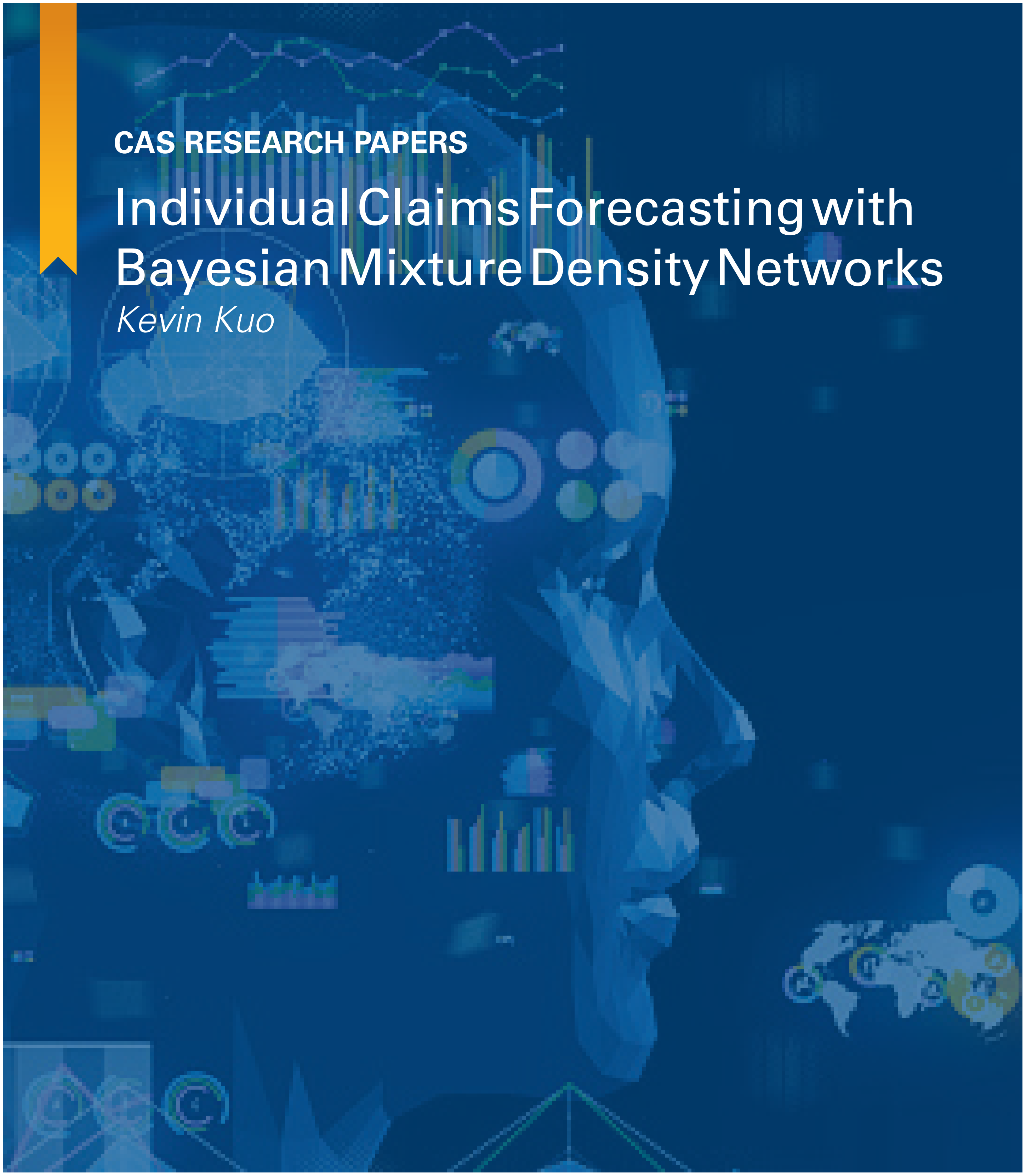

CAS RESEARCH PAPERS

Individual Claims Forecasting with Bayesian Mixture Density Networks

Kevin Kuo

CASUALTY ACTUARIAL SOCIETY

Individual Claims Forecasting with Bayesian Mixture Density Networks

Kevin Kuo

February 2020

Abstract

We introduce an individual claims forecasting framework utilizing Bayesian mixture density networks that can be used for claims analytics tasks such as case reserving and triaging. The proposed approach enables incorporating claims information from both structured and unstructured data sources, producing multi-period cash flow forecasts, and generating different scenarios of future payment patterns. We implement and evaluate the modeling framework using publicly available data.

Keywords

Claims reserving, individual claims reserving, loss reserving

Introduction

Individual claims reserving has garnered increasing interest in recent years. While the main benefit cited for performing reserving at the claim level over aggregate loss triangle approaches, such as chain ladder and Bornhuetter-Ferguson, is potential improvement in predictive accuracy, especially in environments with changing portfolio mix (Boumezoued and Devineau 2017), there are additional practical advantages to forecasting individual claim behavior. These include being able to obtain updated views of portfolio risk as claims are reported and optimize adjuster resource allocation based on severity predictions. Although the benefits of individual claims modeling are promising, it has not yet achieved widespread adoption in practice. One contributing reason for the lack of adoption, we hypothesize, is the absence of a modeling framework with features important to practitioners. We suggest that an effective loss reserves modeling framework should be able to:

- Incorporate arbitrary claims information as predictors,
- Produce multi-period forecasts that are sufficiently stable over time, and
- Sample different realizations of future payment patterns that encompass both process uncertainty and model risk.

Companies are capturing increasingly diverse data, such as unstructured text from claims adjusters' notes and photographs of damages, that can potentially be predictive. Timing of cash flows and being able to sample from different future states have both business decision making and regulatory applications. A desirable characteristic of the forecasts is that they are stable over time, as management is averse to volatility in loss reserves figures from one accounting period to the next. A subjective criterion not listed above, which may affect adoption of an approach, is that the models should be implementable without the need for extensive bespoke feature engineering or specification of complex assumptions for underlying stochastic processes.

To the best of our knowledge, no existing loss reserving framework implements all of the above features. In this paper, we propose an extensible individual claims forecasting framework towards satisfying many of these criteria, utilizing ideas from Bayesian neural networks (BNN) (Neal 2012) and mixture density networks (MDN) (Bishop 1994). While we discuss these concepts in detail later in the paper, at a high level,

- BNNs are non-linear supervised learning models that capture complex interactions among inputs, with prior distributions on model parameters; and
- MDNs are mixture models for conditional densities, where the mixture model parameters are the outputs of the neural network.

Concretely, our contributions are:

- Development of an individual claims forecasting framework based on Bayesian Mixture Density Networks (BMDN).
- Implementation of the proposed framework using publicly available claims-level data, which provides a baseline for future work to compare against.

Related Work

Claims-level reserving is a fast-moving research area, and (Boumezoued and Devineau 2017) and (Taylor 2019) provide recent surveys. Many of the current works in the area utilize machine learning (ML) techniques. Wüthrich (Wüthrich 2018a) introduces using machine learning algorithms to incorporate diverse claims characteristics inputs. It demonstrates a simple model where regression trees are used to predict the number of payments, and it suggests extensions such as compound modeling to predict severity and bootstrap simulation to obtain prediction uncertainty. More recently, (Duval and Pigeon 2019; Lopez, Milhaud, and Thérond 2019; Baudry and Robert, n.d.) also utilize tree-based techniques for individual claims reserving. Another direction of research for the individual claims reserving problem are the generative approaches of, for example, (Antonio and Plat 2014; Pigeon, Antonio, and Denuit 2013, 2014). In this latter category of methodologies, a set of distributional assumptions are posited for the different drivers of claims, such as time to the next payment and its amount. These distributions are fit to the data, then, with the obtained parameters, the modeler is able to perform simulations of future development paths by sampling from the fitted distributions. While this approach provides a natural way to obtain samples of future cash flow paths, the distributional assumptions may be too rigid in some cases. It is also difficult to incorporate individual claim characteristics; to differentiate among different characteristics, one would have to segment the claims and fit separate models to each group.

In formulating our framework, we also draw inspiration from machine learning approaches to aggregate triangle data, including (Gabrielli, Richman, and Wüthrich 2019; Gabrielli 2019), which embed a classical parametric loss reserving model into neural networks, and the DeepTriangle (Kuo 2019) framework, whose neural network architecture we adapt for individual claims data.

Preliminaries

We begin with a description of the loss reserving problem then briefly discuss BNN and MDN, two ideas that we incorporate into our claims forecasting neural network. As neural networks have been utilized and discussed extensively in recent loss reserving literature (Gabrielli, Richman, and Wüthrich 2019; Gabrielli 2019; Kuo 2019; Wüthrich 2018b; Gabrielli and V Wüthrich 2018), we defer discussion of neural network fundamentals to those works and the standard reference (Goodfellow, Bengio, and Courville 2016). To aid the practicing actuary in consuming this paper, we expand on certain concepts when we introduce the proposed neural network architecture in 4.4.

We remark that in sections 3.2 and 3.3, we discuss Bayesian inference and mixture density networks, respectively, in general, and provide details on our specific choices of distributions later on in the paper.

The Loss Reserving Problem

Figure 1 shows a diagram of the development of a typical claim. We first point out that there can be a time difference, known as the reporting lag, between an accident's occurrence and its reporting. The accidents which have been reported to the insurer, but not yet settled, are known as reported but not settled (RBNS) or, equivalently, incurred but not enough reported (IBNER) claims, while the accidents which have occurred but are yet unknown to the insurer are known as incurred but not reported (IBNR) claims. The reserving actuary is interested in estimating the ultimate loss amounts associated with accidents that have already occurred. As demonstrated in Figure 1, it is possible for a closed claim to re-open, and it is also possible for a claim to be closed without any cash flows.

In this paper, we are concerned with RBNS/IBNER, but not IBNR, claims, due to limitations of available data, as for IBNR claims we do not have individual claim feature information available. Also, the claims we study encompass closed claims to allow for the possibility of claim re-opening.

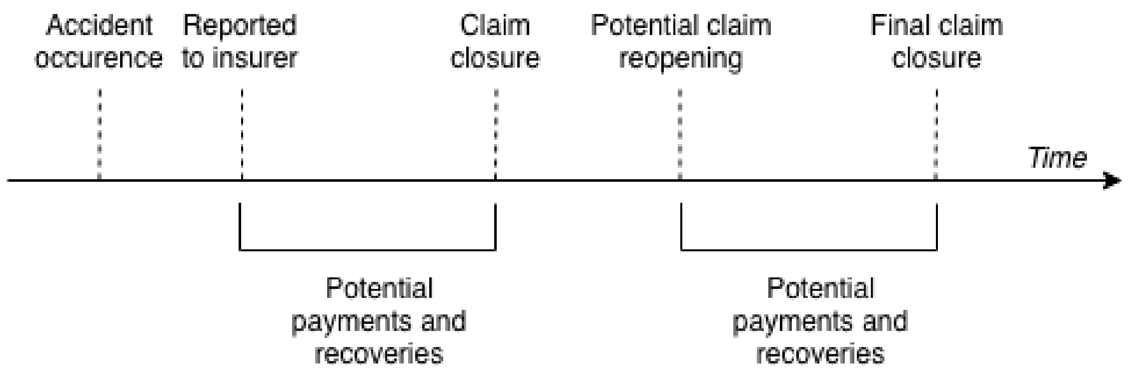
Accident occurence
Reported to insurer
Claim closure
Potential claim reopening
Final claim closure
Time
Potential payments and recoveries
Potential payments and recoveries

Figure 1. Development of a claim

Bayesian Inference on Neural Networks

In this section, we briefly discuss Bayesian inference on neural networks, following the discourse in (Graves 2011) and (Blundell et al. 2015).

Consider a neural network parameterized by weights w that takes input x and returns output y. We can view this general formulation as a probabilistic model $P(y|x,w)$; as an example, in the case of linear regression where $y \in \mathbb{R}$, mean squared loss is specified, and constant variance is assumed, $P(y|x,w)$ corresponds to a Gaussian distribution.

Rather than treating w as a fixed unknown parameter, we adapt a Bayesian perspective and treat w as a random variable with some prior distribution $P(w)$. The task is then to compute the posterior distribution $P(w|x,y)$ given the training data. We can then calculate the posterior predictive distribution $P(y^*|x^*) = \mathbb{E}_{P(w|x,y)}[P(y^*|x^*,w)]$, where x^* is a new data point to be scored and Y^* the corresponding unknown response. However, determining $P(w|x,y)$ analytically is intractable, and convergence of Markov chain Monte Carlo (MCMC) to the actual posterior for nontrivial neural networks is too slow to be feasible. Variational inference provides a workaround to this problem by approximating the posterior with a more tractable distribution $q(w|\theta)$, which is often chosen to come from the mean-field family, i.e., $q(z) = \prod_i q\,(z_i)$, but can be more general (Blundell et al. 2015). We then formulate an optimization problem to find the parameters θ. Specifically, we minimize the Kullback-Leibler (KL) divergence from the true posterior distribution P to the approximate distribution q. KL divergence is defined in general for probability distributions P to Q as

$$D_{KL}(Q\|P) = \int q(x)\log\frac{q(x)}{p(x)}dx. \qquad (1)$$

Our optimization problem can then be stated as

$$\begin{aligned}\theta^* &= \underset{\theta}{\operatorname{argmin}} D_{KL}(q(w|\theta)\|P(w|\mathcal{D})) \qquad (2)\\ &= \underset{\theta}{\operatorname{argmin}} \int q(w|\theta)\log\frac{q(w|\theta)}{P(w|\mathcal{D})}dw \qquad (3)\\ &= \underset{\theta}{\operatorname{argmin}} \int q(w|\theta)\log\frac{q(w|\theta)}{P(w)P(\mathcal{D}|w)}dw \qquad (4)\\ &= \underset{\theta}{\operatorname{argmin}} D_{KL}(q(w|\theta)\|P(w)) - \mathbb{E}_{q(w|\theta)}[\log P(\mathcal{D}|w)] \qquad (5)\end{aligned}$$

where $\mathcal{D} = (x_i, y_i)_i$ is the training dataset. We remark that, in Equation (2), the $P(\mathcal{D})$ term resulting from applying Bayes' theorem to $P(w|\mathcal{D})$ disappears because it is irrelevant to the optimization.

Equation (1) then gives us the optimization objective[1]

$$\mathcal{F}(\mathcal{D},\theta) = D_{KL}(q(w|\theta)\|P(w)) - \mathbb{E}_{q(w|\theta)}[\log P(\mathcal{D}|w)] \qquad (6)$$

In practice, during training, where $\mathcal{D}$ is randomly split into M mini-batches $\mathcal{D}_1, \dots, \mathcal{D}_M$, we compute $\mathcal{F}(\mathcal{D},\theta) = \sum_1^M \mathcal{F}_i\,(\mathcal{D}_i,\theta)$, where

$$\mathcal{F}_i(\mathcal{D}_i,\theta) = \frac{|\mathcal{D}_i|}{|\mathcal{D}|}D_{KL}(q(w|\theta)\|P(w)) - \mathbb{E}_{q(w|\theta)}[\log P(\mathcal{D}_i|w)] \qquad (7)$$

which we approximate with

$$\mathcal{F}_i(\mathcal{D}_i,\theta) \approx \sum_{j=1}^{|\mathcal{D}_i|}\frac{1}{|\mathcal{D}|}(\log q(w^{(j)}|\theta) - \log P(w^{(j)})) - \log P(\mathcal{D}_i|w^{(j)}) \qquad (8)$$

[1] We note that $-\mathcal{F}(\mathcal{D},\theta)$ is often referred to as the evidence lower bound (ELBO) in the machine learning literature.

where the $w^{(j)}$ are sampled independently from $q(w|\theta)$. In other words, we sample from the weights distribution just once for each training sample.

Mixture Density Networks

In practical applications, the response variables we try to predict often have multimodal distributions and exhibit heteroscedastic errors. This is particularly relevant in forecasting claims cash flows since, in a given time period, there could be a large payment or little or no payment. An MDN allows the output to follow a mixture of arbitrary distributions and estimate each of its parameters with the neural network. Recall that a mixture distribution has a distribution function of the form

$$F(z) = \sum_{i=1}^{n} w_i \, P_i(z), \qquad (9)$$

where each $w_i \geq 0$, $\sum w_i = 1$, and n is the number of component distributions P_i. Letting $\mathcal{P}$ denote the union of the sets of parameters for the distributions $P_1, \dots, P_n$, the neural network must then output $n + |\mathcal{P}|$ values. The first n values determine the categorical distribution of the mixing weights, and the $|\mathcal{P}|$ outputs parameterize the component distributions.

By obtaining distributions rather than single points as prediction outputs, we also gain a straightforward mechanism to quantify the uncertainty of individual cash flow forecasts.

We emphasize now the difference between the uncertainty captured by specifying a distribution as the neural network's output, as discussed here, and the uncertainty in the weight distribution, as discussed in Section 3.2. The former corresponds to the irreducible pure randomness, while the latter reflects uncertainty in parameter estimation.

Data and Model

In this section, we describe the data used for our experiments and the proposed model. The dataset used and relevant code are available on GitHub[2]. The experiments are implemented using the R programming language (R Development Core Team 2011) and TensorFlow (Abadi et al. 2015).

Data

We utilize the individual claims history simulator of (Gabrielli and V Wüthrich 2018) to generate data for our experiments. For each claim, we have the following static information: line of business, labor sector of the injured, accident year, accident quarter, age of the injured, part of body injured, and the reporting year. In addition, we also have 12 years of claim development information, in the form of cash flows and claims statuses (whether the claim is open or not). The generated claims history exhibit behaviors such as negative cash flows for recoveries and late reporting, which mimic realistic scenarios.

Since we only have claims data and not policy level and exposure data, we study only reported claims.

Experiment Setup

We first simulate approximately 500,000[3] claims using the simulator, which provides us with the full development history of these claims. The claims cover accident years 1994 to 2005. Since we are concerned with reported claims, we remove claims with report date after 2005, which leaves us with N = 497,516 claims. In this paper, we assume that each claim is fully developed at development year 11 (note that in the dataset the first development year is denoted year 0). More formally, we can represent the dataset as the collection

$$\mathcal{D} = \{(X^{(j)}, \left(C_i^{(j)}\right)_{0\le i\le 11}, \left(S_i^{(j)}\right)_{0\le i\le 11}\right) : j \in 1, \dots, N\Big\}, \qquad (10)$$

[2] https://github.com/kasaai/bnn-claims

[3] The number of claims generated is stochastic; in our case, we draw 500,904 claims.

where X, (C_i), and (S_i) denote the static claim features, incremental cash flow sequences, and claim status sequences, respectively, and j indexes the claims.

To create the training and testing sets, we select year 2005 as the evaluation year cutoff. For the training set, any cash flow information available after 2005 is removed. In symbols, we have

$$\mathcal{D}_{\text{train}} = \left\{ \left(X^{(j)}, \left(C_i^{(j)} \right), \left(S_i^{(j)} \right) \right) : i + \mathrm{AY}^{(j)} \leq 2005, j \in 1, \dots, N \right\}, \qquad (11)$$

where $\mathrm{AY}^{(j)}$ denotes the accident year associated with claim j.

For each claim in the training set, we create a training sample for each time period after development year 0. The response variable consists of cash flow information available as of the end of each time period until the evaluation year cutoff, and predictors are derived from information available before the time period. For a claim j, we have the following input-output pairs:

$$\left\{ \left(\left(X^{(j)}, \left(C_0^{(j)}, \dots, C_i^{(j)} \right), \left(S_0^{(j)}, \dots, S_i^{(j)} \right) \right), \left(C_{i+1}^{(j)}, \dots, C_{k^{(j)}}^{(j)} \right) \right) : i = 0, \dots, k^{(j)} - 1 \right\}, \qquad (12)$$

where $k^{(j)}$ denotes the latest development year for which data is available for claim j in the training set. As an example, if a claim has an accident year of 2000, five training samples are created. The first training sample has cash flows from 2001 to 2005 for the response and one cash flow value from 2000 for the predictor, while the last training sample has only the cash flow in year 2005 for the response and cash flows from 2000 to 2004 for the predictor.

We note here that we do not predict future claim statuses. As we will discuss in Section 4.4.4, our output distributions, which contain point masses at zero, can accommodate behaviors of both open and closed claims.

The training samples in Equation (12) undergo additional transformations before they are used for training our model. We discuss these transformations in detail in the next section.

Feature Engineering

We exhibit the predictors used and associated transformations in Table 1. In the raw simulated data, each time period has a cash flow amount along with a claim open status indicator associated with it. We take the cash flow value and derive two variables from it: the paid loss and the recovery, corresponding to the positive part and negative part of the cash flow, respectively. In other words, for each claim and for each time step, at most one of paid loss and recovery can be nonzero. For predictors, both the paid losses and recoveries are centered and scaled with respect to all instances of their values in the training set. The response cash flow values are not normalized.

Claim status indicator, which is a scalar value of 0 or 1, is one-hot encoded (i.e., represented using dummy variables); for each training sample, a claim status value is available for each time step.

Development year is defined as the number of years since the accident occurred. It is then scaled to [0,1].

The static claim characteristic variables include age, line of business, occupation of the claimant, and the injured body part. Of these, age is numeric while the others are categorical. We center and scale the age variable and integer-index the others, which are fed into embedding layers (Guo and Berkhahn 2016), discussed in the next section.

As noted in the previous section, the response variable and the sequence predictors can have different lengths (in terms of time steps) from one sample to the next. To facilitate computation, we pre-pad and post-pad the sequences with a predetermined masking value (we use 99999) so that all sequences have a fixed length of 11.

Table 1. Predictor variable and transformations

Variable	Type	Preprocessing
Paid loss history	Numeric sequence	Centered and scaled
Recovery history	Numeric sequence	Centered and scaled
Claim status history	Categorical sequence	One-hot encoded
Development year	Integer	Scaled to [0, 1]
Age	Numeric	Centered and scaled
Line of business	Categorical	Indexed
Claim code (occupation)	Categorical	Indexed
Injured part	Categorical	Indexed

Claims Forecasting Model

We utilize an encoder-decoder architecture with sequence output similar to the model proposed by (Kuo 2019). The architecture is illustrated in Figure 2. We first provide a brief overview of the architecture, then provide details on specific components later in the section.

The output in our case is the future sequence of distributions of loss payments, and the input is the cash flow and claim status history along with static claim characteristics.

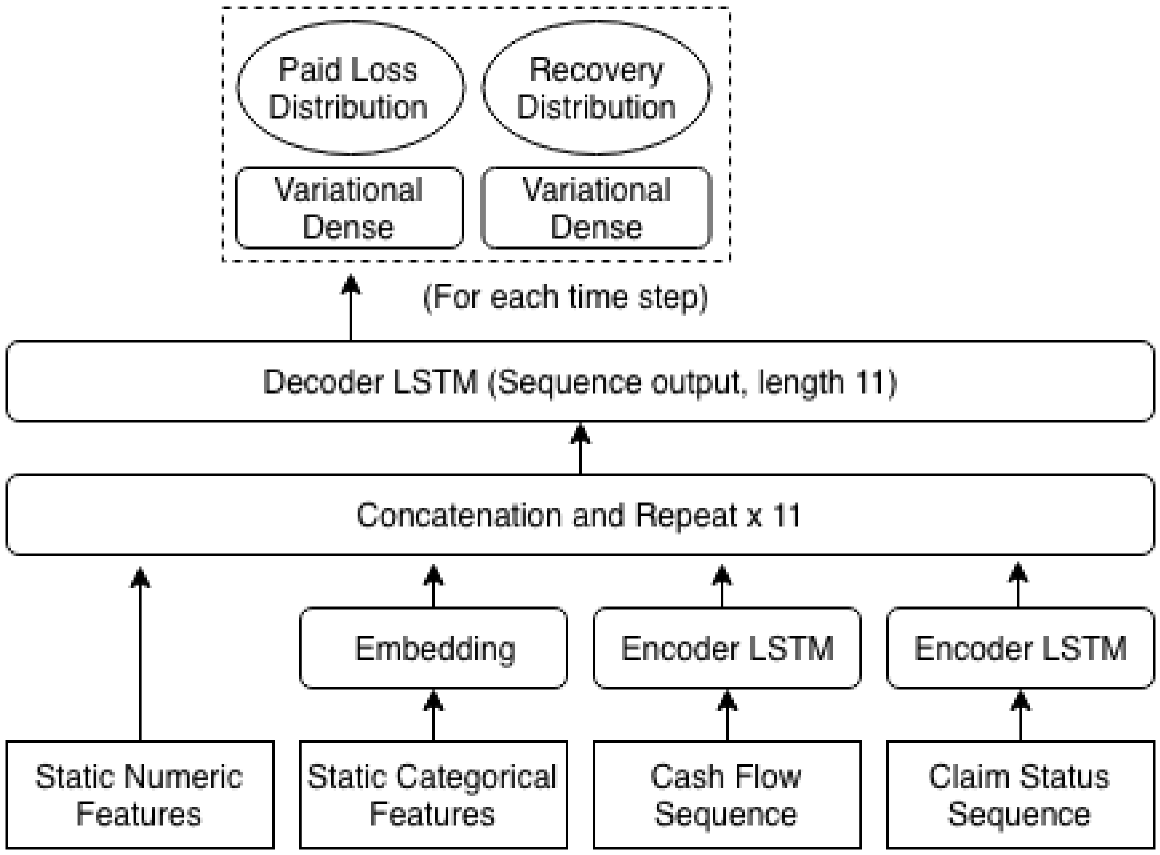

Figure 2. Claims forecasting model architecture. Note that the weights for the variational dense layers are shared across time steps.

Static categorical variable inputs, such as labor sector of the injured, are indexed, then connected to embedding layers (Guo and Berkhahn 2016) and sequential data, including payments and claim statuses, are connected to long short-term memory (LSTM) layers (Hochreiter and Schmidhuber 1997).

The encoded inputs are concatenated together, then repeated 11 times before being passed to a decoder LSTM layer which returns a sequence. The length of this output sequence is so chosen to match our requirement to forecast a maximum of 11 steps into the future. Each time step of this sequence is connected to two dense variational layers, each of which parameterizes an output distribution, corresponding to paid losses and recoveries, respectively. The weights of the dense variational layer (for paid loss and recovery) are each shared across the time steps. In other words, for each training sample, we output two 11-dimensional random variables, each of which can be considered as a collection of 11 independent but non-identically distributed random variables.

We use the same loss function for each output and weight them equally for model optimization. The loss function is the negative log-likelihood given the target data with an adjustment for variable output lengths which we discuss in detail in Section 4.4.5.

In the remainder of this section, we discuss in more detail embedding layers, LSTM, our choices of the variational and distribution layers, the loss function, and model training.

Embedding Layer

An embedding layer maps each level of a categorical variable to a fixed-length vector in n-dimensional Euclidean space. The value of n is a hyperparameter chosen by the modeler; in our case we select $n = 2$ for all embedding layers, which means we map each factor level to a point in $\mathbb{R}^2$. In contrast to data-preprocessing dimensionality techniques such as principal component analysis (PCA) or t-distributed stochastic neighbor embedding (t-SNE), the values of the embeddings are learned during training of the neural network.

Long Short-Term Memory

LSTM is a type of recurrent neural network (RNN) architecture that is appropriate for sequential data, such as natural language or time series (our use case). Empirically, LSTM is more apt at handling data with longer term dependencies than simple RNN (LeCun, Bengio, and Hinton 2015). In our model, for both of the sequence input encoders and the decoder, we utilize a single layer of LSTM with 3 hidden units. We found that these relatively small modules provided reasonable results but did not perform comprehensive tuning to test deeper and larger architectures.

Dense Variational Layer

We choose Gaussians for both the prior and surrogate posterior distributions over the weights of the dense layers. The prior distribution is constrained to have zero mean and unit variance while for the surrogate posterior both the mean and variance are trainable. Symbolically, if we let w denote the weights associated with each dense layer, we have

$$P(w) = \mathcal{N}(0, I) \text{ and} \qquad (13)$$

$$q(w|\theta) = \mathcal{N}\big(\mu(\theta), \sigma(\theta)\big), \qquad (14)$$

where θ represents trainable parameters. The KL term associated with estimation, as described in Section 3.2, is added to the neural network loss during optimization. The dense layers each output four units to parameterize the output distributions which we discuss next.

Output Distribution

For each of the 11 time steps we forecast, we parameterize two distributions, one for paid losses and one for recoveries. Each of the distributions is chosen to be a mixture of a shifted log-normal distribution and a deterministic degenerate distribution localized at zero, representing cash flow and no cash flow, respectively. Denoting $\{v_1, \dots, v_4\}$ to be the output of the preceding dense variational layer, we have the following as the distribution function:

$$F(y) = w_1(v_1, v_2)P(y; v_3, (\alpha + \lambda\sigma(\beta v_4))^2) + w_2(v_1, v_2)F_0(y). \qquad (15)$$

We now describe the components of Equation (15). First,

$$w_i = \frac{e^{v_i}}{e^{v_1} + e^{v_2}} \quad (i = 1,2) \qquad (16)$$

are the normalized mixing weights. $P(y; \mu, \sigma^2)$ denotes the distribution function for a log-normal distribution with location and scale parameters μ and σ, respectively, shifted to the left by 0.001 to accommodate zeros in the data. In other words, if $Y \sim LN(\mu, \sigma^2)$, then $Y - 0.001 \sim P$. This modification is necessary because we have zero cash flows in the response sequences, and computing the likelihood becomes problematic since the support of the log-normal distribution does not contain zero.[4]

$$F_0(y) = \begin{Bmatrix} 1, & \text{if } y \geq 0 \\ 0, & \text{if } y < 0 \end{Bmatrix} \qquad (17)$$

is the distribution function for the deterministic distribution, and σ is the sigmoid function defined as

$$\sigma(t) = \frac{1}{1 + e^{-t}}. \qquad (18)$$

In Equation (15), the constants $\alpha, \beta > 0$ are included for numerical stability and can be tuned as hyperparameters, although we set them to 0.001 and 0.01, respectively, for our experiments. The purpose of λ is also numerical stability, as the scale parameter of the log-normal distribution is hard to learn in practice, so we bound it above by a constant. In our experiments we fix it to be 0.7.

The net loss prediction for each development year is then the difference of the paid loss and recovery amounts. We assume that the output distributions, conditional on their parameters, are independent across the time steps, which facilitates the derivation of the loss function in the next section. We remark that this is a weakness of our approach, since the independence assumption is not fulfilled in practice.

Loss Function

We calculate the log-likelihood loss for each output by computing the log probability of the true labels with respect to the output distributions. As mentioned in Section 4.3, the output sequences may contain masking values that need to be adjusted. Specifically, we marginalize out the components with the masking values. Formally, for a given training sample, let M denote the masking constant, $Y = (Y_1, \dots, Y_{11})$ denote a single

[4] As of the writing of this paper, TensorFlow Probability attempts to evaluate all terms in the likelihood of the mixture.

output sequence, $Y_{\mathcal{T}_M} = (Y_i : Y_i = M)$, and, abusing notation, also let $\mathcal{T}_M = \{i : Y_i = M\}$. Then,

$$\begin{aligned} f_{Y \setminus Y_{\mathcal{T}_M}}(Y) &= \int_{Y_{\mathcal{T}_M}} f(Y) dY_{\mathcal{T}_M} \\ &= \int_{Y_{\mathcal{T}_M}} f(Y \setminus Y_{\mathcal{T}_M}, Y_{\mathcal{T}_M}) dY_{\mathcal{T}_M} \\ &= f_{Y \setminus Y_{\mathcal{T}_M}}(Y \setminus Y_{\mathcal{T}_M}) \int_{Y_{\mathcal{T}_M}} f_{Y_{\mathcal{T}_M}}(Y_{\mathcal{T}_M}) dY_{\mathcal{T}_M} \quad \text{(by independence)} \\ &= f_{Y \setminus Y_{\mathcal{T}_M}}(Y \setminus Y_{\mathcal{T}_M}) \\ &= \prod_{i \notin \mathcal{T}_M} f_{Y_i}(Y_i) \quad \text{(by independence).} \end{aligned}$$

The adjusted log-likelihood for a single training sample then becomes

$$\begin{aligned} \log \mathcal{L}(Y) &= \log f_{Y \setminus Y_{\mathcal{T}_M}}(Y) \\ &= \log \prod_{i \notin \mathcal{T}_M} f_{Y_i}(Y_i) \\ &= \sum_{i=1}^{T} \log f_{Y_i}(Y_i), \end{aligned}$$

where T is $\min \mathcal{T}_M - 1$ if $\mathcal{T}_M$ is nonempty and 11 otherwise.

Hence, the log-likelihood associated with a training sample is the sum of the log probabilities of the non-masked elements. The negative log-likelihood, summed with the KL term associated with the variational layers, discussed in Section 3.2, comprise the total network loss for optimization. The contribution of a single training sample to the loss is then

$$\sum_{i=1}^{T} \log f_{Y_i}(Y_i) + \frac{1}{|\mathcal{D}|} \sum_{k=1}^{2} D_{KL}\left(q_k(w_k|\theta_k) \| P_k(w_k)\right), \qquad (19)$$

where $k = 1{,}2$ correspond to variational layers for parameterizing the paid loss and recovery distributions, respectively.

Training and Scoring

We use a random subset of the training set consisting of 5% of the records as the validation set for determining early stopping and scheduling the learning rate. For optimizing the neural network, we use stochastic gradient descent with an initial learning rate of 0.01 and a minibatch size of 100,000, and we halve the learning rate when there is no improvement in the validation loss for five epochs. Training is stopped early when the validation loss does not improve for ten epochs; we cap the maximum number of epochs at 100. Here, an *epoch* refers to a complete iteration through our training dataset, and the *minibatch* size refers to how many training samples we randomly sample for each gradient descent step.

To forecast an individual claim, we construct a scoring data point by using all data available to us as of the evaluation date cutoff. In other words, the last elements of the predictor cash flow sequences correspond to the actual value from the cutoff year. Hence, in the output sequence, the first element corresponds to the prediction of the cash flow distribution of the year after the cutoff.

Recall that our model weights are a random variable, so each time we score a new data point, we are sampling from the distribution of model weights and expect to obtain different parameters for the distributions of cash flows. The output distributions themselves are also stochastic and can be sampled from. Following (Kendall and Gal 2017), we refer to the former variability as *epistemic* uncertainty and the latter as *aleatoric* uncertainty. Epistemic uncertainty reflects uncertainty about our model which can be reduced given more training data, while aleatoric uncertainty reflects the irreducible noise in the observations. In actuarial science literature, these concepts are also known as parameter estimation uncertainty and process uncertainty, respectively (Wuthrich 2017).

Generating a distribution of future paths of cash flows amounts to repeating the procedure of drawing a model from its distribution, calculating the output distribution parameters using the drawn model, then drawing a sample of cash flows from the distribution. Since we decompose the cash flows into paid losses and recoveries, we can obtain net amounts by subtracting the generated recoveries from the generated paid losses.

If we were only interested in point estimates, we can avoid sampling the output distributions and simply compute their means, since given the model weights we have closed form expressions for the distributions given by Equation (15). Note that we would still have to sample the model weights.

Results and Discussion

We evaluate our modeling framework by qualitatively inspecting samples paths generated for an individual claim and also comparing the aggregate estimate of unpaid claims with a chain ladder estimate. We note that, to the best of our knowledge, prior to the current paper, there is not a benchmark for individual claims forecasts. We also discuss the extensibility of our approach to accommodate company specific data elements and expert knowledge.

Individual Claim Forecasts

Figure 3 shows various posterior densities, obtained by sampling the model weights 1,000 times, of parameters for the output distributions of a single claim. Our model assigns a higher probability of payment in the next year along with more variability around that probability. It can be seen that the expected probability of a payment decreases as we forecast further into the future. Given that a loss payment occurs, we see from the middle and bottom plots that both the expected value and the variability for both the mean and variance of the log-normal distributions increase with time. In other words, for this particular claim, loss payments become less likely as time goes on, but if they occur, they tend to be more severe and the model is less certain about the severity distribution.

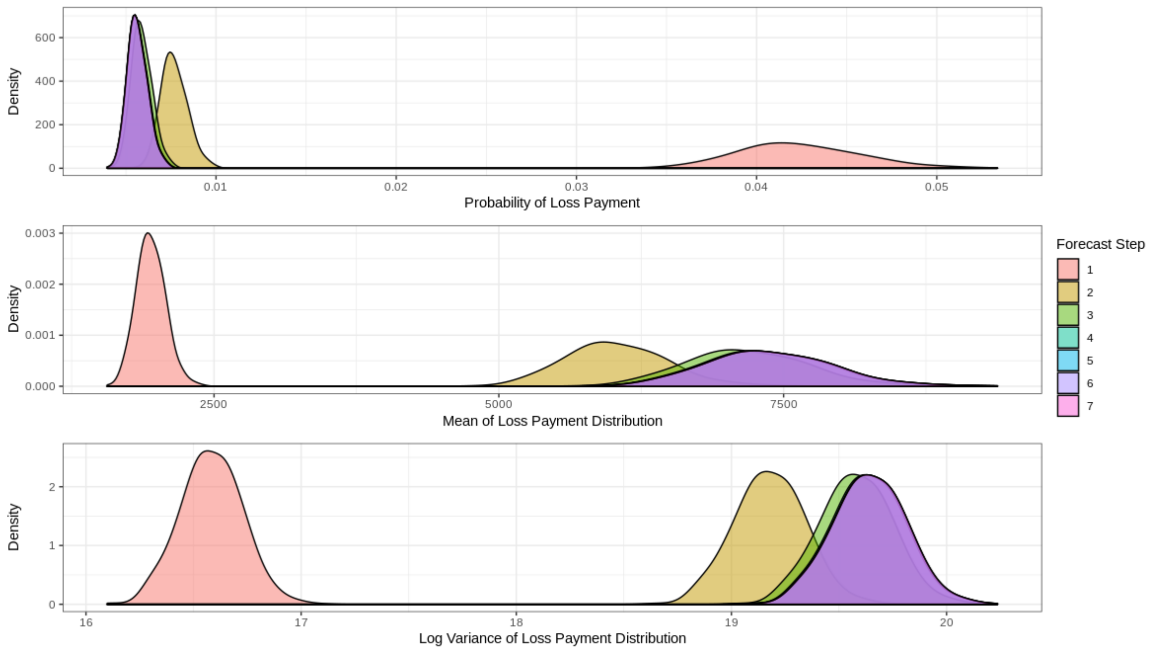

Figure 3. Posterior distributions for a single claim at development year 4. (Top) Payment probability. (Middle) Mean of loss payment distribution. (Bottom) Log variance of payment distribution.

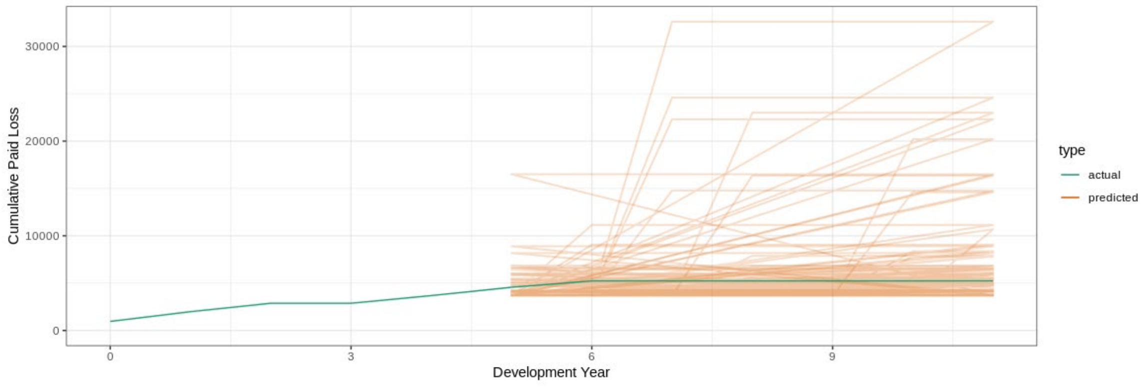

Figure 4. 1,000 samples of cumulative cash flow paths for a single claim. The training data includes development year 4 and the predictions begin in development year 5.

In Figure 4 we show plausible future developments of the claim. Note that one thousand samples are drawn, so the handful of scenarios that present large payments represents a small portion of the paths, which is consistent with the distributions of parameters we see above.

Aggregate estimates

To compute a point estimate for the total unpaid losses, we score each claim once (i.e., sample from the weights distribution once) to obtain a distribution for each time step for paid losses and recoveries. We then compute the means of the paid loss and recovery distributions and take the differences to obtain the net cash flow amount. The final unpaid loss estimate is then the sum of all the net amounts up to and including development year 11. Since there is randomness in the neural network weight initialization, we instantiate and train the model 10 times, and take the average of the predicted future paid losses across the 10-model ensemble. In practice, one could distribute the prediction procedure over a computing cluster and draw more samples for an even more stable estimate.

For the chain ladder development method benchmark, we aggregate the data into a report year triangle (to exclude IBNR), calculate ultimate losses using all-year weighted age-to-age factors, then subtract the already paid amounts to arrive at the unpaid estimate. The unpaid amounts from the two approaches are then compared to the actual unpaid amounts. The results are shown in Table 2.

Model	Forecast	Error
Actual	109,216,388	–
Chain Ladder	108,040,572	−1.08%
Ours (10-Model Ensemble)	102,502,710	−6.15%

Table 2: Aggregate forecast results.

While we are not improving on chain ladder estimates at the aggregate level for this particular simulated dataset, our approach is able to provide individual claims forecasts, which are interesting in their own right.

Extensibility

One of the primary advantages of neural networks is their flexibility. Considering the architecture described in Section 4.4, we can include additional predictors by appending additional input layers. These inputs can be unstructured, and we can leverage appropriate techniques to process them before combining with the existing features. For example, we may utilize convolutional layers to transform image inputs and recurrent layers to transform audio or text inputs.

The forms of the output distributions can also be customized. We choose a log-normal mixture for our dataset, but depending on the specific book of business, one may want to specify a different distribution, such as a Gamma or a Gaussian. As we have done in constraining the scale parameter of the log-normal distribution, the modeler also has the ability to bound or fix specific parameters as situations call for them.

Conclusion

We have introduced a framework for individual claims forecasting that can be utilized in loss reserving. By leveraging Bayesian neural networks and stochastic output layers, our approach provides ways to learn uncertainty from the data. It is also able to produce cash flow estimates for multiple future time periods. Experiments confirm that the approach gives reasonable results and provides a viable candidate for future work on individual claims forecasting to benchmark against.

There are a few potential avenues for enhancements and extensions, including larger scale simulations to evaluate the quality of uncertainty estimates, evaluating against new datasets, and relaxing simplifying assumptions, such as the independence of time steps to obtain more internally consistent forecasts. One main limitation of the proposed approach and experiments is that we focus on reported claims only. With access to policy level data, a particularly interesting direction of research would be to extend the approach to also estimate IBNR claims.

Acknowledgments

We thank Sigrid Keydana, Daniel Falbel, Mario Wüthrich, and volunteers from the Casualty Actuarial Society (CAS) for helpful discussions. This work is supported by the CAS.

References

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, et al. 2015. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems." https://www.tensorflow.org/.

Antonio, Katrien, and Richard Plat. 2014. "Micro-Level Stochastic Loss Reserving for General Insurance." *Scandinavian Actuarial Journal* 2014 (7):649–69.

Baudry, Maximilien, and Christian Y. Robert. n.d. "A Machine Learning Approach for Individual Claims Reserving in Insurance." *Applied Stochastic Models in Business and Industry*.

Bishop, Christopher M. 1994. "Mixture Density Networks."

Blundell, Charles, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. "Weight Uncertainty in Neural Networks." *arXiv:1505.05424 [Cs, Stat]*, May. http://arxiv.org/abs/1505.05424.

Boumezoued, Alexandre, and Laurent Devineau. 2017. "Individual Claims Reserving: A Survey."

Duval, Francis, and Mathieu Pigeon. 2019. "Individual Loss Reserving Using a Gradient Boosting-Based Approach." *Risks* 7 (3):79. https://doi.org/10.3390/risks7030079.

Gabrielli, Andrea. 2019. "A Neural Network Boosted Double over-Dispersed Poisson Claims Reserving Model." SSRN Scholarly Paper ID 3365517. Rochester, NY: Social Science Research Network.

Gabrielli, Andrea, Ronald Richman, and Mario V. Wüthrich. 2019. "Neural Network Embedding of the over-Dispersed Poisson Reserving Model." *Scandinavian Actuarial Journal*, 1–29.

Gabrielli, Andrea, and Mario V Wüthrich. 2018. "An Individual Claims History Simulation Machine." *Risks* 6 (2):29.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT press.

Graves, Alex. 2011. "Practical Variational Inference for Neural Networks." In *Advances in Neural Information Processing Systems*, 2348–56.

Guo, Cheng, and Felix Berkhahn. 2016. "Entity Embeddings of Categorical Variables." *arXiv Preprint arXiv:1604.06737*.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. "Long Short-Term Memory." *Neural Computation* 9 (8):1735–80.

Kendall, Alex, and Yarin Gal. 2017. "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" In *Advances in Neural Information Processing Systems*, 5574–84.

Kuo, Kevin. 2019. "DeepTriangle: A Deep Learning Approach to Loss Reserving." *Risks 7* (3):97.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. "Deep Learning." *Nature* 521 (7553):436.

Lopez, Olivier, Xavier Milhaud, and Pierre-E. Thérond. 2019. "A Tree-Based Algorithm Adapted to Microlevel Reserving and Long Development Claims." *ASTIN Bulletin: The Journal of the IAA*, 1–22.

Neal, Radford M. 2012. *Bayesian Learning for Neural Networks*. Springer Science & Business Media.

Pigeon, Mathieu, Katrien Antonio, and Michel Denuit. 2013. "Individual Loss Reserving with the Multivariate Skew Normal Framework." *ASTIN Bulletin: The Journal of the IAA* 43 (3):399–428.

———. 2014. "Individual Loss Reserving Using PaidIncurred Data." *Insurance: Mathematics and Economics* 58:121–31.

R Development Core Team, RFFSC. 2011. *R: A Language and Environment for Statistical Computing*. R foundation for statistical computing Vienna, Austria.

Taylor, Greg. 2019. "Loss Reserving Models: Granular and Machine Learning Forms." *Risks* 7 (3):82.

Wuthrich, Mario V. 2017. "Non-Life Insurance: Mathematics & Statistics." *Available at SSRN 2319328*.

Wüthrich, Mario V. 2018a. "Machine Learning in Individual Claims Reserving." *Scandinavian Actuarial Journal* 2018 (6):465–80.

———. 2018b. "Neural Networks Applied to ChainLadder Reserving." *European Actuarial Journal* 8 (2):407–36.