected claim frequencies, residuals (computed as the numbers of claims minus the expected claim frequencies) are concentrated around integer values. For severities modeled by the Gamma distribution, gradients obey a translated Gamma distribution, which can be markedly skewed and rules out the least-squares principle. For general Tweedie response corresponding to compound Poisson sums with Gamma-distributed summands, the abundance of zeroes and the mixture of Gamma distributions for positive outcomes also result in highly dissymmetric gradients and poor performances of least-squares estimation. These comments are based on the general expressions (3.7) for gradients obtained from Tweedie responses. This clearly shows that applying the least-squares principle on gradients is not effective and exposes actuaries to the same deficiency that led to the massive adoption of GLMs in the 1990s, in lieu of Gaussian

linear models.

In likelihood-based boosting, the loss function is minus the log-likelihood, or equivalently the deviance, so that any distribution within the Exponential Dispersion family can be used. Likelihood-based boosting was designed to solve problems appearing in GAMs by adopting stagewise training of the base learners. This is the case with the GAMBoost procedure proposed by Tutz and Binder (2006). The idea of likelihood-based boosting is to include base-learners that directly maximize an overall likelihood in each boosting step. Given a starting value or estimate from a previous boosting step, likelihood-based boosting approaches use base learners for estimating a new effect correcting the previous estimation in a log-likelihood that contains the previous score as a fixed offset. To speed up the calculations, base learners are generally estimated using one step of the iterative re-weighted least squares algorithm, using the current score as offset. Compared to gradient boosting, likelihood-based boosting thus works with the responses, not with gradients.

The present contribution is closely related to generalized additive modeling by likelihood-based boosting proposed by Tutz and Binder (2006). These authors suggested to resort to penalized regression splines and penalized stumps (i.e. trees with only two terminal nodes) as weak learners and to perform a single Fisher scoring step to estimate the corresponding parameters. In the present paper, we show that Fisher scoring is even not needed when trees are used as weak learners under Tweedie loss with log-link, provided the responses are re-weighted and re-scaled at each boosting step. We also allow for more types of weak learners, including neural networks and larger trees (compared to stumps), but no penalty is included. The aim of this paper is to show that responses can be used directly and that there is no need to replace them with gradients as long as Tweedie loss is adopted, with log-link.

For Normally-distributed responses, including an offset is the same as fitting a model to the residuals from the previous boosting step, and likelihood maximization by a base learner becomes standard least-squares estimation with respect to these residuals. In this special case, likelihood-based boosting thus coincides with gradient boosting. For the kind of responses encountered in actuarial applications, such as claim frequencies and severities on individual policies, likelihood-based boosting improves on gradient boosting by recognizing the particular shape of the responses (whereas gradient boosting applies a least-squares principle, whatever the range of the gradients).

This paper aims to show that there is often no need to boost gradients in insurance applications, since boosting can easily be performed directly with responses under Tweedie loss function and log-link. Given that this setting is now widely adopted by actuaries to conduct their analyzes, the results of this paper are widely applicable to insurance. From a methodological point of view, we extend in Proposition 3.1 the results obtained by Wüthrich and Buser (2019) in the Poisson case to the whole Tweedie family under log-link. In the Poisson case, this link is the canonical one and Wüthrich and Buser (2019) demonstrated that there was no need to resort to gradient boosting when using regression trees as weak learners. Indeed, at each iteration, the structure of the new tree is obtained by least-squares under gradient boosting, facing the well-known problems with low counts, while the Poisson deviance loss can be used without difficulty provided exposure-to-risks are adapted at each step. Gradient boosting trees thus introduce an extra step which leads to an unnecessary approximation in the Poisson case. In this paper, we show that this property extends to

the whole Tweedie family by an appropriate modification of weights at each boosting step, provided the analysis is conducted under log-link. As an illustration, we use trees, GLMs and neural networks as weak learners.

Lee and Lin (2018) introduced Delta Boosting Machine as a new member of the boosting family, with application to general insurance. See also Lee (2020). The present contribution is intimately connected with the work by Lee and Lin (2018) who considered various loss functions (whereas we concentrate on the Tweedie one in this paper) but restricted their study to trees as weak learners (whereas we also allow here for GLMs and Neural Networks). In their Theorem 3, Lee and Lin (2018) established that it is optimal to partition data on the basis of quantities called individual deltas (precisely defined in formula (4) of their paper) under Tweedie loss and log-link. It turns out that individual deltas are precisely the log of modified responses used at each step of the likelihood-based boosting procedure developed in this paper. The analysis conducted here indeed confirms the previous finding by Lee and Lin (2018), that gradients should be replaced with individual deltas when trees are used as weak learners, and strengthens it in two ways under Tweedie loss and log-link: first by considering also other types of weak learners and second by showing that gradients are in fact not needed because likelihood-based boosting is feasible by iteratively re-weighting and re-scaling the response. The contribution of this paper is thus methodological, showing that responses can be used instead of gradients so that the procedure is more transparent. Numerical illustrations demonstrate the feasibility of the approach, with similar or better performances on the Wasa data set used by Ohlsson and Johansson (2010). Even if no general conclusion can be drawn from an analysis conducted on a particular data set, this shows that the proposed method can be implemented in practice.

The remainder of this paper is organized as follows. Section 2 recalls the boosting principle and its forward stagewise additive modeling. The particular Normal and Poisson cases are considered there, showing that gradients are not needed to conduct boosting under the corresponding deviance. Section 3 extends the results in the Poisson case to the whole Tweedie family, under log-link function. Section 4 is devoted to a case study using trees, GLMs and neural networks as weak learners. The final Section 5 summarizes the results and concludes.

2 Boosting

2.1 Insurance pricing

In actuarial pricing, the aim is to evaluate the pure premium as accurately as possible. The target is the conditional expectation $\mu(\boldsymbol{X}) = \mathrm{E}[Y|\boldsymbol{X}]$ of the response Y (claim number or claim amount for instance) given the available information summarized in a vector $\boldsymbol{X}$ of features $X_1, X_2, \ldots, X_p$. The function $\boldsymbol{x} \mapsto \mu(\boldsymbol{x}) = \mathrm{E}[Y|\boldsymbol{X} = \boldsymbol{x}]$ is unknown to the actuary and is approximated by a working predictor $\boldsymbol{x} \mapsto \widehat{\mu}(\boldsymbol{x})$ entering premium calculation.

Lack of accuracy for $\widehat{\mu}(\boldsymbol{x})$ is defined by the generalization error

$$Err(\widehat{\mu}) = \mathrm{E}\left[L(Y, \widehat{\mu}(\boldsymbol{X}))\right],$$

where $L(.,.)$ is the loss function measuring the discrepancy between its two arguments and the

expected value is over the joint distribution of $(Y, \boldsymbol{X})$. We aim to find a function $\boldsymbol{x} \mapsto \widehat{\mu}(\boldsymbol{x})$ of the features minimizing the generalization error.

We denote by

$$\mathcal{D} = \{(y_1, \boldsymbol{x}_1), (y_2, \boldsymbol{x}_2), \ldots, (y_n, \boldsymbol{x}_n)\}, \tag{2.1}$$

the set of observations used to fit the model $\widehat{\mu}$, called training set. An estimate $\widehat{\mu}(\boldsymbol{x})$ to $\mu(\boldsymbol{x})$ is obtained by applying a training procedure to the training set $\mathcal{D}$. Common training procedures restrict $\mu(\boldsymbol{x})$ to parametrized classes of functions. More specifically, a supervised training procedure assumes that μ belongs to a family of candidate models of restricted structure. Model selection is then defined as finding the best model within this family on the basis of the training set. Of course, the true model is usually not a member of the family under consideration, depending on the restricted structure imposed to the candidate models. However, when these restrictions are flexible enough, some candidate models could be sufficiently close to the true model to provide an accurate pricing structure.

2.2 Ensemble learning methods

Ensemble techniques assume structural models of the form

$$\mu(\boldsymbol{x}) = g^{-1}\left(\text{score}(\boldsymbol{x})\right) = g^{-1}\left(\sum_{m=1}^{M} T(\boldsymbol{x}; \mathbf{a}_m)\right), \tag{2.2}$$

where g is the link function and $T(\boldsymbol{x}; \mathbf{a}_m)$, $m = 1, 2, \ldots, M$, are usually simple functions of the features $\boldsymbol{x}$, characterized by parameters $\mathbf{a}_m$. In (2.2), the score is the function of features $\boldsymbol{x}$ mapped to $\mu(\boldsymbol{x})$ by the inverse of the link function g. Estimating a score of the form

$$\text{score}(\boldsymbol{x}) = \sum_{m=1}^{M} T(\boldsymbol{x}; \mathbf{a}_m),$$

by minimizing the corresponding training sample estimate of the generalized error

$$\min_{\{\mathbf{a}_m\}_1^M} \sum_{i=1}^{n} L\left(y_i, g^{-1}\left(\sum_{m=1}^{M} T(\boldsymbol{x}_i; \mathbf{a}_m)\right)\right), \tag{2.3}$$

is in general infeasible. It requires computationally intensive numerical optimization techniques. One way to overcome this problem is to approximate the solution to (2.3) by using a greedy forward stagewise approach, also called boosting.

2.3 Forward stagewise additive modeling

Forward stagewise additive modeling consists in sequentially fitting a single function and adding it to the expansion of prior fitted terms. Each fitted term is not readjusted as new terms are added into the expansion, contrarily to a stepwise approach where previous terms are each time readjusted when a new one is added. Specifically, we start by computing

$$\widehat{\mathbf{a}}_1 = \underset{\mathbf{a}_1}{\text{argmin}} \sum_{i=1}^{n} L\left(y_i, g^{-1}\left(\widehat{\text{score}}_0(\boldsymbol{x}_i) + T(\boldsymbol{x}_i; \mathbf{a}_1)\right)\right), \tag{2.4}$$

where $\widehat{\text{score}_0}(\boldsymbol{x})$ is an initial guess (for instance, just an intercept). Then, at each iteration $m \geq 2$, we solve the subproblem

$$\widehat{\mathbf{a}}_m = \underset{\mathbf{a}_m}{\text{argmin}} \sum_{i=1}^{n} L\left(y_i, g^{-1}\left(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i; \mathbf{a}_m)\right)\right) , \tag{2.5}$$

with

$$\widehat{\text{score}}_{m-1}(\boldsymbol{x}) = \widehat{\text{score}}_{m-2}(\boldsymbol{x}) + T(\boldsymbol{x}; \widehat{\mathbf{a}}_{m-1}).$$

This leads to the following algorithm:

Algorithm 1 Forward Stagewise Additive Modeling.

---

1. Initialization :

Initialize $\widehat{\text{score}_0}(\boldsymbol{x})$ to be a constant. For instance:

$$\widehat{\text{score}_0}(\boldsymbol{x}) = \underset{\beta}{\text{argmin}} \sum_{i=1}^{n} L(y_i, g^{-1}(\beta)).$$

2. Main procedure :

For $m = 1$ to M **do**

a) Compute $\widehat{\mathbf{a}}_m$ as defined in Equation (2.4).

b) Update $\widehat{\text{score}}_m(\boldsymbol{x}) = \widehat{\text{score}}_{m-1}(\boldsymbol{x}) + T(\boldsymbol{x}; \widehat{\mathbf{a}}_m)$.

End for

3. **Output:**

$$\widehat{\mu}_{\mathcal{D}}^{\text{boost}}(\boldsymbol{x}) = g^{-1}\left(\widehat{\text{score}}_M(\boldsymbol{x})\right).$$

The forward stagewise additive modeling described in Algorithm (1) is also called boosting. Boosting is thus an iterative method based on the idea that combining many simple functions should result in a powerful one. In a boosting context, the simple functions $T(\boldsymbol{x}; \mathbf{a}_m)$ are called weak learners or base learners.

There is a large variety of weak learners available for boosting models. For instance, commonly used weak learners are wavelets, multivariate adaptive regression splines, smoothing splines, classification trees and regression trees or neural networks. For instance, the "parameters" $\mathbf{a}_m$ represent the splitting variables and their split values as well as the corresponding predictions in the terminal nodes for regression trees. The two next subsections detail the step 2.1 of the forward stagewise additive approach for squared error loss and Poisson deviance loss functions.

2.4 Squared error loss

The most common criterion to estimate the model is the squared-error loss

$$L(y, \widehat{\mu}(\boldsymbol{x})) = (y - \widehat{\mu}(\boldsymbol{x}))^2 ,$$

together with the identity link function. Then, step 2.1 in Algorithm (1) simplifies to

$$\begin{aligned}\widehat{\mathbf{a}}_m &= \underset{\mathbf{a}_m}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - \widehat{\operatorname{score}}_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i;\mathbf{a}_m)\right)^2 \\ &= \underset{\mathbf{a}_m}{\operatorname{argmin}} \sum_{i=1}^{n} \left(r_{i,m} + T(\boldsymbol{x}_i;\mathbf{a}_m)\right)^2,\end{aligned}$$

where $r_{i,m}$ is the residual of the model after $m-1$ iterations on the ith observation. One sees that the term $T(\boldsymbol{x};\mathbf{a}_m)$ actually fits the residuals obtained after $m-1$ iterations. Finding the solution to (2.5) is thus no harder than for a single weak learner. It amounts to fit a weak learner on the working training set

$$\mathcal{D}^{(m)} = \{(r_{i,m}, \boldsymbol{x}_i), i = 1, \ldots, n\}.$$

Notice that minimizing the squared-error loss function is equivalent to maximizing the log-likelihood of responses, y, that are assumed Normal. For non-Gaussian responses, such as low count Poisson data, the squared-error loss is then less appropriate than deviance loss function, as detailed in the next section.

2.5 Poisson deviance loss

The squared loss function of the previous paragraph is also the deviance of normally distributed observations. In actuarial applications, such as claim frequency modeling, another distribution of interest is the Poisson one. In this case, we have to consider (2.5) with the Poisson deviance loss

$$L(y, \widehat{\mu}(\boldsymbol{x})) = 2\left(y \ln \frac{y}{\widehat{\mu}(\boldsymbol{x})} - (y - \widehat{\mu}(\boldsymbol{x}))\right),$$

and the log-link function (which is the canonical link function for the Poisson distribution). In actuarial studies, this choice is often made to model the number of claims, so that we also account for the length of the observation period e referred to as the exposure-to-risk. In such a case, one observation of the training set can be described by the claims count y_i, the features $\boldsymbol{x}_i$ and the exposure-to-risk e_i, so that we have

$$\mathcal{D} = \{(y_i, \boldsymbol{x}_i, e_i), i = 1, \ldots, n\}.$$

In this context, (2.5) becomes

$$\begin{aligned}\widehat{\mathbf{a}}_m &= \underset{\mathbf{a}_m}{\operatorname{argmin}} \sum_{i=1}^{n} L\left(y_i, e_i \exp\left(\widehat{\operatorname{score}}_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i;\mathbf{a}_m)\right)\right) \\ &= \underset{\mathbf{a}_m}{\operatorname{argmin}} \sum_{i=1}^{n} L\left(y_i, e_i \exp\left(\widehat{\operatorname{score}}_{m-1}(\boldsymbol{x}_i)\right) \exp\left(T(\boldsymbol{x}_i;\mathbf{a}_m)\right)\right),\end{aligned}$$

which can be expressed as

$$\widehat{\mathbf{a}}_m = \underset{\mathbf{a}_m}{\operatorname{argmin}} \sum_{i=1}^{n} L\left(y_i, e_{i,m} \exp\left(T(\boldsymbol{x}_i;\mathbf{a}_m)\right)\right),$$

with

$$e_{i,m} = e_i \exp\left(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i)\right) ,$$

for $i = 1, \ldots, n$. At iteration m, $e_{i,m}$ is a constant and can be regarded as a working exposure-to-risk applied to observation i.

Again, solving (2.5) is thus no harder than for a single weak learner. This is equivalent to building a single weak learner with the Poisson deviance loss and the log-link function on the working training set

$$\mathcal{D}^{(m)} = \{(y_i, \boldsymbol{x}_i, e_{i,m}) , i = 1, \ldots, n\} .$$

This observation has been first made by Wüthrich and Buser (2019). In comparison, gradient boosting, introduces an extra step which leads to an unnecessary approximation. For the sake of completeness, gradient boosting algorithm is recalled in appendix A. We can see there that at each iteration, the structure of the new estimator is obtained by approximating the gradient of the generalized error in the least-squares (step 2.b, see Appendix A) senses instead of minimizing directly the Poisson deviance loss. For low counts, such as those encountered in personal insurance lines, this exposes gradient boosting to the same problems that lead actuaries to adopt GLM for pricing.

In the next section, we establish that the result valid in the Poisson case also holds more generally, for the whole Tweedie family under log-link.

3 Boosting with Tweedie loss function under log-link

3.1 Tweedie family of distributions

In practice, actuaries often use distributions that belong to the Tweedie class together with the log-link function for modeling responses. The log-link function is chosen mainly because of the multiplicative structure it produces for the resulting estimates. The Tweedie class regroups the members of the Exponential Dispersion family having power variance functions $V(\mu) = \mu^\xi$ for some ξ.

Specifically, the Tweedie class contains continuous distributions such as the Normal, Gamma and Inverse Gaussian distributions. It also includes the Poisson and compound Poisson-Gamma distributions. Compound Poisson-Gamma distributions can be used for modeling annual claim amounts, having positive probability at zero and a continuous distribution on the positive real numbers. In practice, annual claim amounts are often decomposed into claim numbers and claim severities and separate analyses of these quantities are conducted. Typically, the Poisson distribution is used for modeling claim counts and the Gamma or Inverse Gaussian distributions for claim severities. We refer the reader to Delong et al. (2021) for a through presentation of Tweedie models and their application to insurance.

Table 3.1 gives a list of all Tweedie distributions. Negative values of ξ gives continuous distributions on the whole real axis. For $0 < \xi < 1$, no member of the Exponential Dispersion family exists. Only the cases $\xi \geq 1$ are thus interesting for applications to insurance.

	Type	Name
$\xi < 0$	Continuous	-
$\xi = 0$	Continuous	Normal
$0 < \xi < 1$	Non existing	-
$\xi = 1$	Discrete	Poisson
$1 < \xi < 2$	Mixed, non-negative	Compound Poisson-Gamma
$\xi = 2$	Continuous, positive	Gamma
$2 < \xi < 3$	Continuous, positive	-
$\xi = 3$	Continuous, positive	Inverse Gaussian
$\xi > 3$	Continuous, positive	-

Table 3.1: Tweedie distributions.

3.2 Tweedie deviance-based boosting under log-link

From e.g. Denuit et al. (2019a, Table 4.7 pp 153), Tweedie deviance loss function is given by

$$L(y,\widehat{\mu}(\boldsymbol{x})) = \begin{cases} (y-\widehat{\mu}(\boldsymbol{x}))^2 & \text{if } \xi = 0\,, \\ 2\left(y\ln\frac{y}{\widehat{\mu}(\boldsymbol{x})} - (y-\widehat{\mu}(\boldsymbol{x}))\right) & \text{if } \xi = 1, \\ 2\left(-\ln\frac{y}{\widehat{\mu}(\boldsymbol{x})} + \frac{y}{\widehat{\mu}(\boldsymbol{x})} - 1\right) & \text{if } \xi = 2, \\ 2\left(\frac{\max\{y,0\}^{2-\xi}}{(1-\xi)(2-\xi)} - \frac{y\widehat{\mu}(\boldsymbol{x})^{1-\xi}}{1-\xi} + \frac{\widehat{\mu}(\boldsymbol{x})^{2-\xi}}{2-\xi}\right) & \text{otherwise } (\xi > 0)\,. \end{cases} \tag{3.1}$$

For $\xi = 0$, we recover the L2 loss function whereas $\xi = 1$ and 2 correspond to the Poisson and Gamma deviance functions, respectively.

Let us revisit the Poisson case ($\xi = 1$) to understand the extension to the whole Tweedie family. In the Poisson case, we have noticed that

$$L\left(y_i, e_i\exp\left(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i;\mathbf{a}_m)\right)\right) = L\left(y_i, e_{i,m}\exp\left(T(\boldsymbol{x}_i;\mathbf{a}_m)\right)\right)\,, \tag{3.2}$$

with $e_{i,m} = e_i\exp\left(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i)\right)$. For a Poisson response Y, we know that the actuary is allowed to work either with the observed claim count Y or with the observed claim rate, or claim frequency $\widetilde{Y} = Y/e$ provided the weight $\nu = e$ enters the analysis. The distribution of the claim rate $\widetilde{Y}$ still belongs to the Tweedie class and is called the Poisson rate distribution. See Property 2.5.1 in Denuit et al. (2019a) for more details. This is reflected by the fact that the Poisson deviance loss satisfies

$$L\left(y_i, e_i\exp\left(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i;\mathbf{a}_m)\right)\right) = \nu_i L\left(\tilde{y}_i, \exp\left(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i;\mathbf{a}_m)\right)\right)\,, \tag{3.3}$$

with $\nu_i = e_i$ and $\tilde{y}_i = \frac{y_i}{e_i}$. Working with the claim rates $\tilde{y}_i$ then yields

$$\begin{aligned}
&\nu_i L(\tilde{y}_i, \exp(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i; \mathbf{a}_m))) \\
&= \nu_i 2 \left(\tilde{y}_i \ln \left(\frac{\tilde{y}_i}{\exp(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i; \mathbf{a}_m))} \right) - \left(\tilde{y}_i - \exp(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i; \mathbf{a}_m)) \right) \right) \\
&= \nu_i \exp(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i)) \, 2 \left(\frac{\tilde{y}_i}{\exp(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i))} \ln \left(\frac{\tilde{y}_i}{\exp(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i)) \exp(T(\boldsymbol{x}_i; \mathbf{a}_m))} \right) \right. \\
&\quad \left. - \left(\frac{\tilde{y}_i}{\exp(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i))} - \exp(T(\boldsymbol{x}_i; \mathbf{a}_m)) \right) \right) \\
&= \nu_i \exp(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i)) L \left(\frac{\tilde{y}_i}{\exp(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i))}, \exp(T(\boldsymbol{x}_i; \mathbf{a}_m)) \right) \\
&= \nu_{i,m} L \left(\tilde{r}_{i,m}, \exp(T(\boldsymbol{x}_i; \mathbf{a}_m)) \right) , \qquad (3.4)
\end{aligned}$$

with

$$\begin{aligned}
\nu_{i,m} &= \nu_i \exp(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i)) \, , \\
\tilde{r}_{i,m} &= \frac{\tilde{y}_i}{\exp(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i))} \, . \qquad (3.5)
\end{aligned}$$

The mth iteration of the boosting procedure reduces to build a single weak learner on the working training set

$$\mathcal{D}^{(m)} = \{(\nu_{i,m}, \tilde{r}_{i,m}, \boldsymbol{x}_i), i = 1, \ldots, n\},$$

using the Poisson deviance loss and the log-link function. The weights are each time updated together with the responses that are assumed to follow Poisson rate distributions.

Actually, this approach extends to the whole class of Tweedie distributions. Let us denote

$$\begin{aligned}
\nu_{i,m} &= \nu_i \exp(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i))^{2-\xi} \, , \\
\tilde{r}_{i,m} &= \frac{y_i}{\exp(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i))} .
\end{aligned}$$

Notice that $\ln \tilde{r}_{i,m}$ corresponds to individual deltas defined in equation (4) in Lee and Lin (2018) under Tweedie loss and log-link. In accordance with Theorem 3 by these authors, we also conclude from the next result that gradients should be abandoned and replaced with iteratively re-scaled responses $\tilde{r}_{i,m}$. Precisely, the next result shows that when we work with the log-link function and a response that belongs to the Tweedie class (and so with a loss function of the form (3.1)), solving (2.5) amounts to build a single weak learner on the working training set

$$\mathcal{D}^{(m)} = \{(\nu_{i,m}, \tilde{r}_{i,m}, \boldsymbol{x}_i), i = 1, \ldots, n\} \, .$$

Proposition 3.1. *Consider the deviance loss function* (3.1)*. Then,* (2.5) *with the log-link function, that is*

$$\widehat{\mathbf{a}}_m = \underset{\mathbf{a}_m}{\text{argmin}} \sum_{i=1}^{n} \nu_i L \left(y_i, \exp \left(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i; \mathbf{a}_m) \right) \right) \, ,$$

can be rewritten as

$$\widehat{\mathbf{a}}_m = \underset{\mathbf{a}_m}{\operatorname{argmin}} \sum_{i=1}^{n} \nu_{i,m} L\left(\tilde{r}_{i,m}, \exp\left(T(\boldsymbol{x}_i; \mathbf{a}_m)\right)\right). \tag{3.6}$$

Proof. The case $\xi = 1$ has already been discussed. Turning to the Normal case $\xi = 0$, we have

$$\begin{aligned}
&\nu_i L(y_i, \exp(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i; \mathbf{a}_m))) \\
&= \nu_i \Big(y_i - \exp\left(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i; \mathbf{a}_m)\right) \Big)^2 \\
&= \nu_i \exp(2\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i)) \left(\frac{y_i}{\exp(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i))} - \exp(T(\boldsymbol{x}_i; \mathbf{a}_m)) \right)^2 \\
&= \nu_i \exp(2\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i)) L\left(\frac{y_i}{\exp(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i))}, \exp(T(\boldsymbol{x}_i; \mathbf{a}_m)) \right) \\
&= \nu_{i,m} L\left(\tilde{r}_{i,m}, \exp(T(\boldsymbol{x}_i; \mathbf{a}_m))\right),
\end{aligned}$$

where $\tilde{r}_{i,m}$ is defined by Equation (3.5). Now, for the Gamma case $\xi = 2$, it comes

$$\begin{aligned}
&\nu_i L\left(y_i, \exp\left(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i; \mathbf{a}_m)\right)\right) \\
&= \nu_i \Bigg(-2 \ln \left(\frac{y_i}{\exp\left(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i; \mathbf{a}_m)\right)} \right) \\
&\qquad +2 \left(\frac{y_i}{\exp\left(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i; \mathbf{a}_m)\right)} - 1 \right) \Bigg) \\
&= \nu_i \left(-2 \ln \left(\frac{\tilde{r}_{i,m}}{\exp\left(T(\boldsymbol{x}_i; \mathbf{a}_m)\right)} \right) + 2 \left(\frac{\tilde{r}_{i,m}}{\exp\left(T(\boldsymbol{x}_i; \mathbf{a}_m)\right)} - 1 \right) \right) \\
&= \nu_i L\left(\tilde{r}_{i,m}, \exp\left(T(\boldsymbol{x}_i; \mathbf{a}_m)\right)\right).
\end{aligned}$$

Finally, when $\xi \notin \{0, 1, 2\}$, we get

$$\begin{aligned}
&\nu_i L(y_i, \exp(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i; \mathbf{a}_m))) \\
&= \nu_i \, 2 \left(\frac{\max(y_i, 0)^{2-\xi}}{(1-\xi)(2-\xi)} - \frac{y_i \exp(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i; \mathbf{a}_m))^{1-\xi}}{1-\xi} \right. \\
&\qquad \left. + \frac{\exp(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i) + T(\boldsymbol{x}_i; \mathbf{a}_m))^{2-\xi}}{2-\xi} \right) \\
&= \nu_i \exp(\widehat{\text{score}}_{m-1}(\boldsymbol{x}_i))^{2-\xi} 2 \left(\frac{\max(\tilde{r}_{i,m}, 0)^{2-\xi}}{(1-\xi)(2-\xi)} - \frac{\tilde{r}_{i,m} \exp(T(\boldsymbol{x}_i; \mathbf{a}_m))^{1-\xi}}{1-\xi} + \frac{\exp(T(\boldsymbol{x}_i; \mathbf{a}_m))^{2-\xi}}{2-\xi} \right) \\
&= \nu_{i,m} L\left(\tilde{r}_{i,m}, \exp(T(\boldsymbol{x}_i; \mathbf{a}_m))\right),
\end{aligned}$$

which completes the proof. □

Therefore, when we work with the log-link function and a response that obeys a distribution belonging to the Tweedie class, which are the most relevant cases for application in

insurance, likelihood boosting should be preferred to gradient boosting. The latter procedure introduces an extra step which is unnecessary and leads to an approximation that can be easily avoided with boosting. This extends the point made by Wüthrich and Buser (2019) for the Poisson distribution to the whole Tweedie class.

On the other hand, the gradient boosting applies a least-squares principle on the gradients of the deviance. As the sum of least squares corresponds to the log-likelihood of a Gaussian model, choosing this optimization criterion is equivalent to assume that gradients are realizations of a Normal random variable. But under Tweedie deviance loss function and log-link, the negative gradient in Step 2.1 of the Algorithm (2) in Appendix A is given by

$$-\frac{\partial L\left(y,\mu(\boldsymbol{x})\right)}{\partial \mu(\boldsymbol{x})}\frac{\partial \mu(\boldsymbol{x})}{\partial \text{score}(\boldsymbol{x})}=\frac{2}{\mu(\boldsymbol{x})^{\xi-2}}\left(\frac{y}{\mu(\boldsymbol{x})}-1\right), \tag{3.7}$$

$\xi \geq 0$. This equation emphasizes that the gradient inherits from the distribution y that is scaled and translated. Except in the normal case ($\xi = 1$), applying the least squares principle is then ineffective and slows down the convergence compared to the Tweedie boosting as illustrated in the following section.

4 Numerical illustrations

4.1 Data set

To illustrate our results, we compare the ability of boosting and gradient boosting models to estimate the expected claim frequency and expected claim severity for motorcycle drivers. The data set comes from the Swedish insurance company *Wasa* and has been collected in 1999. The database is available on the companion website of the book of Ohlsson and Johansson (2010) and contains information about motorcycle insurance policies over the period 1994-1998. A contract is described by quantitative and categorical variables. The quantitative variables are the insured's age and the ancienty of the vehicle. The categorical variables are the policyholder's gender, the geographic zone and the category of the vehicle. The category of the vehicle is based on the ratio, noted EV, power in KW $\times 100$ / vehicle weight in kg + 75, rounded to the nearest integer. The data set reports the number of claims, the total claim costs and the duration of the contract for each policy. Table 4.1 summarizes the information provided by categorical features. The database comprises $n = 62,436$ policies (after removing contracts with a null duration). The total exposure is equal to 65,217 years and 693 claims are reported.

Rating factors	Class	Class description
Gender	M	Male
	K	Female
Geographic area	1	Central and semi-central parts of Sweden's three largest cities
	2	Suburbs plus middle-sized cities
	3	Lesser towns, except those in 5 or 7
	4	Small towns and countryside
	5	Northern towns
	6	Northen countryside
	7	Gotland (Sweden's largest island)
Vehicle class	1	EV ratio -5
	2	EV ratio 6-8
	3	EV ratio 9-12
	4	EV ratio 13-15
	5	EV ratio 16-19
	6	EV ratio 20-24
	7	EV ratio 25-

Table 4.1: Rating factors of motorcycle insurances. Source: Ohlsson and Johansson (2010).

Tables 4.2 and 4.3 report the averages and standard deviations of claim frequency and severity. Claims are more frequent and more expensive in urban and sub-urban environment than in other areas. In this data set, female drivers have a higher claim frequency and severity than men. This conclusion must nevertheless be nuanced as the portfolio counts only 9,482 female insureds and different trends are observed if we refine this analysis by e.g. geographic areas. We also observe that the most powerful vehicles (classes 6 and 7) cause on average more accidents than others. Without surprise, the vehicle and owner's ages are important discriminating factors. Young drivers and recent vehicles cause accidents more frequently and claim costs are higher for recent motorcycles.

Rating factors	Class	Average Frequency	st. dev. Frequency	Average Severity	st. dev. Severity
Gender	M	0.013	0.213	17 217	22793
	K	0.031	1.023	24 434	34713
Geographic area	1	0.0681	1.2527	29 648	38 282
	2	0.0287	0.3872	28 799	37 121
	3	0.0209	0.4118	18 310	27 689
	4	0.0227	1.2668	18 910	30 281
	5	0.0105	0.2312	11 638	15 067
	6	0.0124	0.2450	16 622	21 778
	7	0.0027	0.0522	650	-
Vehicle class	1	0.0210	0.4402	21 907	38 779
	2	0.0264	0.3449	15 201	21 895
	3	0.0275	1.5218	30 013	46 787
	4	0.0227	0.6304	22 108	28 882
	5	0.0248	0.3152	21 600	28 072
	6	0.0496	0.7988	23 999	27 510
	7	0.0347	0.4376	27 826	26 670

Table 4.2: Averages and standard deviations (st. dev.) of claim frequency and severity by modalities of rating factors.

	Average Frequency	st. dev. Frequency	Average Severity	st. dev. Severity
Owner's age ≤ 30	0.082	1.8813	25622	31208
$31 \leq$Owner's age ≤ 50	0.0119	0.2589	22876	35708
$51 \leq$ Owner's age	0.0104	0.2396	19823	37716
Vehicule age ≤ 5	0.0588	1.655	35109	43009
Vehicle age ≥ 6	0.0177	0.5068	14888	20430

Table 4.3: Averages and standard deviations (st. dev.) of claim frequency and severity by ranges of owner and vehicle ages.

In this paper, we analyze the motorcycle insurance data set from Ohlsson and Johansson (2010) with boosting and gradient boosting. We consider three weak learners: trees, randomized GLMs and neural networks. We use a training set made of 80% of the observation to build the models and we use the remaining 20% of the observations to compute out-of-sample estimates of the deviance. Each time, we assess model performances (in-sample and out-of-sample estimates of the deviance) compared to gradient boosting based on the same weak learners. At each iteration of boosting, we estimate a weak learner to data with adjusted exposure using a tweedie likelihood. In the gradient boosting, we fit a weak learner to gradients with a Gaussian likelihood. As the $\widehat{\beta}_m$'s (step 2.c of the algorithm in Appendix A, Equation (5.2)) admit closed form expressions, both algorithms displays then comparable computation times.

4.2 Regression trees as weak learners

In this section, we use binary regression trees as weak learners. That is, we consider weak learners $T(\boldsymbol{x}; \mathbf{a}_m)$ of the form

$$T(\boldsymbol{x}; \mathbf{a}_m) = \sum_{t \in \mathcal{T}_m} c_{t,m} \mathrm{I}\left[\boldsymbol{x} \in \chi_t^{(m)}\right], \tag{4.1}$$

where $\left\{\chi_t^{(m)}\right\}_{t \in \mathcal{T}_m}$ is the partition of the feature space χ induced by the regression tree $T(\boldsymbol{x}; \mathbf{a}_m)$ and $\{c_{t,m}\}_{t \in \mathcal{T}_m}$ the corresponding predictions for the score. For regression trees, the "parameters" $\mathbf{a}_m$ represent the splitting variables and their split values as well as the corresponding predictions in the terminal nodes, that is,

$$\mathbf{a}_m = \left\{c_{t,m}, \chi_t^{(m)}\right\}_{t \in \mathcal{T}_m}.$$

With regression trees, a shrinkage parameter $0 \leq \gamma < 1$ is often used to slow the learning rate of the training procedure, so that instead of adding tree $T(\boldsymbol{x}; \mathbf{a}_m)$ to the score in Step 2.2 of Algorithm (1), we rather add $\gamma T(\boldsymbol{x}; \mathbf{a}_m)$. Small values for γ work best but result in larger computation time since more regression trees are necessary. Empirically, values for the shrinkage parameter γ smaller than 10% yield dramatic improvements for regression estimation. We run the models of Section 4.2 with shrinkage parameter $\gamma = 1\%$. We refer the reader to the seminal book of Breiman et al. (1984) for details about the recursive partitioning algorithm used to build binary regression trees.

4.2.1 Claims frequency

Let $Y_1, \ldots, Y_n$ be the numbers of claims observed in the data set. Precisely, Y_i corresponds to the number of claims caused by the i^{th} policyholder during a time interval with length e_i. The numbers of claims $Y_1, \ldots, Y_n$ are assumed to be independent and Poisson distributed, so that we adopt the Poisson deviance as loss function. We also work with the log-link function, which is the canonical link in the Poisson case.

We use the `R` function `rpart` from the `R` package `rpart` of Therneau and Atkinson (2018) to produce each of the M constituent trees for boosting models and gradient boosting models. For the gradient boosting models, the command `rpart` is of course only used at each iteration to determine the structure of the tree with the squared error loss, the predictions in the terminal nodes being well estimated using the Poisson deviance as loss function. The size of the trees is controlled with the variable `maxdepth=D`. A regression tree with `maxdepth=D` has $2^{\mathtt{D}}$ terminal nodes, each terminal node having $\mathtt{D}-1$ ancestors. Note that specifying the number of terminal nodes or the interaction depth of a tree is not possible with `rpart`.

We run boosting trees and gradient boosting trees on the training set for $\mathtt{D} = 1, 2, 3$. The whole training set is used at each iteration to produce the new tree and we use shrinkage parameter $\gamma = 1\%$. Figures 4.1 and 4.2 display the Poisson deviance (in-sample estimates in Figure 4.1 and out-of-sample estimates in Figure 4.2) with respect to the number of iterations. For each value of `D`, one sees from Figure 4.2 that the maximum number of iterations M is taken large enough. The comparison of deviances in Figures 4.1 and 4.2

shows that the boosting algorithm slightly outperforms the gradient boosting approach for $\mathtt{D} = 1$ and provides better results for $\mathtt{D} = 2, 3$.

The main reason explaining this gap of performance is that gradient boosting applies a least-squares principle on its gradients. As mentioned in Section 2.4, this is implicitly equivalent to maximize the log-likelihood of gradient residuals assumed to be normally distributed. This assumption in general does not hold for low count data and prevents gradient boosting to significantly outperform boosting.

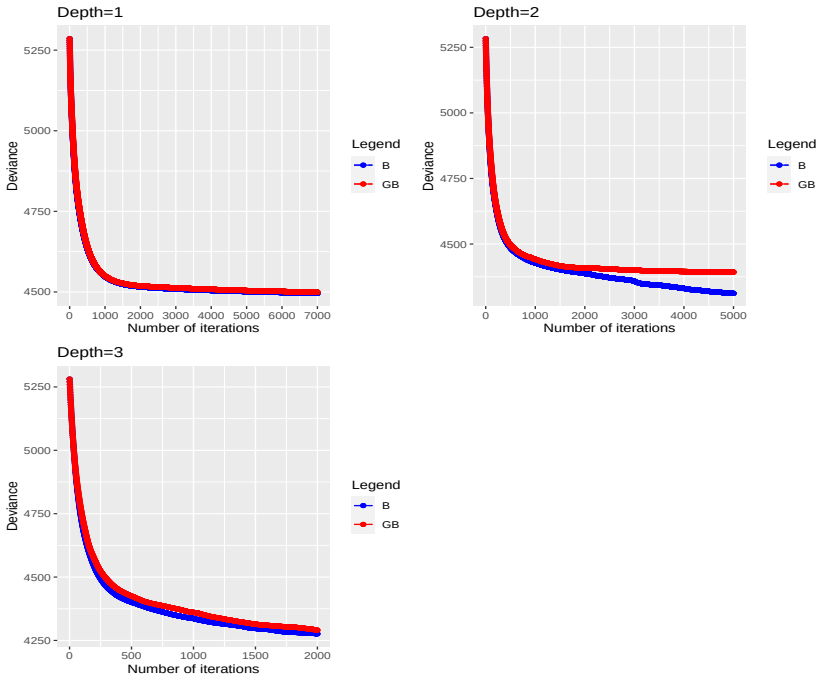

Figure 4.1: Claims frequency: comparison of in-sample deviances obtained with boosting trees (B) and gradient boosting trees (GB).

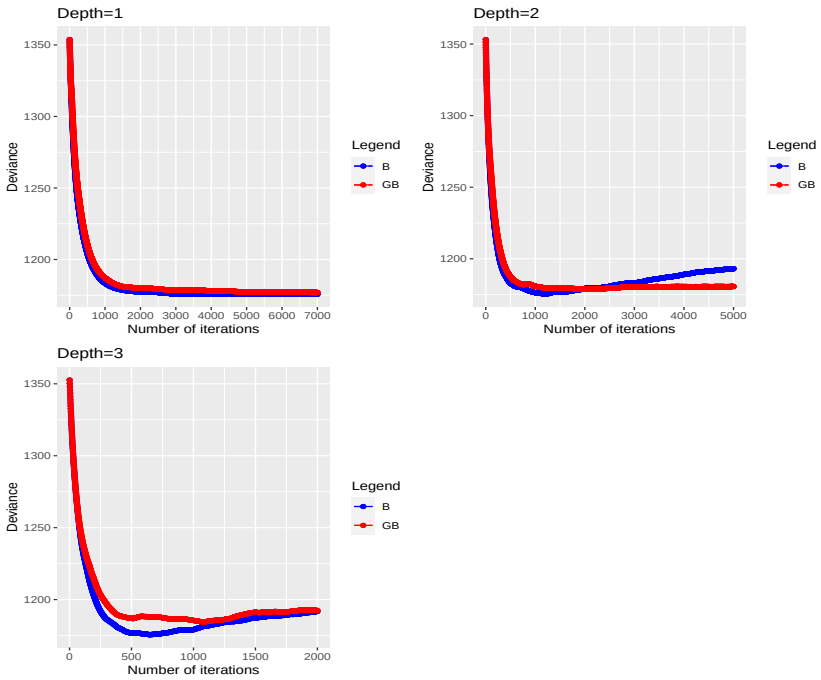

Figure 4.2: Claims frequency: comparison of out-of-sample deviances obtained with boosting trees (B) and gradient boosting trees (GB).

4.2.2 Average claims severity

Let us now study the claims severities. Here, we only consider policyholders i who made at least one claim during the observation period e_i, which represents only 666 drivers in the data set. Among these 666 drivers, 639 made one claim and 27 two claims. For these 666 drivers, we denote by Y_i the corresponding average claims severity and we use as weight ν_i the number of claims. The average claims severities are assumed to be independent and Gamma distributed, so that we work with the Gamma deviance as loss function and we adopt the log-link function.

Again, we use the command `rpart` to produce each of the M constituent trees for boosting models and gradient boosting models. Note that we rely on the `R` package `distRforest` of Henckaerts (2020) to use the Gamma deviance as loss function. We control the size of the trees with `maxdepth=D`, the whole training set is used at each iteration to produce the new tree and we use a shrinkage parameter $\gamma = 1\%$.

We run Boosting trees and gradient boosting trees for $\texttt{D} = 1, 2, 3$. In Figures 4.3 and 4.4, we show the Gamma deviance computed on the training set and on the validation set, respectively, with respect to the number of iterations. We note from Figure 4.4 that the maximum number of iterations M is taken large enough. Both approaches provide almost identical results for the in-sample estimates of the deviance. The similar performance of both methods is explained by the fact that the distribution of gradients in the Gamma regression is closer to a Normal than in the Poisson case. However, when looking at the out-of-sample estimates of the deviance, one sees that the boosting algorithm outperforms the gradient boosting approach for $\texttt{D} = 2, 3$ and that both approaches perform similarly for $\texttt{D} = 1$.

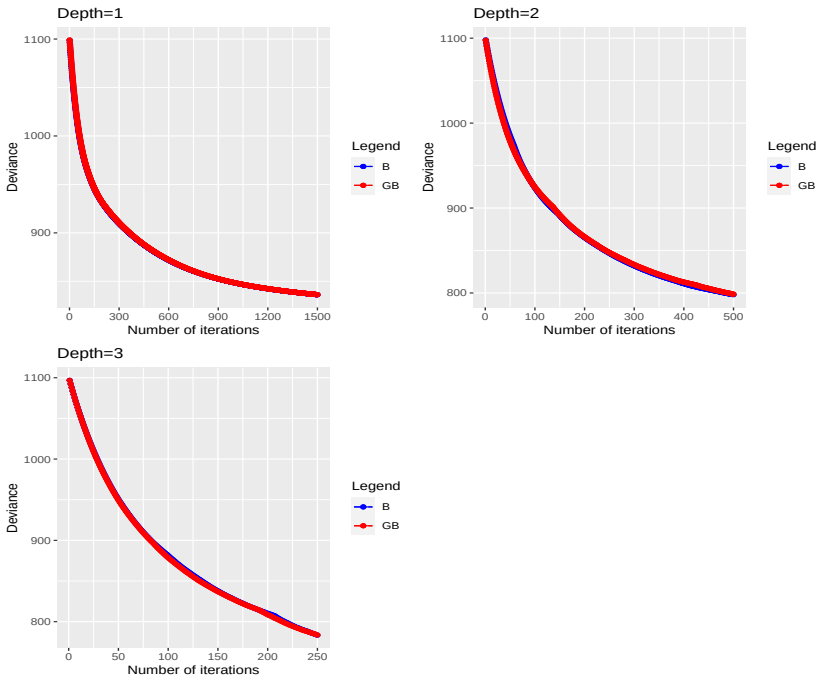

Figure 4.3: Claims severity: comparison of in-sample deviances obtained with boosting trees (B) and gradient boosting trees (GB).

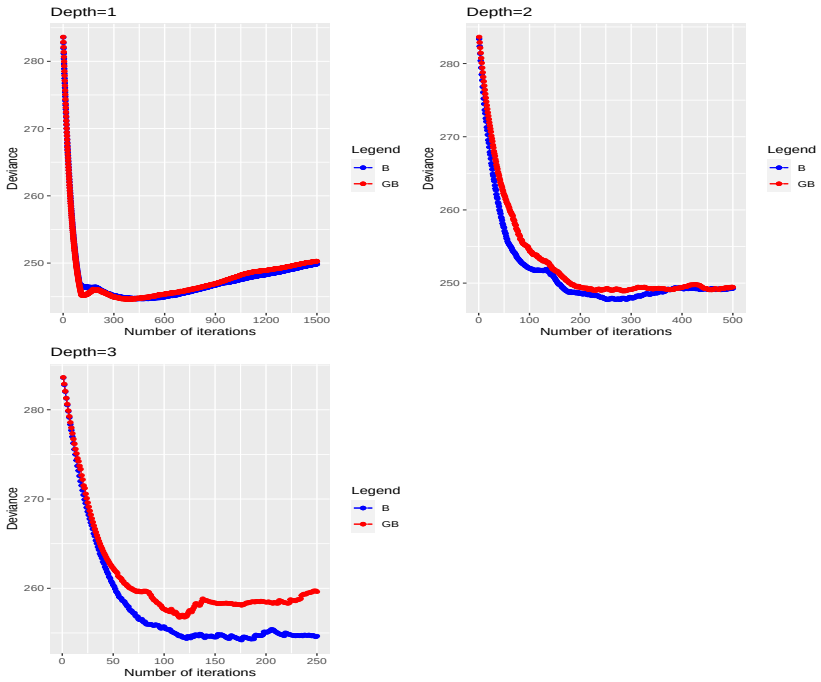

Figure 4.4: Claims severity: comparison of out-of-sample deviances obtained with boosting trees (B) and gradient boosting trees (GB).

4.3 GLMs as weak learners

Tutz and Binder (2006, 2007) introduced GAMBoost as a procedure for fitting GAMs using likelihood-based boosting. GAMBoost can cope with large numbers of possibly uninformative features, whilst avoiding over-fitting. This procedure has been implemented in the `R` package `GAMBoost`, using P-splines as weak learners. To speed the calculations, parameters involved in weak learners are estimated using only one iteration of the IRLS algorithm applied in the GLM and GAM settings (which is intuitively acceptable, since base learners must be weak).

In this section, we consider a randomized GLM as base learner. In the m^{th} iteration of the boosting algorithm, we randomly select a subset $\mathcal{I}_m$ of $n_{glm} < p$ features among the p available ones and regress the response Y on them. As explained in Chapter 6 of Denuit et al. (2019b), the randomization of the training procedure is a powerful technique for reducing the variance of the estimator. The functions $T(\boldsymbol{x}; \mathbf{a}_m)$, $m = 1, 2, \ldots, M$ are then linear combinations of the M subsets of covariates:

$$T(\boldsymbol{x}; \mathbf{a}_m) \quad = \quad a_{m,0} + \sum_{k \in \mathcal{I}_m} a_{m,k}\, x_k \,,$$

where $\mathbf{a}_m$ is a vector of dimension n_{glm} that is estimated by a standard GLM procedure. We also weaken the estimator by limiting the number of maximum iterations performed by the GLM routine, as in `GAMBoost` except that we allow here for more than just one IRLS step. We compare boosting and gradient boosting on the data set from the company *Wasa*. In order to capture non-linearity between the response and continuous features, we convert them into categories. This approach is also standard in the industry. We class policyholders by age into 20 subsets of comparable size and vehicle ages are split in 10 categories. In this setting, an insurance policy described by a set of 33 binary features.

4.3.1 Claims frequency

We assume that the numbers of claims $Y_1, \ldots, Y_n$ are independent and Poisson distributed. The loss function is the Poisson deviance and we adopt the log-link function, which is the canonical link in the Poisson case. We run 200 iterations for both boosting and gradient boosting and choose a shrinkage parameter $\gamma = 5\%$.

At each epoch, a GLM is estimated with 5, 8, 10 and 13 randomly selected covariates. This allows us to weaken the estimator, in the same spirit than random forests. We also limit to 30, the maximum number of iteratively reweighted least squares (IWLS) for estimating a single GLM to keep under control the computation time.

Figures 4.5 and 4.6 respectively reports the evolution of the in- and out-of- sample deviance with respect to the number of gradient boosting iterations. For less than 150 iterations, the boosting outperforms gradient boosting on the training set but achieves a higher deviance on the validation set. After 150 iterations, both methods have similar in- and out-of- sample performances.

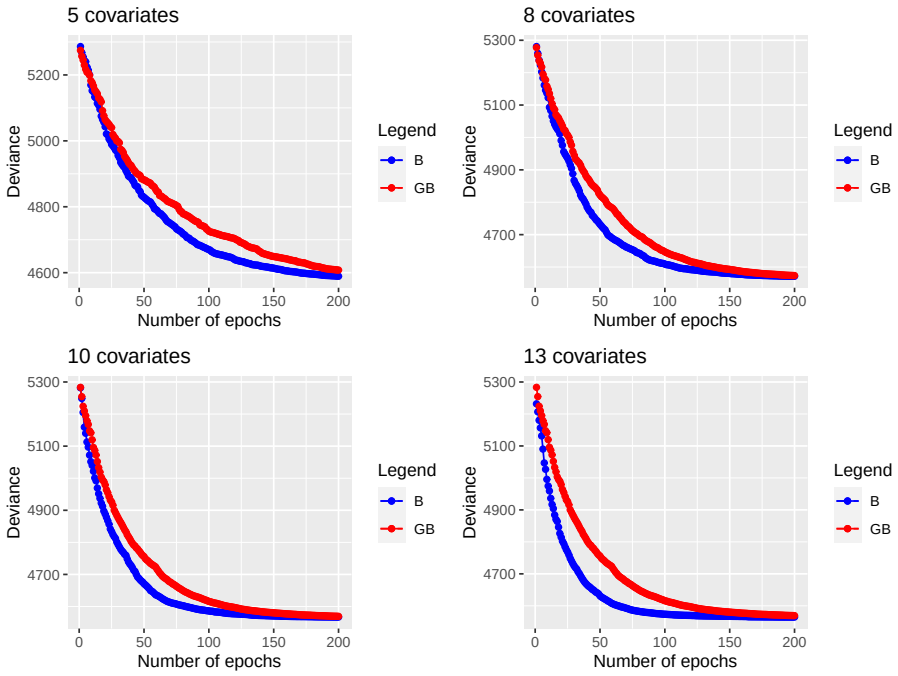

Figure 4.5: Claims frequency: comparison of in-sample deviances obtained with boosting (B) and gradient boosting (GB).

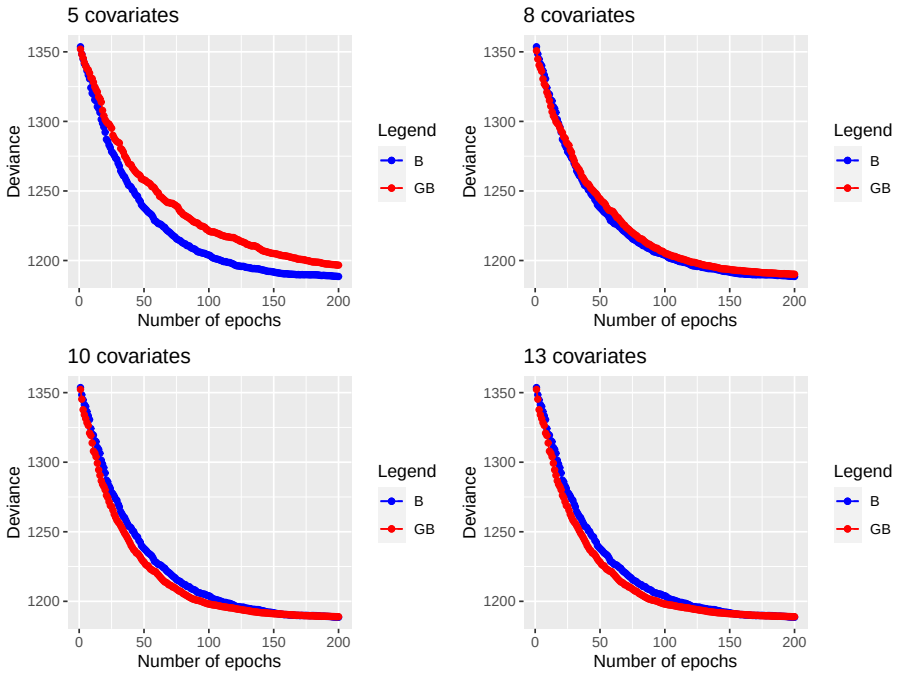

Figure 4.6: Claims frequency: comparison of out-of-sample deviances obtained with boosting (B) and gradient boosting (GB).

4.3.2 Average claims severity

We study here the average claim severities Y_i using the corresponding number of reported claims as weight ν_i. The responses are independent and Gamma distributed and we consider a log-link function. Since the size of the training dataset is limited to 528 contracts, we run 400 iterations for both boosting and gradient boosting with a shrinkage parameter $\gamma = 1\%$. Figures 4.7 and 4.8 show the evolution of the in- and out-of- sample deviance with respect to the number of iterations. On the training dataset, the boosting and gradient boosting behave in a similar manner for base learners with 8 and 10 covariates. For 5 and 13 covariates, the boosting outperforms its gradient counterpart. The evolution of out-of-sample deviances reveals that both methods overfit data after around 100 iterations for 8, 10 and 13 covariates in the GLM. The predictive power of the gradient boosting appears in this case slightly better than the one of the boosting algorithm for 5 and 8 covariates. For 10 and 13 covariates, B and GB have respectively similar predictive performances up to 75 and 100 iterations. In this setting, the boosting method quicker overfits the validation set than gradient boosting. Nevertheless, we should avoid to generalize this conclusion given the limited size (138 contracts) of the validation dataset.

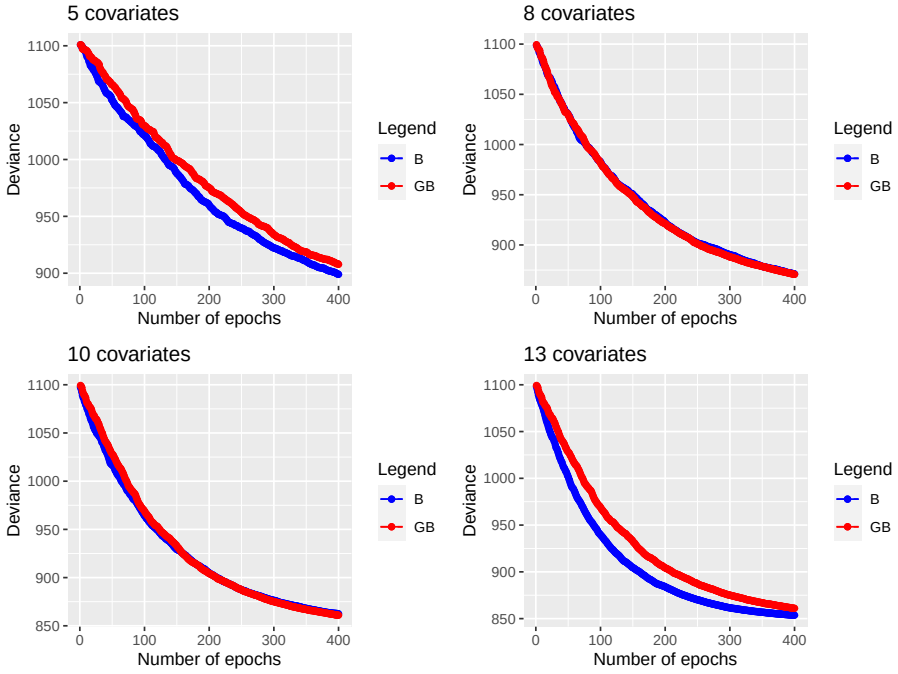

Figure 4.7: Claims severity: comparison of in-sample deviances obtained with boosting (B) and gradient boosting (GB).

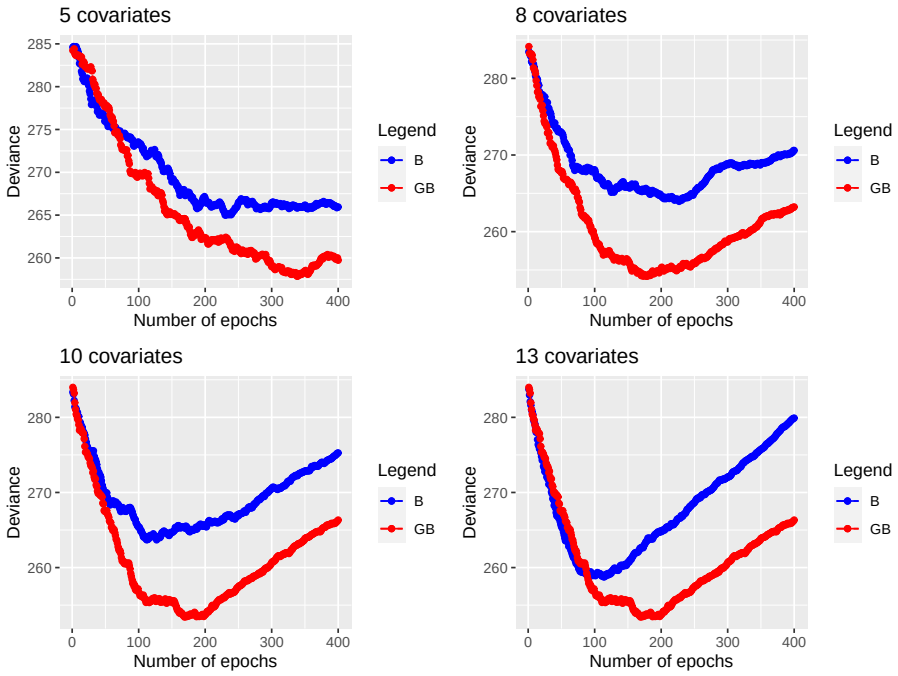

Figure 4.8: Claims severity: comparison of out-of-sample deviances obtained with boosting (B) and gradient boosting (GB).

4.4 Neural networks as weak learners

In this section, we use a shallow neural network as base learner. This network involves three layers, as illustrated in Figure 4.9. The first input layer dispatches the input signals, a vector $\boldsymbol{x}$ of dimension p, between neurons of the intermediate layer. In this second layer, signals are linearly mixed and pass through an activation function. The outputs of intermediate neurons next activate the output neuron that sends back $T(\boldsymbol{x};\mathbf{a})$.

The number of neurons in the j^{th} layer is denoted as n_j^{net} where $j \in \{1,2\}$ refers to the intermediate and output layers, respectively. The activation functions of neurons, denoted here as $\phi_{i,j}(.)$ are identified by two indices: i for the position in a layer and j for the layer. The output of the i^{th} neuron in the first layer, denoted by $\widehat{y}_{i,1}$, is computed as follows:

$$\widehat{y}_{i,1} \quad = \quad \phi_{i,1}\left(a_{i0}^1 + \sum_{k=1}^{p} a_{i,k}^1 x_k\right) \qquad i = 1, ..., n_1^{net},$$

where $a_{i,k}^1$ are the weights for the k^{th} explanatory variables, received by the i^{th} neuron. The output of the network is:

$$T(\boldsymbol{x};\mathbf{a}) \quad = \quad \phi_{1,2}\left(a_{10}^2 + \sum_{k=1}^{n_1^{net}} a_{1,k}^2 \widehat{y}_{k,1}\right),$$

where $\mathbf{a}$ is here the vector of neural weights.

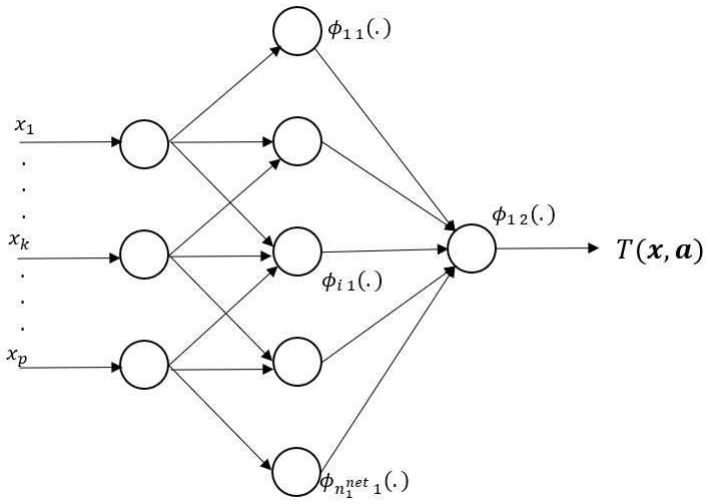

Figure 4.9: Illustration of a shallow neural network used as base learner in the Boosting algorithm.

We compare boosting and gradient boosting on the data set from the company *Wasa*, described in Section 4.1. In order to avoid numerical issues, we rescale the owner's age and ancienty of the vehicle and convert their domain on the pavement $[0,1] \times [0,1]$. The categorical features are converted into binary dummy variables. An insurance contract is therefore described by 2 quantitative variables and 13 binary features.

4.4.1 Claims frequency

Again, we assume that the numbers of claims $Y_1, \ldots, Y_n$ are independent and Poisson distributed. The loss function is the Poisson deviance and we adopt the log-link function, which is the canonical link in the Poisson case. At each epoch, a shallow network is estimated with KERAS for R[1]. The intermediate layer counts from 1 to 4 neurons with hyperbolic tangent activation functions[2]. In order to weaken the predictive power of base learners, we only run 50 iterations of the RMSPROP algorithm to estimate neural weights. This is a gradient descent algorithm which has by default in Keras, a very low learning rate of 0.001 and requires a large number of iterations before converging. Furthermore, we fuzzify the procedure by working with small batches of 2000 contracts randomly chosen among the 49 949 policies of the training set. As we perform 50 iterations, we pass on average only twice times the dataset through the neural network. As the estimation of neural networks is time consuming, we choose a higher shrinkage rate ($\gamma = 60\%$) and run only 50 boosting and gradient boosting iterations.

The comparison of deviances in Figure 4.10 confirms that the boosting algorithm outperforms the gradient boosting approach on the training dataset for most of basic learners. Out-sample deviances, plotted in Figure 4.11, shows that the predictive power of boosted

[1]Chollet, F., & others. (2015). https://github.com/rstudio/keras/

[2]Notice that the 1 Neuron NN differs from the GLM by the fact that covariates pass through a first layer with hyperbolic tangent activation function.

neural networks is also better than its gradient boosted counterpart for most of configurations.

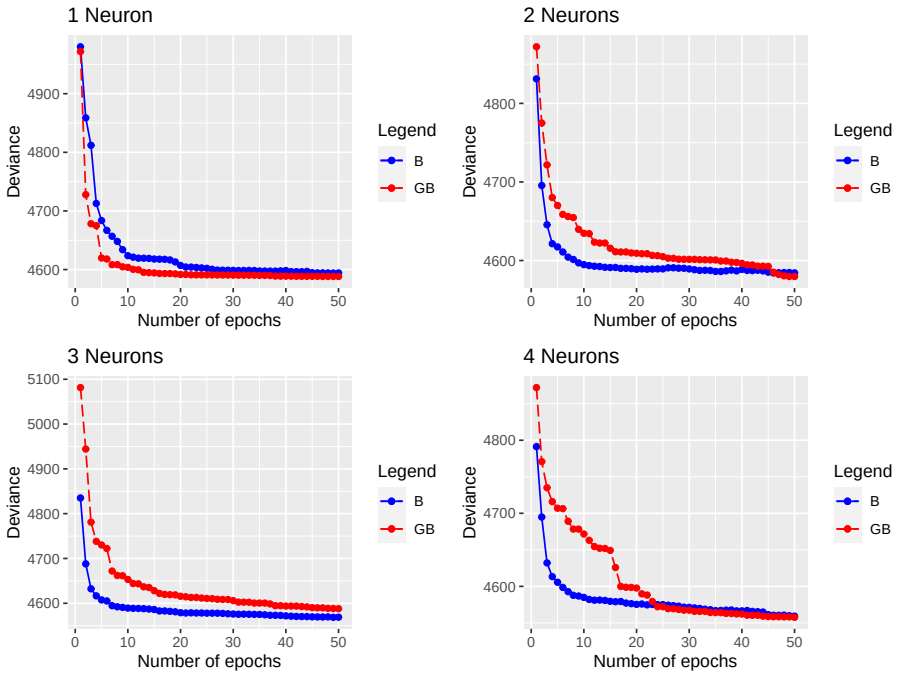

Figure 4.10: Claims frequency: comparison of in-sample deviances obtained with boosting (B) and gradient boosting (GB).

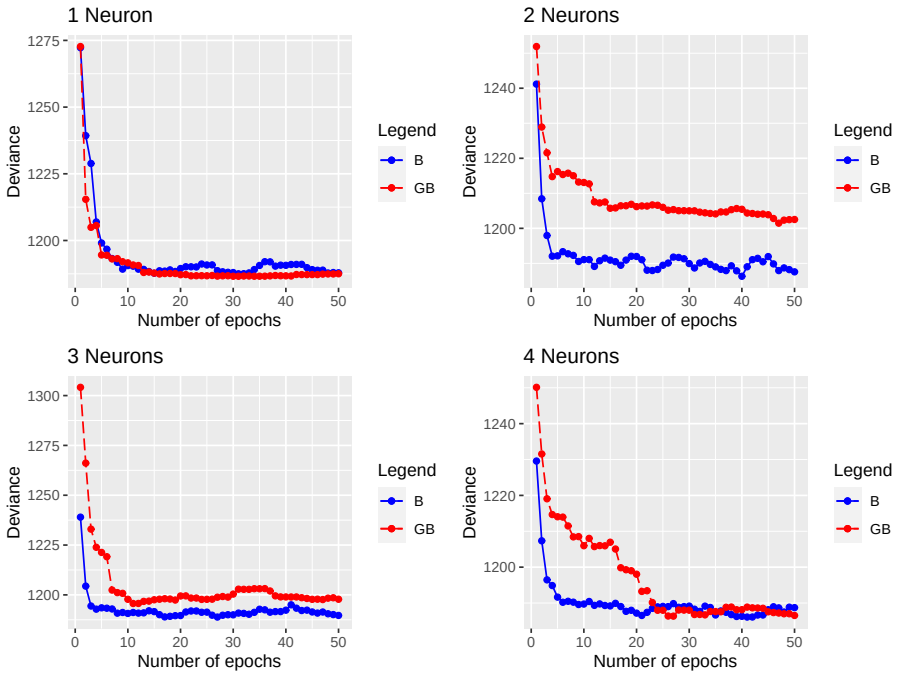

Figure 4.11: Claims frequency: comparison of out-sample deviances obtained with boosting (B) and gradient boosting (GB).

4.4.2 Average claims severity

We study here the average claim severities Y_i using the number of reported claims as weight ν_i. We assume that $Y_1, \ldots, Y_n$ are independent and Gamma distributed. Thus, we work with the Gamma deviance as loss function and adopt the log-link. We run 50 iterations for both boosting and gradient boosting algorithms with $\gamma = 60\%$. The base learner is still a shallow neural net with a hidden layer with 1 to 4 neurons and hyperbolic tangent activation functions. As the data set is small, we run 1000 steps of the RMSprop algorithm on the whole data set. The evolution of deviances with respect to iterations are shown in Figures 4.12 and 4.13. We cannot conclude that gradient boosting outperforms boosting. As for GLM base learner, both methods overfit the dataset after a few iterations.

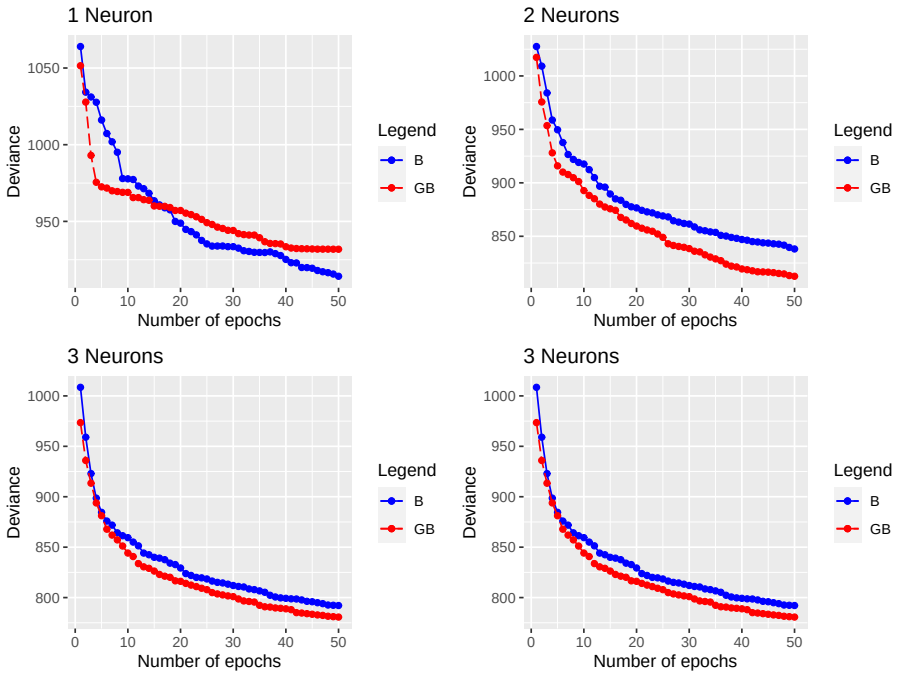

Figure 4.12: Claims severity: comparison of in-sample deviances obtained with boosting (B) and gradient boosting (GB).

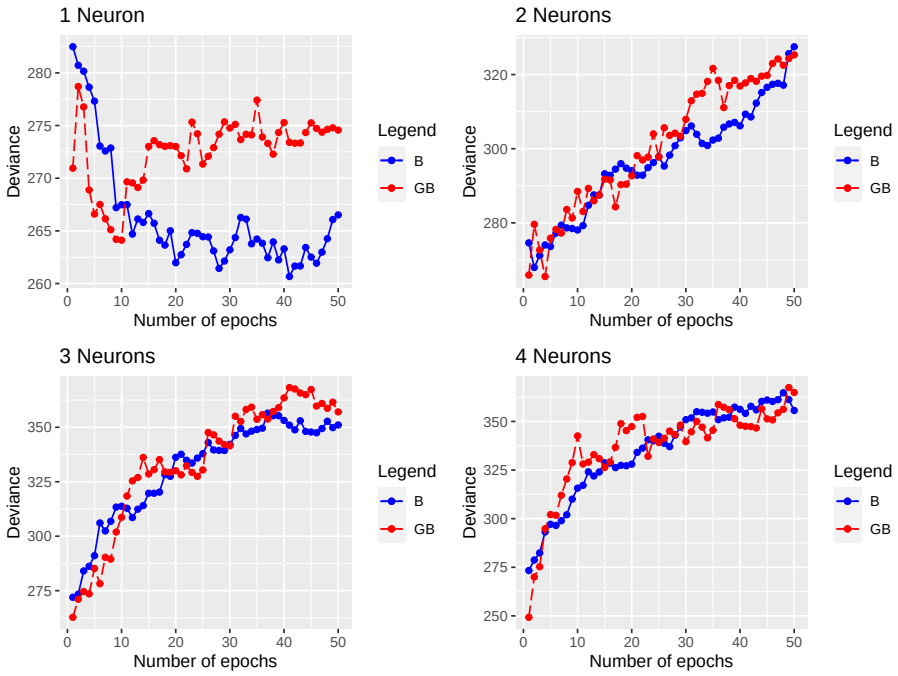

Figure 4.13: Claims severity: comparison of out-sample deviances obtained with boosting (B) and gradient boosting (GB).

5 Conclusions

The gradient boosting algorithm has outstanding performances in a wide variety of regression problems. Instead of maximizing the log-likelihood, gradient boosting applies a least-squares principle on its gradients. To some extent, this amounts to the assumption that gradients are normally distributed as least squares are inversely proportional to the Gaussian log-likelihood. In some circumstances, such as for low count datasets, it is then preferable to consider log-likelihood based boosting as argued in this article for insurance applications.

When trees are used as weak learners, response boosting achieves similar or better performances than gradient boosting on the particular Wasa data set. The conclusions of the numerical study are more contrasted with GLM and NN weak learners where gradient boosting sometimes performs better than response boosting, or is of similar quality. Even if it is impossible to draw general conclusions from a single data set, the numerical analysis conducted in Section 4 demonstrates that boosting responses instead of gradients is feasible from a practical point of view and suggests that the proposed approach is particularly effective for claim frequencies (low count responses). Further numerical analyzes should be performed for claim severities (because of the limited number of observed severities comprised in the Wasa data set) as well as for GLM and NN weak learners.

This work contributes to the literature in two directions. From a methodological point of view, it extends the results of Wüthrich and Buser (2019) to the whole class of Tweedie distributions. We show indeed that gradient boosting introduces an extra step which leads to an unnecessary approximation of gradients. Instead, the boosting algorithm is performed by a simple modification of weights at each iteration. From an empirical point of view, the analysis of the claim frequencies and severities of motorcycle insurances comprised in the Wasa data set shows that boosting achieves similar or better performances than its gradient boosting counterpart when trees are used as weak learners while the conclusions are more contrasted with GLM and neural weak learners. Since trees are widely used in insurance studies, the approach proposed in this paper turns out to be useful in practice since it ensures a higher level of transparency to end-users with comparable computation time.

Appendix A: Gradient boosting

Gradient boosting is a forward stagewise additive approach in which the estimator has the following form:

$$\mu_M(\boldsymbol{x}) = g^{-1}\left(\sum_{k=0}^{M} \beta_k\, T(\boldsymbol{x};\mathbf{a}_k)\right),$$

where $(\beta_k)_{k=0,\dots,M} \in \mathbb{R}$. In this framework, the functions $(T(\boldsymbol{x};\mathbf{a}_k))_{k=0,\dots,M}$ approximates the negative gradient of the generalized error, in the least-squares sense. This leads to the following algorithm:

Algorithm 2 Gradient Boosting algorithm.

1. Initialization :

Initialize the constant estimator $\widehat{\mu}_0 = g^{-1}(\widehat{a}_0)$:

$$\widehat{a}_0 = \arg\min_{a_0} \sum_{i=1}^{n} L\left(y_i, g^{-1}(a_0)\right) . \tag{5.1}$$

Set $\widehat{\beta}_0 = 1$ and $T(\boldsymbol{x}_i; \widehat{a}_0) = \widehat{a}_0$.

2. Main procedure :

For $m = 0$ to maximum epoch, $M - 1$ **do**

a) Calculate the negative gradient $\boldsymbol{g} = (g_i)_{i=1,\ldots,n}$ of the generalized error:

$$g_i \;=\; -\frac{\partial L\left(y_i, \mu_m(\boldsymbol{x}_i)\right)}{\partial \mu_m(\boldsymbol{x}_i)} \frac{\partial \mu_m(\boldsymbol{x}_i)}{\partial \mathrm{score}_m(\boldsymbol{x}_i)} ,$$

where $\mu_m(\boldsymbol{x}_i) = g^{-1}\left(\mathrm{score}_m(\boldsymbol{x}_i)\right)$ and $\mathrm{score}_m(\boldsymbol{x}_i) = \sum_{k=0}^{m} T(\boldsymbol{x}_i; \mathbf{a}_k)$.

b) Estimate the weights $\mathbf{a}_{m+1}$ of $T(\boldsymbol{x}_i; \mathbf{a}_{m+1})$ regressing $(g_i)_{i=1,\ldots,n}$:

$$\widehat{\mathbf{a}}_{m+1} = \arg\min_{\mathbf{a}_{m+1}} \sum_{i=1}^{n} \left(g_i - T(\boldsymbol{x}_i; \mathbf{a}_{m+1})\right)^2 .$$

c) Compute $\widehat{\beta}_{m+1}$ such that

$$\widehat{\beta}_{m+1} = \arg\min_{\beta} \sum_{i=1}^{n} \left(L\left(y_i, g^{-1}\left(\widehat{\mathrm{score}}_m(\boldsymbol{x}_i) + \beta\, T(\boldsymbol{x}_i; \widehat{\mathbf{a}}_{m+1})\right)\right)\right) . \tag{5.2}$$

d) Update the estimator $\widehat{\mu}_{m+1}(\boldsymbol{x})$ as follows:

$$\widehat{\mu}_{m+1}(\boldsymbol{x}) \;=\; g^{-1}\left(\widehat{\mathrm{score}}_m(\boldsymbol{x}) + \widehat{\beta}_{m+1}\, T(\boldsymbol{x}; \widehat{\mathbf{a}}_{m+1})\right) .$$

End for

Acknowledgements

The authors thank anonymous Referees and the Editor for their constructive comments which helped to improve this paper.

References

- Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J. (1984). Classification and Regression Trees. Wadsworth Statistics/Probability Series.

- Delong, L., Lindholm, M., Wüthrich, M. V. (2021). Making Tweedie's compound Poisson model more accessible. European Actuarial Journal 11, 185-226.

- Denuit, M., Hainaut, D., Trufin, J. (2019a). Effective Statistical Learning Methods for Actuaries I: GLM and Extensions. Springer Actuarial Lecture Notes Series.

- Denuit, M., Hainaut, D., Trufin, J. (2019b). Effective Statistical Learning Methods for Actuaries III: Neural Networks and Extensions. Springer Actuarial Lecture Notes Series.

- Denuit, M., Hainaut, D., Trufin, J. (2020). Effective Statistical Learning Methods for Actuaries II: Tree-based Methods and Extensions. Springer Actuarial Lecture Notes Series.

- Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. Annals of Statistics 29(5), 1189-1232.

- Friedman, J., Hastie, T. and Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). Annals of Statistics 28(2), 337-407.

- Guelman, L. (2012). Gradient boosting trees for auto insurance loss cost modeling and prediction. Expert Systems with Applications 39, 3659-3667.

- Henckaerts, R. (2020). distRforest: Distribution-based random forest. https://www.github.com/henckr/distRforest.

- Henckaerts, R., Cote, M-P., Antonio, K., Verbelen, R. (2021). Boosting insights in insurance tariff plans with tree-based machine learning methods. North American Actuarial Journal 25, 255-285.

- Lee, S.C. (2020). Delta boosting implementation of Negative Binomial regression in actuarial pricing. Risks 8, 1-19.

- Lee, S.C., Lin, S. (2018). Delta boosting machine with application to general insurance. North American Actuarial Journal 22, 405-425.

- Liu, Y., Wang, B., Lv, S. (2014). Using multi-class AdaBoost tree for prediction frequency of auto insurance. Journal of Applied Finance and Banking 4, 45-53.

- Mayr, A., Binder, H., Gefeller, O., Schmid, M. (2014a). The evolution of boosting algorithms - From machine learning to statistical modelling. Methods of Information in Medicine 53, 419-427.

- Mayr, A., Binder, H., Gefeller, O., Schmid, M. (2014b). Extending statistical boosting - An overview of recent methodological developments. Methods of Information in Medicine 53, 428-435.

- Ohlsson, E., Johansson, B. (2010). Non-Life Insurance Pricing with Generalized Linear Models. Springer.

- Pesantez-Narvaez, J., Guillen, M., Alcaniz, M. (2019). Predicting motor insurance claims using telematics data- XGBoost versus logistic regression. Risks 7, 1-16.

- Therneau, T. M., Atkinson, B. (2018). rpart: Recursive partitioning and regression trees. https://cran.r-project.org/package=rpart.

- Tutz, G., Binder, H. (2006). Generalized additive modelling with implicit variable selection by likelihood based boosting. Biometrics 51, 961-971.

- Tutz, G., Binder, H. (2007). Boosting ridge regression. Computational Statistics and Data Analysis 51, 6044-6059.

- Wüthrich, M. V., Buser, C. (2019). Data Analytics for Non-Life Insurance Pricing. Lecture notes available at SSRN http://dx.doi.org/10.2139/ssrn.2870308.

- Yang, Y., Qian, W., Zou, H. (2018). Insurance premium prediction via gradient tree-boosted Tweedie compound Poisson models. Journal of Business & Economic Statistics 36, 456-470.