RESPONSE VERSUS GRADIENT BOOSTING TREES, GLMs AND NEURAL NETWORKS UNDER TWEEDIE LOSS AND LOG-LINK

Hainaut, D., Trufin, J. and M. Denuit

LIDAM Discussion Paper ISBA/2021/12

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

Louvain-la-Neuve, Belgium

November 28, 2020

Abstract

Thanks to its outstanding performances, boosting has rapidly gained wide acceptance among actuaries. To speed 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 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 improved performances compared to gradient boosting when trees, GLMs and neural networks are used as weak learners.

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 GLMs or 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. 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. 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. 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. 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). Pesantez-Narvaez et al. (2019) employed XGBoost to predict the occurrence of claims using telematics data. Henckaerts et al. (2020) 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-offsetted procedure applied to the original data. 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. 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. 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, 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-offsetting 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. This is why likelihood-based boosting as introduced by Tutz and Binder (2006) is superior to gradient boosting in this case.

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. 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 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.

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 the results obtained by WÄEthrich 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 WÄEthrich 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.

The remainder of this paper is organized as follows. Section 2 recalls the boosting principle and its forward stagewise additive modeling. The particular Poisson case is 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.

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 $\mu(\mathbf{X}) = \mathbb{E}[Y|\mathbf{X}]$ of the response Y (claim number or claim amount for instance) given the available information summarized in a vector $\mathbf{X}$ of features $X_1, X_2, \dots, X_p$. The function $\mathbf{x} \mapsto \mu(\mathbf{x}) = \mathbb{E}[Y|\mathbf{X} = \mathbf{x}]$ is unknown to the actuary and is approximated by a working predictor $\mathbf{x} \mapsto \hat{\mu}(\mathbf{x})$ entering premium calculation.

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

$$Err(\hat{\mu}) = \mathbb{E} [L(Y, \hat{\mu}(\mathbf{X}))],$$

where $L(., .)$ is the loss function measuring the discrepancy between its two arguments and the expected value is over the joint distribution of $(Y, \mathbf{X})$. We aim to find a function $\mathbf{x} \mapsto \hat{\mu}(\mathbf{x})$ of the features minimizing the generalization error.

We denote by

$$\mathcal{D} = \{(y_1, \mathbf{x}_1), (y_2, \mathbf{x}_2), \dots, (y_n, \mathbf{x}_n)\} \quad (2.1)$$

the set of observations used to fit the model $\hat{\mu}$, called training set. An estimate $\hat{\mu}(\mathbf{x})$ to $\mu(\mathbf{x})$ is obtained by applying a training procedure to the training set $\mathcal{D}$. Common training procedures restrict $\mu(\mathbf{x})$ 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

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

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

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

by minimizing the corresponding training sample estimate of the generalized error

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

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

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

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

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

with

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

This leads to the following algorithm:

Algorithm 1: Forward Stagewise Additive Modeling.

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

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

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

2.1 Compute

$$\hat{\mathbf{a}}_m = \underset{\mathbf{a}_m}{\operatorname{argmin}} \sum_{i=1}^n L(y_i, g^{-1}(\widehat{\text{score}}_{m-1}(\mathbf{x}_i) + T(\mathbf{x}_i; \mathbf{a}_m))) . \quad (2.6)$$

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

End for

3. Output: $\hat{\mu}_{\mathcal{D}}^{\text{boost}}(\mathbf{x}) = g^{-1}(\widehat{\text{score}}_M(\mathbf{x}))$.
-

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 $T(\mathbf{x}; \mathbf{a}_m)$ 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.

2.4 Squared error loss

Consider the squared-error loss

$$L(y, \hat{\mu}(\mathbf{x})) = (y - \hat{\mu}(\mathbf{x}))^2$$

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

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

where r_{im} is the residual of the model after $m-1$ iterations on the i th observation. One sees that the term $T(\mathbf{x}; \mathbf{a}_m)$ actually fits the residuals obtained after $m-1$ iterations. Finding the solution to (2.6) is thus no harder than for a single weak learner. It amounts to fit a weak learner on the working training set

$$\mathcal{D}^{(m)} = \{(r_{mi}, \mathbf{x}_i), i = 1, \dots, n\}.$$

2.5 Poisson deviance loss

Consider (2.6) with the Poisson deviance loss

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

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 $\mathbf{x}_i$ and the exposure-to-risk e_i , so that we have

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

In this context, (2.6) becomes

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

which can be expressed as

$$\hat{\mathbf{a}}_m = \underset{\mathbf{a}_m}{\operatorname{argmin}} \sum_{i=1}^n L(y_i, e_{mi} \exp(T(\mathbf{x}_i; \mathbf{a}_m)))$$

with

$$e_{mi} = e_i \exp(\widehat{\text{score}}_{m-1}(\mathbf{x}_i))$$

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

Again, solving (2.6) 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

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

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 the appendix. 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 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.

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

Tweedie deviance loss function is given by

$$L(y, \hat{\mu}(\mathbf{x})) = \begin{cases} (y - \hat{\mu}(\mathbf{x}))^2 & \text{for } \xi = 0 \\ 2 \left(y \ln \frac{y}{\hat{\mu}(\mathbf{x})} - (y - \hat{\mu}(\mathbf{x})) \right) & \text{for } \xi = 1 \\ 2 \left(-\ln \frac{y}{\hat{\mu}(\mathbf{x})} + \frac{y}{\hat{\mu}(\mathbf{x})} - 1 \right) & \text{for } \xi = 2 \\ 2 \left(\frac{\max\{y, 0\}^{2-\xi}}{(1-\xi)(2-\xi)} - \frac{y\hat{\mu}(\mathbf{x})^{1-\xi}}{1-\xi} + \frac{\hat{\mu}(\mathbf{x})^{2-\xi}}{2-\xi} \right) & \text{else.} \end{cases} \quad (3.1)$$

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

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

with $e_{mi} = e_i \exp(\widehat{\text{score}}_{m-1}(\mathbf{x}_i))$. 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 $\tilde{Y} = Y/e$ provided the weight $\nu = e$ enters the analysis. The distribution of the claim rate $\tilde{Y}$ 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

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

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

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

with

$$\nu_{mi} = \nu_i \exp(\widehat{\text{score}}_{m-1}(\mathbf{x}_i))$$

and

$$\tilde{r}_{mi} = \frac{\tilde{y}_i}{\exp(\widehat{\text{score}}_{m-1}(\mathbf{x}_i))}.$$

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

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

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

$$\nu_{mi} = \nu_i \exp(\widehat{\text{score}}_{m-1}(\mathbf{x}_i))^{2-\xi}$$

and

$$\tilde{r}_{mi} = \frac{y_i}{\exp(\widehat{\text{score}}_{m-1}(\mathbf{x}_i))}.$$

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 (2.6) amounts to build a single weak learner on the working training set

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

Proposition 3.1. *Consider the deviance loss function (3.1). Then, (2.6) with the log-link function, that is*

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

can be rewritten as

$$\hat{\mathbf{a}}_m = \underset{\mathbf{a}_m}{\operatorname{argmin}} \sum_{i=1}^n \nu_{mi} L(\tilde{r}_{mi}, \exp(T(\mathbf{x}_i; \mathbf{a}_m))). \tag{3.5}$$

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

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

Now, for the Gamma case $\xi = 2$, it comes

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

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

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

which completes the proof. $\square$

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, 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 WÄCethrich and Buser (2019) for the Poisson distribution to the whole Tweedie class.

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 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 9482 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
Vehicle 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) by boosting trees, randomized GLMs and neural networks. In the former case, estimates are obtained in closed form, by averaging working responses. In the latter cases, we show that forward stagewise fitting is possible using appropriate computational tools. Each time, we assess model performances compared to gradient boosting based on the same weak learners.

4.2 Regression trees as weak learners

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

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

where $\{\chi_t^{(m)}\}_{t \in \mathcal{T}_m}$ is the partition of the feature space χ induced by the regression tree $T(\mathbf{x}; \mathbf{a}_m)$ and $\{c_{tm}\}_{t \in \mathcal{T}_m}$ 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,

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

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 $Y_1, \dots, Y_n$ 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 $Y_1, \dots, Y_n$ 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` 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^D terminal nodes, each terminal node having $D - 1$ 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 whole data set for $D = 1, 2, 3$. The whole data set is used at each iteration to produce the new tree and we do not use a shrinkage parameter. Figure 4.1 displays the Poisson deviance (computed on the whole data set) with respect to the number of iterations. For each value of D , we select the maximum number of iterations M so to minimize the 1000-fold cross validation error of a gradient boosting model with an interaction depth `ID` equals to D , so that a gradient boosting model with M trees with `maxdepth=D` will unlikely overfit the data set. This latter task is performed using the R package `gbm3` that is optimized to compute K -fold cross validation errors for large values of K . The comparison of deviances in Figure 4.1 shows that the boosting algorithm slightly outperforms the gradient boosting approach for $D = 1, 2$ while neither approach outperforms the other for $D = 3$.

	Number of iterations, M								
	1	5	10	15	20	25	30	35	40
Depth=1	6242.73	5849.04	5751.93	5733.645	5715.54	5706.71	5695.24	5684.52	5680.67
	Number of iterations, M								
	1	2	3	4	5	6	7	8	9
Depth=2	6054.67	5879.72	5802.18	5773.84	5741.78	5724.74	5702.26	5689.71	5661.12
	Number of iterations, M								
	10	11	12	13					
Depth=2	5653.61	5647.82	5636.27	5623.85					
	Number of iterations, M								
	1	2	3	4	5	6	7	8	9
Depth=3	5900.68	5790.09	5726.67	5668.36	5634.67	5607.95	5577.98	5549.13	5525.84

Table 4.4: Claims frequency: deviance of gradient boosting using a regression tree as base learner.

	Number of iterations, M								
	1	5	10	15	20	25	30	35	40
Depth=1	6242.74	5828.83	5743.39	5720.62	5704.52	5694.52	5687.06	5681.11	5673.96
	Number of iterations, M								
	1	2	3	4	5	6	7	8	9
Depth=2	6055.05	5853.52	5795.87	5754.62	5728.51	5695.27	5673.23	5663.50	5655.05
	Number of iterations, M								
	10	11	12	13					
Depth=2	5650.56	5637.18	5620.83	5612.29					
	Number of iterations, M								
	1	2	3	4	5	6	7	8	9
Depth=3	5897.34	5768.46	5712.57	5676.71	5641.74	5606.91	5579.50	5567.76	5551.39

Table 4.5: Claims frequency: deviance of boosting using a regression tree as base learner.

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

4.2.2 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` to use the Gamma deviance as loss function. We control the size of the trees with `maxdepth=D`, the whole data set is used at each iteration to produce the new tree and we do not use a shrinkage parameter.

We run Boosting trees and gradient boosting trees for $D = 1, 2, 3$. In Figure 4.2, we show the Gamma deviance (computed on the whole data set) with respect to the number of iterations. For each value of D , we select the maximum number of iterations M so as to minimize the 666-fold cross validation error of a gradient boosting model with an interaction depth ID equals to D , so that a gradient boosting model with M trees with `maxdepth=D` will

	Number of iterations, M									
	1	2	3	4	5	6	7	8	9	10
Depth=1	1233.46	1179.84	1160.22	1147.95	1134.70	1119.96	1107.60	1100.46	1093.62	1084.16
	Number of iterations, M									
	1	2	3	4	5	6	7	8	9	
Depth=2	1158.80	1125.78	1077.65	1044.21	1026.93	1020.34	1008.73	990.05	987.33	
	Number of iterations, M									
	1	2	3	4						
Depth=3	1112.10	1038.98	1015.71	980.88						

Table 4.6: Claims severity: deviance of gradient boosting using a regression tree as base learner.

	Number of iterations, M									
	1	2	3	4	5	6	7	8	9	10
Depth=1	1196.99	1177.04	1150.03	1129.21	1115.55	1102.55	1089.47	1082.49	1076.04	1070.18
	Number of iterations, M									
	1	2	3	4	5	6	7	8	9	
Depth=2	1151.87	1098.01	1054.06	1043.68	1018.53	997.95	987.79	980.64	967.28	
	Number of iterations, M									
	1	2	3	4						
Depth=3	1089.63	1010.12	965.14	935.41						

Table 4.7: Claims severity: deviance of boosting using a regression tree as base learner.

unlikely overfit the data set. This latter task is performed using the R package `gbm3` that is optimized to compute K -fold cross validation errors for large values of K . The comparison of deviances in Figure 4.2 shows that the boosting algorithm outperforms the gradient boosting approach.

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 $T(\mathbf{x}; \mathbf{a}_m)$, $m = 1, 2, \dots, M$ are then linear

Figure 4.2: Claims severity: comparison of deviances obtained with boosting trees (B) and gradient boosting trees (GB).

Number of	Number of epochs, M					
covariates	0	1	5	10	15	20
5	6647.56	6551.28	6375.18	6296.06	6200.4	5947.27
8	6647.56	6564.8	6168.55	6085.84	6017.72	5871.75
10	6647.56	6331.98	5941.73	5818.67	5793.33	5773.2
13	6647.56	6279.25	6065.17	5875.97	5839.77	5812.01

Table 4.8: Claims frequency: deviance of gradient boosting using a randomized GLM as base learner.

Number of	Number of epochs, M					
covariates	0	1	5	10	15	20
5	6647.56	6230.97	5855.28	5797.05	5778.72	5745.52
8	6647.56	6383.19	5812.14	5753.59	5747.76	5746.38
10	6647.56	6215.65	5819.52	5750.4	5748.53	5747.22
13	6647.56	6008.42	5746.69	5742.68	5741.08	5740.72

Table 4.9: Claims frequency: deviance of boosting using a randomized GLM as base learner.

combinations of the M subsets of covariates:

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

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. 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 $Y_1, \dots, Y_n$ 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 20 iterations for both boosting and gradient boosting. At each epoch, a GLM is estimated with 5, 8, 10 and 13 randomly selected covariates. We also limit to 30 the number of GLM iterations in order to weaken the estimator.

Tables 4.8 and 4.9 report the deviances after 1, 5, 10, 15 and 20 epochs of the boosting and gradient boosting algorithms. They are also plotted in Figure 4.3. These graphs confirm that the boosting outperforms gradient boosting for most of chosen weak learners.

Number of	Number of epochs, M					
covariates	0	1	3	5	7	10
1	1378.46	1347.77	1267.56	1222.25	1119.69	1103.33
2	1378.46	1197.89	1144.52	1124.4	1101.09	1090.89
3	1378.46	1259.24	1130.83	1101.07	1093.87	1084.58
4	1378.46	1294.38	1104.25	1089.92	1084.48	1083.98

Table 4.10: Claims severity: deviance of gradient boosting using a GLM as base learner.

Figure 4.3: Claims frequency: comparison of 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 dataset is limited to 666 contracts, we only run 10 iterations for both boosting and gradient boosting. Tables 4.14 and 4.15 present the deviances after 0,1,3, 5, 7 and 10 epochs of the boosting and gradient boosting algorithms. Figure 4.7 allows to compare the evolution of deviances. In this case, the boosting and the gradient boosting methods perform in a similar way.

Number of	Number of epochs, M					
covariates	0	1	3	5	7	10
1	1378.46	1367.04	1217.03	1169.22	1157.86	1134.33
2	1378.46	1224.57	1146.01	1106.2	1097.3	1096.85
3	1378.46	1244.79	1137.03	1114.41	1109.67	1108.05
4	1378.46	1164.87	1119.14	1110.96	1097.95	1096.58

Table 4.11: Claims severity: deviance of boosting using a GLM as base learner.

Figure 4.4: Claims severity: comparison of 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 4.5. The first input layer dispatches the input signals, a vector $\mathbf{x}$ 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 $T(\mathbf{x}; \mathbf{a})$.

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 $\phi_{i,j}(\cdot)$ 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 $\hat{y}_{i,1}$, is computed as follows:

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

Figure 4.5: Illustration of a shallow neural network used as base learner in the Boosting algorithm.

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:

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

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

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 $Y_1, \dots, Y_n$ 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 20 iterations for both boosting and gradient boosting. At each epoch, a shallow network is estimated with KERAS. The intermediate layer counts from 1 to 4 neurons with hyperbolic tangent activation functions. Neural weights are computed after 100 steps of the RMSprop algorithm, with shuffled batches of 2,000 contracts. Tables 4.12 and 4.13 respectively report the deviances after 1, 5, 10, 15 and 20 epochs of the boosting and gradient boosting algorithms. The comparison of deviances in Figure 4.6 confirms that the boosting algorithm outperforms the gradient boosting approach for most of basic learners, whatever the considered number of epochs. The performances

Number of	Number of epochs, M					
Neurons	0	1	5	10	15	20
1	6647.56	6014.63	5841.1	5802.48	5783.22	5781.64
2	6647.56	6117.58	5822.54	5809.71	5794.28	5782.05
3	6647.56	6034.26	5883.65	5844.82	5817.16	5790.01
4	6647.56	6004.18	5881.32	5806.62	5784.56	5777.63

Table 4.12: Claims frequency: deviance of gradient boosting using a shallow neural network as base learner.

Number of	Number of epochs, M					
Neurons	0	1	5	10	15	20
1	6647.56	6055.23	5871.15	5807.66	5796.21	5786.96
2	6647.56	5924.27	5812.24	5781.65	5777.83	5772.17
3	6647.56	5877.23	5791.95	5775.1	5773.63	5767.01
4	6647.56	5824.44	5779.61	5774.42	5769.65	5763

Table 4.13: Claims frequency: deviance of boosting using a shallow neural network as base learner.

are only comparable for the one neuron base learner which is a very weak learner. Running more than 5 iterations of the boosting does not improve the deviance.

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

Number of	Number of epochs, M				
Neurons	0	1	3	5	7
1	1378.46	1274.93	1207.29	1183.88	1163.54
2	1378.46	1220.27	1130.87	1082.12	1054.52
3	1378.46	1165.81	1071.81	1051.39	1025.45
4	1378.46	1127.98	1061.03	1013.69	987.28

Table 4.14: Claims severity: deviance of gradient boosting using a shallow neural network as base learner.

Number of	Number of epochs, M				
Neurons	0	1	3	5	7
1	1378.46	1298.45	1206.37	1173.86	1170.9
2	1378.46	1204.67	1116	1088.66	1054.9
3	1378.46	1181.96	1083.63	1046.77	1028.39
4	1378.46	1167.74	1050.16	995.32	967.92

Table 4.15: Claims severity: deviance of boosting using a shallow neural network as base learner.

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 $Y_1, \dots, Y_n$ are independent and Gamma distributed. Thus, we work with the Gamma deviance as loss function and adopt the log-link. The data set comprises only 693 claims, reported by 666 insureds. Therefore we only run 7 iterations for both boosting and gradient boosting. 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. Tables 4.14 and 4.15 respectively report the deviances after 1, 3, 5 and 7 epochs of the boosting and gradient boosting algorithms. The comparison of deviances is illustrated in Figure 4.7. Again, we cannot conclude that gradient boosting outperforms boosting.

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

Appendix: Gradient boosting

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

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

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

Algorithm 2: Gradient boosting.

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

$$\hat{a}_0 = \arg \min_{a_0} \sum_{i=1}^n L(y_i, g^{-1}(a_0)) . \quad (4.2)$$

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

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

2.1 Calculate the negative gradient $\mathbf{g} = (g_i)_{i=1,\dots,n}$ of the generalized error:

$$g_i = -\frac{\partial L(y_i, \mu_m(\mathbf{x}_i))}{\partial \mu_m(\mathbf{x}_i)} \frac{\partial \mu_m(\mathbf{x}_i)}{\partial \text{score}_m(\mathbf{x}_i)}$$

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

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

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

2.3 Compute $\hat{\beta}_{m+1}$ such that

$$\hat{\beta}_{m+1} = \arg \min_{\beta} \sum_{i=1}^n (L(y_i, g^{-1}(\widehat{\text{score}}_m(\mathbf{x}_i) + \beta T(\mathbf{x}_i; \hat{\mathbf{a}}_{m+1}))) . \quad (4.3)$$

2.4 Update the estimator $\hat{\mu}_{m+1}(\mathbf{x})$ as follows:

$$\hat{\mu}_{m+1}(\mathbf{x}) = g^{-1}(\widehat{\text{score}}_m(\mathbf{x}) + \hat{\beta}_{m+1} T(\mathbf{x}; \hat{\mathbf{a}}_{m+1})) .$$

End for

References

- Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J. (1984). Classification and Regression Trees. Wadsworth Statistics/Probability Series.
- 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.
- Guelman, L. (2012). Gradient boosting trees for auto insurance loss cost modeling and prediction. Expert Systems with Applications 39, 3659-3667.
- Henckaerts, R., Cote, M-P., Antonio, K., Verbelen, R. (2020). Boosting insights in insurance tariff plans with tree-based machine learning methods. North American Actuarial Journal, in press.

- 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.
- 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.