An individual claims reserving model for reported claims

Andrea Gabrielli¹

Received: 28 May 2020 / Revised: 29 September 2020 / Accepted: 18 March 2021 /
Published online: 12 April 2021
© The Author(s) 2021

Abstract

We present a claims reserving technique that uses claim-specific feature and past payment information in order to estimate claims reserves for individual reported claims. We design one single neural network allowing us to estimate expected future cash flows for every individual reported claim. We introduce a consistent way of using dropout layers in order to fit the neural network to the incomplete time series of past individual claims payments. A proof of concept is provided by applying this model to synthetic as well as real insurance data sets for which the true outstanding payments for reported claims are known.

Keywords Claims reserving · Individual claims · RBNS reserves · Neural networks · Multi-task learning · Dropout · Time series · Micro reserving

1 Introduction

Claims reserving is of major importance in general insurance. The main task is to estimate the amount required to cover future payments for claims that have already occurred. Traditionally, these claims reserves are calculated using aggregate data such as upper claims reserving triangles. The most popular method on aggregate data is Mack's chain-ladder model [26]. Working on aggregate data simplifies the modeling task considerably, however, it also leads to a considerable loss of information. Using individual claims data has the potential to improve accuracy of claims reserving. Therefore, developing new claims reserving techniques that go beyond modeling aggregate data has become increasingly popular in actuarial science.

Surveys of recently developed claims reserving techniques can be found in [4] and [33]. The individual claims reserving models introduced in the last couple of years can broadly be separated into models with and without the use of machine learning methods. For semi-parametric individual claims reserving models not using machine learning techniques we refer to [1, 6, 19, 21, 23, 27, 28, 34, 36]. Disadvantages of

✉ Andrea Gabrielli
andrea.gabrielli@bluewin.ch

¹ RiskLab, Department of Mathematics, ETH Zurich, 8092 Zurich, Switzerland

these models are that they often assume too rigid assumptions, making it difficult to apply them in practice. More recently, machine learning has also found its way into claims reserving research, see [2, 8–10, 12, 22, 25, 35] for approaches based on regression trees, gradient boosting machines and neural networks.

We remark that claims reserves can be split into an RBNS (reported but not settled) part and an IBNR (incurred but not reported) part. Here, we focus on modeling RBNS reserves, which consist of estimated expected future payments for claims that have already been reported. Usually, the biggest share of the claims reserves are due to such RBNS claims.

Throughout this manuscript, for the sake of reproducibility, we mainly use synthetic insurance data. However, we complement this analysis by an application on real Swiss accident insurance data. The synthetic data is generated from the individual claims history simulation machine [16]. For every individual reported claim we are equipped with claim-specific feature information such as age or labor sector of the insured. After a claim gets reported, it can trigger multiple (yearly) payments. The time lags (on a yearly scale) between payments and reporting date are called payment delays. At the end of the last considered accident year, for every reported claim we have claim specific feature information and past payment information. The main goal is to use this information in order to estimate expected future payments for every individual reported claim. This corresponds to estimating RBNS reserves.

1.1 A single neural network model

When setting up an individual claims reserving model, a main question is whether modeling every payment delay period separately, resulting in one-period models, or whether jointly modeling all payment delay periods, resulting in a single multi-period model. Here, we design one single neural network in order to jointly model all payment delay periods. Such an approach allows us to learn from one payment delay period to the other. From the machine-learning based individual claims reserving techniques available in the literature, only [22] also offers a joint model, all the other approaches are based on one-period models.

Our single neural network models the expected payments in all considered payment delay periods on the basis of claim-specific feature and past payment information, separately for every individual reported claim. This single neural network consists of several subnets. Every subnet serves the purpose of modeling the expected payment in the corresponding payment delay period. The separate subnets allow for sufficient flexibility to jointly model all considered payment delay periods in one neural network. However, the subnets are still connected to each other by sharing some of the neural network parameters. This increases stability of the neural network, and allows us to learn from one payment delay period to the other. The procedure of simultaneously learning multiple tasks is called multi-task learning, a concept introduced in [5]; for an overview we refer to [30]. For readers not familiar with neural networks, we refer to [11, 13, 17, 24, 31].

Fitting the single neural network to the observed payments then allows us to jointly estimate the expected future payments, separately for every individual

reported claim. This provides us with an estimate of RBNS reserves based on individual claims information.

1.2 Past payment information

Using past payments as covariate information in order to estimate expected future payments is not straightforward. Indeed, to the best of our knowledge, all machine-learning based individual claims reserving techniques available in the literature which are built from one-period models either do not consider past payments as covariate information, or do not provide a consistent model by clearly distinguishing between observed payments (realizations of random variables) and forecasted payments (expectations) in their one-period prediction models.

The key problem of using past payments as covariate information in order to estimate expected future payments is the incomplete time series structure of past payments. Individual claims are equipped with different amounts of past payment information, depending on the reporting years of the claims. For every observed payment for a given individual claim, of course also the previous payments for this individual claim are known. Thus, a single multi-period model blindly fitted to the observed data can rely on full past payment information. However, when estimating expected future payments, not all past payment information is known. Consequently, the resulting estimation might suffer from a systematic bias, as the individual claims reserving model cannot deal with incomplete past payment information.

In [22], which is also based on a single multi-period model, the issue of having different lengths of past payment information is solved by using multiple replications for every individual claim during model fitting. For every replication of a particular individual claim, a different knowledge of past payment information is assumed. This artificial augmentation of the number of individual claims is superfluous in terms of additional information. Moreover, this technique substantially increases the time required to fit the model.

Therefore, we propose a different approach here. The problem of the incomplete time series structure of past payment information is addressed by combining embedding layers and dropout layers. We assume that expected future payments depend on past payments only through the order of magnitude of these payments. We then introduce the additional label “no information” for past payments variables, and set the neural network embedding weights for this label to zero. This does not result in a loss of generality, as there are no restrictions for the other labels. By applying dropout layers on past payment information during neural network fitting, i.e. by randomly setting some of the corresponding neurons to zero, we mimic the situation of missing parts of the past payment information. This forces the neural network to learn how to cope with the situation of not knowing the full past payment information.

For implementation we use the statistical software package R, see [29]. The R code can be downloaded from GitHub: <https://github.com/gabrielliandrea/neuralnetworkindividualrbnsclaimsreserving>.

1.3 Outline

In Sect. 2 we introduce the data used in this manuscript. In Sect. 3 we set up the model framework for individual claims reserving. In Sect. 4 we describe the single neural network architecture of the individual RBNS claims reserving model in detail. In Sect. 5 we train this neural network by applying the dropout technique on past payment information, and we calculate the individual RBNS claims reserves. In Sect. 6 we analyze the individual RBNS claims reserves in terms of neural network stability and in terms of sensitivity with respect to claim-specific feature and past payment information.

2 Data

In this manuscript we mainly use a synthetic insurance portfolio generated from the individual claims history simulation machine [16]. We generate exactly the same insurance portfolio as in [14, 15]. This portfolio is divided into six lines of business (LoBs). As we use the same model separately for every LoB, we refrain from using a LoB component in our notation.

For all six LoBs, we consider individual claims from $I = 12$ accident years. For our RBNS claims reserving exercise we assume that we are at the end of the last considered accident year I , that is we only consider those claims which are reported until the end of calendar year I . We assume that these n reported claims are labeled with $v = 1, \dots, n$, see Table 1 for the numbers of individual reported claims until time I for all six LoBs.

Every individual reported claim v is equipped with corresponding feature information $\mathbf{x}_v = (x_{v,1}, \dots, x_{v,d}) \in \mathbb{R}^d$ with $d = 6$ consisting of

- $x_{v,1}$: claims code `cc` which is categorical with labels in $\mathcal{X}_1 \subset \{1, \dots, 53\}$;
- $x_{v,2}$: accident year `AY` in $\mathcal{X}_2 = \{1, \dots, 12\}$;
- $x_{v,3}$: accident quarter `AQ` in $\mathcal{X}_3 = \{1, \dots, 4\}$;
- $x_{v,4}$: age of the injured `age` (in 5 years age buckets) in $\mathcal{X}_4 = \{20, 25, \dots, 70\}$;
- $x_{v,5}$: injured body part `inj_part`, which is categorical with labels in $\mathcal{X}_5 \subset \{1, \dots, 99\}$;
- $x_{v,6}$: reporting delay `RepDel` in $\mathcal{X}_6 = \{0, \dots, 11\}$.

The feature claims code `cc` denotes the labor sector of the insured. We remark that not all the values in the set $\{1, \dots, 53\}$ are attained. In fact, we have $|\mathcal{X}_1| = 51$ different observed values. Originally, the claims have accident years `AY` in $\{1994, \dots, 2005\}$. In order to ease notation, we relabel these $I = 12$ accident years with values $\{1, \dots, 12\}$. The accident quarter `AQ` labels $\{1, 2, 3, 4\}$ denote the time periods January–March,

Table 1 Total numbers of individual reported claims until time I in the six LoBs

	LoB 1	LoB 2	LoB 3	LoB 4	LoB 5	LoB 6
Number of individual reported claims	248,550	248,718	99,229	247,691	247,290	98,808

April–June, July–September and October–December, respectively, of the accident. The individual claims history simulation machine [16] provides age of the injured age from age 15 to age 70 on a yearly scale. Assuming that the approximate age is sufficient, we simplify our model by summarizing the ages in 5 years age buckets $\{20, 25, \dots, 70\}$. For the feature injured body part inj_part we observe $|\mathcal{X}_5| = 46$ different values. The reporting delay RepDel denotes the time lag (on a yearly scale) between accident year and reporting year. The individual claims history simulation machine [16] assumes that all claims are fully settled after development period $J = 11$, without considering a tail development factor. The accident year $x_{v,2}$ and the reporting delay $x_{v,6}$ determine the reporting year $r_v = x_{v,2} + x_{v,6} \in \{1, \dots, I + J\}$ of claim v , for all $v = 1, \dots, n$. We define $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \dots \times \mathcal{X}_d \subset \mathbb{R}^d$ to be the feature space of the d -dimensional features $\mathbf{x}_1, \dots, \mathbf{x}_n$.

In Fig. 1 we provide the distribution of the portfolio of LoB 1 for all six features, only considering individual claims which have been reported until calendar year I . We approximately have uniform distributions for accident year AY and accident quarter AQ . For reporting delay RepDel we observe a quite imbalanced distribution. Most of the claims have reporting delay 0, and the proportions of claims with reporting delay bigger than 1 are very small.

For every individual reported claim $v = 1, \dots, n$ the individual claims history simulation machine [16] additionally provides us with the complete cash flow

$$\mathbf{Y}_v = (Y_{v,0}, \dots, Y_{v,J-x_{v,6}}) \in \mathbb{R}^{J-x_{v,6}+1}$$

Fig. 1 Distribution of the portfolio of LoB 1 for all six features cc , AY , AQ , age , inj_part and RepDel , only considering individual claims which have been reported until calendar year I

of total yearly payments. Considering $Y_{v,j}$, subscript $j = 0, \dots, J - x_{v,6}$ indicates the payment delay after reporting year r_v . If a total yearly payment $Y_{v,j}$ is equal to 0, then there is no net payment in payment delay period j for claim v . A positive total yearly payment indicates a net payment from the insurance company to the policyholder; a negative total yearly payment a net recovery payment from the policyholder to the insurance company. As we only consider claims developments until development period J , the last yearly payment of claim v is $Y_{v,J-x_{v,6}}$. We remark that the individual claims history simulation machine [16] generates synthetic data with complete knowledge of the cash flows. As we assume to be at the end of calendar year I , all payment information beyond calendar year I is only used for back-testing our predictions.

In Table 2 we present an example claim from LoB 1. In the left part of Table 2 we see the $d = 6$ feature components of this example claim. In particular, as this claim is reported in year $r_v = 9 + 0 = 9$, we know the corresponding payments for payment delay periods $j = 0, 1, 2, 3$. This past payment information is given in the middle part of Table 2. The future payments in the right part of Table 2 correspond to payment delay periods $j = 4, \dots, J$. This is exactly the payment information we only use for back-testing purposes.

In Fig. 2 we provide descriptive statistics for the payments observed in LoB 1 up to calendar year I . In the top row of Fig. 2 we present the empirical probabilities of having a (positive) payment as well as the empirical mean of the logarithmic (positive) payments for all payment delay periods $j = 0, \dots, 11$. We deduce that the probability of a payment decreases with increasing payment delay j , and that the mean logarithmic payment increases for $j = 0, 1, 2, 3$ and is more or less constant for later payment delays $j = 4, \dots, 11$. In the bottom row of Fig. 2 we show the densities of the logarithmic (positive) payments for all payment delay periods $j = 0, \dots, 11$. We observe approximate normal distributions on the log-scale, i.e. log-normal distributions on the original scale. For payment delays $j = 4, \dots, 11$ the densities have a very similar shape, with slightly decreasing variance. For more details on the generated dataset we refer to Appendix A of [15].

In RBNS claims reserving we are interested in the true outstanding payments for reported claims given by

$$R_{\text{RBNS}}^{\text{true}} = \sum_{v=1}^n \sum_{j=I-r_v+1}^{J-x_{v,6}} Y_{v,j}.$$

Our main goal is to estimate the expected outstanding payments $\mathbb{E}[R_{\text{RBNS}}^{\text{true}} | \mathcal{D}_I]$, conditionally given $\mathcal{D}_I$, where $\mathcal{D}_I$ reflects past payment information at time I . In order to tackle this RBNS prediction problem on an individual claims level, we need to define an appropriate model. This is done in the next section.

Table 2 Example claim from LoB 1

cc	AY	AQ	Age	inj_part	RepDel	Pay00	Pay01	Pay02
45	9	4	60	52	0	2337	3526	2227
Pay03	Pay04	Pay05	Pay06	Pay07	Pay08	Pay09	Pay10	Pay11
0	0	1922	1306	0	0	0	0	0

3 Model assumptions

In this section we set up the model framework for individual RBNS claims reserving. We assume that we work on a sufficiently rich probability space $(\Omega, \mathcal{F}, \mathbb{P})$, and that the cash flow

$$Y_v(\mathbf{x}_v) = (Y_{v,0}(\mathbf{x}_v), \dots, Y_{v,J-x_{v,6}}(\mathbf{x}_v)) \in \mathbb{R}^{J-x_{v,6}+1}$$

is a random vector on this probability space depending on the features $\mathbf{x}_v \in \mathbb{R}^d$, for all individual claims $v = 1, \dots, n$. In our run-off situation we assume that we are at the end of calendar year I . Thus, the available information at time I is given by

$$\mathcal{D}_I = \{Y_{v,j}(\mathbf{x}_v) \mid v = 1, \dots, n, j = 0, \dots, J, r_v + j \leq I\}.$$

Our goal is to predict expected future payments, conditionally given information $\mathcal{D}_I$. We assume that expected future payments do not depend on exact values of past payments, but only on the simplified information “no payment”, “recovery payment”, “small payment”, “medium payment”, “big payment” and “extremely big payment”. To this end, we define function $g : \mathbb{R} \mapsto \{0, 1, 2, 3, 4, 5\}$ by

$$g(y) = \begin{cases} 0, & \text{if } y = 0, \\ 1, & \text{if } y < 0, \\ 2, & \text{if } 0 < y \leq 5000, \\ 3, & \text{if } 5000 < y \leq 20,000, \\ 4, & \text{if } 20,000 < y \leq 100,000, \\ 5, & \text{if } y > 100,000. \end{cases} \quad (3.1)$$

Thus, available past payments are summarized using six labels $\{0, 1, 2, 3, 4, 5\}$. We introduce the additional label 6 corresponding to “no information”. This label is used for past payments which are unknown at the time of estimation. Consequently, for all individual reported claims $v = 1, \dots, n$, the past payment information used for estimating the expected payment in payment delay period j is a j -dimensional vector in $\{0, \dots, 6\}^j$. We can now introduce the model assumptions used for our individual RBNS claims reserving model.

Fig. 2 Top row: empirical probabilities of (positive) payments and empirical mean of logarithmic (positive) payment sizes in LoB 1 for all payment delay periods $j = 0, \dots, J$. Bottom row: densities of the logarithmic (positive) payments in LoB 1 for all payment delay periods $j = 0, \dots, J$ (color figure online)

Model Assumptions 1 We assume that

- (a1) the cash flows $Y_1(\mathbf{x}_1), \dots, Y_n(\mathbf{x}_n)$ are independent;
- (a2) for all individual claims $v = 1, \dots, n$, payment delays $j = 1, \dots, J - x_{v,6}$ and $k = 0, \dots, j - 1$ we have

$$\mathbb{E}[Y_{v,j}(\mathbf{x}_v) | Y_{v,0}(\mathbf{x}_v), \dots, Y_{v,k}(\mathbf{x}_v)] = \mathbb{E}[Y_{v,j}(\mathbf{x}_v) | g(Y_{v,0}(\mathbf{x}_v)), \dots, g(Y_{v,k}(\mathbf{x}_v))], \quad \mathbb{P}\text{-a.s.};$$

- (a3) for all individual claims $v = 1, \dots, n$, payment delays $j = 0, \dots, J - x_{v,6}$, $k = 0, \dots, j - 1$ and $y_{v,0}, \dots, y_{v,k} \in \{0, \dots, 5\}$ we have

$$\begin{aligned} \mathbb{P}[Y_{v,j}(\mathbf{x}_v) > 0 | g(Y_{v,0}(\mathbf{x}_v)) = y_{v,0}, \dots, g(Y_{v,k}(\mathbf{x}_v)) = y_{v,k}] \\ = p_j(\mathbf{x}_v, y_{v,0}, \dots, y_{v,k}, 6, \dots, 6), \end{aligned}$$

where $p_j : \mathcal{X} \times \{0, \dots, 6\}^j \rightarrow [0, 1]$ is a probability function;

- (a4) for all individual claims $v = 1, \dots, n$, payment delays $j = 0, \dots, J - x_{v,6}$, $k = 0, \dots, j - 1$ and $y_{v,0}, \dots, y_{v,k} \in \{0, \dots, 5\}$ we have

$$\begin{aligned} \log(Y_{v,j}(\mathbf{x}_v) | \{Y_{v,j}(\mathbf{x}_v) > 0, g(Y_{v,0}(\mathbf{x}_v)) = y_{v,0}, \dots, g(Y_{v,k}(\mathbf{x}_v)) = y_{v,k}\}) \\ \sim \mathcal{N}(\mu_j(\mathbf{x}_v, y_{v,0}, \dots, y_{v,k}, 6, \dots, 6), \sigma_j^2), \end{aligned}$$

where $\mu_j : \mathcal{X} \times \{0, \dots, 6\}^j \rightarrow \mathbb{R}$ is a regression function and $\sigma_j^2 > 0$ a variance parameter;

- (a5) for all individual claims $v = 1, \dots, n$, payment delays $j = 0, \dots, J - x_{v,6}$, $k = 0, \dots, j - 1$ and $y_{v,0}, \dots, y_{v,k} \in \{0, \dots, 5\}$ we have

$$\mathbb{P}[Y_{v,j}(\mathbf{x}_v) < 0 \mid g(Y_{v,0}(\mathbf{x}_v)) = y_{v,0}, \dots, g(Y_{v,k}(\mathbf{x}_v)) = y_{v,k}] = p_-,$$

and

$$\mathbb{E}[Y_{v,j}(\mathbf{x}_v) \mid Y_{v,j}(\mathbf{x}_v) < 0, g(Y_{v,0}(\mathbf{x}_v)) = y_{v,0}, \dots, g(Y_{v,k}(\mathbf{x}_v)) = y_{v,k}] = \mu_-,$$

for some constants $p_- \in [0, 1]$ and $\mu_- < 0$.

We remark that in assumption (a1) we assume that individual claims $v = 1, \dots, n$ are independent from each other. This is a common assumption in individual claims reserving. It allows us to treat all individual claims separately.

Assumption (a2) has already been mentioned above. It implies that expected future payments only depend on past payments through the simplifying function $g(\cdot)$ defined in (3.1).

While assumption (a3) is rather natural, assumption (a4) is more delicate, as claim size modeling in general insurance is difficult. However, we emphasize that the model is not used to simulate future payments, and that we are only interested in expected values. In light of the bottom row of Fig. 2, the log-normal assumption can be justified here. As we are only interested in expected values, the log-normal assumption could also be used if we had observed different payment distributions. However, if desired, the log-normal model of assumption (a3) can of course also be replaced by any other claim size distribution, for example the gamma distribution. With a different claim size distribution only the calculation of the expected payments, see Eq. (3.2) below, and the loss function used for training the model, see Sect. 5.2, would change. We remark that in assumptions (a3) and (a4) we assume that the payments $Y_{v,0}(\mathbf{x}_v), \dots, Y_{v,k}(\mathbf{x}_v)$ are known, for $k < j$. Thus, we can use the past payment information $g(Y_{v,0}(\mathbf{x}_v)), \dots, g(Y_{v,k}(\mathbf{x}_v))$. On the other hand, the payments $Y_{v,k+1}(\mathbf{x}_v), \dots, Y_{v,j-1}(\mathbf{x}_v)$ are not known, and the past payment information corresponding to payment delay periods $k + 1, \dots, j - 1$ is set to the value 6, corresponding to the class “no information”. This implies that this model can be used regardless of the amount of past payment information.

According to assumption (a5), we use a rather crude model for recoveries. We assume that the probability of a recovery and the mean recovery payment are described by feature independent values. The proportion of the recovery payments to the positive payments until time I is roughly only 0.06% for all LoBs. Thus, we choose to keep our model as simple as possible, and to focus on the positive payments. For data sets where recovery payments play a more important role, we could use assumptions (a3) and (a4) also for recoveries.

Using Model Assumptions 1, we have

$$\begin{aligned}
\mathbb{E}[R_{\text{RBNS}}^{\text{true}} | \mathcal{D}_I] &= \sum_{v=1}^n \sum_{j=l-r_v+1}^{J-x_{v,6}} \mathbb{E}\left[Y_{v,j}(\mathbf{x}_v) \mid Y_{v,0}(\mathbf{x}_v), \dots, Y_{v,l-r_v}(\mathbf{x}_v)\right] \\
&= \sum_{v=1}^n \sum_{j=l-r_v+1}^{J-x_{v,6}} \mathbb{E}\left[Y_{v,j}(\mathbf{x}_v) \mid g(Y_{v,0}(\mathbf{x}_v)), \dots, g(Y_{v,l-r_v}(\mathbf{x}_v))\right] \\
&= \sum_{v=1}^n \sum_{j=l-r_v+1}^{J-x_{v,6}} (p_j(\mathbf{x}_v, g(Y_{v,0}(\mathbf{x}_v)), \dots, g(Y_{v,l-r_v}(\mathbf{x}_v)), 6, \dots, 6) \\
&\quad \cdot \exp\left\{\mu_j(\mathbf{x}_v, g(Y_{v,0}(\mathbf{x}_v)), \dots, g(Y_{v,l-r_v}(\mathbf{x}_v)), 6, \dots, 6) + \sigma_j^2/2\right\} \\
&\quad + p_- \mu_-).
\end{aligned} \tag{3.2}$$

In the next section we model the $J + 1$ probability functions $\{p_j(\cdot)\}_{j=0,\dots,J}$ and the $J + 1$ regression functions $\{\mu_j(\cdot)\}_{j=0,\dots,J}$ with a single neural network.

4 Neural network architecture

The neural network presented here jointly models the $J + 1$ probability functions $\{p_j(\cdot)\}_{j=0,\dots,J}$ and the $J + 1$ regression functions $\{\mu_j(\cdot)\}_{j=0,\dots,J}$. The advantage of such a joint model is that we enable learning from one probability/regression function to the other via parameter sharing. That is, we can exploit commonalities and differences between the individual tasks, which can increase model performance. However, setting up a joint model for $2(J + 1) = 24$ different functions is a balancing act: on the one hand, the neural network needs to be flexible in order to capture the structure in the data as good as possible for every modeled function; on the other hand, we have to be careful to avoid overfitting, i.e. to avoid learning all the peculiarities of the training data. Therefore, our neural network framework will have characteristics providing flexibility, and characteristics contributing to stability.

In a first step, we define the embedding layers used in this neural network, see Sect. 4.1. In a second step, we create $J + 1 = 12$ subnets within the neural network, where subnet j models probability function $p_j(\cdot)$ and regression function $\mu_j(\cdot)$ used in payment delay period j , see Sect. 4.2. The basic idea is to create separate feed-forward neural networks for every one of these $J + 1$ subnets, allowing for flexibility in a joint model of all probability functions $\{p_j(\cdot)\}_{j=0,\dots,J}$ and regression functions $\{\mu_j(\cdot)\}_{j=0,\dots,J}$ in one neural network. However, the probability/regression functions are still connected to each other by shared embedding weights. The shared embedding weights increase model stability and allow us to learn from one payment delay period to the other. For simplicity, apart from claim-specific feature and past payment information used, these $J + 1$ subnets have exactly the same design in terms of number of hidden layers and numbers of neurons in these hidden layers.

Finally, in Sect. 4.3 we emphasize the differences between the (neural network based) data generating mechanism and the (neural network based) individual RBNS claims reserving model.

4.1 Embedding layers

For all $j = 0, \dots, J$ probability function $p_j(\cdot)$ and regression function $\mu_j(\cdot)$ are functions defined on $\mathcal{X} \times \{0, \dots, 6\}^j$. As we model all probability functions $\{p_j(\cdot)\}_{j=0, \dots, J}$ and regression functions $\{\mu_j(\cdot)\}_{j=0, \dots, J}$ in one neural network, the input of the neural network is of the form $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \{0, \dots, 6\}^J$, where $\mathbf{x} = (x_1, \dots, x_d) \in \mathcal{X}$ reflects the claim-specific feature information and $\mathbf{y} = (y_0, \dots, y_{J-1}) \in \{0, \dots, 6\}^J$ the claim-specific past payment information.

This input $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \{0, \dots, 6\}^J$ consists of the categorical features claims code `cc`, injured body part `inj_part` and the $J = 11$ past payment informations, as well as on the continuous (but categorized) features accident year `AY`, accident quarter `AQ`, age `age` and reporting delay `RepDel`. In neural networks, embedding layers introduced in [3] provide an easy and elegant way to treat categorical input variables. They enable us to learn a low-dimensional representation of the considered categorical variable, as every level of this categorical variable is mapped to a vector in $\mathbb{R}^p$, for some $p \in \mathbb{N}$. The entries of these embedding vectors are then simply parameters of the neural network, and consequently they have to be trained.

Even though embedding layers have originally been introduced in order to deal with categorical input variables, they can also be used for continuous variables that have been summarized into categories. The four continuous (but categorized) features accident year `AY` (12 categories), accident quarter `AQ` (4 categories), age `age` (11 categories) and reporting delay `RepDel` (12 categories summarized to 3 categories, see below) do not consist of too many categories. Therefore, believing that the gain in flexibility outweighs the loss of stability, we use embedding layers also for these input variables.

We choose two-dimensional embeddings for all inputs $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \{0, \dots, 6\}^J$ except for the accident year `AY` represented by x_2 , for which we use a three-dimensional embedding. We come back to the reason for this choice in Sect. 4.2.4, below. For all $l = 1, 3, \dots, d$ the two-dimensional embedding of the levels $u_{l,1}, \dots, u_{l,|\mathcal{X}_l|} \in \mathcal{X}_l$ is defined by

$$\begin{aligned} \alpha_l : \mathcal{X}_l &\rightarrow \mathbb{R}^2, \\ u_{l,k} &\mapsto \alpha_l(u_{l,k}) = (\alpha_{l,1}(u_{l,k}), \alpha_{l,2}(u_{l,k})), \end{aligned}$$

for all $k = 1, \dots, |\mathcal{X}_l|$. The set $\{\alpha_{l,1}(u_{l,k}), \alpha_{l,2}(u_{l,k})\}_{k=1, \dots, |\mathcal{X}_l|}$ is the set of parameters to be learned for feature l representing the claims code `cc` ($l = 1$), the accident quarter `AQ` ($l = 3$), the age of the injured `age` ($l = 4$), the part of the body injured `inj_part` ($l = 5$) and the reporting delay `RepDel` ($l = d = 6$). For accident year `AY` represented by x_2 we use a three-dimensional embedding

$$\begin{aligned} \alpha_2 : \mathcal{X}_2 &\rightarrow \mathbb{R}^3, \\ u_{2,k} &\mapsto \alpha_2(u_{2,k}) = (\alpha_{2,1}(u_{2,k}), \alpha_{2,2}(u_{2,k}), \alpha_{2,3}(u_{2,k})), \end{aligned}$$

where $\{\alpha_{2,1}(u_{2,k}), \alpha_{2,2}(u_{2,k}), \alpha_{2,3}(u_{2,k})\}_{k=1, \dots, |\mathcal{X}_2|}$ is the set of parameters to be learned for feature accident year `AY`.

In Fig. 1 we have seen that in LoB 1 reporting delays bigger than 1 occur very rarely. We have a similar situation for the other LoBs. Therefore, we set $\alpha_{d,1}(r) = \alpha_{d,1}(2)$ and $\alpha_{d,2}(r) = \alpha_{d,2}(2)$ for all $r = 3, \dots, 11$, i.e. reporting delays bigger than 1 are summarized into one category, and are treated in the same way.

For all J elements of the past payment information $\mathbf{y} = (y_0, \dots, y_{J-1}) \in \{0, \dots, 6\}^J$ we use a two-dimensional embedding

$$\begin{aligned}\beta &: \{0, \dots, 6\} \rightarrow \mathbb{R}^2, \\ u &\mapsto \beta(u) = (\beta_1(u), \beta_2(u)),\end{aligned}$$

where $\{\beta_1(u), \beta_2(u)\}_{u=0, \dots, 6}$ is the set of parameters to be learned for the past payment information. We remark that one could define separate embedding layers for every element of the past payment information $\mathbf{y} = (y_0, \dots, y_{J-1}) \in \{0, \dots, 6\}^J$. However, for all $k = 0, \dots, J-1$, element y_k corresponds to the payment in payment delay period k . As we observe less and less payments for later payment delays, with separate embedding layers for elements of the past payment information the model can become unstable. Moreover, it may also make sense to use the same embedding weights for all elements of past payment information, as it allows the neural network to transfer model structure from one payment delay period to the other.

We recall that $y_k = 0, 1, \dots, 5$ indicates the order of magnitude of the payment in payment delay period k . If this information is not known at time I , then we have $y_k = 6$. While the embedding weights $\beta(0), \dots, \beta(5)$ have to be trained, we set $\beta(6) = (\beta_1(6), \beta_2(6)) = (0, 0)$, and we declare these two embedding weights to be non-trainable, i.e. they do not change during model fitting. As there are no restrictions for the other embedding weights, this choice does not result in a loss of generality. It only fixes the origin of the embedding. With this choice we can use dropout layers during neural network fitting in order to mimic the situation of not knowing the full past payment information, see Sect. 5.3.

Summarizing, the embedding layers map the input variables $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \{0, \dots, 6\}^J$ to

$$(\alpha_1(x_1), \dots, \alpha_d(x_d), \beta(y_0), \dots, \beta(y_{J-1})) \in \mathbb{R}^{2(d+J)+1}.$$

4.2 Subnet j of the neural network

The single neural network consists of $J+1$ subnets, where, for all $j = 0, \dots, J$, subnet j models probability function $p_j(\cdot)$ and regression function $\mu_j(\cdot)$. In Fig. 3 we provide the model architecture of subnet $j = 3$. The five colors reflect the different parts of the model. The input variables $(x_1, \dots, x_d, y_0, y_1, y_2) \in \mathcal{X} \times \{0, \dots, 6\}^3$ are given in blue color, the two-dimensional embedding weights from the previous section in green color. The first set of embedding weights of the input variables $(x_1, \dots, x_d, y_0, y_1, y_2) \in \mathcal{X} \times \{0, \dots, 6\}^3$ is processed through a classical feed-forward neural network with two hidden layers. This is the orange part of the model in Fig. 3. Probability function $p_3(\cdot)$ and regression function $\mu_3(\cdot)$ both share this feed-forward neural network. In order to increase the task-specific flexibility, the feed-forward

neural network is processed through additional hidden layers. On the one hand, the feed-forward neural network is followed by a third hidden layer which is used to model probability function $p_3(\cdot)$. Additionally to this third hidden layer, probability function $p_3(\cdot)$ is directly linked to the second set of embedding weights of the input variables $(x_1, \dots, x_d, y_0, y_1, y_2) \in \mathcal{X} \times \{0, \dots, 6\}^3$. This is the red part of the model. On the other hand, the feed-forward neural network is followed by a fourth hidden layer which is used to model regression function $\mu_3(\cdot)$. Completely analogously to above, regression function $\mu_3(\cdot)$ is also directly linked to the second set of embedding weights of the input variables $(x_1, \dots, x_d, y_0, y_1, y_2) \in \mathcal{X} \times \{0, \dots, 6\}^3$. This is the magenta part of the model. The box labeled “second embedding weights” appears twice in Fig. 3 only for graphical reasons.

The individual parts of this neural network model are described in detail below. As all $J + 1$ subnets have exactly the same neural network design, we present subnet j for a general $j = 0, \dots, J$. The only difference between the $J + 1$ subnets are the input variables used. For subnet j we use input variables of the form $(\mathbf{x}, y_0, \dots, y_{j-1}) = (x_1, \dots, x_d, y_0, \dots, y_{j-1}) \in \mathcal{X} \times \{0, \dots, 6\}^j$.

4.2.1 Feed-forward neural network

This part of the model corresponds to the orange part in Fig. 3. The $(d + j)$ -dimensional vector

$$\mathbf{z}_0^{(j)} = (\alpha_{1,1}(x_1), \alpha_{2,1}(x_2), \dots, \alpha_{d,1}(x_d), \beta_1(y_0), \dots, \beta_1(y_{j-1})) \in \mathbb{R}^{d+j}$$

serves as input of a standard feed-forward neural network with two hidden layers. The superscript (j) in $\mathbf{z}_0^{(j)}$ refers to subnet j , and the subscript 0 to the input layer of the feed-forward neural network of this subnet j .

We remark that the input vector $\mathbf{z}_0^{(j)} \in \mathbb{R}^{d+j}$ only contains the corresponding first embedding values $\alpha_{1,1}(x_1), \dots, \alpha_{d,1}(x_d), \beta_1(y_0), \dots, \beta_1(y_{j-1})$ of the inputs $x_1, \dots, x_d, y_0, \dots, y_{j-1}$. The corresponding second embedding values $\alpha_{1,2}(x_1), \dots, \alpha_{d,2}(x_d), \beta_2(y_0), \dots, \beta_2(y_{j-1})$ are not processed through the feed-forward neural network but are directly linked to the outputs $p_j(\cdot)$ and $\mu_j(\cdot)$ of this subnet via a skip connection in order to speed up learning, see below.

As activation function of the feed-forward neural network we choose the hyperbolic tangent activation function $\phi = \tanh$. This activation function has the advantage that it allows to calculate gradients in an efficient manner and, as it is bounded, that the activations in the neurons do not explode. These two properties are useful in neural network fitting as they can considerably accelerate gradient descent methods.

The first and second hidden layer of this feed-forward neural network of subnet j consist of q_1 and q_2 neurons, respectively, and are denoted by $\mathbf{z}_1^{(j)}$ and $\mathbf{z}_2^{(j)}$, respectively. From the universal approximation property of neural networks, see [7] and [20], one hidden layer is sufficient to approximate any continuous and compactly supported function arbitrarily well. The drawback of choosing only one hidden layer is that we may require too many neurons, which makes the model more difficult to calibrate. Moreover, additional hidden layers stimulate the learning of interactions

Fig. 3 Model architecture of subnet $j = 3$. The five colors (blue, green, orange, red and magenta) reflect the different parts of the model (color figure online)

between input features. Therefore, we choose two hidden layers for the feed-forward neural network part. We come back to the numbers of neurons q_1 and q_2 below.

4.2.2 Probability function $p_j(\cdot)$

The model for probability function $p_j(\cdot)$ corresponds to the red part in Fig. 3. As the second hidden layer $\mathbf{z}_2^{(j)}$ is used by both probability function $p_j(\cdot)$ and regression function $\mu_j(\cdot)$, we introduce additional task-dependent hidden layers. The second hidden layer $\mathbf{z}_2^{(j)}$ is processed to a third hidden layer $\mathbf{z}_3^{(j)}$ with q_3 neurons, which is used together with the second embedding values $\alpha_{1,2}(x_1), \dots, \alpha_{d,2}(x_d), \beta_2(y_0), \dots, \beta_2(y_{j-1})$ of the inputs $x_1, \dots, x_d, y_0, \dots, y_{j-1}$ to define probability function $p_j(\cdot)$. As $p_j(\cdot)$ models a probability, its range has to be in the unit interval $[0,1]$. Therefore, as output function we use the logistic function, which we denote here by $\varphi(\cdot)$. Probability function $p_j(\cdot)$ is then defined by

$$p_j(\mathbf{x}, y_0, \dots, y_{j-1}) = \varphi\left(\gamma_{3,0}^{(j)} + \sum_{l=1}^d \gamma_{3,l}^{(j)} \alpha_{l,2}(x_l) + \sum_{l=0}^{j-1} \delta_{3,l}^{(j)} \beta_2(y_l) + \langle \boldsymbol{\eta}_3^{(j)}, \mathbf{z}_3^{(j)} \rangle\right) \in (0, 1), \quad (4.1)$$

where $\gamma_{3,0}^{(j)}, \dots, \gamma_{3,d}^{(j)}, \delta_{3,0}^{(j)}, \dots, \delta_{3,j-1}^{(j)} \in \mathbb{R}$ and $\boldsymbol{\eta}_3^{(j)} \in \mathbb{R}^{q_3}$ are parameters of the neural network, and the operation $\langle \cdot, \cdot \rangle$ denotes the usual scalar product in Euclidean space.

We remark that as in classical neural network modeling, the output neuron containing the probability function $p_j(\cdot)$ depends on the previous hidden layer $\mathbf{z}_3^{(j)}$. However, in order to speed up learning, we directly link the second embedding values $\alpha_{1,2}(x_1), \dots, \alpha_{d,2}(x_d), \beta_2(y_0), \dots, \beta_2(y_{j-1})$ of the inputs $x_1, \dots, x_d, y_0, \dots, y_{j-1}$ to

the output neuron. This step can be interpreted as a skip connection between the embedding layers and the output neuron. Skip connections have originally been introduced in [18], and help to speed up training of deep neural networks by linking non-consecutive layers. The direct dependence of probability function $p_j(\cdot)$ on the second embedding values $\alpha_{1,2}(x_1), \dots, \alpha_{d,2}(x_d), \beta_2(y_0), \dots, \beta_2(y_{j-1})$ of the inputs $x_1, \dots, x_d, y_0, \dots, y_{j-1}$ can also be viewed as GLM (generalized linear model) step, which is additionally supported by the neural network through the hidden layer $z_3^{(j)}$.

4.2.3 Regression function $\mu_j(\cdot)$

In order to also have a task-dependent layer for regression function $\mu_j(\cdot)$, the second hidden layer $z_2^{(j)}$ is processed to a fourth hidden layer $z_4^{(j)}$ with q_4 neurons. Thus, the third hidden layer and the fourth hidden layer are not consecutive but parallel layers. Regression function $\mu_j(\cdot)$ is defined using the fourth hidden layer $z_4^{(j)}$ together with the second embedding values $\alpha_{1,2}(x_1), \alpha_{3,2}(x_3), \dots, \alpha_{d,2}(x_d), \beta_2(y_0), \dots, \beta_2(y_{j-1})$ of the inputs $x_1, x_3, \dots, x_d, y_0, \dots, y_{j-1}$ as well as the third embedding value $\alpha_{2,3}(x_2)$ of the input x_2 representing accident year AY. As $\mu_j(\cdot)$ models the mean parameter of a log-normal distribution, its range is the whole real line. We define

$$\begin{aligned} \mu_j(\mathbf{x}, y_0, \dots, y_{j-1}) &= \gamma_{4,0}^{(j)} + \sum_{\substack{l=1 \\ l \neq 2}}^d \gamma_{4,l}^{(j)} \alpha_{l,2}(x_l) + \gamma_{4,2}^{(j)} \alpha_{2,3}(x_2) \\ &\quad + \sum_{l=0}^{j-1} \delta_{4,l}^{(j)} \beta_2(y_l) + \langle \boldsymbol{\eta}_4^{(j)}, \mathbf{z}_4^{(j)} \rangle \in \mathbb{R}, \end{aligned} \quad (4.2)$$

where $\gamma_{4,0}^{(j)}, \dots, \gamma_{4,d}^{(j)}, \delta_{4,0}^{(j)}, \dots, \delta_{4,j-1}^{(j)} \in \mathbb{R}$ and $\boldsymbol{\eta}_4^{(j)} \in \mathbb{R}^{q_4}$ are parameters of the neural network. The peculiarity of input value x_2 representing accident year AY is explained next.

4.2.4 Feature accident year AY

For the feature accident year AY we face a very particular problem. The single neural network has to be trained using individual claims information available at time I . This implies that for payment delay period $j = 0$ we have payment observations $Y_{v,0}$ for all accident years $\{1, \dots, I\}$. But this is not the case for later payment delay periods $j = 1, \dots, J$. For example for the last considered payment delay period J we only have observations $Y_{v,J}$ for individual claims v with accident year $x_{v,2} = 1$. Therefore, in subnet $j = J$ the neural network weights corresponding to the neurons containing the embedding weights of the accident years are only trained for accident year equal to 1. Thus, we cannot control the effect of these weights on later accident years. However, we would like to estimate expected payments in this last payment delay period J also for claims with accident years greater than 1.

This problem is circumvented by sharing neural network weights for the feature accident year. For all subnets $j = 0, \dots, J$ we use the same q_1 weights linking the

embedding value $\alpha_{2,1}(x_2)$ to the first hidden layer, and the same two weights linking the embedding values $\alpha_{2,2}(x_2)$ and $\alpha_{2,3}(x_2)$ to the probability function and the regression function, respectively. This ensures that these weights can be trained for all accident years, yielding a better control for this feature component. As this reduces flexibility of the probability/regression models w.r.t. the feature accident year AY , we use as countermeasure three embedding weights for this feature component. The first embedding weight $\alpha_{2,1}(x_2)$ is used as input of the feed-forward neural networks, the second embedding weight $\alpha_{2,2}(x_2)$ as GLM part for probability functions $\{p_j(\cdot)\}_{j=0,\dots,J}$, and the third embedding weight $\alpha_{2,3}(x_2)$ as GLM part for regression functions $\{\mu_j(\cdot)\}_{j=0,\dots,J}$.

4.2.5 Numbers of neurons

Usually, one selects the numbers of neurons by training the neural network for a set of possible numbers of neurons and opting for the combination for which one observes the lowest out-of-sample loss. However, here we directly choose small numbers of neurons in order to decrease the risk of over-fitting and to control run times and stability. Following the guideline $q_1 > q_2 > q_3 + q_4$, reflecting the idea that information processed through the hidden layers should be condensed until it reaches the output, we decide to use

$$q_1 = 40, \quad q_2 = 30, \quad q_3 = q_4 = 10.$$

We remark that we choose small numbers of neurons and that the neural network is performing multi-task learning, which already acts as a regularization technique. Therefore, apart from early stopping, see Sects. 5.4 and 5.5, we refrain from using additional regularization techniques.

4.2.6 Output of the neural network model

Summarizing, for an input $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \{0, \dots, 6\}^J$, the output of the single neural network are the $J + 1$ probability functions $\{p_j(\cdot)\}_{j=0,\dots,J}$ and the $J + 1$ regression functions $\{\mu_j(\cdot)\}_{j=0,\dots,J}$. Training this neural network allows us to learn from one probability/regression function to the other. This is done in the next section.

We remark that even though all $J + 1$ probability functions and all $J + 1$ regression functions are modeled in a single neural network, we cannot train all $2(J + 1)$ output functions for every individual claim $v = 1, \dots, n$. For an individual claim v with reporting year $r_v \in \{1, \dots, I\}$, we can only train the probability functions $p_0(\cdot), \dots, p_{I-r_v}(\cdot)$, and the regression functions $\mu_j(\cdot)$ for those payment delay periods $j = 0, \dots, I - r_v$ for which we have observed a positive payment. This requires a special discussion of the choice of loss function, see Sect. 5.2.

4.3 Differences of the claims reserving model to the data generating model

The synthetic data introduced in Sect. 2 and used in this paper is generated from the individual claims history simulation machine [16]. In this section we highlight the

differences between this data generating mechanism (based on 35 neural networks) and the individual RBNS claims reserving model (based on one single neural network). We observe three crucial differences.

The data generating mechanism of the individual claims history simulation machine [16] is based on multiple steps which are performed for every individual claim. In a first step, the features `LoB`, `cc`, `AY`, `AQ`, `age` and `inj_part` are used to model the reporting delay. In a second step, the reporting delay is added to the features and a payment indicator is modeled. This payment indicator is a binary variable indicating whether we observe at least one payment or whether there is no payment. In a third step, we model the exact number of payments. Again augmenting the feature space with the new variable, in a fourth step we model the total individual claim size. In a last step, we add the total individual claim size to the features and we model a payout pattern according to which the total individual claim size is split to the respective development delay periods. For full details of the construction of the individual claims history simulation machine we refer to [16].

In the individual RBNS claims reserving model presented in this paper we model every `LoB` separately and we start with the features `cc`, `AY`, `AQ`, `age` and `inj_part` and `RepDel`. Then, instead of modeling one single payment indicator for all development delay periods, we separately model payment indicators for all considered payment delay periods. Here we have a first crucial structural difference. Moreover, instead of modeling a total individual claim size and allocating it to the development delay periods according to a payout pattern, we separately model the payment sizes for all considered payment delay periods. This is a second crucial difference.

A third crucial difference is the treatment of the features. In the individual claims history simulation machine [16] every label of a categorical feature is replaced by the sample mean of the considered response variable restricted to the corresponding feature label. Then, a MinMax scaler is applied to all features. In the individual RBNS claims reserving model we use embedding layers for all features.

We conclude that the individual RBNS claims reserving model presented in this paper does not exploit the structure of the data generating mechanism. Thus, the validity of the individual RBNS claims reserving model is not reduced. In fact, in Sect. 5.9 we also successfully apply the model to a real insurance data set.

5 Individual RBNS claims reserves

There remains to estimate the $J + 1$ probability functions $\{p_j(\cdot)\}_{j=0,\dots,J}$, the $J + 1$ regression functions $\{\mu_j(\cdot)\}_{j=0,\dots,J}$, the $J + 1$ variance parameters $\{\sigma_j^2\}_{j=0,\dots,J}$, the recovery probability p_- and the recovery mean μ_- . Training of the single neural network modeling the $J + 1$ probability functions $\{p_j(\cdot)\}_{j=0,\dots,J}$ and the $J + 1$ regression functions $\{\mu_j(\cdot)\}_{j=0,\dots,J}$ is described in Sects. 5.1–5.5. In Sect. 5.6 we estimate the variance parameters $\{\sigma_j^2\}_{j=0,\dots,J}$. For the recovery probability p_- and the recovery mean μ_- we refer to Sect. 5.7. The resulting individual RBNS claims reserves are

presented in Sect. 5.8. Moreover, in Sect. 5.9 we present the results of an application of the individual RBNS claims reserving model on real accident insurance data.

In this section we use notation $Y_{v,j}$ instead of $Y_{v,j}(\mathbf{x}_v)$ for payments observed at time I , as here we are not interested in modeling the observed payments as random variables, but rather in using these payments in order to fit the model and estimate the expected future payments.

The single neural network is trained using the Keras library within R. Neural network modeling with Keras involves some randomness, for example for the initialization of the neural network weights or the choice of the mini-batches. In order to get reproducible results, we set the Keras seed, in a random but reproducible way.

5.1 Parameter initialization

Before training a neural network, it might make sense to choose particular starting values for some parameters of the neural network. Such an initialization allows to exploit information which is already available, and it can speed up training. For all subnets $j = 0, \dots, J$, in (4.1) and (4.2) we initially set

$$\gamma_{3,l}^{(j)} = \gamma_{4,l}^{(j)} = 0, \quad \delta_{3,k}^{(j)} = \delta_{4,k}^{(j)} = 0, \quad \boldsymbol{\eta}_3^{(j)} = \mathbf{0} \in \mathbb{R}^{q_3} \quad \text{and} \quad \boldsymbol{\eta}_4^{(j)} = \mathbf{0} \in \mathbb{R}^{q_4}, \quad (5.1)$$

for all $l = 1, \dots, d$ and $k = 0, \dots, j-1$. Moreover, we define

$$a_j = \frac{\sum_{v=1}^n \mathbb{1}_{\{Y_{v,j} > 0\}} \mathbb{1}_{\{r_v + j \leq I\}}}{\sum_{v=1}^n \mathbb{1}_{\{r_v + j \leq I\}}} \quad \text{and} \quad b_j = \frac{\sum_{v=1}^n \log(Y_{v,j}) \mathbb{1}_{\{Y_{v,j} > 0\}} \mathbb{1}_{\{r_v + j \leq I\}}}{\sum_{v=1}^n \mathbb{1}_{\{Y_{v,j} > 0\}} \mathbb{1}_{\{r_v + j \leq I\}}},$$

where a_j is the empirical probability of having a positive payment in payment delay period j among all claims v for which $Y_{v,j}$ is known at time I , and b_j is the empirical mean logarithmic payment size in payment delay period j among all claims v for which $Y_{v,j}$ is known at time I and $Y_{v,j} > 0$. Then, for all subnets $j = 0, \dots, J$ we set

$$\gamma_{3,0}^{(j)} = \log\left(\frac{a_j}{1 - a_j}\right) \quad \text{and} \quad \gamma_{4,0}^{(j)} = b_j. \quad (5.2)$$

With initializations (5.1) and (5.2), the probability function $p_j(\cdot)$ and the regression function $\mu_j(\cdot)$ are initially given by

$$p_j(\mathbf{x}, y_0, \dots, y_{j-1}) = a_j \quad \text{and} \quad \mu_j(\mathbf{x}, y_0, \dots, y_{j-1}) = b_j,$$

for all $j = 0, \dots, J$. This initialization implies that we start from the homogeneous model not considering any covariate information.

5.2 Loss function

In order to train the neural network using information $\mathcal{D}_I$, we have to define a loss function. As we jointly model the $J + 1 = 12$ probability functions $\{p_j(\cdot)\}_{j=0,\dots,J}$ and the $J + 1 = 12$ regression functions $\{\mu_j(\cdot)\}_{j=0,\dots,J}$, the loss function of the

neural network consists of the individual loss functions of these $2(J+1) = 24$ tasks. As we do not have complete payment information for all individual claims, and as we do not have payments in all payment delay periods of the individual claims, not all of the output functions of the single neural network can be trained for every individual claim.

Let $j = 0, \dots, J$. For probability function $p_j(\cdot)$ of subnet j the available response variables are given by

$$\left\{ \mathbb{1}_{\{Y_{v,j} > 0\}} \mid v = 1, \dots, n, r_v + j \leq I \right\}.$$

We define the binary cross-entropy loss function

$$\begin{aligned} \mathcal{L}_{j,1}(p_j(\cdot), \mathcal{D}_I) = & -\frac{2}{n} \sum_{v=1}^n \left(\log(p_j(\mathbf{x}_v, g(Y_{v,0}), \dots, g(Y_{v,j-1}))) \mathbb{1}_{\{Y_{v,j} > 0\}} \right. \\ & \left. + \log(1 - p_j(\mathbf{x}_v, g(Y_{v,0}), \dots, g(Y_{v,j-1}))) (1 - \mathbb{1}_{\{Y_{v,j} > 0\}}) \right) \mathbb{1}_{\{r_v + j \leq I\}}. \end{aligned}$$

We remark that those claims v for which $r_v + j > I$, i.e. for which $Y_{v,j} \notin \mathcal{D}_I$, do not contribute to the loss function. Thus, we only use information known at time I .

For regression function $\mu_j(\cdot)$ of subnet j the available response variables are given by

$$\{\log(Y_{v,j}) \mid v = 1, \dots, n, r_v + j \leq I, Y_{v,j} > 0\}.$$

We define the mean square loss function

$$\begin{aligned} \mathcal{L}_{j,2}(\mu_j(\cdot), \mathcal{D}_I) = & \frac{1}{n} \sum_{v=1}^n \left(\mu_j(\mathbf{x}_v, g(Y_{v,0}), \dots, g(Y_{v,j-1})) - \log(Y_{v,j}) \right)^2 \\ & \times \mathbb{1}_{\{Y_{v,j} > 0\}} \mathbb{1}_{\{r_v + j \leq I\}}. \end{aligned}$$

The mean square loss function $\mathcal{L}_{j,2}(\mu_j(\cdot), \mathcal{D}_I)$ only uses information $\mathcal{D}_I$ known at time I . Moreover, claims v for which $Y_{v,j} \leq 0$ do not contribute to the loss function, as we model the positive payment size conditionally given there is a positive payment.

Considering our example claim of Table 2, we see that this claim can be used to train probability functions $p_0(\cdot), p_1(\cdot), p_2(\cdot), p_3(\cdot)$ as well as regression functions $\mu_0(\cdot), \mu_1(\cdot), \mu_2(\cdot)$. In particular, we use

- the embedded features $(\alpha_1(45), \alpha_2(9), \alpha_3(4), \alpha_4(60), \alpha_5(52), \alpha_6(0)) \in \mathbb{R}^{13}$ in order to train $p_0(\cdot)$ and $\mu_0(\cdot)$, with corresponding responses 1 and $\log(2337)$, respectively;
- the embedded features $(\alpha_1(45), \alpha_2(9), \alpha_3(4), \alpha_4(60), \alpha_5(52), \alpha_6(0)) \in \mathbb{R}^{13}$ and past payment information $(\beta(g(2337))) = (\beta(2)) \in \mathbb{R}$ in order to train $p_1(\cdot)$ and $\mu_1(\cdot)$, with corresponding responses 1 and $\log(3526)$, respectively;
- the embedded features $(\alpha_1(45), \alpha_2(9), \alpha_3(4), \alpha_4(60), \alpha_5(52), \alpha_6(0)) \in \mathbb{R}^{13}$ and past payment information $(\beta(g(2337)), \beta(g(3526))) = (\beta(2), \beta(2)) \in \mathbb{R}^2$ in

order to train $p_2(\cdot)$ and $\mu_2(\cdot)$, with corresponding responses 1 and $\log(2227)$, respectively;

- the embedded features $(\alpha_1(45), \alpha_2(9), \alpha_3(4), \alpha_4(60), \alpha_5(52), \alpha_6(0)) \in \mathbb{R}^{13}$ and past payment information $(\beta(g(2337)), \beta(g(3526)), \beta(g(2227))) = (\beta(2), \beta(2), \beta(2)) \in \mathbb{R}^3$ in order to train $p_3(\cdot)$, with corresponding response 0.

For our example claim of Table 2 we cannot train $\mu_3(\cdot)$ as we did not observe a payment in payment delay period $j = 3$. Moreover, we cannot train $p_4(\cdot), \dots, p_J(\cdot)$ and $\mu_4(\cdot), \dots, \mu_J(\cdot)$ because these functions correspond to future payments which are unknown at time I .

Training the single neural network requires simultaneous minimization of the loss functions $\{\mathcal{L}_{j,1}(p_j(\cdot), \mathcal{D}_I), \mathcal{L}_{j,2}(\mu_j(\cdot), \mathcal{D}_I)\}_{j=0, \dots, J}$. In multi-task learning this is achieved by minimizing a weighted sum of the considered individual loss functions. Therefore, we define the global loss function of the single neural network by

$$\mathcal{L}(\{p_j(\cdot), \mu_j(\cdot)\}_{j=0, \dots, J}, \mathcal{D}_I) = \sum_{j=0}^J w_{j,1} \mathcal{L}_{j,1}(p_j(\cdot), \mathcal{D}_I) + w_{j,2} \mathcal{L}_{j,2}(\mu_j(\cdot), \mathcal{D}_I), \quad (5.3)$$

where, for all $j = 0, \dots, J$,

$$w_{j,1} = \left(-\frac{2}{n} \sum_{v=1}^n \left(\log(a_j) \mathbb{1}_{\{Y_{v,j} > 0\}} + \log(1 - a_j)(1 - \mathbb{1}_{\{Y_{v,j} > 0\}}) \right) \mathbb{1}_{\{r_v + j \leq I\}} \right)^{-1} \quad (5.4)$$

and

$$w_{j,2} = \left(\frac{1}{n} \sum_{v=1}^n (b_j - \log(Y_{v,j}))^2 \mathbb{1}_{\{Y_{v,j} > 0\}} \mathbb{1}_{\{r_v + j \leq I\}} \right)^{-1}. \quad (5.5)$$

The weights defined in (5.4) and (5.5) ensure that all the $2(J+1)$ individual loss functions $\{\mathcal{L}_{j,1}(p_j(\cdot), \mathcal{D}_I), \mathcal{L}_{j,2}(\mu_j(\cdot), \mathcal{D}_I)\}_{j=0, \dots, J}$ live on the same scale, i.e. that they contribute equally to the overall loss function. This normalization step is important to make sure that all tasks of the neural network can be learned.

We train the single neural network by minimizing the global loss function given in (5.3). This can be achieved by iteratively applying (a version of) the stochastic gradient descent method. Here, we choose mini-batches of size 10,000 and the optimizer nadam within the Keras library; we refer to Sections 8.1.3 and 8.5.2 of [17] for theoretical background.

Overall, the single neural network has a total of

$$\begin{aligned}
& 2(|\mathcal{X}_1| + |\mathcal{X}_3| + |\mathcal{X}_4| + |\mathcal{X}_5|) \\
& + 3|\mathcal{X}_2| + 2(|\mathcal{X}_6| - 9) + 2 \cdot 6 + q_1 + 2 + \sum_{j=0}^J ((d+j)q_1 \\
& + (q_1 + 1)q_2 + (q_2 + 1)(q_r + q_s) + 2(d+j) + q_r + q_s) = 28,568
\end{aligned}$$

trainable parameters. Because of this large number of parameters, we have to be careful to not over-fit the model to the observations $\mathcal{D}_I$. Therefore, we apply early stopping. This means that we have to very carefully select the number of gradient descent steps. This will be done in Sects. 5.4 and 5.5.

5.3 Training of a time series model with dropout

Training the single neural network described in Sect. 4 with loss function from Sect. 5.2, we face a structural problem: For every payment $Y_{v,j}$ known at time I , also the previous payments $Y_{v,0}, \dots, Y_{v,j-1}$ are known at time I . Thus, for the corresponding past payment information we have $g(Y_{v,0}), \dots, g(Y_{v,j-1}) \in \{0, \dots, 5\}$, with embedding weights $(\beta_l(g(Y_{v,0})), \dots, \beta_l(g(Y_{v,j-1}))) \in \mathbb{R}^j$, $l = 1, 2$. In particular, during training the neural network never has to deal with past payment information category 6, corresponding to the category “no information”, and the embedding weights $\beta_1(6) = \beta_2(6) = 0$. Thus, the neural network does not learn how to correctly process the embedding weights $\beta_1(6) = \beta_2(6) = 0$. However, when estimating expected future payments, the past payment information contains category 6, with corresponding embedding weights $\beta_1(6) = \beta_2(6) = 0$.

Usually, this consistency problem is circumvented by using the individual claims multiple times, assuming a different amount of past payment information for every replication. This enables the neural network to also learn how to process the embedding weights $\beta_1(6) = \beta_2(6) = 0$. However, this technique leads to a massive increase of data, slowing down the neural network fitting, without any gain of additional information. Therefore, we opt for a different strategy here, exploiting the properties of dropout layers and using every observation only once.

Originally, dropout layers have been introduced in [32] as a regularization technique to prevent neural networks from overtraining individual neurons. If we use a dropout layer with a dropout rate of 10%, then in every gradient descent step, every neuron of the layer previous to the dropout layer is temporarily set to 0 with a probability of 10%, independently of the other neurons. This forces the neural network to not rely too much on individual neurons.

We apply dropout for embedding weights $(\beta_l(g(Y_{v,0})), \dots, \beta_l(g(Y_{v,j-1}))) \in \mathbb{R}^j$, $l = 1, 2$, of the past payment information $g(Y_{v,0}), \dots, g(Y_{v,j-1}) \in \{0, \dots, 5\}$ for payment $Y_{v,j}$. Hence, in every gradient descent step some of the embedding weights are set to 0. Due to the choice of the non-trainable embedding weights $\beta(6) = (\beta_1(6), \beta_2(6)) = (0, 0)$, for $k = 0, \dots, j-1$ an embedding value $\beta_l(g(Y_{v,k})) = 0$ exactly reflects that $g(Y_{v,k}) = 6$, i.e. that the payment $Y_{v,k}$ is unknown.

When randomly setting embedding weights to 0, we would like to mimic the situation encountered when estimating expected future payments. For

every subnet $j \in \{2, \dots, J\}$ and every individual claim $v = 1, \dots, n$ for which $Y_{v,j} \in \mathcal{D}_I$, in every gradient descent step instead of the embedding weights $(\beta_l(g(Y_{v,0})), \dots, \beta_l(g(Y_{v,j-1}))) \in \mathbb{R}^j$, $l = 1, 2$, we use

$$\left\{ \begin{array}{ll} (\beta_l(g(Y_{v,0})), 0, \dots, 0) \in \mathbb{R}^j, & \text{with probability } \frac{1}{j}, \\ (\beta_l(g(Y_{v,0})), \beta_l(g(Y_{v,1})), 0, \dots, 0) \in \mathbb{R}^j, & \text{with probability } \frac{1}{j}, \\ \vdots & \vdots \\ (\beta_l(g(Y_{v,0})), \dots, \beta_l(g(Y_{v,j-2})), 0) \in \mathbb{R}^j, & \text{with probability } \frac{1}{j}, \\ (\beta_l(g(Y_{v,0})), \dots, \beta_l(g(Y_{v,j-1}))) \in \mathbb{R}^j, & \text{with probability } \frac{1}{j}. \end{array} \right. \quad (5.6)$$

For $l = 1, 2$ exactly the same entries of the two embedding vectors are set to 0. The choices of the embedding vector in (5.6) are independent for different subnets $j \in \{2, \dots, J\}$ and individual claims $v = 1, \dots, n$. We do not need to apply dropout for subnets $j = 0$ and $j = 1$ because for subnet $j = 0$ no past payment information is used, and for subnet $j = 1$ the only past payment $Y_{v,0}$ is known for all claims.

By using structure (5.6) we mimic the situation we also encounter when estimating expected future payments. For an individual claim v and payment delay j with $I - r_v < j \leq J - x_{v,6}$, we rely on past payment information $(g(Y_{v,0}), \dots, g(Y_{v,I-r_v}), 6, \dots, 6) \in \{0, \dots, 6\}^j$ with embedding weights $(\beta_l(g(Y_{v,0})), \dots, \beta_l(g(Y_{v,I-r_v})), 0, \dots, 0) \in \mathbb{R}^j$, $l = 1, 2$. Moreover, considering subnet $j = 2, \dots, J$, then among the individual claims v for which we need to predict the expected payment in payment delay period j , the past payment $Y_{v,k}$, $k < j$, is known for roughly $\frac{j-k}{j} \cdot 100\%$ of the claims (assuming a roughly uniform distribution among reporting years). Going back to a payment $Y_{v,j} \in \mathcal{D}_I$ used for training and a $Y_{v,k}$ with $k < j$, summing up the probabilities given in (5.6) for the cases for which $\beta_l(g(Y_{v,k}))$ is not set to 0, we exactly get

$$\sum_{m=k}^{j-1} \frac{1}{j} = \frac{j-k}{j}.$$

We conclude that during neural network training, in subnet $j = 2, \dots, J$ the embedding value $\beta_l(g(Y_{v,k}))$ is used in roughly $\frac{j-k}{j} \cdot 100\%$ of the cases, exactly as during estimation procedure.

5.4 First training step: embedding weights

We divide neural network training into two steps. In a first step, we train all neural network weights, but we only store the embedding weights introduced in Sect. 4.1. The embedding weights are a crucial ingredient in a neural network, as they allow us to distinguish between categorical levels of the features. We interpret training of embedding weights as choosing reasonable representations of the feature variables. Having chosen the embedding weights, we start the actual

neural network training. Thus, in a second step, we keep the learned embedding weights fixed, and re-train the remaining neural network weights.

We still need to determine the numbers of epochs for both the first and the second training step. An epoch refers to considering all available data exactly once in the gradient descent iteration. As we use the stochastic gradient descent method *nadam* in Keras, one epoch corresponds to several gradient descent steps. In order to determine the numbers of epochs for the first training step, we randomly allocate 80% of the n individual claims to a training set and the remaining 20% to a validation set. We then train the neural network only using the training set. After every epoch, we calculate the value of the loss function (5.3) on the training set and the validation set. The results for the six LoBs are given in Fig. 4. The blue points indicate the values of the loss function on the training set (in-sample) and the red points the losses on the validation set (out-of-sample). Generally speaking, the neural network is learning (true) model structure of the data as long as the validation loss decreases. Once this validation loss flattens out or starts to increase, we have reached the phase of over-fitting, where we are not learning (true) model structure of the data anymore but rather the noisy part in the training set, which is undesirable. The chosen numbers of epochs for the six LoBs are given on line (i) of Table 3, see also the green lines in Fig. 4. Note that for simplicity we only consider numbers of epochs in $\{k \cdot 10 \mid k \in \mathbb{N}\}$.

After this training/validation analysis, we train the neural network using all n individual claims for the chosen number of epochs. The time required for this training on a personal MacBook Pro with CPU @ 2.20GHz (6 CPUs) with 16GB RAM is given on line (ii) of Table 3 for all six LoBs. The resulting values of the embedding weights are stored, and kept constant during the second training step.

Fig. 4 Training loss (in blue) and validation loss (in red) in the first training step for all six LoBs. The green lines indicate the chosen numbers of epochs for the first training step (color figure online)

Table 3 Chosen numbers of epochs in the first training step and corresponding time required for training

	LoB 1	LoB 2	LoB 3	LoB 4	LoB 5	LoB 6
(i) Number of epochs	30	30	40	90	50	30
(ii) Time (s)	205	215	112	570	345	92

5.5 Second training step: neural network weights

In the second training step we keep the embedding weights fixed, i.e. we only train the remaining neural network weights, which are initially set to the same values as before the first training step. Apart from this, we proceed as above. We use the same splitting of the n individual claims into a training set and a validation set. The resulting training and validation losses are given in Fig. 5, for all six LoBs. If we compare the validation losses of Figs. 4 and 5, we observe smaller values for the second training step. This indicates that determining the embedding weights beforehand leads to a better out-of-sample performance. The chosen numbers of epochs for the six LoBs in the second training step are given on line (i) of Table 4, see also the green lines in Fig. 5. Finally, we train the neural network using all n individual claims for the chosen number of epochs.

We remark that we do not use a single number of epochs in order to estimate the probability/regression functions $\{p_j(\cdot)\}_{j=0,\dots,J}$ and $\{\mu_j(\cdot)\}_{j=0,\dots,J}$. Let $\mathcal{E}$ denote the chosen numbers of epochs. In Figs. 4 and 5 we observe that training and validation losses are rather erratic. In order to stabilize the results, we train the neural network for $\mathcal{E} + 2$ epochs and determine the probability functions $\{p_j(\cdot)\}_{j=0,\dots,J}$ and the regression functions $\{\mu_j(\cdot)\}_{j=0,\dots,J}$ for all 5 numbers of epochs $\{\mathcal{E} - 2, \mathcal{E} - 1, \mathcal{E}, \mathcal{E} + 1, \mathcal{E} + 2\}$. The estimated probability functions $\{\hat{p}_j(\cdot)\}_{j=0,\dots,J}$ and the estimated regression functions $\{\hat{\mu}_j(\cdot)\}_{j=0,\dots,J}$ are then given by the corresponding averaged values. The time required for this training on a personal MacBook Pro with CPU @ 2.20GHz (6 CPUs) with 16GB RAM is given on line (ii) of Table 4 for all six LoBs.

5.6 Variance parameters

In order to estimate the variance parameters $\{\sigma_j^2\}_{j=0,\dots,J}$, we choose a pragmatic approach. We define

$$\hat{\sigma}_j^2 = \max \left\{ 2 \log \left(\frac{\sum_{v=1}^n Y_{v,j} \mathbb{1}_{\{r_v+j \leq I\}}}{\sum_{v=1}^n \exp\{\hat{\mu}_j(\mathbf{x}_v, g(Y_{v,0}), \dots, g(Y_{v,j-1}))\}} \mathbb{1}_{\{r_v+j \leq I\}} \right), 10^{-9} \right\}, \quad (5.7)$$

for all $j = 0, \dots, J$. The max term in (5.7) ensures that all variance parameters are positive, and for our six LoBs it is only needed once (for LoB 4 and $j = J$). With definition (5.7), and if $\hat{\sigma}_j^2 > 10^{-9}$, we have

$$\sum_{v=1}^n \exp\{\hat{\mu}_j(\mathbf{x}_v, g(Y_{v,0}), \dots, g(Y_{v,j-1})) + \hat{\sigma}_j^2/2\} \mathbb{1}_{\{r_v+j \leq I\}} = \sum_{v=1}^n Y_{v,j} \mathbb{1}_{\{r_v+j \leq I\}},$$

for all $j = 0, \dots, J$. This implies that for every payment delay period j with $\hat{\sigma}_j^2 > 10^{-9}$, the aggregated estimated expected past payments are equal to the aggregated observed past payments. In particular, this step can be interpreted as a bias regularization.

For illustrative purposes, the estimated variance parameters for LoB 1 are given in Table 5.

5.7 Recoveries

In Model Assumptions 1 we assume in (a5) that the probability of a recovery and the mean recovery payment are the same, respectively, for all claims and payment delays.

In order to estimate recovery probability p_- , we define

$$M_{r,j} = \sum_{\substack{v=1, \dots, n \\ r_v = r}} \mathbb{1}_{\{Y_{v,j} < 0\}},$$

for all reporting years $r = 1, \dots, I$ and payment delays $j = 0, \dots, I - r$. In particular, $M_{r,j}$ is the number of individual claims with reporting year r having a recovery

Fig. 5 Training losses (in blue) and validation losses (in red) in the second training step for all six LoBs. The green lines indicate the chosen numbers of epochs for the second training step (color figure online)

Table 4 Chosen numbers of epochs in the second training step and corresponding time required for training

	LoB 1	LoB 2	LoB 3	LoB 4	LoB 5	LoB 6
(i) Number of epochs	50	40	50	120	60	100
(ii) Time (s)	317	250	139	705	370	253

payment in payment delay period j . Moreover, we define the vector $N = (N_1, \dots, N_I)$ of the numbers of reported claims per reporting year $r = 1, \dots, I$. We estimate recovery probability p_- by

$$\hat{p}_- = \frac{\sum_{r=1}^I \sum_{j=0}^{I-r} j \cdot M_{r,j}}{\sum_{r=1}^I \sum_{j=0}^{I-r} j \cdot N_r}. \quad (5.8)$$

Note that in (5.8) we use a weighting with payment delay j . The intuition behind this weighting is that when predicting future recoveries, we need to consider payment delay period j for exactly j reporting years, i.e. recovery predictions are more often needed for higher payment delays. We try to take that into account when estimating recovery probability p_- by using the weighting with the payment delay.

In order to estimate mean recovery payment μ_- , we define

$$S_{r,j} = \sum_{\substack{v=1, \dots, n \\ r_v = r}} Y_{v,j} \mathbb{1}_{\{Y_{v,j} < 0\}},$$

for all reporting years $r = 1, \dots, I$ and payment delays $j = 0, \dots, I - r$. In particular, $S_{r,j}$ is the total recovery payment for individual claims with reporting year r in payment delay period j . We estimate mean recovery payment μ_- by

$$\hat{\mu}_- = \frac{\sum_{r=1}^I \sum_{j=0}^{I-r} j \cdot S_{r,j}}{\sum_{r=1}^I \sum_{j=0}^{I-r} j \cdot M_{r,j}}.$$

Finally, we estimate the expected future recovery payments by

$$\sum_{v=1}^n \sum_{j=I-r_v+1}^{J-x_{v,6}} \hat{p}_- \hat{\mu}_-,$$

see line (i) of Table 6 for the results. If we compare these values to the true future recovery payments, see line (ii) of Table 6, we observe that, on average, our crude approach works sufficiently well here. Moreover, if we compare the numbers in Table 6 to the estimated RBNS reserves and the true outstanding payments in Table 8, below, we see that the recovery payments are almost negligible for our data set.

Table 5 Estimated variance parameters $\{\hat{\sigma}_j^2\}_{j=0,\dots,J}$ in LoB 1

j	0	1	2	3	4	5	6	7	8	9	10	11
$\hat{\sigma}_j^2$	1.15	0.81	0.74	0.55	0.53	0.37	0.28	0.16	0.14	0.25	0.22	0.32

5.8 Results

As an illustration, we first show the results for the example claim of Table 2 with feature information $\mathbf{x} = (45, 9, 4, 60, 52, 0) \in \mathbb{R}^d$ and past payment information $\mathbf{y} = (2, 2, 2, 0, 6, 6, 6, 6, 6, 6, 6, 6) \in \mathbb{R}^J$. For the prediction procedure, the output of the neural network for this example claim are the estimated probability/regression functions

$$\begin{array}{ll}
 \bullet \hat{p}_4(\mathbf{x}, g(2), g(2), g(2), g(0)) & \bullet \hat{\mu}_4(\mathbf{x}, g(2), g(2), g(2), g(0)) \\
 \bullet \hat{p}_5(\mathbf{x}, g(2), g(2), g(2), g(0), g(6)) & \bullet \hat{\mu}_5(\mathbf{x}, g(2), g(2), g(2), g(0), g(6)) \\
 \vdots & \vdots \\
 \bullet \hat{p}_J(\mathbf{x}, g(2), g(2), g(2), g(0), g(6), \dots, g(6)) & \bullet \hat{\mu}_J(\mathbf{x}, g(2), g(2), g(2), g(0), g(6), \dots, g(6))
 \end{array}$$

The corresponding values are given in the first two lines of Table 7. Together with the estimated variance parameters $\{\hat{\sigma}_j\}_{j=4,\dots,J}$, the estimated recovery probability $\hat{p}$ and the estimated recovery mean $\hat{\mu}$, we can calculate the expected future payments for this example claim, see line (iii) of Table 7. We emphasize that these are results for just one single claim, i.e. a comparison between the estimated expected future payments and the true future payments does not make sense here.

Calculating the expected future payments for all claims, the individual RBNS claims reserves $R_{\text{RBNS}}^{\text{IC}}$ (IC = individual claims) defined by

$$\begin{aligned}
 R_{\text{RBNS}}^{\text{IC}} &= \hat{\mathbb{E}}[R_{\text{RBNS}}^{\text{true}} \mid \mathcal{D}_I] \\
 &= \sum_{v=1}^n \sum_{j=I-r_v+1}^{J-x_{v,6}} (\hat{p}_j(\mathbf{x}_v, g(Y_{v,0}), \dots, g(Y_{v,I-r_v}), 6, \dots, 6) \\
 &\quad \cdot \exp \left\{ \hat{\mu}_j(\mathbf{x}_v, g(Y_{v,0}), \dots, g(Y_{v,I-r_v}), 6, \dots, 6) + \hat{\sigma}_j^2/2 \right\} \\
 &\quad + \hat{p}_- \hat{\mu}_-),
 \end{aligned}$$

see also (3.2), are given on line (i) of Table 8, for all six LoBs. We see that we get quite close to the true outstanding payments for reported claims given on line (ii). On lines (iii) and (iv) we provide the absolute and relative biases of the individual RBNS claims reserves $R_{\text{RBNS}}^{\text{IC}}$. We observe that we always stay within 2% of the true outstanding payments. For additional analyses of the individual RBNS claims reserves we refer to Sect. 6.

5.9 Results of an application on real accident insurance data

In the previous sections we applied the individual RBNS claims reserving model on synthetic data generated from the individual claims history simulation machine [16]. In this section we perform a small back-test on real insurance data. We choose the real Swiss accident insurance data which was used to develop the individual claims history simulation machine [16]. This data set originally consists of four LoBs, and the individual claims are exactly of the same style as the ones simulated from the individual claims history simulation machine [16]. For confidentiality reasons, we only present (scaled) results. For all four LoBs, we randomly select two disjoint subsamples of individual reported claims, see Table 9 for the numbers of considered claims.

The (scaled) results of the application of the individual RBNS claims reserving model are shown in Table 10. We observe that, in general, the model also performs well on this real insurance data set. We remark that for LoBs 2, 3 and 4 we have biases of different signs for the two subsamples, even though they are similar in structure. This shows that the individual RBNS claims reserving model does not suffer from a systematic bias. Nevertheless, we are interested in detecting what causes these biases with different signs. Here, we focus on LoB 2.

Table 6 Estimated expected future recovery payments and true future recovery payments (in thousands)

	LoB 1	LoB 2	LoB 3	LoB 4	LoB 5	LoB 6
(i) Estimated expected future recovery payments	– 89	– 106	– 47	– 201	– 151	– 93
(ii) True future recovery payments	– 121	– 92	– 38	– 221	– 213	– 93

Table 7 Values of the estimated probability and regression functions as well as expected payments in the future payment delay periods $j = 4, \dots, J$ for the example claim of Table 2

Payment delay period j	4	5	6	7	8	9	10	11	Total
(i) $\hat{p}_j(\cdot)$	0.035	0.036	0.029	0.017	0.004	0.003	0.004	0.004	–
(ii) $\hat{\mu}_j(\cdot)$	6.965	6.787	6.964	7.178	6.532	7.353	7.595	7.258	–
(iii) Expected payment	49	38	35	24	3	5	10	7	171

Table 8 Individual RBNS claims reserves $R_{\text{RBNS}}^{\text{IC}}$, true outstanding payments $R_{\text{RBNS}}^{\text{true}}$ and absolute and relative biases (with absolute values in thousands)

	LoB 1	LoB 2	LoB 3	LoB 4	LoB 5	LoB 6
(i) IC RBNS reserves $R_{\text{RBNS}}^{\text{IC}}$	38,416	36,155	16,548	67,225	68,967	30,661
(ii) True outstanding payments $R_{\text{RBNS}}^{\text{true}}$	38,093	35,500	16,275	68,037	69,810	30,069
(iii) Bias $R_{\text{RBNS}}^{\text{IC}} - R_{\text{RBNS}}^{\text{true}}$	323	655	273	– 812	– 843	592
(iv) Bias in %	0.8%	1.8%	1.7%	– 1.2%	– 1.2%	2.0%

On lines (i)–(iv) of Table 11 we repeat the individual RBNS reserves $R_{\text{RBNS}}^{\text{IC}}$, the true outstanding payments $R_{\text{RBNS}}^{\text{true}}$ and the absolute and relative biases for the two subsamples of LoB 2 from Table 10. On lines (v), (vii), (ix) and (xi) of Table 11 we show the (scaled) absolute biases coming from all payments done in payment delay periods $j = 1, j = 2, j = 3$ and $j = 4, \dots, 11$, respectively. On lines (vi), (viii), (x) and (xii) of Table 11 we provide the biases (in %) relative to the complete true outstanding payments $R_{\text{RBNS}}^{\text{true}}$ which are caused by payment delay periods $j = 1, j = 2, j = 3$ and $j = 4, \dots, 11$, respectively. We observe that the major part of the overall bias of 4.5% in subsample 1 is due to payment delay period $j = 1$, which is responsible for a bias of 2.3%. A similar observation (with opposite sign) holds true for subsample 2. There, the overall bias of -0.9% is mainly due to payment delay period $j = 1$, which causes a bias of -1.6% .

In Table 12 we analyze the origins of the biases in payment delay period $j = 1$ for the two subsamples. On lines (i) and (ii) we provide the (scaled) estimated and true average probability of a payment in payment delay period $j = 1$, respectively. Note that for these probabilities we only consider projections and future payments after time point I , respectively. That is, for the estimated average payment probability we take the average of the payment probabilities estimated by our neural network model for the claims reported in year I . For the true average payment probability we take the proportion of claims reported in year I for which we observe a payment in payment delay period $j = 1$. We see that for subsample 1 we overestimate this probability. This is one reason for the positive bias observed in subsample 1. Additionally, we also slightly overestimate the payment size, see lines (iii) and (iv) of Table 12, where we provide the (scaled) estimated and true average logarithmic payment size in payment delay period $j = 1$, respectively. Analogously to the probabilities on lines (i) and (ii) we only consider projections and future payments after time point I , respectively. To allow for a fair comparison, for the estimated average logarithmic payment size on line (iii) we only consider those individual claims for which we actually have a payment in this payment delay period. This explains the positive bias in subsample 1. In subsample 2 we underestimate the payment probability in payment delay period $j = 1$. This effect gets reduced by a slight overestimation of the average logarithmic payment size. Similar observations hold true for the other LoBs.

6 Analysis of the individual RBNS claims reserves

The results of the individual RBNS claims reserves $R_{\text{RBNS}}^{\text{IC}}$ presented in Table 8 (and Table 10) seem convincing in the sense that they are very close to the true outstanding payments for reported claims $R_{\text{RBNS}}^{\text{true}}$. In this section we take a closer look at the individual RBNS claims reserves. First, we check the stability of the single neural network. In a second step, we investigate the sensitivity of the individual RBNS claims reserves with respect to the variance parameters $\{\sigma_j^2\}_{j=0,\dots,J}$. Finally, we analyze whether the neural network is indeed able to capture the claim-specific feature and past payment information.

Table 9 Total numbers of individual reported claims in the two subsamples for all four LoBs

	LoB 1	LoB 2	LoB 3	LoB 4
(i) Subsample 1	218,112	218,004	277,622	278,323
(ii) Subsample 2	218,414	217,917	278,291	277,689

Table 10 Individual RBNS claims reserves R_{RBNS}^{IC} , true outstanding payments R_{RBNS}^{true} and absolute and relative biases (with scaled absolute values in thousands) for the two subsamples of all four LoBs of a real accident insurance data set

		LoB 1	LoB 2	LoB 3	LoB 4
(i) Subsample 1	IC RBNS reserves R_{RBNS}^{IC}	15,102	19,444	26,996	22,831
(ii) Subsample 1	True outstanding payments R_{RBNS}^{true}	15,147	18,605	25,542	22,608
(iii) Subsample 1	Bias $R_{RBNS}^{IC} - R_{RBNS}^{true}$	- 45	839	1454	223
(iv) Subsample 1	Bias in %	- 0.3%	4.5%	5.7%	1.0%
(v) Subsample 2	IC RBNS reserves R_{RBNS}^{IC}	13,350	17,852	27,464	22,481
(vi) Subsample 2	True outstanding payments R_{RBNS}^{true}	13,529	18,023	27,938	23,156
(vii) Subsample 2	Bias $R_{RBNS}^{IC} - R_{RBNS}^{true}$	- 179	- 171	- 474	- 675
(viii) Subsample 2	Bias in %	- 1.3%	- 0.9%	- 1.7%	- 2.9%

Table 11 Individual RBNS claims reserves R_{RBNS}^{IC} , true outstanding payments R_{RBNS}^{true} , overall absolute and relative biases as well as absolute and relative biases corresponding to payment delay periods $j = 1$, $j = 2$, $j = 3$ and $j = 4, \dots, 11$ (with scaled absolute values in thousands) for the two subsamples of LoB 2 of the real accident insurance data set

LoB 2	Subsample 1	Subsample 2
(i) IC RBNS reserves R_{RBNS}^{IC}	19,444	17,852
(ii) True outstanding payments R_{RBNS}^{true}	18,605	18,023
(iii) Bias $R_{RBNS}^{IC} - R_{RBNS}^{true}$	839	- 171
(iv) Bias in %	4.5%	- 0.9%
(v) Absolute bias due to period $j = 1$	531	- 282
(vi) Relative bias due to period $j = 1$	2.9%	- 1.6%
(vii) Absolute bias due to period $j = 2$	- 24	- 83
(viii) Relative bias due to period $j = 2$	- 0.1%	- 0.5%
(ix) Absolute bias due to period $j = 3$	76	- 71
(x) Relative bias due to period $j = 3$	0.4%	- 0.4%
(xi) Absolute bias due to periods $j = 4, \dots, 11$	257	265
(xii) Relative bias due to periods $j = 4, \dots, 11$	1.4%	1.5%

6.1 Stability of the neural network

We challenge the stability of the single neural network in two ways. On the one hand, we calculate the individual RBNS claims reserves using different numbers of

epochs in the second training step. On the other hand, we calculate the individual RBNS claims reserves using different seeds in the second training step.

The individual RBNS claims reserves $R_{\text{RBNS}}^{\text{IC}}$ presented on line (i) of Table 8 are calculated based on $\mathcal{E}$ training epochs in the second training step, where $\mathcal{E}$ has been chosen according to the training/validation analysis in Fig. 5, and is given on line (i) of Table 4. The validation losses (in red) in Fig. 5 are rather erratic, and slightly smaller or bigger numbers of epochs can also be justified. In order to analyze the stability of the neural network, we calculate the individual RBNS claims reserves $R_{\text{RBNS}}^{\text{IC}}$ for the 41 epochs $\mathcal{E} - 20, \dots, \mathcal{E}, \dots, \mathcal{E} + 20$. More precisely, for every $\mathcal{E}^* \in \{\mathcal{E} - 20, \dots, \mathcal{E}, \dots, \mathcal{E} + 20\}$, probability functions $\{p_j(\cdot)\}_{j=0, \dots, J}$ and regression functions $\{\mu_j(\cdot)\}_{j=0, \dots, J}$ are estimated by averaging over the results of the 5 epochs $\{\mathcal{E}^* - 2, \mathcal{E}^* - 1, \mathcal{E}^*, \mathcal{E}^* + 1, \mathcal{E}^* + 2\}$. The relative biases of the resulting 41 individual RBNS claims reserves with respect to the true outstanding payments $R_{\text{RBNS}}^{\text{true}}$ are presented with blue dots in Fig. 6, for all six LoBs. The dashed green lines indicate zero bias, the dashed red lines a bias of $\pm 5\%$. The numbers of epochs chosen beforehand in Sect. 5.5 are indicated by the dashed vertical lines. We observe that we stay rather close to the true outstanding payments $R_{\text{RBNS}}^{\text{true}}$. Therefore, even though the choice of the number of epochs remains crucial, there is some leeway. We remark that one can increase stability of the individual RBNS claims reserves by averaging over a higher number of epochs.

As a second stability check we repeat the second training step for 100 different seeds, always using the number of epochs $\mathcal{E}$ given on line (i) of Table 4 and averaging probability functions $\{p_j(\cdot)\}_{j=0, \dots, J}$ and regression functions $\{\mu_j(\cdot)\}_{j=0, \dots, J}$ over $\{\mathcal{E} - 2, \mathcal{E} - 1, \mathcal{E}, \mathcal{E} + 1, \mathcal{E} + 2\}$. The boxplots of the relative biases of the resulting 100 individual RBNS claims reserves with respect to the true outstanding payments are given in Fig. 7, for all six LoBs. The dashed green lines indicate zero bias, the dashed red lines a bias of $\pm 5\%$. We observe that even though we did not choose seed-dependent numbers of epochs, we predominantly stay within 5% of the true outstanding payments. Moreover, considering the medians of the biases, we can conclude that the individual RBNS claims reserving model does not suffer from a systematic bias.

Table 12 Estimated and true (scaled) average probability of a payment in payment delay period $j = 1$ as well as estimated and true average logarithmic (scaled) payment size in payment delay period $j = 1$ for the two subsamples of LoB 2 of the real accident insurance data set

LoB 2	Subsample 1	Subsample 2
(i) Estimated average probability of a payment in period $j = 1$	3.86%	3.29%
(ii) True average probability of a payment in period $j = 1$	3.63%	3.67%
(iii) Estimated average logarithmic payment size in period $j = 1$	4.08	4.04
(iv) True average logarithmic payment size in period $j = 1$	4.04	4.02

6.2 Sensitivity with respect to the variance parameters

In Sect. 5.6 we estimated the variance parameters $\{\sigma_j^2\}_{j=0,\dots,J}$ such that the aggregated estimated expected past payments are equal to the aggregated observed past payments, for every payment delay period $j = 0, \dots, J$. Here, we are interested in the effect of small changes in the variance parameters on the resulting individual RBNS claims reserves. As the payments in payment delay period $j = 0$ are all known, the individual RBNS claims reserves do not depend on the variance parameter σ_0^2 . In Fig. 8 we plot the bias of the individual RBNS claims reserves of LoB 1 against the relative changes of the variance parameters, for payment delay periods $j = 1, \dots, 9$. The dashed green lines indicate zero bias. The estimated variance parameters $\{\hat{\sigma}_j^2\}_{j=0,\dots,J}$ are indicated by the dashed vertical lines. Not surprisingly, we see that the first few variance parameters $\sigma_1^2, \dots, \sigma_4^2$ have quite some influence on the resulting individual RBNS claims reserves, whereas the later variance parameters $\sigma_5^2, \dots, \sigma_9^2$ are only of minor importance. Also for variance parameters σ_{10}^2 and σ_{11}^2 we would observe almost flat lines.

6.3 Individual feature and past payment information

The neural network individual RBNS claims reserving model works very well in terms of small biases with respect to the true outstanding payments $R_{\text{RBNS}}^{\text{true}}$. We may wonder if this is really due to the ability of the neural network to correctly process the provided claim-specific feature and past payment information. To answer this question, we analyze the performance of the individual RBNS claims reserving model on smaller subsets of the n individual claims.

For every feature $l = 1, \dots, d$ and every level $u_{l,k} \in \mathcal{X}_l = \{u_{l,1}, \dots, u_{l,|\mathcal{X}_l|}\}$ of this feature l , we can estimate the average reserves for individual claims with level $u_{l,k}$ of feature l . These estimated average reserves can be compared to the true average outstanding payments. For LoB 1 the results for all $d = 6$ features claims code `cc`, accident year `AY`, accident quarter `AQ`, age `age`, injured body part `inj_part` and reporting delay `RepDel` are presented in logarithmic scale in the first two rows of Fig. 9. The (categorical) levels of the $d = 6$ features are represented by the blue numbers of their corresponding classes. The green diagonals reflect the identity between the estimated average reserves from the individual RBNS claims reserving model and the true average outstanding payments. We see that in general the observations are close to the diagonals. This implies that the neural network is correctly processing the claim-specific feature information. Moreover, by considering the ranges of the y-axes, we see that the feature `inj_part` is the most important among the $d = 6$ features (excluding the accident year `AY`) in terms of estimating expected future payments.

Similarly, for all payment delay periods $j = 0, \dots, J - 1$ and past payment information $u \in \{0, \dots, 6\}$ we can estimate the average reserves for individual claims v with past payment information $g(Y_{v,j}) = u$, and compare them to the true average outstanding payments. For LoB 1 the results for payment delay periods $j = 0, 2, 9$ are presented in logarithmic scale in the third row of Fig. 9. We observe that generally

Fig. 6 Relative biases (blue dots) of the individual RBNS claims reserves for epochs in the set $\{\mathcal{E} - 20, \dots, \mathcal{E}, \dots, \mathcal{E} + 20\}$, for all six LoBs. The dashed green lines indicate zero bias, the dashed red lines a bias of $\pm 5\%$. The numbers of epochs $\mathcal{E}$ chosen beforehand in Sect. 5.5 are indicated by the dashed vertical lines (color figure online)

Fig. 7 Boxplots of the relative biases of the individual RBNS claims reserves for 100 different seeds in the second training step, for all six LoBs. The dashed green lines indicate zero bias, the dashed red lines a bias of $\pm 5\%$ (color figure online)

the neural network is also able to correctly process the claim-specific past payment information. In particular, on average the neural network knows how to deal with past payment information class 6 corresponding to “no information”. We remark that for payment delay period $j = 0$ the classes 1 (“recovery payment”) and 6 are not available as there are no recovery payments in this payment delay period and as the payments in this payment delay period are known for all individual claims. The ranges of the y-axes indicate that the past payment information is the main driver of the size of expected future payments. This closes the analysis of the individual RBNS claims reserves.

7 Conclusion

We presented a claims reserving technique on the basis of claim-specific feature and past payment information in order to estimate RBNS claims reserves. For every individual reported claim, the expected future cash flow is estimated using a single neural network. This neural network consists of several subnets, where every subnet models expected payments for the individual claims for a particular payment delay period. Training this single neural network allows us to transfer model structure from one payment delay period to the other. As individual claims are not equipped with the same amount of past payment information, we combined embedding layers and dropout layers in order to consistently fit the neural network to the incomplete time series of past payments for individual reported claims.

We applied this individual RBNS claims reserving model to a synthetic insurance data set with complete information consisting of six lines of business generated from the individual claims history simulation machine [16]. We observed that the individual RBNS claims reserves stayed within 2% of the true outstanding payments for all lines of business. Challenging the stability of the individual RBNS claims reserving model by choosing different numbers of epochs and different Keras seeds during neural network fitting, we observed that we predominantly stayed within 5% of the true outstanding payments. Another analysis showed that the neural network on average correctly processes the claim-specific feature and past payment information.

Fig. 8 Sensitivity of the individual RBNS claims reserves of LoB 1 with respect to the variance parameters $\{\sigma_j^2\}_{j=1,\dots,9}$. The dashed green lines indicate zero bias. The estimated variance parameters $\{\hat{\sigma}_j^2\}_{j=1,\dots,9}$ are indicated by the dashed vertical lines (color figure online)

In particular, the neural network trained with the dropout technique applied on past payment information knows how to deal with the situation of not knowing the full past payment information. Finally, we also observed that, in general, the individual RBNS claims reserving model also performed well on a real insurance data set.

In this paper we did not touch upon uncertainty estimates. Uncertainty of claims reserves is usually measured in terms of (conditional) root-mean square error of prediction, which consists of process variance and parameter estimation error. Within our model the process variance could be estimated using a slight extension of Model Assumptions 1. The parameter estimation error is more tricky. One approach would be to bootstrap the observed payments. However, our log-normal payment size model is only appropriate for modeling expected payments. It does not allow reasonable simulations of payments, as it would not generate sufficiently extreme

Fig. 9 Comparison of the estimated average reserves for all levels of the $d = 6$ features claims code `cc`, accident year `AY`, accident quarter `AQ`, age `age`, injured part `inj_part` and reporting delay `RepDel` and of the past payment information in payment delay periods $j = 0, 2, 9$ to the true average outstanding payments in LoB 1 (blue numbers). The green diagonals indicate the identity between the estimated average reserves and the true average outstanding payments (color figure online)

observations. Thus, one would need to choose a more sophisticated payment size model.

Finally, a complete picture of the claims reserves would include IBNR reserves. We believe that IBNR reserves are ideally estimated on the basis of policy data. As the individual claims history simulation machine [16] does not provide policy data, we focused only on RBNS reserves.

Acknowledgements The author would like to kindly thank Prof. Dr. Mario Wüthrich for providing his insights, for his support and for reading previous versions of this manuscript.

Funding Open Access funding provided by ETH Zurich.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Antonio K, Plat R (2014) Micro-level stochastic loss reserving for general insurance. *Scand Actuar J* 2014(7):649–669
2. Baudry M, Robert CY (2019) A machine learning approach for individual claims reserving in insurance. *Appl Stoch Models Bus Ind* 35(5):1127–1155
3. Bengio Y, Ducharme R, Vincent P, Jauvin C (2003) A neural probabilistic language model. *J Mach Learn Res* 3:1137–1155
4. Boumezoued A, Devineau L (2017) Individual claims reserving: a survey. 2017. hal-01643929
5. Caruana R (1997) Multitask learning. *Mach Learn* 28(1):41–75
6. Crevecœur J, Antonio K (2019) A generalized reserving model: bridging the gap between pricing and individual reserving. [arXiv:1910.12692](https://arxiv.org/abs/1910.12692), version of October 28, 2019
7. Cybenko G (1989) Approximation by superpositions of a sigmoidal function. *Math Control Signal Syst* 2:303–314
8. De Felice M, Moriconi F (2019) Claim watching and individual claims reserving using classification and regression trees. *Risks* 7(4):102
9. Delong L, Lindholm M, Wüthrich MV (2020) Collective reserving using individual claims data. *SSRN Manuscript* ID 3582398, version of April 22, 2020
10. Delong L, Wüthrich MV (2020) Neural networks for the joint development of individual payments and claim incurred. *Risks* 8(2):33
11. Denuit M, Hainaut D, Trufin J (2019) Effective statistical learning methods for actuaries III: neural networks and extensions. Springer
12. Duval F, Pigeon M (2019) Individual loss reserving using a gradient boosting-based approach. *Risks* 7(3):79
13. Ferrario A, Noll A, Wüthrich MV (2018) Insights from inside neural networks. *SSRN Manuscript* ID 3226852, Version of November 14, 2018
14. Gabrielli A (2020) A neural network boosted double overdispersed Poisson claims reserving model. *ASTIN Bull* 50(1):25–60
15. Gabrielli A, Richman R, Wüthrich MV (2020) Neural network embedding of the over-dispersed Poisson reserving model. *Scand Actuar J* 1:1–29

16. Gabrielli A, Wüthrich MV (2018) An individual claims history simulation machine. *Risks* 6(2):29
17. Goodfellow I, Bengio Y, Courville A (2016) *Deep learning*. MIT Press, Cambridge
18. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp 770–778
19. Hiabu M, Margraf C, Martínez-Miranda MD, Nielsen JP (2016) The link between classical reserving and granular reserving through double chain-ladder and its extensions. *Br Actuar J* 21(1):97–116
20. Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. *Neural Netw* 2:359–366
21. Jessen AH, Mikosch T, Samorodnitsky G (2011) Prediction of outstanding payments in a Poisson cluster model. *Scand Actuar J* 2011(3):214–237
22. Kuo K (2020) Individual claims forecasting with Bayesian mixture density networks. [arXiv:2003.02453](https://arxiv.org/abs/2003.02453), version of March 5, 2020
23. Larsen CR (2007) An individual claims reserving model. *ASTIN Bull* 37(1):113–132
24. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. *Nature* 521(7553):436–444
25. Lopez O, Milhaud X, Thérond P-E (2019) A tree-based algorithm adapted to microlevel reserving and long development claims. *ASTIN Bull* 49(3):741–762
26. Mack T (1993) Distribution-free calculation of the standard error of chain ladder reserve estimates. *ASTIN Bull* 23(2):213–225
27. Martínez-Miranda MD, Nielsen JP, Verrall RJ, Wüthrich MV (2015) The link between classical reserving and granular reserving through double chain-ladder and its extensions. *Scand Actuar J* 2015(5):383–405
28. Pigeon M, Antonio K, Denuit M (2013) Individual loss reserving with the multivariate skew normal framework. *ASTIN Bull* 43(3):399–428
29. R Core Team (2018) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna
30. Ruder S (2017) An overview of multi-task learning in deep neural networks. [arXiv:1706.05098](https://arxiv.org/abs/1706.05098), version of June 15, 2017
31. Schmidhuber J (2015) Deep learning in neural networks: an overview. *Neural Netw* 61:85–117
32. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res* 15:1929–1958
33. Taylor G (2019) Loss reserving models: granular and machine learning forms. *Risks* 7(3):82
34. Taylor G, McGuire G, Sullivan J (2008) Individual claim loss reserving conditioned by case estimates. *Ann Actuar Sci* 3(1–2):215–256
35. Wüthrich MV (2018) Machine learning in individual claims reserving. *Scand Actuar J* 2018(6):465–480
36. Zhao XB, Zhou X, Wang JL (2009) Semiparametric model for prediction of individual claim loss reserving. *Insur Math Econ* 45:1–8

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.