Claim Reserving: Classical versus Machine Learning Methods

Claim Reserving: Classical versus Machine Learning Methods

Pengfei(Frank) Cai

A thesis

submitted to the Department of Mathematics & Statistics

and the school of graduate studies

of McMaster University

in partial fulfillment of the requirements

for the degree of

Master of Science

Master of Science (2021) McMaster University

(Statistics) Hamilton, Ontario, Canada

TITLE:	Claim Reserving: Classical versus Machine Learning Methods
AUTHOR:	Pengfei Cai
SUPERVISORS:	Dr. Anas Abdallah and Dr. Traian Pirvu
NUMBER OF PAGES:	xi, 83

Abstract

In this thesis, we investigate loss reserving with classical methods and deep learning approach. Claim reserving is a crucial task in non-life insurance. Insurance companies have historical data on claim amounts. These data are usually aggregated to loss triangles, which can be analyzed to predict the claim reserve. The reserves for different business lines are related and a copula regression model can link the claims of different business lines. We apply the models on real data and obtain both reserve estimates and risk capitals. Product copula, Gaussian copula, Frank copula and Student's t copula are used to model the dependence of the two business lines. The AIC of Student's t copula model is 1.3 percent smaller than the AIC of the Gaussian copula model. The AIC of Frank copula model and Product copula model is within 0.9 percent of the Gaussian copula model's AIC. The Gaussian copula model generates the largest risk capital gain among all the copula models. Neural networks are popular machine learning methods and have been applied to the loss reserving problem. The DeepTriangle model is a deep learning framework for forecasting paid losses. At each accident year and development year for which we have data, we predict future losses based on observed history. Each training and testing sample is associated with an accident year - development year pair. The input for the training and testing sample is the observed incremental paid loss and claims outstanding as of the accident - development year pair. To capture the dependency between two lines of business, we use the incremental paid loss from two business lines as input to the deep triangle model. The incre-

mental paid loss from the personal auto line and the incremental paid loss from the commercial auto line correspond to the first and the second component of the sample. The predicted reserve from the deep triangle model is within two percent, three percent and nine percent of the predictions from the Gaussian copula, Frank copula and Student's t copula models, respectively.

Acknowledgements

First of all, I would like to express my greatest gratitude to my supervisors Dr. Anas Abdallah and Dr. Traian Pirvu for their patience and support during the research study. Their feedback and insights help me develop the research work. They guided me and offered me suggestions in all the time of research of this thesis.

I want to thank my committee member Dr. Anas Abdallah, Dr. Pratheepa Jeganathan and Dr. Trian Pirvu for their comments on the thesis. I'm indebted to them for their encouragement and help.

Finally, I'm also grateful for all the unconditional supports from my family and my friends. This thesis would not have been possible without them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Insurance companies need to set up funds for future compensation of policy holders who have made claims. This amount is called the claims reserve. It is important that the claims reserve is calculated carefully. On the basis of historical data one can estimate the expected outstanding claims. The reserving methods used in practice are frequently deterministic. For instance, the claims reserve is often obtained with the chain-ladder method (Mack, 1993). Hence, people calculate the reserve estimates without explicit reference to a stochastic model.

However, due to poor data quality or even lack of data, the uncertainty of the best estimate can be quite high. In order to manage this risk it is important that the claim reserve estimate is complemented by some measure of uncertainty.

In Chapter 2, we introduce the common loss models for reserving. For the statistical approach, we first specify a model and then find an estimate of the outstanding claims under that model. Standard statistical software packages can be used to obtain maximum likelihood estimates for the model parameters. The model can be used to find the variability of the estimate. The object of these papers is to find a model under which the estimate is the same as the one given by the chain-ladder method, see e.g. Verrall (2000), Mack and Venter (2000) and Verrall and England (2000).

The variability of the claims reserve can be obtained analytically or by simulation. For example, Mack (1993) derived the mean squared error of prediction for the chain-ladder method. The variability of the reserve can be difficult to calculate analytically since the reserve estimators are complex functions. England and Verrall (1999, 2002) introduced the bootstrapping method for the chain-ladder. The mean squared error of prediction can be estimated with bootstrap simulation.

Insurance companies may have multiple lines of business. Assuming that there is no dependency between different lines of business, the total reserve is the sum of the reserves for separate lines of business. However, there may be dependence between multiple lines of business in practice. The model of Shi and Frees (2011) capture the dependence between two lines of business with a copula that links the claims with the same accident and development year. We present the copula regression models and applications in Chapter 3. Product copula, Gaussian copula, Frank copula and Student's t copula are used to model the dependence between the personal auto line and the commercial auto line. Tail value at risk (TVaR) is calculated as the risk measure for the portfolio. Compared with the silo model, the Gaussian copula model results in the largest risk capital gain among all the copula models.

Neural networks are popular methods in machine learning. Feed-forward neural networks are inspired by the functionality of the brains. The generic architecture of feed-forward neural networks are described in Chapter 4. It first computes an intermediate variable z from the input x. z is formed by applying weights w and bias b to the input x. The network layer is also equipped with an activation function designed to reproduce the output of interest. The activation function is non-linear and applied to the intermediate variable z. For example, in a regression model, the activation can be the ReLu function, whereas in a classification model with two classes, the sigmoid activation is used since it provides an output which can be interpreted as a probability.

To fit the neural network, a loss function $L(y, \hat{y})$, measuring the quality of the predicted output of the model, $\hat{y}$, compared to the observations y is specified. The loss function $L()$ can be chosen flexibly based on the problem. For regression, common loss functions are the mean squared error (MSE) or the mean absolute error (MAE), while for classification the cross-entropy loss is often used. The model is then fit by back-propagation (Rumelhart et al., 1986) and gradient descent. The gradient descent updates the parameters of the model by computing the gradient and adjusting the parameters in the direction of the gradient. For each round of gradient descent, a small step in the direction of the gradient is taken to ensure that the minimum of the loss function is found.

The application of deep learning techniques to loss reserving problems seems to be a rapidly emerging field. Recently, Kuo (2019) proposed the DeepTriangle model for loss reserving based on deep neural networks. In Chapter 5, following Kuo (2019), we present the training data and testing data setup for the deep triangle model. Each training sample is associated with an accident year - development year pair. A mask value is used if there is no data at that time step. The sequences consists of 9 time steps. At each accident year and development year for which we have data, we predict future incremental paid loss and claims outstanding based on the observed history. We applied Kuo's DeepTriangle model to one of the loss triangles from the commercial line. The model is trained for a maximum of 1000 epochs with early stopping scheme. We keep the data with development lag less or equal than 10 years for the predictions. The same model is also applied to a dataset from a major US property-casualty insurer. The first component of the sample is the incremental paid loss from the personal line of business, and the second component of the sample corresponds to the incremental paid loss from the commercial line of business. Kuo's DeepTriangle can predict the reserves for the personal line and commercial line simultaneously. We run the DeepTriangle 100 times and use the average as the predicted reserve.

Chapter 2

Loss reserving

2.1 Loss triangles

Insurance companies have data bases with historical information on incurred claims. These information can be the number of claims, the origin year of the accidents, the year of the payments and claim amounts. These data can be analyzed to predict the claim reserve. Loss triangles are usually used to aggregate the data.

Let us denote the aggregated incremental payments of all claims with accident year $i(0 \leq i \leq I)$ and development year $j(0 \leq j \leq J)$ by $Y_{i,j}$. Here $Y_{i,j}$ refers to all payments in development year j for the claims occurred in year i. The corresponding cumulative payments for the claims with accident year i after j development years are denoted as

$$C_{i,j} = \sum_{k=0}^{j} Y_{i,k} \tag{2.1}$$

We now consider such an incremental triangle of paid claims $\{Y_{i,j};\ \text{i,j}\in \nabla\}$, where the business has been observed during I years, i.e. ∇={i=0,...,I;j=0,...,I-i} (Table 2.1). The indexes i and j of the paid claims refer to the origin year and the development year, respectively. In addition, the diagonals of ∇ are the calendar

years.

If we assume that the claims are settled within the J observed years, then we would like to predict the claim amounts in the lower, unobserved future triangle $\{Y_{i,j};\ \text{i,j} \in \nabla\}$, where Δ={i=1,...,I;j=I-i+1,...,J}, see Table 2.2. The sum $R = \sum_{\Delta} Y_{ij}$ is the outstanding claims for which the insurance company must hold a reserve.

Table 2.1: The triangle of observed incremental claims

	Development year					
Origin year	0	1	2	...	J-1	J
0	Y_{00}	Y_{01}	Y_{02}	...	$Y_{0,J-1}$	$Y_{0,J}$
1	Y_{10}	Y_{11}	Y_{12}	...	$Y_{1,J-1}$	
2	Y_{20}	Y_{21}	Y_{22}	...		
...	...	...	...			
I-1	$Y_{I-1,0}$	$Y_{I-1,1}$				
I	$Y_{I,0}$					

Further, the claims reserves for accident year i can be shown as

$$R_i = \sum_{j=I+1-i}^{J} Y_{i,j} \quad i \in \{I+1-J, .., I\} \tag{2.2}$$

The total reserve for the outstanding payments at time I are given by

$$R = \sum_{i=1}^{I} \sum_{j=I-i+1}^{J} Y_{i,j} \tag{2.3}$$

We have observations $Y_{i,j}(i + j \leq I)$ in the upper triangle. The goal is to predict the total reserve R. We use the predicted mean as the estimate for $Y_{i,j}$ in lower triangle. The means $\mu_{i,j}$ of $Y_{i,j}$ are related to the explanatory variable: accident year i and development year j.

Table 2.2: The triangle of future incremental claims

	Development year					
Origin year	0	1	2	...	J-1	J
0						
1						$Y_{1,J}$
2					$Y_{2,J-1}$	$Y_{2,J}$
...					...	...
I-1			$Y_{I-1,2}$	...	$Y_{I-1,J-1}$	$Y_{I-1,J}$
I		$Y_{I,1}$	$Y_{I,2}$	...	$Y_{I,J-1}$	$Y_{I,J}$

2.2 Mack's Chain-ladder

Mack's chain-ladder model (Mack, 1993) is probably the most popular loss reserving method in theory and practice. Individual claim payments are aggregated by accident year and development year. Let indexes $0 \leq i \leq I$ denotes the accident years and $0 \leq j \leq J$ denotes the development years. $C_{i,j}$ is the cumulative claims payments with accident year i and development year j. $C_{i,J}$ is the total ultimate claim amount of accident year i.

Model Assumptions

- Cumulative claims $C_{i,j}$ of different accident years i are independent.
- There exist factors $f_0, ..., f_{J-1} \geq 0$ and variance parameters $\sigma_0^2, ..., \sigma_{J-1}^2 \geq 0$ such that for all $0 \leq i \leq I$ and $1 \leq j \leq J$ we have

$$E[C_{i,j}|C_{i,j-1}] = f_{j-1}C_{i,j-1} \tag{2.4}$$

$$Var(C_{i,j}|C_{i,j-1}) = \sigma_{j-1}^2 C_{i,j-1} \tag{2.5}$$

- Here we do not make assumptions on the explicit distribution of $C_{i,j}$ given $C_{i,j-1}$. We just made assumptions on the first two moments.

Parameter estimation

The parameters f_j and σ_j^2 are estimated by

$$\hat{f}_j = \frac{\sum_{i=0}^{I-j-1} C_{i,j+1}}{\sum_{i=0}^{I-j-1} C_{i,j}} = \sum_{i=0}^{I-j-1} \frac{C_{i,j}}{\sum_{k=0}^{I-j-1} C_{k,j}} \frac{C_{i,j+1}}{C_{i,j}} \tag{2.6}$$

$$\hat{\sigma}_j^2 = \frac{1}{I-j-1} \sum_{i=0}^{I-j-1} C_{i,j}(\frac{C_{i,j+1}}{C_{i,j}} - \hat{f}_j)^2 \tag{2.7}$$

If we denote the individual development factors by $F_{i,j+1} = \frac{C_{i,j+1}}{C_{i,j}}$, then the factor estimates $\hat{f}_j$ are weighted averages of $F_{i,j+1}$, that is,

$$\hat{f}_j = \sum_{i=0}^{I-j-1} \frac{C_{i,j}}{\sum_{k=0}^{I-j-1} C_{k,j}} F_{i,j+1} \tag{2.8}$$

Based on the estimations of f_j, we can show that the claim reserve for accident year i is

$$\hat{R}_i = C_{i,J} - C_{i,I-i} = C_{i,I-i}(\hat{f}_{I-i}...\hat{f}_{J-1} - 1) \tag{2.9}$$

Finally, the total outstanding losses is estimated by

$$\hat{R} = \sum_{i=1}^{I} \hat{R}_i \tag{2.10}$$

2.3 Generalized linear models - background

In this section, we first introduce the generalized linear models (GLM) as a class of statistical models. As a generalization of the classical linear models, this class includes for example the Poisson model, log-normal model and the gamma model.

2.3.1 Exponential family of distributions

Before applying the generalized linear models in claims reserve problems, we describe the theoretical background of the GLM. The theory of the generalized linear models is introduced in Nelder and Wedderburn (1972). The response variable can be any distribution from the exponential family. The probability density function of the exponential family has the general form

$$f(y) = c(y, \phi)exp\{\frac{y\theta - a(\theta)}{\phi}\}, \tag{2.11}$$

where θ is the canonical parameter and ϕ is the dispersion parameter. Different choice of the function $a(\theta)$ and $c(y, \phi)$ determine the actual probability function such as the normal, Poisson or gamma.

Lemma 1. *For exponential family response distributions, we have the following identity*

$$a''(\theta) = Var(y)/\phi. \tag{2.12}$$

Proof. We first define $f'(y)$ and $f''(y)$ as the first and second derivatives of $f(y)$ with respect to θ. Then

$$f'(y) = f(y)\left(\frac{y - a'(\theta)}{\phi}\right) \tag{2.13}$$

$$f''(y) = f(y)\left(\frac{y - a'(\theta)}{\phi}\right)^2 - f(y)\frac{a''(\theta)}{\phi} \tag{2.14}$$

Integrating both sides of each of these expressions with respect to y yields

$$0 = \frac{E(y) - a'(\theta)}{\phi} \tag{2.15}$$

$$0 = \frac{E[(y - a'(\theta))^2]}{\phi^2} - \frac{a''(\theta)}{\phi}. \tag{2.16}$$

The left hand sides are zeros since

$$\int f'(y)dy = \frac{\partial}{\partial\theta}\int f(y)dy, \tag{2.17}$$

$$\int f''(y)dy = \frac{\partial^2}{\partial\theta^2}\int f(y)dy, \tag{2.18}$$

where $\int f(y)dy = 1$ and assuming integration and differentiation can be interchanged. □

The mean and variance of y is given by

$$E(y) = a'(\theta), \quad Var(y) = \phi a''(\theta), \tag{2.19}$$

where $a'(\theta)$ and $a''(\theta)$ are the first and second derivatives of $a(\theta)$ with respect to θ.

distribution	θ	$a(\theta)$	ϕ	$E(y)$	$V(\mu) = \frac{Var(y)}{\phi}$
N(μ, σ^2)	μ	$\frac{1}{2}\theta^2$	σ^2	μ	1
Poisson(μ)	$ln\mu$	e^θ	1	μ	μ
Gamma(μ, ν)	$-\frac{1}{\mu}$	$-ln(-\theta)$	$\frac{1}{\nu}$	μ	μ^2

Table 2.3: Exponential family distributions and their parameters

The exponential family includes a number of the well-known distributions, as illustrated in Table 2.3.

2.3.2 Model introduction

The generalized linear model is

$$f(y) = c(y,\phi)exp\{\frac{y\theta - a(\theta)}{\phi}\}, \quad g(\mu) = x^T\beta. \tag{2.20}$$

Here we assume observations y are independent. The first equation describes the distribution of the random variable y. $g(\mu)$ is called the link function and

it determines how the mean of y is related to the explanatory variables x. For example, the link function for the normal linear model is identity and the relationship is described as $\mu = x^T\beta$. In GLM, it is generalized to $g(\mu) = x^T\beta$, where the function g is monotonic and differentiable. Given x, μ is determined by $g(\mu)$. Then θ is calculated from $a'(\theta) = \mu$.

Let $\eta = g(\mu) = x^T\beta$, then η is the so-called linear predictor. Generalized linear models are extensions of classical linear models that allow the mean of the response variable y to depend on linear predictors through a link function.

The choice of a suitable link function g depends on the specific distribution assumption for the response variable y. For each distribution in the exponential family, there is a so-called canonical link function. The canonical link function g relates the canonical parameter θ directly to the linear predictor η,

$$\eta = g(\mu) = \theta \tag{2.21}$$

Since $a'(\theta) = \mu$, this implies that $g(a'(\theta)) = \theta$. Therefore the canonical link function is given by $g = (a')^{-1}$.

With the canonical link function, we have the linear model $\theta = \eta$ for the canonical parameter. For the over-dispersed Poisson model, a log link is a natural choice in the insurance reserving context (Wüthrich and Merz, 2008).

Here we summarize the steps in generalized linear modeling. Given a response variable y such as incremental claim amounts, constructing a GLM consists of the following steps:

- Choose a response distribution f(y). This is guided by the nature of the response variable.
- Choose a link $g(\mu)$. The choice is suggested by the relationship between the response and explanatory variables.

- Collect observations $y_1, .., y_n$ on the response y. Successive observations are assumed to be independent.
- Fit the model by estimating β. The fitting is usually done using software such as R, which implements maximum likelihood estimation or its variants.
- Given the estimates of β, generate predictions of y for different explanatory variable x.

2.3.3 Maximum likelihood estimation

The estimators of the model parameters are obtained by maximum likelihood estimation (MLE). MLE is a method of estimating the parameters of a probability distribution by maximizing a likelihood function, so that the observed data is most probable under the assumed model. The parameter that maximizes the likelihood function is called the maximum likelihood estimate. It is convenient to work with the natural logarithm of the likelihood function, called the log-likelihood.

In case of the GLM's, the corresponding log-likelihood function is

$$\ell(\beta, \phi) = \sum_{i=1}^{n} \ln f(y_i; \beta, \phi) = \sum_{i=1}^{n} \{\ln c(y_i, \phi) + \frac{y_i \theta_i - a(\theta_i)}{\phi}\} \tag{2.22}$$

For the maximum likelihood estimators of model parameters β, we have

$$\frac{\partial \ell}{\partial \beta_j} = \sum_{i=1}^{n} \frac{\partial \ell}{\partial \theta_i} \frac{\partial \theta_i}{\partial \beta_j}. \tag{2.23}$$

These two terms can be further written as

$$\frac{\partial \ell}{\partial \theta_i} = \frac{y_i - a'(\theta_i)}{\phi} = \frac{y_i - \mu_i}{\phi}, \quad \frac{\partial \theta_i}{\partial \beta_j} = \frac{\partial \theta_i}{\partial \eta_j} \frac{\partial \eta_i}{\partial \beta_j} = \frac{\partial \theta_i}{\partial \eta_i} x_{ij}, \tag{2.24}$$

where $\eta_i = x_i^T \beta$ and x_{ij} is component i of x_j.

Setting $\frac{\partial \ell}{\partial \beta_j} = 0$ yields the first order conditions for likelihood maximization:

$$\sum_{i=1}^{n} \frac{\partial \theta_i}{\partial \eta_i} x_{ij}(y_i - \mu_i) = 0 \Leftrightarrow X'D(y - \mu) = 0, \tag{2.25}$$

where D is the diagonal matrix with diagonal entries $\frac{\partial \theta_i}{\partial \eta_i}$,

$$(\frac{\partial \theta_i}{\partial \eta_i})^{(-1)} = \frac{\partial \eta_i}{\partial \theta_i} = \frac{\partial \eta_i}{\partial \mu_i}\frac{\partial \mu_i}{\partial \theta_i} = g'(\mu_i)a''(\theta_i) = g'(\mu_i)V(\mu_i) \tag{2.26}$$

Here D is diagonal with entries $\{g'(\mu_i)V(\mu_i)\}^{-1}$. β is implicit in these equations and it is related to μ and D. The equations in 2.25 are called the estimation equation for β.

The estimation equation for β is usually difficult to solve directly except for the normal distribution with identity link (De Jong and Heller, 2008). We can use an iterative algorithm such as Newton-Raphson method to compute the maximum likelihood estimation numerically. To obtain the numerical solutions for MLE estimation of the parameters, we can use GLM software packages in R. The ChainLadder package in R provides various GLM models and corresponding parameter estimations.

2.4 Loss models

The application of the GLM's to the claims reserving problem will be introduced. The GLM's provide us a wide range of distributions from the exponential family. With these distributions and link functions, we can estimate reserves by the loss triangle of an insurance company. We assume that the incremental payments $Y_{i,j}$ are independent. Accident year and development year are used as two factors for the regression model.

At the end of calendar year I, we have the following data available

$$Y_{i,j}; 0 \leq i \leq I, 0 \leq j \leq J, i + j \leq I \tag{2.27}$$

2.4.1 Normal model

We start with the normally distributed response variables and their combinations with the identity link function. The probability density function of a normally distributed random variable with mean μ and variance σ^2 is

$$f(y) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\{-\frac{(y-\mu)^2}{2\sigma^2}\}, \quad y \in (-\infty, +\infty) \tag{2.28}$$

One of the models for dependent variables with normal distribution is given by the following assumptions.

Incremental claims amounts $Y_{i,j}$ are independently normally distributed random variables with

$$E(Y_{i,j}) = \mu_{i,j}, \quad and \quad var(Y_{i,j}) = \phi \tag{2.29}$$

The relation between the linear predictor and the mean $\mu_{i,j}$ is defined as

$$\eta_{i,j} = \mu_{i,j} = \alpha_i + \beta_j + \gamma, \tag{2.30}$$

with $\alpha_0 = \beta_0 = 0$.

The parameters α_i and β_j denote the effects of the i-th accident year and j-th development year on the expected value of the incremental claims, respectively. γ is the intercept. Hence, the expected incremental claim is

$$E(Y_{i,j}) = \mu_{i,j} = \alpha_i + \beta_j + \gamma \tag{2.31}$$

As can be seen, this model with the identity link is a classical linear model with normally distributed error terms. Here the dispersion parameter ϕ is equal to σ^2.

2.4.2 Log-normal model

The log-normal model assumes that the incremental claims $Y_{i,j}$ are independent log-normally distributed with

$$log(Y_{i,j}) \sim \mathcal{N}(\mu_{i,j}, \sigma^2) \tag{2.32}$$

The density function for a log-normally distributed random variable y with mean μ and variance σ^2 is

$$f(y) = \frac{1}{\sqrt{2\pi\sigma^2}y} exp\{-\frac{(log(y)-\mu)^2}{2\sigma^2}\}, \quad y \in (0, +\infty) \tag{2.33}$$

We consider the form $\mu_{i,j} = \eta_{i,j}$ for a log-normal distribution with location parameter $\mu_{i,j}$ and scale parameter σ.

Here

$$\eta_{i,j} = \alpha_i + \beta_j + \gamma, \tag{2.34}$$

$$E(Y_{i,j}) = e^{\mu_{i,j}+\sigma^2/2} \quad and \quad Var(Y_{i,j}) = e^{2\mu_{i,j}+\sigma^2}(e^{\sigma^2}-1) \tag{2.35}$$

Note that the above model is not a GLM. It models the log of the incremental claims. The first step is to transform the incremental claims by taking their natural logarithm. A model is then fitted to the transformed values with least squares regression. For instance, we can use lm function in R. Within the linear regression framework we forecast the incremental claims. μ and σ are the mean and standard deviation of the logarithm, respectively. It is different from other models in that the logarithm of the incremental claims is used as the response. However, in other models, the incremental claims, themselves, are used as the response for the models.

2.4.3 Over-dispersed Poisson model

For the over-dispersed Poisson (ODP) model, the variance is not equal, but proportional to the mean. It provides the same estimate as the Chain Ladder method, which is proved in Mack (1993). The probability function of the Poisson distribution is

$$f(y) = \frac{\mu^y}{y!}e^{-\mu}, \quad y = 0, 1, 2, ..., \mu > 0 \tag{2.36}$$

μ is the expected value of Y. The model for an incremental loss triangle is defined as follows: Incremental claims $Y_{i,j}$ are independent over-dispersed Poisson random variables, with

$$E(Y_{i,j}) = \mu_{i,j} \quad and \quad Var(Y_{i,j}) = \phi\mu_{i,j}, \tag{2.37}$$

where ϕ is a scalar.

The linear predictor is related to the mean $\mu_{i,j}$ by a log link function

$$\eta_{i,j} = log(\mu_{i,j}) = \alpha_i + \beta_j + \gamma, \tag{2.38}$$

with $\alpha_0 = \beta_0 = 0$.

This formula for the linear predictor can be rewritten as

$$\eta_{i,j} = \gamma X_0 + \beta^T C + \alpha^T L, \tag{2.39}$$

where

- γ is the intercept;
- X_0 is a vector of ones;
- α is the vector of α_i for $i \in \{1, ..., n\}$;
- β is the vector of β_j for $j \in \{1, ..., n\}$;

- C is the vector of c_j for $j \in \{1, ..., n\}$;
- L is the vector of l_i for $i \in \{1, ..., n\}$,

with $l_k = 1$ for $Y_{i,j}$ if $i = k$ (and 0 otherwise), and $c_k = 1$ for $Y_{i,j}$ if $j = k$ (and 0 otherwise).

Hence, the expected incremental claim and the corresponding variance is

$$E(Y_{i,j}) = \mu_{i,j} = e^{\alpha_i + \beta_j + \gamma}, \tag{2.40}$$

$$Var(Y_{i,j}) = \phi \mu_{i,j} = \phi e^{\alpha_i + \beta_j + \gamma} \tag{2.41}$$

2.4.4 Gamma model

The gamma model assumes that the incremental claims $Y_{i,j}$ are independent and follows gamma distribution $G(k, \theta_{i,j})$. Here k is the shape parameter and $\theta_{i,j}$ is the scale parameter.

The probability density function for a gamma random variable y is

$$f(y) = \frac{1}{\Gamma(k)\theta^k} y^{k-1} e^{-y/\theta}, \quad y > 0 \tag{2.42}$$

where k is the shape parameter and θ is the scale parameter.

$$E(Y) = k\theta, \quad Var(Y) = k\theta^2 \tag{2.43}$$

As pointed out by Wuthrich and Merz (2008), a log link $\eta_{i,j} = log(k\theta_{i,j})$ is a natural choice in loss reserving.

$$\eta_{i,j} = \alpha_i + \beta_j + \gamma, \tag{2.44}$$

with $\alpha_0 = \beta_0 = 0$.

$$E(Y_{i,j}) = k\theta_{i,j} \qquad Var(Y_{i,j}) = k\theta_{i,j}^2 \tag{2.45}$$

The expected incremental claim is

$$E(Y_{i,j}) = k\theta_{i,j} = e^{\alpha_i+\beta_j+\gamma} \tag{2.46}$$

2.4.5 Log likelihood function

The formula for the linear predictor η can be rewritten as:

$$\eta = x^T\beta \tag{2.47}$$

Here the parameter vector β is as follows

$$\beta = (\gamma, \alpha_1, ..., \alpha_n, \beta_1, ..., \beta_n)^T \tag{2.48}$$

x contains categorical variables accident year and development year. x can be written as

$$x = (1, a_1, ..., a_n, b_1, ..., b_n)^T, \tag{2.49}$$

where $a_i = 1$ if $i = k_1$ (and 0 otherwise), and $b_j = 1$ if $j = k_2$ (and 0 otherwise) for accident year k_1 and development year k_2.

With the above notation, we obtain the log-likelihood functions.

- Normal distribution

$$\ell(\beta) = \sum_{k=1}^{n}(-\frac{(y_k - x_k^T\beta)^2}{2\sigma^2} - \frac{1}{2}\ln\sigma^2 - \frac{1}{2}\ln(2\pi)) \tag{2.50}$$

- Poisson distribution

$$\ell(\beta) = \sum_{k=1}^{n} (y_k x_k^T \beta - exp(x_k^T \beta) - \ln y_k!) \tag{2.51}$$

- Gamma distribution

$$\ell(\beta) = \sum_{k=1}^{n} (-y_k \kappa x_k^T \beta + (\kappa - 1) \ln y_k - \ln(\Gamma(\kappa)) + \kappa \ln(\kappa x_k^T \beta)) \tag{2.52}$$

Here n is the number of the observed data. In this case, we have $n = I * (J + 1)/2$. y_k is the observed data in the upper triangle.

2.4.6 Estimating the dispersion parameter

After the estimation of β parameter, we can determine the means for each cell of the triangle. The estimated means are then used to compute the dispersion parameter. The dispersion parameter ϕ is usually estimated by the Pearson χ^2 statistic divided by a difference between the number of observations and the number of model parameters. The generalized Pearson χ^2 statistic is used for measuring the discrepancy of a fit and is defined by

$$\chi^2 = \sum_{k=1}^{n} \frac{(y_k - \hat{\mu}_k)^2}{V(\hat{\mu}_k)}, \tag{2.53}$$

where $V(\hat{\mu}_k) = \text{Var}(y_k)/\phi$ and $\hat{\mu}_k$ is the predicted mean.

The estimation for ϕ is

$$\phi = \frac{\chi^2}{n - p}, \tag{2.54}$$

where n is the number of observations and p is the number of model parameters. In this case, we have $n = I * (J + 1)/2$ and $p = I + J + 1$.

For normal distribution, we have $V(\mu_k) = 1$ and

$$\phi = \frac{\chi^2}{n-p} = \frac{1}{n-p}\sum_{k=1}^{n}\frac{(y_k - \hat{\mu}_k)^2}{V(\hat{\mu}_k)} = \frac{1}{n-p}\sum_{k=1}^{n}(y_k - \hat{\mu}_k)^2 \tag{2.55}$$

For Poisson distribution,$Var(y_k) = V(\mu_k)\phi = \mu_k\phi$

so $V(\mu_k) = \mu_k$ and

$$\phi = \frac{\chi^2}{n-p} = \frac{1}{n-p}\sum_{k=1}^{n}\frac{(y_k - \hat{\mu}_k)^2}{V(\hat{\mu}_k)} = \frac{1}{n-p}\sum_{k=1}^{n}\frac{(y_k - \hat{\mu}_k)^2}{\hat{\mu_k}} \tag{2.56}$$

For Gamma distribution,

$$Var(y_k) = V(\mu_k)\phi = \mu_k^2\phi \tag{2.57}$$

so $V(\mu_k) = \mu_k^2$ and

$$\phi = \frac{\chi^2}{n-p} = \frac{1}{n-p}\sum_{k=1}^{n}\frac{(y_k - \hat{\mu}_k)^2}{V(\hat{\mu}_k)} = \frac{1}{n-p}\sum_{k=1}^{n}\frac{(y_k - \hat{\mu}_k)^2}{\hat{\mu_k}^2} \tag{2.58}$$

2.5 Model selection

To evaluate the goodness-of-fit of the models, we can use statistical tests. For example, Kolmogorov-Smirnov test and Cramer-von Mises test can be used to assess the fit. A large p-value indicates the model fitting is reasonable.

The Akaike Information Criterion (Akaike, 1974) is used to choose between different types of models that seek an appropriate trade-off between goodness of fit and model complexity. It balance the goodness of fit of a model with a penalty term for the number of parameters p. AIC is defined as:

$$AIC = -2\ell + 2p, \tag{2.59}$$

where ℓ is the log-likelihood of the model. A good fit means a high value for

the likelihood, and therefore a low value for -2ℓ. Given a collection of models for the data, AIC estimates the quality of each model, relative to other models. The model with the lowest value of AIC is selected.

Bayesian Information Criterion introduced by Schwarz (1978) can also be used to choose models. Its definition is as following:

$$BIC = -2\ell + p * log(n), \tag{2.60}$$

where ℓ is the log-likelihood of the model, p the number of parameters and n is the number of records in the data. The model with the lowest value of BIC is selected.

2.6 Mean squared error of prediction

Mean squared error of prediction (MSEP) is used to measure the uncertainty of the estimated claims reserves. It includes the process variance (variability of any random variable) and the estimation or parameter variance (variability from the model parameter estimate).

First of all, we denote $\hat{R}$ as an estimation for the claim reserve R. Then it is possible to get the following decomposition:

$$\begin{aligned} MSEP(\hat{R}) &= E[(R-\hat{R})^2] \\ &= E[(R-E(R)+E(R)-\hat{R})^2] \\ &= E[(R-E(R))^2] + E[(\hat{R}-E(R))^2] \\ &= Var(R) + Var(\hat{R}) \end{aligned} \tag{2.61}$$

Here Var(R) is the process variance and Var($\hat{R}$) is the parameter variance. The covariance term is canceled out because of the assumption of independence between past observations and future predictions.

England and Verrall (1999) show the analytical expression of the mean squared error of prediction. Denoting the triangle of predicted claims contributing to the final claim reserve by Δ, the MSEP of the total reserve is given by

$$\begin{aligned} MSEP(\hat{R}) &= Var(R) + Var(\hat{R}) \\ &= \sum_{i,j\in\Delta} \phi\mu_{i,j}^{\rho} + \sum_{i,j\in\Delta} \mu_{i,j}^2 Var(\eta_{i,j}) + 2 \sum_{\substack{i_1j_1\in\Delta \\ i_2j_2\in\Delta \\ i_1j_1\neq i_2j_2}} \mu_{i_1j_1}\mu_{i_2j_2} Cov(\eta_{i_1j_1}, \eta_{i_2j_2}), \end{aligned} \tag{2.62}$$

where $\rho = 0$ for the normal model, $\rho = 1$ for the over-dispersed Poisson model and $\rho = 2$ for the gamma model.

The process variance $Var(R) = \sum_{i,j\in\Delta} \phi\mu_{i,j}^{\rho}$ can be calculated easily. The estimation variance $Var(\hat{R})$ is difficult to calculate and can be estimated with bootstrap simulation. Bootstrap method is used to generate a large number (say, N) of reserve estimates. The bootstrap standard error is the standard deviation of the N bootstrap reserve estimates. The bootstrap standard error is an estimate of the square root of the estimation variance. With the MSEP, we can obtain the confidence intervals for the reserve estimates. Based on Central Limit Theorem, we approximate the reserve by means of a normal distribution with expected value given by the predicted mean and the standard deviation given by the standard error of prediction.

2.7 Bootstrapping

The bootstrap technique is a resampling method which repeatedly resamples the original data and makes inferences from the resamples. For loss reserving, bootstrapping approach can be used to simulate the full predictive distribution of the reserves. The quantiles of the predictions can be computed based on the simulation results.

As in Efron and Tibshirani (1993) we let $\{y_1, y_2, ..., y_n\}$ be a random sample from an unknown distribution F with parameter θ. θ can be estimated as a function of the given random sample, i.e. $\hat{\theta} = s(y_1, ..., y_n)$. Given $\hat{\theta}$, the bootstrap sample $\{y_1^*, y_2^*, ..., y_n^*\}$ is then drawn from the distribution F. A bootstrap replication of $\hat{\theta}$ is given by $\hat{\theta}^* = s(y_1^*, y_2^*, ..., y_n^*)$.

The applications of the bootstrap method in the context of reserves were proposed in Lowe (1994) and Taylor et al. (2008). The bootstrap algorithm for loss triangles are as follows:

- Given the parameters estimated on the upper triangle data with GLM, we generate samples for the upper triangle.
- Estimate the parameters with the same method on this new upper triangle data.
- With these new estimated parameters, we generate predictions for the lower triangle.
- Repeat the above steps n times.

In the context of the claim reserving, the resampling is applied to the residuals of the model. To implement a bootstrap analysis we need to choose a model and an adequate residual. The resampling is based on the hypothesis that the residuals are independent and identically distributed. There is no difference to resample the residuals or the residuals multiplied by a constant, as long as we take that fact into consideration in the generation of the pseudo data. Our starting point will be the Pearson residuals defined by

$$r_{ij}^{(p)} = \frac{y_{ij} - \hat{\mu}_{ij}}{\sqrt{\hat{var}(Y_{ij})}} = \frac{y_{ij} - \hat{\mu_{ij}}}{\sqrt{\hat{\phi} V(\hat{\mu}_{ij})}} \tag{2.63}$$

Since ϕ is constant for the data set, we can use

$$r_{ij}^{(p*)} = \frac{y_{ij} - \hat{\mu}_{ij}}{\sqrt{V(\hat{\mu}_{ij})}} \tag{2.64}$$

instead of $r_{ij}^{(p)}$ in the bootstrap procedure. At this stage, the scale parameter is ignored. When using a normal model it is obvious that these residuals are equivalent to the classical residuals, $y_{ij} - \hat{\mu}_{ij}$, since $V(\mu_{ij}) = 1$.

As in the classical linear model (Efron and Tibshirani, 1993), it is more adequate to work with the standardized Pearson residuals. The standardized Pearson residuals can be considered as identically distributed. The well known standardized Pearson residuals are given by

$$r_{ij}^{(p**)} = \frac{r_{ij}^{(p)}}{\sqrt{1 - h_{ij}}}, \tag{2.65}$$

where the factor h_{ij} is the corresponding element of the diagonal of the hat matrix. For the classical linear model, this hat matrix is given by

$$H = X(X^T X)^{-1} X^T \tag{2.66}$$

and it can be generalized for a GLM using

$$H = X(X^T W X)^{-1} X^T W, \tag{2.67}$$

where W is a diagonal matrix with generic element given by

$$\left(V(\mu_{ij}) \left(\frac{\partial \eta_{ij}}{\partial \mu_{ij}} \right)^2 \right)^{-1}. \tag{2.68}$$

Chapter 3

Claim reserving using copulas

The reserves for different lines of business are often related. Shi and Frees (2011) propose a copula regression model which links the claims of different business lines. This model assumes that claims from different triangles with the same accident year and development period are dependent. The copula regression model can be estimated with maximum likelihood estimation. Dependence modeling allows analysts in risk capital analysis to quantify the diversification effect. We first replicate the results of Shi and Frees (2011) regarding the estimation of the claims reserve. Then we calculate the risk capital for different copula models.

3.1 Copula model

Assume that an insurance portfolio consists of N triangles. Define $Y_{ij}^{(n)}$ as the incremental claims in the ith accident year and jth development year. The superscript (n), $n \in \{1, ..., N\}$, indicates the nth run-off triangle. Assume that $Y_{ij}^{(n)}$ is from a parametric distribution:

$$F_{ij}^{(n)} = \text{Prob}(Y_{ij}^{(n)} \leq y_{ij}^{(n)}) = F^{(n)}(y_{ij}^{(n)}; \eta_{ij}^{(n)}, \gamma^{(n)}), \quad n = 1, ..., N. \tag{3.1}$$

Here $F^{(n)}(\cdot)$ is the distribution function for incremental losses from different

lines of business. The parameter $\eta_{ij}^{(n)}$ denotes the systematic component, which determines the location. $\eta_{ij}^{(n)}$ is a linear function of explanatory variables, that is $\eta_{ij}^{(n)} = x_{ij}^{(n)}\beta^{(n)}$. $x_{ij}^{(n)}$ includes the predictive variables in the nth triangle, and $\beta^{(n)}$ are the coefficients to be estimated. $\gamma^{(n)}$ summarizes the additional parameters in the distribution of $Y_{ij}^{(n)}$.

The copula model assumes that claims from different triangles with the same accident year and development period are dependent. This means that two different lines of business $Y_{ij}^{(1)}$ and $Y_{ij}^{(2)}$ are related for a given pair (i, j).

According to Sklar's theorem (Nelsen, 2006), a copula can uniquely represent the joint distribution of n random variables $Y^{(1)}, ..., Y^{(n)}$ as

$$F(y^{(1)}, ..., y^{(n)}) = C(F^{(1)}, ..., F^{(n)}; \theta), \tag{3.2}$$

where $C(\cdot; \theta)$ denotes the copula function with dependence parameter θ, and $F^{(1)}, ..., F^{(n)}$ are the marginal distribution functions of $Y^{(1)}, ..., Y^{(n)}$.

We consider a simple case when an insurance portfolio consists of two lines of business (N=2). The joint distribution of incremental claims $Y_{ij}^{(1)}, Y_{ij}^{(2)}$ can be uniquely represented by a copula function as

$$F_{ij}(y_{ij}^{(1)}, y_{ij}^{(2)}) = \text{Prob}(Y_{ij}^{(1)} \leq y_{ij}^{(1)}, Y_{ij}^{(2)} \leq y_{ij}^{(2)}) = C(F_{ij}^{(1)}, F_{ij}^{(2)}; \theta), \tag{3.3}$$

where $C(\cdot; \theta)$ is the copula function with parameter θ, and $F_{ij}^{(1)}$ and $F_{ij}^{(2)}$ are the marginal distribution functions.

Model 3.3 can be estimated using the likelihood based estimation method. Let $c(\cdot)$ denote the probability density function for the copula distribution function $C(\cdot)$. The log-likelihood function for the insurance portfolio is:

$$L = \sum_{i=1}^{I} \sum_{j=1}^{I-i+1} \log c(F_{ij}^{(1)}, F_{ij}^{(2)}; \theta) + \sum_{i=1}^{I} \sum_{j=1}^{I-i+1} (\log f_{ij}^{(1)} + \log f_{ij}^{(2)}), \tag{3.4}$$

where $f_{ij}^{(n)}$ denotes the density function of the marginal distribution $F_{ij}^{(n)}$ for $n = 1, 2$.

Maximizing the log-likelihood of model 3.3 results in estimates of the parameters, which are used to calculate the claim reserve. This changes the estimates of the parameters for each triangle compared to the parameter estimates of the independence model.

3.2 Reserve prediction

The dataset, which are from the Schedule P of the National Association of Insurance Commissioners database, is the same as the one that Shi and Frees (2011) use. The claim triangles consist of two lines of business, personal auto (Table 3.1) and commercial auto (Table 3.2). For the marginal distribution, we use the log-normal and the gamma distributions for personal and commercial lines of business (Shi and Frees, 2011). We consider the form $\eta_{ij}^{(n)} = \mu_{ij}^{(n)}$ for a log-normal distribution with location parameter $\mu_{ij}^{(n)}$ and shape parameter $\sigma^{(n)}$. For a gamma distribution with location parameter $\mu_{ij}^{(n)}$ and shape parameter $\phi^{(n)}$, we use the exponential link $\mu_{ij}^{(n)} = \frac{\exp(\eta_{ij}^{(n)})}{\phi^{(n)}}$, which is the same as Abdallah et al. (2015).

Table 3.1: Incremental paid losses for personal auto line

year	premium	1	2	3	4	5	6	7	8	9	10
1988	4711333	1376384	1211168	535883	313790	168142	79972	39235	15030	10865	4086
1989	5335525	1576278	1437150	652445	342694	188799	76956	35042	17089	12507	
1990	5947504	1763277	1540231	678959	364199	177108	78169	47391	25288		
1991	6354197	1779698	1498531	661401	321434	162578	84581	53449			
1992	6738172	1843224	1573604	613095	299473	176842	106296				
1993	7079444	1962385	1520298	581932	347434	238375					
1994	7254832	2033371	1430541	633500	432257						
1995	7739379	2072061	1458541	727098							
1996	8154065	2210754	1517501								
1997	8435918	2206886									

We first use the Gaussian copula to model dependence between two lines of

Table 3.2: Incremental paid losses for commercial auto line

year	premium	1	2	3	4	5	6	7	8	9	10
1988	267666	33810	45318	46549	35206	23360	12502	6602	3373	2373	778
1989	274526	37663	51771	40998	29496	12669	11204	5785	4220	1910	
1990	268161	40630	56318	56182	32473	15828	8409	7120	1125		
1991	276821	40475	49697	39313	24044	13156	12595	2908			
1992	270214	37127	50983	34154	25455	19421	5728				
1993	280568	41125	53302	40289	39912	6650					
1994	344915	57515	67881	86734	18109						
1995	371139	61553	132208	20923							
1996	323753	112103	33250								
1997	221448	37554									

business. The two-dimensional Gaussian copula is:

$$C(u_1, u_2) = \Phi_\Sigma(\Phi^{-1}(u_1), \Phi^{-1}(u_2)), \quad u_1, u_2 \in (0, 1) \tag{3.5}$$

where Σ is the 2×2 matrix with 1 on the diagonal and ρ otherwise. ρ is the linear correlation between $\Phi^{-1}(u_1)$ and $\Phi^{-1}(u_2)$. Φ denotes the cumulative distribution function (cdf) of a standard normal distribution while Φ_Σ is the cdf for a bivariate normal distribution with zero mean and covariance matrix Σ. The copula can also be written as

$$C(u_1, u_2) = \int_{-\infty}^{\Phi^{-1}(u_1)} \int_{-\infty}^{\Phi^{-1}(u_2)} \frac{1}{2\pi\sqrt{1-\rho^2}} \exp\left(-\frac{s_1^2 - 2\rho s_1 s_2 + s_2^2}{2\left(1-\rho^2\right)}\right) ds_1 ds_2 \tag{3.6}$$

The bivariate Frank copula is also used to model the dependence. It is defined as

$$C(u_1, u_2) = \frac{1}{\theta} \log\left(1 + \frac{\left(e^{-\theta u_1} - 1\right)\left(e^{-\theta u_2} - 1\right)}{e^{-\theta} - 1}\right), \quad u_1, u_2 \in (0, 1) \tag{3.7}$$

We use the gjrm function from the R package GJRM (Marra and Radice, 2019) to construct the copula. The reserve estimations for the independence model and the copula model are shown in Table 3.3. The corresponding parameter estimates are shown in Table 3.4. The Log-Likelihood and AIC are provided in Table 3.5.

Table 3.3: Estimation of the reserve for different models

Model	Reserves		
	LoB 1	LoB 2	Total
Product Copula	6 464 083	490 653	6 954 736
Gaussian Copula	6 423 246	495 925	6 919 171
Frank Copula	6 511 360	487 893	6 999 253
Student's t Copula	6 800 554	554 426	7 354 980

Table 3.4: Parameter estimates for different models

	Independence		Gaussian		Frank		Student t	
	LoB 1	LoB 2	LoB 1	LoB 2	LoB 1	LoB 2	LoB 1	LoB 2
Intercept	-1.1367	-1.6703	-1.1311	-1.6585	-1.1404	-1.6137	-1.1334	-1.7434
AY 1989	-0.0327	-0.1286	-0.0319	-0.135	-0.0338	-0.1378	-0.0314	-0.0698
AY 1990	-0.0284	-0.1416	-0.0295	-0.1505	-0.0098	-0.2239	-0.0452	-0.0285
AY 1991	-0.1309	-0.289	-0.131	-0.3009	-0.1204	-0.3531	-0.1473	-0.1920
AY 1992	-0.1747	-0.2722	-0.1749	-0.2866	-0.1685	-0.3355	-0.1935	-0.1373
AY 1993	-0.1745	-0.252	-0.1768	-0.2711	-0.158	-0.3516	-0.2134	-0.0714
AY 1994	-0.1729	-0.1242	-0.1771	-0.1446	-0.1629	-0.2243	-0.2174	0.0093
AY 1995	-0.2234	-0.0891	-0.2365	-0.1001	-0.2224	-0.1317	-0.1424	-0.1531
AY 1996	-0.2444	0.1348	-0.265	0.1676	-0.2435	0.1206	-0.1224	0.3744
AY 1997	-0.2042	-0.1041	-0.2064	-0.1159	-0.1942	-0.1618	-0.2053	-0.0207
Dev 2	-0.2244	0.1955	-0.227	0.2004	-0.2261	0.2414	-0.1863	0.2013
Dev 3	-1.0469	-0.0227	-1.0516	-0.0201	-1.0481	-0.006	-1.0178	-0.0198
Dev 4	-1.6441	-0.4088	-1.6479	-0.4119	-1.6374	-0.4169	-1.6488	-0.3781
Dev 5	-2.254	-1.0477	-2.2592	-1.0558	-2.2458	-1.0979	-2.2749	-0.9663
Dev 6	-3.013	-1.4632	-3.0171	-1.4691	-3.0006	-1.4984	-3.0094	-1.2971
Dev 7	-3.6713	-2.0885	-3.6754	-2.0994	-3.6544	-2.1342	-3.6886	-1.8934
Dev 8	-4.4935	-2.7831	-4.5022	-2.809	-4.4831	-2.8982	-4.5389	-2.6214
Dev 9	-4.9109	-3.1106	-4.9136	-3.1198	-4.9006	-3.164	-4.9122	-3.0569
Dev 10	-5.9134	-4.1705	-5.9157	-4.1823	-5.9034	-4.2282	-5.9146	-4.0870
Scale/Shape	0.0887	10.0925	0.0888	10.0793	0.0908	9.9170	0.0865	11.2900

Table 3.5: Fit statistics for different models

	Copula			
Fit statistics	Independence	Gaussian	Frank	Student's t
Dependence Parameter	.	-0.3656	-2.7977	-0.2657
Log-Likelihood	346.6	350.4	350.3	355.4
AIC	-613.2	-618.9	-618.5	-626.9

3.3 Parametric bootstrap for the copula model

In practice, reserving actuaries are also interested in reserve ranges. Here we describe a simulation method and a parametric bootstrap which are used to generate the predictive distribution of the reserve.

The simulation method in this section is equivalent to the ones described in Shi

and Frees (2011). The simulation procedure is described by the following steps:

- Simulate realizations $u_{ij}^{(n)}(n = 1, ..., N)$ from the corresponding copula regression model for all accident years i and development period j which satisfy $i + j - 1 > I$.

- Transform these simulations to predictions of the lower triangles by $y_{ij}^{(n)} = F^{(n)(-1)}(u_{ij}^{(n)}; \hat{\eta}_{ij}^{(n)}, \hat{\gamma}^{(n)})$. Here $\hat{\eta}_{ij}^{(n)}$ and $\hat{\gamma}^{(n)}$ denote the estimates of the linear predictor and the additional parameters.

- Obtain a prediction of the outstanding claim reserve for the entire insurance portfolio by

$$\sum_{n=1}^{N} \sum_{i=2}^{I} \sum_{j=I-i+2}^{I} \omega_i^{(n)} y_{ij}^{(n)}. \tag{3.8}$$

  Here $\omega_i^{(n)}$ denotes the exposure for the ith accident year in the nth triangle.

Repeat the above steps for the total number of simulations to obtain a predictive distribution of the reserve.

Shi and Frees (2011) expand the above simulation method to include parameter uncertainty by using a bootstrapping approach. In this approach, we generate a new upper triangle for each simulation and fit the corresponding copula regression model to this new upper triangle. Thus, this approach uses different parameters for each simulation. The predictive distribution generated by bootstrapping includes parameter uncertainty. This parametric bootstrap (Shi and Frees, 2011) is described as follows:

- Simulate realizations $u_{ij}^{(n)}(n = 1, ..., N)$ from the corresponding copula regression model for all accident years i and development period j which satisfy $i + j - 1 \leq I$.

- Transform these simulations to estimates of the upper triangles by $y_{ij}^{*(n)} = F^{(n)(-1)}(u_{ij}^{(n)}; \hat{\eta}_{ij}^{(n)}, \hat{\gamma}^{(n)})$. Here $\hat{\eta}_{ij}^{(n)}$ and $\hat{\gamma}^{(n)}$ denote the estimates of the linear

predictor and the additional parameters.

- Obtain estimates of the linear predictors $\hat{\eta}_{ij}^{*(n)}$, the additional parameters $\hat{\gamma}^{*(n)}$ and dependence parameter $\hat{\theta}^*$ by maximizing the log-likelihood of the corresponding copula regression model for the created $y_{ij}^{*(n)}$.

- Using these new parameter estimates, simulate the lower triangle by performing the steps of the simulation procedure.

Repeat the above steps for the total number of simulations to obtain a predictive distribution of the reserve.

Based on the techniques described above, a predictive distribution could be generated for the reserve. In Table 3.6, we show the bootstrapping reserve, bias and standard errors for different models.

Table 3.6: Bootstrap bias for different models

	Reserve	Bootstrap reserve	Bias	Std Error
Product Copula	6 954 736	6 994 653	0.57%	401 431
Gaussian Copula	6 919 171	6 953 469	0.50%	377 904
Frank Copula	6 999 253	7 044 507	0.65%	395 374
Student's t Copula	7 354 980	7 387 915	0.45%	402 437

In table 3.7, we also report the confidence intervals for the predicted reserves. The lower and upper bounds are calculated using the 5th and 95th percentile of the predictive distribution, respectively.

Table 3.7: Confidence intervals for different models

	lower bound	upper bound
Product Copula	6 342 244	7 660 710
Gaussian Copula	6 344 747	7 589 519
Frank Copula	6 448 667	7 700 954
Student t Copula	6 767 624	8 075 700

Insurance companies are also interested in other risk measures such as the Value-at-Risk (VaR) and the Tail Value-at-Risk (TVaR) (Côté et al., 2016). One

way to calculate the risk measure for the entire portfolio (i.e. the personal auto line and the commercial auto line) is to use the simple sum of the risk measures for each subportfolio. This is the result under the silo method. We also report the risk measures calculated from the product copula and the Gaussian copula models. The risk measures are compared in Table 3.8 for different models. The risk measures from both copula models are smaller than the silo method. This is explained by the negative association between the two lines of business. The silo method assumes a perfect positive linear relationship between subportfolios. The Gaussian copula model allows diversification effect in the portfolio.

The predictive distribution of unpaid losses is also helpful to obtain risk capital, which is the amount that insurance companies set aside as a buffer against potential losses from extreme events. It can be calculated as the difference between risk measure and the expected unpaid losses of the portfolio. For the risk measure, we use the TVaR which is more informative than VaR. In practice, the risk capital is calculated as the difference between the risk measure and the reserve. The reserve is usually set at a risk tolerance between 60% and 80%, according to the risk appetite. Here we set the risk tolerance at 60% for the reserve in our analysis. See (Abdallah et al., 2016) for further details on risk capital analysis. Mathematically, the risk capital is calculated as follows:

$$\text{Risk capital} = \text{TVaR} - \text{TVaR}_{60\%}. \tag{3.9}$$

Table 3.8 shows the risk capital for the silo method, product copula model, Gaussian copula model and Student's t copula model. The silo method gives more conservative risk capital, while the Gaussian copula model results in more aggressive risk capital. We show in Table 3.9 that the gain in terms of risk capital is important when we capture the association between the two lines of business. The Gaussian copula model is the best in terms of risk capital gain.

Table 3.8: Risk capital estimation for different methods

Risk measure	TVaR (60%)	TVaR (80%)	TVaR (85%)	TVaR (90%)	TVaR (95%)	TVaR(99%)
Silo	7 437 089	7 665 227	7 747 424	7 856 726	8 029 590	8 400 998
Product copula	7 366 110	7 556 391	7 623 324	7 711 546	7 849 315	8 156 210
Gaussian copula	7 309 225	7 486 336	7 553 536	7 640 866	7 779 089	8 081 251
Frank copula	7 412 858	7 597 721	7 663 806	7 752 307	7 895 450	8 203 132
Student's t copula	7 777 369	7 969 783	8 040 971	8 136 747	8 288 298	8 628 209
Risk capital						
Silo		228 138	310 335	419 637	592 501	963 909
Product copula		190 281	257 214	345 436	483 205	790 100
Gaussian copula		177 111	244 311	331 641	469 864	772 026
Frank copula		184 863	250 948	339 449	482 592	790 274
Student's t copula		192 414	263 602	359 378	510 929	850 840

Table 3.9: Risk capital gain for different methods

Risk capital gain	TVaR (80%)	TVaR (85%)	TVaR (90%)	TVaR (95%)	TVaR(99%)
Product copula vs Silo	16.59%	17.12%	17.68%	18.45%	18.03%
Frank copula vs Silo	18.97%	19.14%	19.11%	18.55%	18.01%
Gaussian copula vs Silo	22.37%	21.28%	20.97%	20.70%	19.91%
Student's t copula vs Silo	15.66%	15.06%	14.36%	13.77%	11.73%

Chapter 4

Overview of neural networks

Artificial Neural Networks (ANNs), sometimes referred to as multilayer perceptrons (MLPs), have become increasingly popular tool in machine learning. The ideas for ANNs were first introduced in McCulloch and Pitts (1943), and the application of backpropagation came in the 1980s, see Werbos (1975) and Rumelhart et al. (1986). Recent advancements in processor speed and memory have enabled widespread use of these models in a diverse set of fields, including medical care, autonomous navigation, and marketing analytics. Neural Networks can be used for discrete classification and continuous value prediction.

4.1 Universal approximation theorem

Let $L \in \mathbb{N}$ and the tuple $(N_1, N_2, ..., N_L) \in \mathbb{N}^L$ denote the number of layers and the number of nodes (neurons) on each layer respectively. Furthermore, we introduce the function

$$\begin{aligned} z^l \quad : \quad & \mathbb{R}^{N_l} \mapsto \mathbb{R}^{N_{l+1}} \quad for \quad 1 \leq l \leq L-1 \\ & x \mapsto w^{l+1}x + b^{l+1} \end{aligned} \tag{4.1}$$

acting between layers. Each entry $w_{i,j}^{l+1}$ denotes the weight connecting node

$i \in N_l$ of layer l with node $j \in N_{l+1}$ of layer $l+1$. b^{l+1} is the so called bias term.

A neural network $F : \mathbb{R}^{N_0} \mapsto \mathbb{R}^{N_L}$ is defined as the composition:

$$F = F_L \circ \cdots \circ F_1 \tag{4.2}$$

where each component is of the form $F_l = \sigma_l \circ z^l$. The function σ_l is the so called activation function. The activation function is typically nonlinear and applied component wise on z^l. The first layer F_1 is the input layer and the last layer F_L is the output layer. $F_2, ..., F_{L-1}$ are the hidden layers.

The following central result of Hornik et al. (1989) justifies the use of neural networks as approximators for any continuous functions.

Theorem 2 (Universal approximation theorem)**.** *Let $\mathcal{NN}^{\sigma}_{d_0,d_1}$ be the set of neural networks with activation function $\sigma : \mathbb{R} \mapsto \mathbb{R}$, input dimension $d_0 \in \mathbb{N}$ and output dimension $d_1 \in \mathbb{N}$. Then, if σ is continuous and non-constant, $\mathcal{NN}^{\sigma}_{d_0,d_1}$ is dense in $L^p(\mu)$ for all finite measures μ.*

Basically, this theorem states that neural networks allow us to approximate any continuous function arbitrarily well if we allow for arbitrarily many neurons N_l in the hidden layer.

4.2 Single layer: the perceptron

The simplest neural network is also known as the perceptron. It contains a single input layer and an output node. The basic structure of the perceptron is shown in Figure 4.1 . The input layer has d neurons which transmit the d features $(x_1, ..., x_d)$ to an output neuron. The output layer applies weight $(w_1, ..., w_d)$ to the input data and generates the linear combination $\sum_{i=1}^{d} w_i x_i$. The sign of this real value is used to make prediction. The prediction $\hat{y}$ is computed as follows:

$$\hat{y} = sign\{\sum_{i=1}^{d} w_i x_i\} \tag{4.3}$$

In many cases, there is a bias component of the prediction. For example, when the feature variables are mean centered, the mean of the binary class prediction from $\{-1, 1\}$ is not 0. This is the case when the binary class distribution is highly imbalanced. It is necessary to include a bias variable b:

$$\hat{y} = sign\{\sum_{i=1}^{d} w_i x_i + b\} \tag{4.4}$$

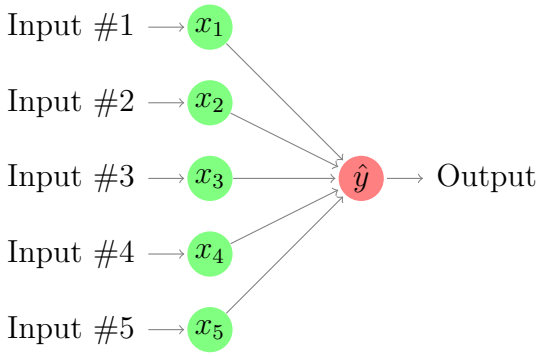

Figure 4.1: Perceptron

The perceptron algorithm was proposed by Rosenblatt (1958) and the optimizations were performed in a heuristic way. It performs well at classifying data sets which are linearly separable. However, it performs poorly on data sets when the data are not linearly separable.

4.3 Activation functions

Activation function (Figure 4.2) is a critical component for a neural network. Different types of functions such as linear, sigmoid, softmax, or Rectified linear unit (ReLU) are used in various layers. A neuron computes two functions, which are weighted summation and the activation. The value computed before applying the activation function is used in the computation of the backpropagation.

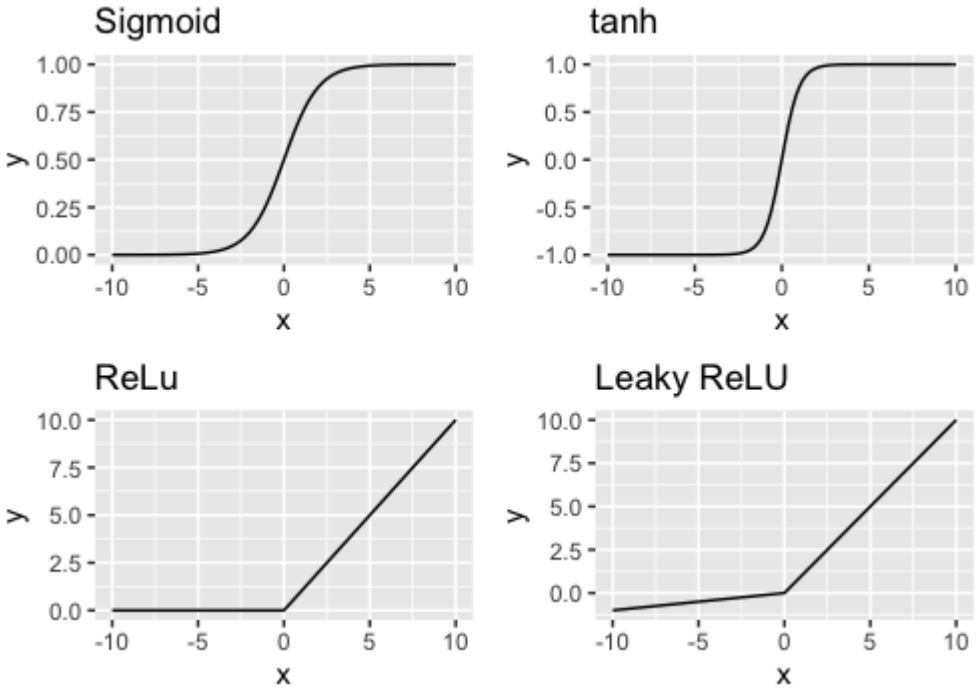

Figure 4.2: Activation function

- The most basic activation function is the linear or identity activation, which provides no nonlinearity. The identity activation is often used in the output node, when the target value is a real number.

- Rectified linear unit or ReLU ranges from 0 to infinity. All the negative values are converted to zero. In other words, it computes the function

$$f(x) = max(0, x). \tag{4.5}$$

  It was found to accelerate the convergence of stochastic gradient descent compared to the sigmoid functions. Compared to the sigmoid neurons which need expensive operations, the ReLU is simply an activation at zero. Neural networks with ReLU tend to learn several times faster than saturating activation functions, such as sigmoid function. However, the ReLU can be fragile and the corresponding units will be zero forever if the learning rate is set too high.

- Compared to the ReLU function, a leaky ReLU will have a small negative slope when $x < 0$. The corresponding function is

$$f(x) = 1(x < 0)(\alpha x) + 1(x >= 0)(x), \tag{4.6}$$

where α is a small constant. Because of the small scaling for the negative values, the gradient would not saturate. The apparent advantage of Leaky ReLU is that it does not saturate in the positive or negative region.

- The sigmoid activation function is a probabilistic approach and ranges in between 0 and 1. Since the range is the minimum and the prediction would be more accurate. The equation for the sigmoid function is

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{4.7}$$

A very undesired property is that the gradient is almost zero when the neuron's activation saturates at either 0 or 1. If the initial weights are too large then the neurons would become saturated and the network may not be able to learn.

4.4 Neural network structure

Feedforward neural network (Figure 4.3) is designed to approximate some function f^*. A feedforward network defines a mapping $y = f(x; \theta)$ and learns the value of the parameter θ which results in the best approximation to f^*. Information flows through the input layer and hidden layers, and finally to the output. There are no feedback connections in which outputs are fed back into the model. The hidden layer consists of many units that act in parallel. Each unit receives input from many other units and computes its own activation value. A neuron can be thought

of as a unit that accepts a number of inputs, combines them and generates an output that is sent to one or more further neurons. Neurons are not all the same. They come in a variety of different types. Neurons can differ in the number and type of their inputs and outputs (Newman, 2018).

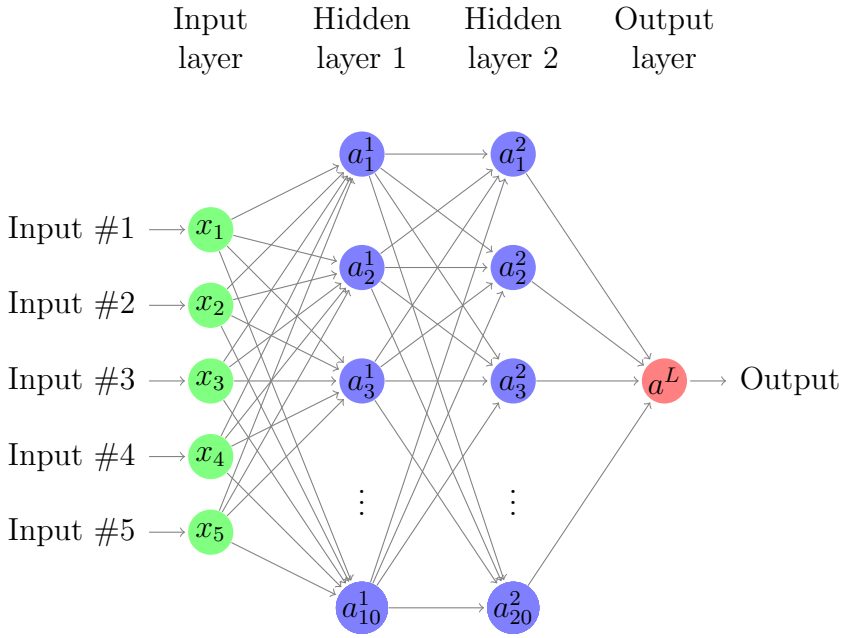

Figure 4.3: A feed forward neural network with two hidden layers

The number of input neurons is corresponding to the number of input features. For regression problems with one real valued output, there is one single node in the output layer. There is one node per class in the output layer for classification problems with multiple classes. Each neuron takes an input vector from the previous layer a^{l-1} and applies a weight vector w_j^l and a bias b_j^l so that

$$z_j^{l,t} = w_j^{l,t} a^{l-1,t} + b_j^{l,t} \tag{4.8}$$

or in matrix notation

$$z^{l,t} = w^{l,t} a^{l-1,t} + b^{l,t} \tag{4.9}$$

The result from layer l is then given by applying a non-linear activation function

$g(z)$,

$$a^{l,t} = g(z^{l,t}) \tag{4.10}$$

The calculation for the first hidden layer is as follows

$$a_j^1 = g(w_j^1 x + b_j^1) \tag{4.11}$$

For the output layer $l = L$, we calculate the prediction

$$\hat{y} = a^L = g(w_j^L a^{L-1} + b_j^L) \tag{4.12}$$

The weights and bias parameters are updated to minimize the error between the training data and the generated estimates. Let m denote the number of training samples and C denotes the cost function which measures the error, and we have

$$C^t(w^l, b^l) = \sum_i^m \mathcal{L}(\hat{y}_i, y_i), \tag{4.13}$$

where the function $\mathcal{L}$ is a measure of loss. The choices of the loss measure can be the L2 or L1 norm. Here $\hat{y}_i$ is the output vector from the neural network for the ith training sample.

For example, the quadratic cost for a single training sample can be written as

$$C^t = \frac{1}{2} \sum_j (y_{i,j} - a_{i,j}^L)^2 \tag{4.14}$$

The number of layers in the model, L and the number of neurons for each layer are considered as hyperparameters. In general, their optimal values are not known a priori. These parameters can be optimized using hyperparameter tuning over a portion of the training data set. Models with many hidden layers are known as deep neural network.

4.5 The back propagation algorithm

An optimization procedure is used to find the minimum of the cost function. Gradient descent methods can be used for this training process. These procedures will require the calculation of the gradient of the cost function. The goal of back-propagation is to compute the partial derivatives $\partial C/\partial w$ and $\partial C/\partial b$ of the cost function (Nielsen, 2015).

Let's first consider the values sent to the current layer before the activation function was applied.

$$z_j^{l,t} = \sum_k w_{j,k}^{l,t} a_k^{l-1,t} + b_j^{l,t} \tag{4.15}$$

Consider the quantity

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} \tag{4.16}$$

With chain rule, we have

$$\frac{\partial C}{\partial w_{j,k}^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{j,k}^l} = \delta_j^l a_k^{l-1} \tag{4.17}$$

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l} = \delta_j^l \tag{4.18}$$

a_i^{l-1} is available for all the neurons in the network. To calculate the gradients, we just need to obtain the value of the δ_j^l's.

We start with the last layer L of the network. By chain rule, it is easy to notice that

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L} g'(z_j^L) \tag{4.19}$$

With the Mean Square Error function, we get

$$\delta_j^L = (a_j^L - y_j)g'(z_j^L) \tag{4.20}$$

The above result can be written in vector form as follows:

$$\Delta^L = \nabla_{a^L} C \odot g'(z^L), \tag{4.21}$$

where $\nabla_{a^L} C = (\frac{\partial C}{\partial a_1^L}, ..., \frac{\partial C}{\partial a_k^L})$ is the gradient of C taken with respect to the elements of a^L and $\odot$ is the Hadamard Product or Schur product.

For two matrices A and B of the same dimension, the elements of the Hadamard product $A \odot B$ are given by

$$(A \odot B)_{ij} = (A)_{ij}(B)_{ij}$$

We can propagate this backwards in the network to obtain δ_j^{L-1}.

$$\begin{aligned} \delta_j^{L-1} &= \frac{\partial C}{\partial z_j^{L-1}} && (4.22) \\ &= \sum_i^k \frac{\partial C}{\partial z_i^L} \frac{\partial z_i^L}{\partial z_j^{L-1}} && (4.23) \\ &= \sum_i^k \delta_i^L \frac{\partial z_i^L}{\partial z_j^{L-1}} && (4.24) \end{aligned}$$

In order to calculate the term $\frac{\partial z_i^L}{\partial z_j^{L-1}}$, we have

$$\frac{\partial z_i^L}{\partial z_j^{L-1}} = \frac{\partial(\sum_k w_{i,k}^L a_k^{L-1} + b_i^L)}{\partial z_j^{L-1}} \tag{4.25}$$

$$= \frac{\partial(\sum_k w_{i,k}^L g(z_k^{L-1}) + b_i^L)}{\partial z_j^{L-1}} \tag{4.26}$$

$$= \frac{\partial(w_{i,j}^L g(z_j^{L-1}))}{\partial z_j^{L-1}} \tag{4.27}$$

$$= w_{i,j}^L g'(z_j^{L-1}) \tag{4.28}$$

Therefore,

$$\delta_j^{L-1} = \sum_i^k \delta_i^L \frac{\partial z_i^L}{\partial z_j^{L-1}} \tag{4.29}$$

$$= \sum_i^k \delta_i^L w_{i,j}^L g'(z_j^{L-1}) \tag{4.30}$$

This formula shows that we can calculate any δ_j^l in the network, assuming we know δ_i^{l+1}. Propagating this method backwards through the layers of the network, we are able to find all the partial derivatives. The derivatives of cost function C with respect to the weights and biases of the network can be calculated.

The algorithm for back propagation is as follows:

- Input: Set the corresponding activation a^1 for the input layer.
- Feedforward: For each $l = 2, 3, ..., L$ compute $z^{l,t} = w^{l,t}a^{l-1,t} + b^{l,t}$ and $a^{l,t} = g(z^{l,t})$
- Output δ^L: Compute $\delta_j^L = (a_j^L - y_j)g'(z_j^L)$.
- Backpropagation: For each $l = L-1, L-2, ..., 2$ compute $\delta_j^l = \sum_i^k \delta_i^{l+1} w_{i,j}^{l+1} g'(z_j^l)$.
- Output: The gradient of the cost function is given by $\frac{\partial C}{\partial w_{j,k}^l} = \delta_j^l a_k^{l-1}$ and $\frac{\partial C}{\partial b_j^l} = \delta_j^l$.

4.6 Training the model

The weights and bias are updated as

$$w^{l,t+1} = w^{l,t} - \alpha \frac{1}{m} \sum_{i}^{m} \frac{\partial C(y_i)}{\partial w^l} \tag{4.31}$$

$$b^{l,t+1} = b^{l,t} - \alpha \frac{1}{m} \sum_{i}^{m} \frac{\partial C(y_i)}{\partial b^l}. \tag{4.32}$$

This means we need a term that sums up all the partial derivatives of the individual sample cost function calculations. Let Δw^l and Δb^l denote these summing up terms. Then the updates can also be written as

$$w^{l,t+1} = w^{l,t} - \alpha[\frac{1}{m} \Delta w^l] \tag{4.33}$$

$$b^{l,t+1} = b^{l,t} - \alpha[\frac{1}{m} \Delta b^l] \tag{4.34}$$

The algorithm for updating the weights and bias is summarized below:

Randomly initialise the weights for each layer w^l

While iterations $<$ iteration limit:

1. Set Δw and Δb to zero
2. For samples 1 to m:
   (a) Perform a feed foward pass through all the L layers. Store the activation function outputs $h^{(\ell)}$
   (b) Use backpropagation to calculate the gradient values for layers 2 to $L - 1$.
   (c) update the Δw^l and Δb^l for each layer
3. Perform a gradient descent step

4.7 A simple network to predict digits

In this section we will implement a neural network that learns to recognize handwritten digits. For this example, a simple neural network is used to perform training and prediction on the MNIST dataset. It consists of images of handwritten digits with associated labels that tell us what the digit is. Each image is 28x28 pixels in size and the image data sample is represented by 784 data points which denote the pixel intensity.

The input data ranges from 0 to 255. It's standard practice to scale the input data. It can help the convergence of the neural network and is especially important if combining different data types. The amount of training data is always more numerous than the testing data, and is usually between 60-80% of the total dataset. We've made test set to be 20% of the total data, leaving 80% to train with.

The output layer needs 10 nodes to predict whether the digits represented by the input pixels is between 0 and 9. When an image of the digit 5 is presented to the neural network, the neuron in the output layer representing 5 has the highest value. Ideally, the output looks like: (0. 0. 0. 0. 0. 1. 0. 0. 0. 0.). In reality, we can see something like this: (0.01 0.1 0.2 0.05 0.3 0.8 0.4 0.03 0.25 0.02)

For the input layer, we need 784 nodes to cover 784 pixels in the image. We need hidden layers to allow for the complexity of the task. The structure of the network $[784, 256, 10]$ The sigmoid activation function will be used. The source code for training the model and making predictions are included in Appendix.

We run the model for 100 epochs on the training data set. During training, we use 20 percent of the training data as a validation data. In Figure 4.4 we plot the decrease of training loss and validation loss. After 84 epochs, the validation loss starts to increase and the training stops. This trained model is used to predict digit labels on the test data.

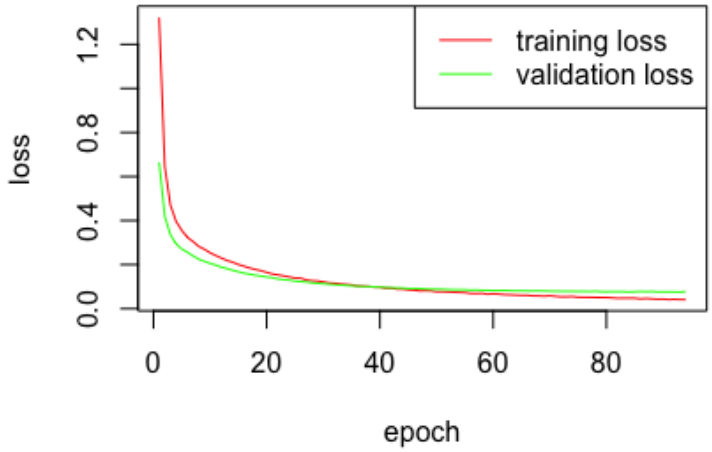

Figure 4.4: Performance of the simple model, the red line shows the training loss and the green line graphs the validation loss.

4.8 Stochastic gradient descent

In gradient descent, we are trying to minimize the cost function of the model by moving the model parameters along the negative direction of the gradient. One need to compute the cost function for all the training data simultaneously and calculate the gradients. For large neural networks, the number of parameters can be on the order of millions, and one needs to simultaneously run all samples forwards and backwards to calculate the backpropagation updates. When we first initialize the weights, they are far from the true values. Even a small number of samples can provide an accurate direction of movement. The gradient is calculated as an expectation, which may be approximated with a small set of samples (Goodfellow et al., 2016).

For the stochastic gradient descent, training process involves randomly picking out a small number n of randomly chosen training inputs. These random training inputs $x_1, x_2, ..., x_n$ are referred as a mini-batch. Given that the sample size n is large enough, the average of the gradient will be roughly equal to the average over

all training data,

$$\frac{\sum_{j=1}^{n} \nabla C}{n} \approx \frac{\sum_{i=1}^{m} \nabla C}{m},$$

where the second sum is over the entire set of training data.

The overall gradient can be estimated by calculating gradient for the randomly chosen mini-batch. The weights and biases in our neural network can be updated with the chosen mini-batch.

$$w^{l,t+1} = w^{l,t} - \alpha \frac{1}{n} \sum_{i}^{n} \frac{\partial C(y_i)}{\partial w^l} \tag{4.35}$$

$$b^{l,t+1} = b^{l,t} - \alpha \frac{1}{n} \sum_{i}^{n} \frac{\partial C(y_i)}{\partial b^l}, \tag{4.36}$$

where the sums are over all the training samples in the current mini-batch. Then we train another randomly chosen mini-batch. An epoch of training is completed after we exhausted all the training inputs. Then we start another epoch training.

The size of the mini batch is regulated by the amount of memory available on the hardware. A small batch size is not efficient in terms of computational time. A very large batch size may not lead to sufficient accuracy of gradient computation. One can use powers of 2 for the mini batch size, because this design can provide the best accuracy on most hardware. For example, the size can be set to 32, 64, 128, or 256.

We can use early stopping to determine the number of training epochs. At the end of each epoch, we compute the value of the loss function on the validation data. When the error on the validation data begins to rise, the training is ended. Early stopping automatically prevents us from overfitting.

4.9 Choice of loss function

Consider a neural network whose input is x, and its output is $y = f_{w,b}(x)$. $f_{w,b}$ is the input-output mapping of the network, which depends on the weight w and bias b. $\hat{y}$ is the predicted value. The error function measures a proximity between the predicted output $\hat{y}$ and the target y. The mean squared error is usually used for regression problem, while the cross-entropy is often useful in classification problems.

- The mean squared error can be written as the expectation of their squared difference

$$C(w,b) = E[(y-\hat{y})^2] \tag{4.37}$$

We search for the pair (w,b) so that it reaches the minimum of the cost function. In other words, we are computing $arg\ min\ C(w,b)$. This can be achieved with steep descent method.

If the measurements of random variables (x,y) are given by (x_1,y_1), (x_2,y_2), ... , (x_n,y_n), then the cost function is defined by the average

$$C(w,b) = \frac{1}{n}\sum_{j=1}^{n}(y_j-\hat{y}_j)^2, \tag{4.38}$$

which can be considered as the empirical mean of the squared difference between y and $\hat{y}$.

- The cross entropy involves two distributions, $p(x)$, the true distribution, and $q(x)$, the estimated distribution. It is summed over the variable x and is defined as

$$H(p,q) = -\sum_{x} p(x)log(q(x)) \tag{4.39}$$

For a single sample, the neural network cross entropy cost function is as follows

$$C(w, b) = -y \cdot log(\hat{y}), \tag{4.40}$$

where $\cdot$ is the inner product.

Chapter 5

Deep triangle

Machine Learning techniques are widely used in the field of data analytics. Different neural network modelling approaches have been developed for loss reserving. More recently, Kuo (2019) proposed to apply machine learning techniques on a large triangle dataset of multilpe P&C companies. The paper (Kuo, 2019) introduces a deep learning approach for loss reserving. This approach can produce forecasts by jointly modeling paid losses and claims outstanding.

5.1 Introduction

The main goal of the DeepTriangle model is to predict future incremental paid losses. It can also predict the outstanding claims at each point. For input to the DeepTriangle model, the incremental paid losses and claims outstandings are normalized by the earned premium. To fit the model, the data as of calendar year 1997 is used for training. The model is trained for a maximum of 1000 epochs with early stopping scheme. Then the trained model is used to predict the loss amounts after calendar year 1997. At each accident year and development year pair, the model predicts future incremental paid loss and claims outstanding based on the observed history data.

5.2 Data and variables

The data are from the National Association of Insurance Commissioners (NAIC) Schedule P triangles (Meyers and Shi, 2011). In Schedule P data, the data are aggregated into accident year-development year records. Our analysis will focus on one insurance company. Table 5.1 and Table 5.2 display the cumulative paid losses and incurred losses, respectively. Each triangle contains data for accident years 1988-1997 and ten development years.

The incremental paid losses and claims outstanding are shown in Table 5.3 and Table 5.4. Claims outstanding is calculated as incurred loss less cumulative paid loss.

Table 5.1: cumulative paid loss

year	premium	1	2	3	4	5	6	7	8	9	10
1988	286 378	54 699	108 337	143 899	164 818	179 538	185 391	188 023	189 759	190 520	193 499
1989	308 908	60 091	119 366	151 151	174 665	185 469	192 213	196 152	198 013	199 997	
1990	326 503	65 860	130 803	172 390	197 977	210 230	219 267	222 428	224 078		
1991	332 616	61 946	121 108	158 880	182 689	195 247	201 854	204 911			
1992	341 890	65 043	128 550	164 433	187 508	199 823	208 008				
1993	355 840	72 295	144 579	185 446	208 388	219 345					
1994	379 781	81 988	151 197	189 630	212 446						
1995	398 755	83 207	152 470	190 974							
1996	406 609	79 699	143 590								
1997	406 516	75 827									

Table 5.2: incurred loss

year	premium	1	2	3	4	5	6	7	8	9	10
1988	286 378	172 262	180 112	187 158	188 775	190 569	190 701	190 344	191 752	192 087	194 099
1989	308 908	192 154	198 550	196 977	197 877	198 265	198 957	200 108	200 044	201 101	
1990	326 503	215 766	218 769	222 694	225 682	226 110	226 247	226 698	226 739		
1991	332 616	208 425	207 219	207 811	211 186	210 054	208 456	207 581			
1992	341 890	216 172	217 665	216 400	214 510	212 937	214 979				
1993	355 840	237 378	237 679	237 253	233 699	231 958					
1994	379 781	249 592	248 849	242 493	243 140						
1995	398 755	256 434	249 225	238 468							
1996	406 609	237 860	227 353								
1997	406 516	220 029									

Let indices $1 \leq i \leq I$ and $1 \leq j \leq I$ denote accident years and development years, respectively. The incremental paid loss is denoted as $x_{i,j}^{paid}$ and the total claims outstanding or case reserve is denoted as $x_{i,j}^{os}$. We define the normalized incremental paid losses as $Y_{i,j}^{paid} = (x_{i,j}^{paid}/NPE_i)$ and normalized claims outstanding

Table 5.3: incremental paid loss

year	premium	1	2	3	4	5	6	7	8	9	10
1988	286 378	54 699	53 638	35 562	20 919	14 720	5 853	2 632	1 736	761	2 979
1989	308 908	60 091	59 275	31 785	23 514	10 804	6 744	3 939	1 861	1 984	
1990	326 503	65 860	64 943	41 587	25 587	12 253	9 037	3 161	1 650		
1991	332 616	61 946	59 162	37 772	23 809	12 558	6 607	3 057			
1992	341 890	65 043	63 507	35 883	23 075	12 315	8 185				
1993	355 840	72 295	72 284	40 867	22 942	10 957					
1994	379 781	81 988	69 209	38 433	22 816						
1995	398 755	83 207	69 263	38 504							
1996	406 609	79 699	63 891								
1997	406 516	75 827									

Table 5.4: claims outstanding

year	premium	1	2	3	4	5	6	7	8	9	10
1988	286 378	117 563	71 775	43 259	23 957	11 031	5 310	2 321	1 993	1 567	600
1989	308 908	132 063	79 184	45 826	23 212	12 796	6 744	3 956	2 031	1 104	
1990	326 503	149 906	87 966	50 304	27 705	15 880	6 980	4 270	2 661		
1991	332 616	146 479	86 111	48 931	28 497	14 807	6 602	2 670			
1992	341 890	151 129	89 115	51 967	27 002	13 114	6 971				
1993	355 840	165 083	93 100	51 807	25 311	12 613					
1994	379 781	167 604	97 652	52 863	30 694						
1995	398 755	173 227	96 755	47 494							
1996	406 609	158 161	83 763								
1997	406 516	144 202									

as $Y_{i,j}^{os} = (x_{i,j}^{os}/NPE_i)$, where NPE_i denotes the net earned premium for accident year i. Then the normalized incremental paid loss triangle and normalized claims outstanding triangle can be written as in Table 5.5 and Table 5.6.

Table 5.5: normalized incremental paid loss

	1	2	3	4	5	6	7	8	9	10
1	Y_{11}^{paid}	Y_{12}^{paid}	Y_{13}^{paid}	Y_{14}^{paid}	Y_{15}^{paid}	Y_{16}^{paid}	Y_{17}^{paid}	Y_{18}^{paid}	Y_{19}^{paid}	Y_{110}^{paid}
2	Y_{21}^{paid}	Y_{22}^{paid}	Y_{23}^{paid}	Y_{24}^{paid}	Y_{25}^{paid}	Y_{26}^{paid}	Y_{27}^{paid}	Y_{28}^{paid}	Y_{29}^{paid}	
3	Y_{31}^{paid}	Y_{32}^{paid}	Y_{33}^{paid}	Y_{34}^{paid}	Y_{35}^{paid}	Y_{36}^{paid}	Y_{37}^{paid}	Y_{38}^{paid}		
4	Y_{41}^{paid}	Y_{42}^{paid}	Y_{43}^{paid}	Y_{44}^{paid}	Y_{45}^{paid}	Y_{46}^{paid}	Y_{47}^{paid}			
5	Y_{51}^{paid}	Y_{52}^{paid}	Y_{53}^{paid}	Y_{54}^{paid}	Y_{55}^{paid}	Y_{56}^{paid}				
6	Y_{61}^{paid}	Y_{62}^{paid}	Y_{63}^{paid}	Y_{64}^{paid}	Y_{65}^{paid}					
7	Y_{71}^{paid}	Y_{72}^{paid}	Y_{73}^{paid}	Y_{74}^{paid}						
8	Y_{81}^{paid}	Y_{82}^{paid}	Y_{83}^{paid}							
9	Y_{91}^{paid}	Y_{92}^{paid}								
10	Y_{101}^{paid}									

The observed data are $x_{i,j}^{paid} : i = 1, ..., I; j = 1, ..., I - i + 1$ and $x_{i,j}^{os} : i = 1, ..., I; j = 1, ..., I - i + 1$. The goal is to predict the future values $\{\hat{x}_{i,j}^{paid} : i =$

Table 5.6: normalized claims outstanding

	1	2	3	4	5	6	7	8	9	10
1	Y^{os}_{11}	Y^{os}_{12}	Y^{os}_{13}	Y^{os}_{14}	Y^{os}_{15}	Y^{os}_{16}	Y^{os}_{17}	Y^{os}_{18}	Y^{os}_{19}	Y^{os}_{110}
2	Y^{os}_{21}	Y^{os}_{22}	Y^{os}_{23}	Y^{os}_{24}	Y^{os}_{25}	Y^{os}_{26}	Y^{os}_{27}	Y^{os}_{28}	Y^{os}_{29}	
3	Y^{os}_{31}	Y^{os}_{32}	Y^{os}_{33}	Y^{os}_{34}	Y^{os}_{35}	Y^{os}_{36}	Y^{os}_{37}	Y^{os}_{38}		
4	Y^{os}_{41}	Y^{os}_{42}	Y^{os}_{43}	Y^{os}_{44}	Y^{os}_{45}	Y^{os}_{46}	Y^{os}_{47}			
5	Y^{os}_{51}	Y^{os}_{52}	Y^{os}_{53}	Y^{os}_{54}	Y^{os}_{55}	Y^{os}_{56}				
6	Y^{os}_{61}	Y^{os}_{62}	Y^{os}_{63}	Y^{os}_{64}	Y^{os}_{65}					
7	Y^{os}_{71}	Y^{os}_{72}	Y^{os}_{73}	Y^{os}_{74}						
8	Y^{os}_{81}	Y^{os}_{82}	Y^{os}_{83}							
9	Y^{os}_{91}	Y^{os}_{92}								
10	Y^{os}_{101}									

$2, ..., I; j = I - i + 2, ..., I\}$ and $\{\hat{x}^{os}_{i,j} : i = 2, ..., I; j = I - i + 2, ..., I\}$. The ultimate losses (UL) for each accident year $i = 1, ..., I$ is calculated as

$$\hat{UL}_i = \sum_{j=1}^{I-i+1} x^{paid}_{i,j} + \sum_{j=I-i+2}^{I} \hat{x}^{paid}_{i,j}, \tag{5.1}$$

where $x^{paid}_{i,j} : j = 1, ..., I - i + 1$ is the observed incremental paid loss and $\hat{x}^{paid}_{i,j} : j = I - i + 2, ..., I$ is the predicted incremental paid loss. The predictive performance will be evaluated on the ultimate losses.

5.3 Training setup

There are 45 samples for the training data. Each training sample is associated with an accident year - development year pair. Training data are associated with accident year i ($1 \leq i \leq 9$) and development year j ($2 \leq j \leq 11 - i$). The input for the training sample associated with accident year i ($1 \leq i \leq 9$) and development year j ($2 \leq j \leq 11-i$) are the sequences $(mask, ..., mask, Y^{paid}_{i,1}, Y^{paid}_{i,2}, ..., Y^{paid}_{i,j-1})$ and $(mask, ..., mask, Y^{os}_{i,1}, Y^{os}_{i,2}, ..., Y^{os}_{i,j-1})$. A mask value is used if there is no data at that time step. The sequences consists of 9 time steps. There is no historic data before development year 1. That's why we use the mask value. The two components of the input for the training data are shown in Table 5.7 and 5.8. The output for the

training sample associated with accident year i ($1 \leq i \leq 9$) and development year j ($2 \leq j \leq 11 - i$) are the sequences $(Y_{i,j}^{paid}, Y_{i,j+1}^{paid}, ..., Y_{i,11-i}^{paid}, mask, ..., mask)$ and $(Y_{i,j}^{os}, Y_{i,j+1}^{os}, ..., Y_{i,11-i}^{os}, mask, ..., mask)$. A mask value is used if there is no data at that time step. The sequences consists of 9 time steps. The mask value is used because we do not have the lower part of the triangle. The two components of the output for the training data are shown in Table 5.9 and 5.10.

The loss function is the average over the predicted time steps of the mean squared error of predictions. For each sample at cell (i, j), the per-sample loss is defined as

$$\frac{1}{I - i + 1 - (j - 1)} \sum_{k=j}^{I-i+1} \frac{(\hat{Y}_{i,k}^{paid} - Y_{i,k}^{paid})^2 + (\hat{Y}_{i,k}^{os} - Y_{i,k}^{os})^2}{2} \tag{5.2}$$

The AMSGRAD (Reddi et al., 2019) variant of ADAM is used for optimization. The learning rate is set to be 0.0005. To fit the model, the data as of calendar year 1997 is used for training. The model is trained for a maximum of 1000 epochs with the following early stopping scheme: if the loss on the validation data does not decrease over a 200-epoch window, training stops.

Table 5.7: First component of input for training data

	1	2	3	4	5	6	7	8	9
1	mask	mask	mask	mask	mask	mask	mask	mask	Y_{11}^{paid}
2	mask	mask	mask	mask	mask	mask	mask	Y_{11}^{paid}	Y_{12}^{paid}
3	mask	mask	mask	mask	mask	mask	Y_{11}^{paid}	Y_{12}^{paid}	Y_{13}^{paid}
4	mask	mask	mask	mask	mask	Y_{11}^{paid}	Y_{12}^{paid}	Y_{13}^{paid}	Y_{14}^{paid}
5	mask	mask	mask	mask	Y_{11}^{paid}	Y_{12}^{paid}	Y_{13}^{paid}	Y_{14}^{paid}	Y_{15}^{paid}
6	mask	mask	mask	Y_{11}^{paid}	Y_{12}^{paid}	Y_{13}^{paid}	Y_{14}^{paid}	Y_{15}^{paid}	Y_{16}^{paid}
7	mask	mask	Y_{11}^{paid}	Y_{12}^{paid}	Y_{13}^{paid}	Y_{14}^{paid}	Y_{15}^{paid}	Y_{16}^{paid}	Y_{17}^{paid}
8	mask	Y_{11}^{paid}	Y_{12}^{paid}	Y_{13}^{paid}	Y_{14}^{paid}	Y_{15}^{paid}	Y_{16}^{paid}	Y_{17}^{paid}	Y_{18}^{paid}
9	Y_{11}^{paid}	Y_{12}^{paid}	Y_{13}^{paid}	Y_{14}^{paid}	Y_{15}^{paid}	Y_{16}^{paid}	Y_{17}^{paid}	Y_{18}^{paid}	Y_{19}^{paid}
10	mask	mask	mask	mask	mask	mask	mask	mask	Y_{21}^{paid}
11	mask	mask	mask	mask	mask	mask	mask	Y_{21}^{paid}	Y_{22}^{paid}
12	mask	mask	mask	mask	mask	mask	Y_{21}^{paid}	Y_{22}^{paid}	Y_{23}^{paid}
13	mask	mask	mask	mask	mask	Y_{21}^{paid}	Y_{22}^{paid}	Y_{23}^{paid}	Y_{24}^{paid}
14	mask	mask	mask	mask	Y_{21}^{paid}	Y_{22}^{paid}	Y_{23}^{paid}	Y_{24}^{paid}	Y_{25}^{paid}
15	mask	mask	mask	Y_{21}^{paid}	Y_{22}^{paid}	Y_{23}^{paid}	Y_{24}^{paid}	Y_{25}^{paid}	Y_{26}^{paid}
16	mask	mask	Y_{21}^{paid}	Y_{22}^{paid}	Y_{23}^{paid}	Y_{24}^{paid}	Y_{25}^{paid}	Y_{26}^{paid}	Y_{27}^{paid}
17	mask	Y_{21}^{paid}	Y_{22}^{paid}	Y_{23}^{paid}	Y_{24}^{paid}	Y_{25}^{paid}	Y_{26}^{paid}	Y_{27}^{paid}	Y_{28}^{paid}
18	mask	mask	mask	mask	mask	mask	mask	mask	Y_{31}^{paid}
19	mask	mask	mask	mask	mask	mask	mask	Y_{31}^{paid}	Y_{32}^{paid}
20	mask	mask	mask	mask	mask	mask	Y_{31}^{paid}	Y_{32}^{paid}	Y_{33}^{paid}
21	mask	mask	mask	mask	mask	Y_{31}^{paid}	Y_{32}^{paid}	Y_{33}^{paid}	Y_{34}^{paid}
22	mask	mask	mask	mask	Y_{31}^{paid}	Y_{32}^{paid}	Y_{33}^{paid}	Y_{34}^{paid}	Y_{35}^{paid}
23	mask	mask	mask	Y_{31}^{paid}	Y_{32}^{paid}	Y_{33}^{paid}	Y_{34}^{paid}	Y_{35}^{paid}	Y_{36}^{paid}
24	mask	mask	Y_{31}^{paid}	Y_{32}^{paid}	Y_{33}^{paid}	Y_{34}^{paid}	Y_{35}^{paid}	Y_{36}^{paid}	Y_{37}^{paid}
25	mask	mask	mask	mask	mask	mask	mask	mask	Y_{41}^{paid}
26	mask	mask	mask	mask	mask	mask	mask	Y_{41}^{paid}	Y_{42}^{paid}
27	mask	mask	mask	mask	mask	mask	Y_{41}^{paid}	Y_{42}^{paid}	Y_{43}^{paid}
28	mask	mask	mask	mask	mask	Y_{41}^{paid}	Y_{42}^{paid}	Y_{43}^{paid}	Y_{44}^{paid}
29	mask	mask	mask	mask	Y_{41}^{paid}	Y_{42}^{paid}	Y_{43}^{paid}	Y_{44}^{paid}	Y_{45}^{paid}
30	mask	mask	mask	Y_{41}^{paid}	Y_{42}^{paid}	Y_{43}^{paid}	Y_{44}^{paid}	Y_{45}^{paid}	Y_{46}^{paid}
31	mask	mask	mask	mask	mask	mask	mask	mask	Y_{51}^{paid}
32	mask	mask	mask	mask	mask	mask	mask	Y_{51}^{paid}	Y_{52}^{paid}
33	mask	mask	mask	mask	mask	mask	Y_{51}^{paid}	Y_{52}^{paid}	Y_{53}^{paid}
34	mask	mask	mask	mask	mask	Y_{51}^{paid}	Y_{52}^{paid}	Y_{53}^{paid}	Y_{54}^{paid}
35	mask	mask	mask	mask	Y_{51}^{paid}	Y_{52}^{paid}	Y_{53}^{paid}	Y_{54}^{paid}	Y_{55}^{paid}
36	mask	mask	mask	mask	mask	mask	mask	mask	Y_{61}^{paid}
37	mask	mask	mask	mask	mask	mask	mask	Y_{61}^{paid}	Y_{62}^{paid}
38	mask	mask	mask	mask	mask	mask	Y_{61}^{paid}	Y_{62}^{paid}	Y_{63}^{paid}
39	mask	mask	mask	mask	mask	Y_{61}^{paid}	Y_{62}^{paid}	Y_{63}^{paid}	Y_{64}^{paid}
40	mask	mask	mask	mask	mask	mask	mask	mask	Y_{71}^{paid}
41	mask	mask	mask	mask	mask	mask	mask	Y_{71}^{paid}	Y_{72}^{paid}
42	mask	mask	mask	mask	mask	mask	Y_{71}^{paid}	Y_{72}^{paid}	Y_{73}^{paid}
43	mask	mask	mask	mask	mask	mask	mask	mask	Y_{81}^{paid}
44	mask	mask	mask	mask	mask	mask	mask	Y_{81}^{paid}	Y_{82}^{paid}
45	mask	mask	mask	mask	mask	mask	mask	mask	Y_{91}^{paid}

Table 5.8: Second component of input for training data

	1	2	3	4	5	6	7	8	9
1	mask	mask	mask	mask	mask	mask	mask	mask	Y_{11}^{os}
2	mask	mask	mask	mask	mask	mask	mask	Y_{11}^{os}	Y_{12}^{os}
3	mask	mask	mask	mask	mask	mask	Y_{11}^{os}	Y_{12}^{os}	Y_{13}^{os}
4	mask	mask	mask	mask	mask	Y_{11}^{os}	Y_{12}^{os}	Y_{13}^{os}	Y_{14}^{os}
5	mask	mask	mask	mask	Y_{11}^{os}	Y_{12}^{os}	Y_{13}^{os}	Y_{14}^{os}	Y_{15}^{os}
6	mask	mask	mask	Y_{11}^{os}	Y_{12}^{os}	Y_{13}^{os}	Y_{14}^{os}	Y_{15}^{os}	Y_{16}^{os}
7	mask	mask	Y_{11}^{os}	Y_{12}^{os}	Y_{13}^{os}	Y_{14}^{os}	Y_{15}^{os}	Y_{16}^{os}	Y_{17}^{os}
8	mask	Y_{11}^{os}	Y_{12}^{os}	Y_{13}^{os}	Y_{14}^{os}	Y_{15}^{os}	Y_{16}^{os}	Y_{17}^{os}	Y_{18}^{os}
9	Y_{11}^{os}	Y_{12}^{os}	Y_{13}^{os}	Y_{14}^{os}	Y_{15}^{os}	Y_{16}^{os}	Y_{17}^{os}	Y_{18}^{os}	Y_{19}^{os}
10	mask	mask	mask	mask	mask	mask	mask	mask	Y_{21}^{os}
11	mask	mask	mask	mask	mask	mask	mask	Y_{21}^{os}	Y_{22}^{os}
12	mask	mask	mask	mask	mask	mask	Y_{21}^{os}	Y_{22}^{os}	Y_{23}^{os}
13	mask	mask	mask	mask	mask	Y_{21}^{os}	Y_{22}^{os}	Y_{23}^{os}	Y_{24}^{os}
14	mask	mask	mask	mask	Y_{21}^{os}	Y_{22}^{os}	Y_{23}^{os}	Y_{24}^{os}	Y_{25}^{os}
15	mask	mask	mask	Y_{21}^{os}	Y_{22}^{os}	Y_{23}^{os}	Y_{24}^{os}	Y_{25}^{os}	Y_{26}^{os}
16	mask	mask	Y_{21}^{os}	Y_{22}^{os}	Y_{23}^{os}	Y_{24}^{os}	Y_{25}^{os}	Y_{26}^{os}	Y_{27}^{os}
17	mask	Y_{21}^{os}	Y_{22}^{os}	Y_{23}^{os}	Y_{24}^{os}	Y_{25}^{os}	Y_{26}^{os}	Y_{27}^{os}	Y_{28}^{os}
18	mask	mask	mask	mask	mask	mask	mask	mask	Y_{31}^{os}
19	mask	mask	mask	mask	mask	mask	mask	Y_{31}^{os}	Y_{32}^{os}
20	mask	mask	mask	mask	mask	mask	Y_{31}^{os}	Y_{32}^{os}	Y_{33}^{os}
21	mask	mask	mask	mask	mask	Y_{31}^{os}	Y_{32}^{os}	Y_{33}^{os}	Y_{34}^{os}
22	mask	mask	mask	mask	Y_{31}^{os}	Y_{32}^{os}	Y_{33}^{os}	Y_{34}^{os}	Y_{35}^{os}
23	mask	mask	mask	Y_{31}^{os}	Y_{32}^{os}	Y_{33}^{os}	Y_{34}^{os}	Y_{35}^{os}	Y_{36}^{os}
24	mask	mask	Y_{31}^{os}	Y_{32}^{os}	Y_{33}^{os}	Y_{34}^{os}	Y_{35}^{os}	Y_{36}^{os}	Y_{37}^{os}
25	mask	mask	mask	mask	mask	mask	mask	mask	Y_{41}^{os}
26	mask	mask	mask	mask	mask	mask	mask	Y_{41}^{os}	Y_{42}^{os}
27	mask	mask	mask	mask	mask	mask	Y_{41}^{os}	Y_{42}^{os}	Y_{43}^{os}
28	mask	mask	mask	mask	mask	Y_{41}^{os}	Y_{42}^{os}	Y_{43}^{os}	Y_{44}^{os}
29	mask	mask	mask	mask	Y_{41}^{os}	Y_{42}^{os}	Y_{43}^{os}	Y_{44}^{os}	Y_{45}^{os}
30	mask	mask	mask	Y_{41}^{os}	Y_{42}^{os}	Y_{43}^{os}	Y_{44}^{os}	Y_{45}^{os}	Y_{46}^{os}
31	mask	mask	mask	mask	mask	mask	mask	mask	Y_{51}^{os}
32	mask	mask	mask	mask	mask	mask	mask	Y_{51}^{os}	Y_{52}^{os}
33	mask	mask	mask	mask	mask	mask	Y_{51}^{os}	Y_{52}^{os}	Y_{53}^{os}
34	mask	mask	mask	mask	mask	Y_{51}^{os}	Y_{52}^{os}	Y_{53}^{os}	Y_{54}^{os}
35	mask	mask	mask	mask	Y_{51}^{os}	Y_{52}^{os}	Y_{53}^{os}	Y_{54}^{os}	Y_{55}^{os}
36	mask	mask	mask	mask	mask	mask	mask	mask	Y_{61}^{os}
37	mask	mask	mask	mask	mask	mask	mask	Y_{61}^{os}	Y_{62}^{os}
38	mask	mask	mask	mask	mask	mask	Y_{61}^{os}	Y_{62}^{os}	Y_{63}^{os}
39	mask	mask	mask	mask	mask	Y_{61}^{os}	Y_{62}^{os}	Y_{63}^{os}	Y_{64}^{os}
40	mask	mask	mask	mask	mask	mask	mask	mask	Y_{71}^{os}
41	mask	mask	mask	mask	mask	mask	mask	Y_{71}^{os}	Y_{72}^{os}
42	mask	mask	mask	mask	mask	mask	Y_{71}^{os}	Y_{72}^{os}	Y_{73}^{os}
43	mask	mask	mask	mask	mask	mask	mask	mask	Y_{81}^{os}
44	mask	mask	mask	mask	mask	mask	mask	Y_{81}^{os}	Y_{82}^{os}
45	mask	mask	mask	mask	mask	mask	mask	mask	Y_{91}^{os}

Table 5.9: First component of output for training data

	1	2	3	4	5	6	7	8	9
1	Y_{12}^{paid}	Y_{13}^{paid}	Y_{14}^{paid}	Y_{15}^{paid}	Y_{16}^{paid}	Y_{17}^{paid}	Y_{18}^{paid}	Y_{19}^{paid}	Y_{110}^{paid}
2	Y_{13}^{paid}	Y_{14}^{paid}	Y_{15}^{paid}	Y_{16}^{paid}	Y_{17}^{paid}	Y_{18}^{paid}	Y_{19}^{paid}	Y_{110}^{paid}	mask
3	Y_{14}^{paid}	Y_{15}^{paid}	Y_{16}^{paid}	Y_{17}^{paid}	Y_{18}^{paid}	Y_{19}^{paid}	Y_{110}^{paid}	mask	mask
4	Y_{15}^{paid}	Y_{16}^{paid}	Y_{17}^{paid}	Y_{18}^{paid}	Y_{19}^{paid}	Y_{110}^{paid}	mask	mask	mask
5	Y_{16}^{paid}	Y_{17}^{paid}	Y_{18}^{paid}	Y_{19}^{paid}	Y_{110}^{paid}	mask	mask	mask	mask
6	Y_{17}^{paid}	Y_{18}^{paid}	Y_{19}^{paid}	Y_{110}^{paid}	mask	mask	mask	mask	mask
7	Y_{18}^{paid}	Y_{19}^{paid}	Y_{110}^{paid}	mask	mask	mask	mask	mask	mask
8	Y_{19}^{paid}	Y_{110}^{paid}	mask	mask	mask	mask	mask	mask	mask
9	Y_{110}^{paid}	mask	mask	mask	mask	mask	mask	mask	mask
10	Y_{22}^{paid}	Y_{23}^{paid}	Y_{24}^{paid}	Y_{25}^{paid}	Y_{26}^{paid}	Y_{27}^{paid}	Y_{28}^{paid}	Y_{29}^{paid}	mask
11	Y_{23}^{paid}	Y_{24}^{paid}	Y_{25}^{paid}	Y_{26}^{paid}	Y_{27}^{paid}	Y_{28}^{paid}	Y_{29}^{paid}	mask	mask
12	Y_{24}^{paid}	Y_{25}^{paid}	Y_{26}^{paid}	Y_{27}^{paid}	Y_{28}^{paid}	Y_{29}^{paid}	mask	mask	mask
13	Y_{25}^{paid}	Y_{26}^{paid}	Y_{27}^{paid}	Y_{28}^{paid}	Y_{29}^{paid}	mask	mask	mask	mask
14	Y_{26}^{paid}	Y_{27}^{paid}	Y_{28}^{paid}	Y_{29}^{paid}	mask	mask	mask	mask	mask
15	Y_{27}^{paid}	Y_{28}^{paid}	Y_{29}^{paid}	mask	mask	mask	mask	mask	mask
16	Y_{28}^{paid}	Y_{29}^{paid}	mask	mask	mask	mask	mask	mask	mask
17	Y_{29}^{paid}	mask	mask	mask	mask	mask	mask	mask	mask
18	Y_{32}^{paid}	Y_{33}^{paid}	Y_{34}^{paid}	Y_{35}^{paid}	Y_{36}^{paid}	Y_{37}^{paid}	Y_{38}^{paid}	mask	mask
19	Y_{33}^{paid}	Y_{34}^{paid}	Y_{35}^{paid}	Y_{36}^{paid}	Y_{37}^{paid}	Y_{38}^{paid}	mask	mask	mask
20	Y_{34}^{paid}	Y_{35}^{paid}	Y_{36}^{paid}	Y_{37}^{paid}	Y_{38}^{paid}	mask	mask	mask	mask
21	Y_{35}^{paid}	Y_{36}^{paid}	Y_{37}^{paid}	Y_{38}^{paid}	mask	mask	mask	mask	mask
22	Y_{36}^{paid}	Y_{37}^{paid}	Y_{38}^{paid}	mask	mask	mask	mask	mask	mask
23	Y_{37}^{paid}	Y_{38}^{paid}	mask	mask	mask	mask	mask	mask	mask
24	Y_{38}^{paid}	mask	mask	mask	mask	mask	mask	mask	mask
25	Y_{42}^{paid}	Y_{43}^{paid}	Y_{44}^{paid}	Y_{45}^{paid}	Y_{46}^{paid}	Y_{47}^{paid}	mask	mask	mask
26	Y_{43}^{paid}	Y_{44}^{paid}	Y_{45}^{paid}	Y_{46}^{paid}	Y_{47}^{paid}	mask	mask	mask	mask
27	Y_{44}^{paid}	Y_{45}^{paid}	Y_{46}^{paid}	Y_{47}^{paid}	mask	mask	mask	mask	mask
28	Y_{45}^{paid}	Y_{46}^{paid}	Y_{47}^{paid}	mask	mask	mask	mask	mask	mask
29	Y_{46}^{paid}	Y_{47}^{paid}	mask	mask	mask	mask	mask	mask	mask
30	Y_{47}^{paid}	mask	mask	mask	mask	mask	mask	mask	mask
31	Y_{52}^{paid}	Y_{53}^{paid}	Y_{54}^{paid}	Y_{55}^{paid}	Y_{56}^{paid}	mask	mask	mask	mask
32	Y_{53}^{paid}	Y_{54}^{paid}	Y_{55}^{paid}	Y_{56}^{paid}	mask	mask	mask	mask	mask
33	Y_{54}^{paid}	Y_{55}^{paid}	Y_{56}^{paid}	mask	mask	mask	mask	mask	mask
34	Y_{55}^{paid}	Y_{56}^{paid}	mask	mask	mask	mask	mask	mask	mask
35	Y_{56}^{paid}	mask	mask	mask	mask	mask	mask	mask	mask
36	Y_{62}^{paid}	Y_{63}^{paid}	Y_{64}^{paid}	Y_{65}^{paid}	mask	mask	mask	mask	mask
37	Y_{63}^{paid}	Y_{64}^{paid}	Y_{65}^{paid}	mask	mask	mask	mask	mask	mask
38	Y_{64}^{paid}	Y_{65}^{paid}	mask	mask	mask	mask	mask	mask	mask
39	Y_{65}^{paid}	mask	mask	mask	mask	mask	mask	mask	mask
40	Y_{72}^{paid}	Y_{73}^{paid}	Y_{74}^{paid}	mask	mask	mask	mask	mask	mask
41	Y_{73}^{paid}	Y_{74}^{paid}	mask	mask	mask	mask	mask	mask	mask
42	Y_{74}^{paid}	mask	mask	mask	mask	mask	mask	mask	mask
43	Y_{82}^{paid}	Y_{83}^{paid}	mask	mask	mask	mask	mask	mask	mask
44	Y_{83}^{paid}	mask	mask	mask	mask	mask	mask	mask	mask
45	Y_{92}^{paid}	mask	mask	mask	mask	mask	mask	mask	mask

Table 5.10: Second component of output for training data

	1	2	3	4	5	6	7	8	9
1	Y^{os}_{12}	Y^{os}_{13}	Y^{os}_{14}	Y^{os}_{15}	Y^{os}_{16}	Y^{os}_{17}	Y^{os}_{18}	Y^{os}_{19}	Y^{os}_{110}
2	Y^{os}_{13}	Y^{os}_{14}	Y^{os}_{15}	Y^{os}_{16}	Y^{os}_{17}	Y^{os}_{18}	Y^{os}_{19}	Y^{os}_{110}	mask
3	Y^{os}_{14}	Y^{os}_{15}	Y^{os}_{16}	Y^{os}_{17}	Y^{os}_{18}	Y^{os}_{19}	Y^{os}_{110}	mask	mask
4	Y^{os}_{15}	Y^{os}_{16}	Y^{os}_{17}	Y^{os}_{18}	Y^{os}_{19}	Y^{os}_{110}	mask	mask	mask
5	Y^{os}_{16}	Y^{os}_{17}	Y^{os}_{18}	Y^{os}_{19}	Y^{os}_{110}	mask	mask	mask	mask
6	Y^{os}_{17}	Y^{os}_{18}	Y^{os}_{19}	Y^{os}_{110}	mask	mask	mask	mask	mask
7	Y^{os}_{18}	Y^{os}_{19}	Y^{os}_{110}	mask	mask	mask	mask	mask	mask
8	Y^{os}_{19}	Y^{os}_{110}	mask	mask	mask	mask	mask	mask	mask
9	Y^{os}_{110}	mask	mask	mask	mask	mask	mask	mask	mask
10	Y^{os}_{22}	Y^{os}_{23}	Y^{os}_{24}	Y^{os}_{25}	Y^{os}_{26}	Y^{os}_{27}	Y^{os}_{28}	Y^{os}_{29}	mask
11	Y^{os}_{23}	Y^{os}_{24}	Y^{os}_{25}	Y^{os}_{26}	Y^{os}_{27}	Y^{os}_{28}	Y^{os}_{29}	mask	mask
12	Y^{os}_{24}	Y^{os}_{25}	Y^{os}_{26}	Y^{os}_{27}	Y^{os}_{28}	Y^{os}_{29}	mask	mask	mask
13	Y^{os}_{25}	Y^{os}_{26}	Y^{os}_{27}	Y^{os}_{28}	Y^{os}_{29}	mask	mask	mask	mask
14	Y^{os}_{26}	Y^{os}_{27}	Y^{os}_{28}	Y^{os}_{29}	mask	mask	mask	mask	mask
15	Y^{os}_{27}	Y^{os}_{28}	Y^{os}_{29}	mask	mask	mask	mask	mask	mask
16	Y^{os}_{28}	Y^{os}_{29}	mask	mask	mask	mask	mask	mask	mask
17	Y^{os}_{29}	mask	mask	mask	mask	mask	mask	mask	mask
18	Y^{os}_{32}	Y^{os}_{33}	Y^{os}_{34}	Y^{os}_{35}	Y^{os}_{36}	Y^{os}_{37}	Y^{os}_{38}	mask	mask
19	Y^{os}_{33}	Y^{os}_{34}	Y^{os}_{35}	Y^{os}_{36}	Y^{os}_{37}	Y^{os}_{38}	mask	mask	mask
20	Y^{os}_{34}	Y^{os}_{35}	Y^{os}_{36}	Y^{os}_{37}	Y^{os}_{38}	mask	mask	mask	mask
21	Y^{os}_{35}	Y^{os}_{36}	Y^{os}_{37}	Y^{os}_{38}	mask	mask	mask	mask	mask
22	Y^{os}_{36}	Y^{os}_{37}	Y^{os}_{38}	mask	mask	mask	mask	mask	mask
23	Y^{os}_{37}	Y^{os}_{38}	mask	mask	mask	mask	mask	mask	mask
24	Y^{os}_{38}	mask	mask	mask	mask	mask	mask	mask	mask
25	Y^{os}_{42}	Y^{os}_{43}	Y^{os}_{44}	Y^{os}_{45}	Y^{os}_{46}	Y^{os}_{47}	mask	mask	mask
26	Y^{os}_{43}	Y^{os}_{44}	Y^{os}_{45}	Y^{os}_{46}	Y^{os}_{47}	mask	mask	mask	mask
27	Y^{os}_{44}	Y^{os}_{45}	Y^{os}_{46}	Y^{os}_{47}	mask	mask	mask	mask	mask
28	Y^{os}_{45}	Y^{os}_{46}	Y^{os}_{47}	mask	mask	mask	mask	mask	mask
29	Y^{os}_{46}	Y^{os}_{47}	mask	mask	mask	mask	mask	mask	mask
30	Y^{os}_{47}	mask	mask	mask	mask	mask	mask	mask	mask
31	Y^{os}_{52}	Y^{os}_{53}	Y^{os}_{54}	Y^{os}_{55}	Y^{os}_{56}	mask	mask	mask	mask
32	Y^{os}_{53}	Y^{os}_{54}	Y^{os}_{55}	Y^{os}_{56}	mask	mask	mask	mask	mask
33	Y^{os}_{54}	Y^{os}_{55}	Y^{os}_{56}	mask	mask	mask	mask	mask	mask
34	Y^{os}_{55}	Y^{os}_{56}	mask	mask	mask	mask	mask	mask	mask
35	Y^{os}_{56}	mask	mask	mask	mask	mask	mask	mask	mask
36	Y^{os}_{62}	Y^{os}_{63}	Y^{os}_{64}	Y^{os}_{65}	mask	mask	mask	mask	mask
37	Y^{os}_{63}	Y^{os}_{64}	Y^{os}_{65}	mask	mask	mask	mask	mask	mask
38	Y^{os}_{64}	Y^{os}_{65}	mask	mask	mask	mask	mask	mask	mask
39	Y^{os}_{65}	mask	mask	mask	mask	mask	mask	mask	mask
40	Y^{os}_{72}	Y^{os}_{73}	Y^{os}_{74}	mask	mask	mask	mask	mask	mask
41	Y^{os}_{73}	Y^{os}_{74}	mask	mask	mask	mask	mask	mask	mask
42	Y^{os}_{74}	mask	mask	mask	mask	mask	mask	mask	mask
43	Y^{os}_{82}	Y^{os}_{83}	mask	mask	mask	mask	mask	mask	mask
44	Y^{os}_{83}	mask	mask	mask	mask	mask	mask	mask	mask
45	Y^{os}_{92}	mask	mask	mask	mask	mask	mask	mask	mask

5.4 Testing setup

There are 9 samples for the testing data. Each testing sample is associated with an accident year - development year pair. Testing data are associated with accident year i ($2 \leq i \leq 10$) and development year j ($j = 12 - i$). For accident year 1, we have all the data from development year 1 to development year 10. The input for the testing sample associated with accident year i ($2 \leq i \leq 10$) and development year j ($j = 12 - i$) are the sequences $(mask, ..., mask, Y_{i,1}^{paid}, Y_{i,2}^{paid}, ..., Y_{i,11-i}^{paid})$ and $(mask, ..., mask, Y_{i,1}^{os}, Y_{i,2}^{os}, ..., Y_{i,11-i}^{os})$. A mask value is used if there is no data at that time step. The sequences also consist of 9 time steps. There is no historic data before development year 1. That's why we use the mask value. The two components of the input for the test data are shown in Table 5.11 and 5.12.

At each accident year and development year for which we have data, we predict future incremental paid loss and claims outstanding based on the observed history. We keep the data with development lag less or equal than 10 years for the predicted output.

Table 5.11: First component of input for testing

	1	2	3	4	5	6	7	8	9
1	Y_{21}^{paid}	Y_{22}^{paid}	Y_{23}^{paid}	Y_{24}^{paid}	Y_{25}^{paid}	Y_{26}^{paid}	Y_{27}^{paid}	Y_{28}^{paid}	Y_{29}^{paid}
2	mask	Y_{31}^{paid}	Y_{32}^{paid}	Y_{33}^{paid}	Y_{34}^{paid}	Y_{35}^{paid}	Y_{36}^{paid}	Y_{37}^{paid}	Y_{38}^{paid}
3	mask	mask	Y_{41}^{paid}	Y_{42}^{paid}	Y_{43}^{paid}	Y_{44}^{paid}	Y_{45}^{paid}	Y_{46}^{paid}	Y_{47}^{paid}
4	mask	mask	mask	Y_{51}^{paid}	Y_{52}^{paid}	Y_{53}^{paid}	Y_{54}^{paid}	Y_{55}^{paid}	Y_{56}^{paid}
5	mask	mask	mask	mask	Y_{61}^{paid}	Y_{62}^{paid}	Y_{63}^{paid}	Y_{64}^{paid}	Y_{65}^{paid}
6	mask	mask	mask	mask	mask	Y_{71}^{paid}	Y_{72}^{paid}	Y_{73}^{paid}	Y_{74}^{paid}
7	mask	mask	mask	mask	mask	mask	Y_{81}^{paid}	Y_{82}^{paid}	Y_{83}^{paid}
8	mask	mask	mask	mask	mask	mask	mask	Y_{91}^{paid}	Y_{92}^{paid}
9	mask	mask	mask	mask	mask	mask	mask	mask	Y_{101}^{paid}

Table 5.12: Second component of input for testing

	1	2	3	4	5	6	7	8	9
1	Y^{os}_{21}	Y^{os}_{22}	Y^{os}_{23}	Y^{os}_{24}	Y^{os}_{25}	Y^{os}_{26}	Y^{os}_{27}	Y^{os}_{28}	Y^{os}_{29}
2	mask	Y^{os}_{31}	Y^{os}_{32}	Y^{os}_{33}	Y^{os}_{34}	Y^{os}_{35}	Y^{os}_{36}	Y^{os}_{37}	Y^{os}_{38}
3	mask	mask	Y^{os}_{41}	Y^{os}_{42}	Y^{os}_{43}	Y^{os}_{44}	Y^{os}_{45}	Y^{os}_{46}	Y^{os}_{47}
4	mask	mask	mask	Y^{os}_{51}	Y^{os}_{52}	Y^{os}_{53}	Y^{os}_{54}	Y^{os}_{55}	Y^{os}_{56}
5	mask	mask	mask	mask	Y^{os}_{61}	Y^{os}_{62}	Y^{os}_{63}	Y^{os}_{64}	Y^{os}_{65}
6	mask	mask	mask	mask	mask	Y^{os}_{71}	Y^{os}_{72}	Y^{os}_{73}	Y^{os}_{74}
7	mask	mask	mask	mask	mask	mask	Y^{os}_{81}	Y^{os}_{82}	Y^{os}_{83}
8	mask	mask	mask	mask	mask	mask	mask	Y^{os}_{91}	Y^{os}_{92}
9	mask	mask	mask	mask	mask	mask	mask	mask	Y^{os}_{101}

5.5 Results and discussion

We run the model on the testing data and generated the lower triangle for both the incremental paid losses and outstanding claims. We keep the data with development lag less or equal than 10 years. Table 5.13 shows the full triangle of incremental paid losses. The full triangle of claims outstanding is displayed in Table 5.14.

Table 5.13: Full triangle of incremental paid loss. The predicted values are in red.

year	1	2	3	4	5	6	7	8	9	10
1988	54699	53638	35562	20919	14720	5853	2632	1736	761	2979
1989	60091	59275	31785	23514	10804	6744	3939	1861	1984	1856
1990	65860	64943	41587	25587	12253	9037	3161	1650	2054	1338
1991	61946	59162	37772	23809	12558	6607	3057	2249	1538	1312
1992	65043	63507	35883	23075	12315	8185	3055	2180	1882	1701
1993	72295	72284	40867	22942	10957	4503	2957	2414	2055	1805
1994	81988	69209	38433	22816	10785	6248	3704	2664	2104	1764
1995	83207	69263	38504	18277	10040	5679	3372	2403	1879	1553
1996	79699	63891	35140	22243	12447	6938	3934	2678	2121	1891
1997	75827	71068	42941	27347	14407	8018	4040	3286	3196	3187

Kuo's Deep Triangle use randomness by design. One of the reason for randomness is the dropout in neural network layers. During training, some number of nodes are dropped out. By dropping a node out, we mean it is temporarily removed from the network. The choice of which nodes to drop is random. Each node is dropped with a given probability, in order to reduce overfiting, see page 6 in Kuo (2019). With limited training data, deep neural network could face

Table 5.14: Full triangle of claims outstanding. The predicted values are in red.

year	1	2	3	4	5	6	7	8	9	10
1988	117563	71775	43259	23957	11031	5310	2321	1993	1567	600
1989	132063	79184	45826	23212	12796	6744	3956	2031	1104	1332
1990	149906	87966	50304	27705	15880	6980	4270	2661	1530	801
1991	146479	86111	48931	28497	14807	6602	2670	1758	903	813
1992	151129	89115	51967	27002	13114	6971	2599	1359	1108	1039
1993	165083	93100	51807	25311	12613	4020	2278	1583	1311	1174
1994	167604	97652	52863	30694	11629	5573	3096	2011	1528	1286
1995	173227	96755	47494	21285	10173	5015	2851	1907	1481	1293
1996	158161	83763	44051	25524	12915	6363	3442	2240	1768	1570
1997	144202	95724	55410	32782	15779	7767	3630	2888	2808	2800

the problem of overfitting. The deep triangle model implement dropout in the *layer_gru* function, which is explained in the appendix. The prediction from the neural network model could be slightly different between runs since the dropout is random. Another prediction for the full triangle of incremental paid loss and claims outstanding is shown in Table 5.15 and Table 5.16.

Table 5.15: Full triangle of incremental paid loss. The predicted values are in blue.

year	1	2	3	4	5	6	7	8	9	10
1988	54699	53638	35562	20919	14720	5853	2632	1736	761	2979
1989	60091	59275	31785	23514	10804	6744	3939	1861	1984	1924
1990	65860	64943	41587	25587	12253	9037	3161	1650	2117	1505
1991	61946	59162	37772	23809	12558	6607	3057	2293	1654	1598
1992	65043	63507	35883	23075	12315	8185	2984	2223	2046	1973
1993	72295	72284	40867	22942	10957	4399	2908	2439	2156	1949
1994	81988	69209	38433	22816	10567	5608	3322	2355	1798	1407
1995	83207	69263	38504	19344	10691	5712	3291	2230	1607	1231
1996	79699	63891	33933	20975	12247	6720	3525	2379	1890	1646
1997	75827	71001	43430	27908	15287	7993	3822	3154	3048	2999

Deep triangle uses incremental paid loss and claims outstanding to predict loss reserve for each line of business. To link the claims of two different business lines, we provide the incremental claims of the two business lines to the deep triangle model. With inputs from two lines of business, the deep triangle model can capture the dependence structure between the two lines.

Table 5.16: Full triangle of claims outstanding. The predicted values are in blue.

year	1	2	3	4	5	6	7	8	9	10
1988	117563	71775	43259	23957	11031	5310	2321	1993	1567	600
1989	132063	79184	45826	23212	12796	6744	3956	2031	1104	1369
1990	149906	87966	50304	27705	15880	6980	4270	2661	1537	847
1991	146479	86111	48931	28497	14807	6602	2670	1743	940	858
1992	151129	89115	51967	27002	13114	6971	2533	1455	1222	1157
1993	165083	93100	51807	25311	12613	4029	2193	1642	1402	1271
1994	167604	97652	52863	30694	11324	4853	2686	1775	1415	1235
1995	173227	96755	47494	22512	10794	4978	2896	1909	1452	1203
1996	158161	83763	42680	24221	12758	6064	3222	2009	1533	1356
1997	144202	96439	55947	33051	16188	7564	3434	2696	2561	2502

Table 5.17: Full triangle of personal line. The predicted values are in red.

year	1	2	3	4	5	6	7	8	9	10
1988	1376384	1211168	535883	313790	168142	79972	39235	15030	10865	4086
1989	1576278	1437150	652445	342694	188799	76956	35042	17089	12507	1287
1990	1763277	1540231	678959	364199	177108	78169	47391	25288	3968	820
1991	1779698	1498531	661401	321434	162578	84581	53449	14423	3061	3022
1992	1843224	1573604	613095	299473	176842	106296	41489	10112	5798	5674
1993	1962385	1520298	581932	347434	238375	114260	42362	17868	11448	9557
1994	2033371	1430541	633500	432257	207632	102930	44102	21960	14674	11913
1995	2072061	1458541	727098	363784	183649	92314	43428	24013	16999	14165
1996	2210754	1517501	777016	422457	235400	128214	60445	29120	20359	17887
1997	2206886	1814038	663834	370789	190152	93078	41316	24885	21977	21480

We apply the deep triangle model to the same data that Shi and Frees (2011) use. The claim triangles consist of two lines of business, personal auto and commercial auto. For the input to the deep triangle model, the incremental claims loss triangles of the two business lines are normalized by earned premium. The model is fit to incremental paid loss ratios of the personal auto line and commercial auto line. The first component of the sample is the incremental paid loss from the personal line of business, and the second component of the sample corresponds to the incremental paid loss from the commercial line of business. Kuo's deep triangle use dropout, which may be interpreted as an ensemble model combination. The predictions are averaged over an ensemble of neural networks (with parameter sharing). It has long been observed that ensembles of models improve predictive performance (Lakshminarayanan et al., 2017). We run the deep triangle model many times and report the average predictions. The predicted full triangles are

Table 5.18: Full triangle of commercial line. The predicted values are in blue.

year	1	2	3	4	5	6	7	8	9	10
1988	33810	45318	46549	35206	23360	12502	6602	3373	2373	778
1989	37663	51771	40998	29496	12669	11204	5785	4220	1910	748
1990	40630	56318	56182	32473	15828	8409	7120	1125	1413	861
1991	40475	49697	39313	24044	13156	12595	2908	2800	1556	806
1992	37127	50983	34154	25455	19421	5728	4957	2345	1176	709
1993	41125	53302	40289	39912	6650	9618	4713	2336	1380	916
1994	57515	67881	86734	18109	18935	10555	5486	3077	2027	1487
1995	61553	132208	20923	31723	16441	10001	5514	3384	2429	1956
1996	112103	33250	49826	29756	16106	10687	6054	3334	2400	2088
1997	37554	48221	29349	16305	9271	5735	3044	1954	1712	1663

shown in Table 5.17 and Table 5.18.

Table 5.19: Estimation of the reserve

number of predictions	Reserves (average)		
	LoB 1	LoB 2	Total
1	6 366 487	393 767	6 760 254
10	6 327 648	386 313	6 713 960
20	6 373 726	392 250	6 765 976
30	6 293 546	384 009	6 677 555
40	6 321 082	385 024	6 706 106
50	6 375 960	385 992	6 761 951
100	6 359 162	386 854	6 746 016

The estimated reserve for the personal line and commercial line are 6 359 162 and 386 854, respectively. Thus, the estimated total reserve is 6 746 016. It is close to the estimated reserve from the Gaussian copula regression result, which is 6 919 171. The difference is 2.5%. We run the prediction for different number of times and use the average as the predicted reserves for the two lines of business. These predictions from various number of runs are displayed in Table 5.19.

Chapter 6

Conclusions and Future Work

The thesis first presents the loss models used in claim reserving, which is a crucial task in non-life insurance. The exponential family of distributions are introduced before moving to the stochastic models. The probability distributions from this family are used to specify the distributions of data in loss triangles. We describe the assumptions for the models and the estimation techniques. The model parameters are estimated with maximum likelihood method. Mean squared error of prediction is used to measure the variability of predicted reserve. We also present the copula regression model for the reserve in the third chapter. The dependence structure between two lines of business is modeled with Gaussian, Frank and Student's t copula. The various copula regression models in this thesis are compared using likelihood-based criteria. Besides, we also compared the risk capitals calculated with different copula models. Compared with the silo method, the Gaussian copula model generates the largest risk capital gain among all the copula models.

The fourth chapter gives an introduction to the feed-forward neural networks. The Universal approximation theorem states that neural networks allow us to approximate any continuous functions arbitrarily well. A loss function is used to measure the error between the predicted output and the target. Gradient descent methods can find the minimum of the cost function. For large networks, the

number of parameters are on the order of millions. Stochastic gradient descent methods estimate the overall gradient by calculating gradient for randomly chosen mini-batch. It replaces the actual gradient by an estimate and reduces the computational burden.

Finally, the last chapter focuses on the DeepTriangle model, which is a deep learning model for predicting paid losses. We present the training data and testing data setup for the deep triangle model. The input for the training and testing sample is the observed history as of the accident year - development year pair. When we use the incremental paid loss from two different business lines as input to the deep triangle model, the dependence structure between the two lines can be captured. We apply Kuo's DeepTriangle model to a dataset from a major US property-casualty insurer, which contains personal auto line and commercial auto line. The incremental paid loss from the two business lines correspond to the two components of the training and testing samples. The prediction is run 100 times and the average is used as the predicted reserve. We compared the predicted reserve from the deep triangle model to the reserve generated by Product copula model, Gaussian copula model, Frank copula model and Student's t copula model in Table 6.1. The deep triangle model generates the reserve within 3 percent difference from the Gaussian copula model and Frank copula model.

Table 6.1: Reserve comparison of various models. DeepTriangle is abbreviated to DT.

	DT vs Product	DT vs Gaussian	DT vs Frank	DT vs Student's t
Percentage difference	3.09%	2.57%	3.75%	9.03%

The deep triangle model predicts similar reserves to the reserves from Gaussian copula and Frank copula models. They all use the same loss triangles from two lines of business. However we do not fully understand why the deep triangle yields comparable results with the other models. We think that is the case because of the way the upper triangle data is organized as training (input and output) data and

testing data. The original upper triangle data is rearranged as 45 training samples and 9 testing samples. Each sample consists of 9 time steps and 2 components. A mask value is used if there is no data at that time step. The model is fit on the training data with a maximum of 1000 epochs. We run the model on the testing data and outputs predictions for the loss amounts in the lower part of the triangles. The data with development lag less or equal than 10 years are kept as the predicted output. Whereas deep triangle uses the aggregate claims triangles, another emerging research field is on the modeling of loss reserves using micro-level approaches. With the time series of reserves and payments on each individual claim, future research may involve studying the use of each individual claim fed into the machine learning models.

Appendix A

R codes

Here we present the source code for the numerical experiments.

```
library(GJRM)
library(reshape)
library(ChainLadder)
library(MASS)

data1 <- read.csv("personal_auto.csv")
#replace NAs with 0
data1[is.na(data1)] <- 0
data1$year <- as.factor(data1$year)

triangle1 <- data1
triangle1[,3:12] <- data1[,3:12]/data1[,2]
#loss ratios
ratios1 <- melt(triangle1, id=c("year","premium"))
ratios1$variable <- as.numeric(ratios1$variable)

## log normal regression for personal
```

```
data1.prp=ratios1[which(ratios1$value>0),]
names(data1.prp) <- c("year","premium","dev","loss")
data1.prp$dev <- as.factor(data1.prp$dev)
data1.prp$premium <- as.numeric(data1.prp$premium)
#log transform
data1.prp$logloss <- log(data1.prp$loss)

data2 <- read.csv("commercial_auto.csv")
data2[is.na(data2)] <- 0
data2$year <- as.factor(data2$year)
triangle2 <- data2
triangle2[,3:12] <- data2[,3:12]/data2[,2]
#loss ratios
ratios2 <- melt(triangle2, id=c("year","premium"))
ratios2$variable <- as.numeric(ratios2$variable)

data2.prp=ratios2[which(ratios2$value>0),]
names(data2.prp) <- c("year","premium","dev","loss")
data2.prp$dev <- as.factor(data2.prp$dev)
data2.prp$premium <- as.numeric(data2.prp$premium)

#copular regression model and loglikelihood and AIC

copular_data_in <- data.frame(x1=data1.prp$year, x2=
   data1.prp$dev,
y1=data1.prp$logloss, y2=data2.prp$loss)
copular_md_n <- gjrm(list(y1 ~ x1+x2,
```

```
                              y2~ x1+x2),
                        data=copular_data_in,
#                        BivD = "N",
                        BivD = "F",
                        margins = c("LN","GA"),
                        Model="B")
conv.check(copular_md_n)
copular_md_n_summ <-  summary(copular_md_n)
# AIC
AIC(copular_md_n)
# loglikelihood
logLik(copular_md_n)
# scale and shape parameter
cbind(copular_md_n_summ$sigma1, (1/copular_md_n_summ$
   sigma2)^2)

data1.lm <-  lm(logloss~year+dev, data=data1.prp)

data1_out.prp=ratios1[which(ratios1$value==0),c(1,2,3)]
names(data1_out.prp) <- c("x1","premium","x2")
data1_out.prp$x2 <- as.factor(data1_out.prp$x2)

data2_out.prp=ratios2[which(ratios2$value==0),c(1,2,3)]
names(data2_out.prp) <- c("x1","premium","x2")
data2_out.prp$x2 <- as.factor(data2_out.prp$x2)

#prediction
```

```
data1_prd_ratio <- data.frame(ratio=exp(predict(copular_
   md_n,
eq=1, newdata=data1_out.prp[,c(1,3)], type="link"))) *
  exp(0.5*copular_md_n_summ$sigma1^2)
data1_reser <- sum(data1_prd_ratio * data1_out.prp$
   premium)

data2_prd_ratio <- data.frame(ratio=exp(predict(copular_
   md_n,
 eq=2, newdata=data2_out.prp[,c(1,3)], type="link")))
data2_reser <- sum(data2_prd_ratio * data2_out.prp$
   premium)

#bootstrap
n_boot <- 5000
reser_boot <- matrix(0, nrow = n_boot, ncol = 1)
# personal auto line
for (i in 1:n_boot) {
  u <- runif(55)
  data1_prd_ratio <- qlnorm(u, meanlog = data1.lm$fitted
     .values, sdlog = data1.lm.summ$sigma*sqrt(36/55))
  data1_in_boot <- data.frame(year=data1.prp$year, dev=
     data1.prp$dev, y1=log(data1_prd_ratio))
  data1_lm_boot <-  lm(y1~year+dev, data=data1_in_boot)
  data1_lm_boot_summ <-  summary(data1_lm_boot)

  u <- runif(45)
```

```
  data1_prd_ratio <- qlnorm(u, meanlog = predict(data1_
     lm_boot, newdata=data1_out.prp[,c(1,3)]), sdlog =
     data1_lm_boot_summ$sigma*sqrt(36/55))
  data1_reser_b <- sum(data1_prd_ratio * data1_out.prp$
     premium)

  reser_boot[i] <- data1_reser_b
}
reser_boot_data1 <- data.frame(reserve=reser_boot)
print(c(mean(reser_boot_data1$reserve))

#commercial auto line
for (i in 1:n_boot) {
  u <- runif(55)
  data2_prd_ratio <- qgamma(u, shape=data2.gm.shape,
                            scale = data2.gm$fitted.
                               values/data2.gm.shape)

  data2_in_boot <- data.frame(year=data2.prp$year, dev=
     data2.prp$dev, loss=data2_prd_ratio)
  data2.gm.boot <-  glm(loss~year+dev, family = Gamma(
     link="log"),data=data2_in_boot)

  data2.gm.boot.shape <- gamma.shape(data2.gm.boot)$
     alpha

  u <- runif(45)
```

```
  data2_prd_ratio <- qgamma(u, shape=data2.gm.boot.shape
     ,
                            scale = exp(predict(data2.gm
                               .boot, newdata=data2_out.
                               prp[,c(1,3)], type="link"
                               ))/data2.gm.boot.shape)
  data2_reser_b <- sum(data2_prd_ratio * data2_out.prp$
     premium)

  reser_boot[i] <- data2_reser_b
}
reser_boot_data <- reser_boot_data1 + reser_boot_data2

print(c(mean(reser_boot_data$reserve),sd(reser_boot_data
   $reserve))

n_boot <- 5000
reser_boot <- matrix(0, nrow = n_boot, ncol = 2)
for (i in 1:n_boot) {
  u <- rCopula(55,normalCopula(copular_md_n_summ$theta,
     dim=2))
  data1_prd_ratio <- qlnorm(u[,1], meanlog = copular_md_
     n$eta1, sdlog = copular_md_n_summ$sigma1)

  data2.gm.shape <- (1/copular_md_n_summ$sigma2)^2
  data2_prd_ratio <- qgamma(u[,2], shape=data2.gm.shape,
```

```
                                scale = exp(copular_md_n$
                                   eta2)/data2.gm.shape)

  copular_data_in_boot <- data.frame(x1=data1.prp$year,
     x2=data1.prp$dev, y1=data1_prd_ratio, y2=data2_prd_
     ratio)
  copular_md_boot <- gjrm(list(y1 ~ x1+x2,
                               y2~ x1+x2),
                          data=copular_data_in_boot,
                          #BivD = "N",
                          BivD = "F",
                          margins = c("LN","GA"),
                          Model="B")
  copular_md_boot_summ <-  summary(copular_md_boot)
  u <- rCopula(45,normalCopula(copular_md_n_summ$theta,
     dim=2))
  data1_prd_ratio <- qlnorm(u[,1], meanlog = predict(
     copular_md_boot, eq=1,
newdata=data1_out.prp[,c(1,3)], type="link"), sdlog =
   copular_md_boot_summ$sigma1)
  data1_reser <- sum(data1_prd_ratio * data1_out.prp$
     premium)

  data2.gm.shape <- (1/copular_md_boot_summ$sigma2)^2
  data2_prd_ratio <- qgamma(u[,2], shape=data2.gm.shape,
                                scale = exp(predict(copular_
                                   md_boot, eq=2, newdata=
```

```
                                    data2_out.prp[,c(1,3)],
                                    type="link"))/data2.gm.
                                    shape)
  data2_reser <- sum(data2_prd_ratio * data2_out.prp$
     premium)
  reser_boot[i,1] <- data1_reser
  reser_boot[i,2] <- data2_reser
}
reser_boot_total <- data.frame(reserve=reser_boot[,1]+
   reser_boot[,2])

# Kuo's deep triangle model use gated recurrent units (
   GRU) to handle time series of incremental paid loss
   and claims outstanding. Code for the model is
   available online https://github.com/kasaai/
   deeptriangle

#GRU layer is appropriate for sequential data. The
   function can be found in Keras library. The layer_gru
    function applies a gated recurrent unit calculation
   to the input object.

layer_gru(
  object,
  units,
  activation = "tanh",
  recurrent_activation = "hard_sigmoid",
```

```
  use_bias = TRUE,
  return_sequences = FALSE,
  return_state = FALSE,
  go_backwards = FALSE,
  stateful = FALSE,
  unroll = FALSE,
  reset_after = FALSE,
  kernel_initializer = "glorot_uniform",
  recurrent_initializer = "orthogonal",
  bias_initializer = "zeros",
  kernel_regularizer = NULL,
  recurrent_regularizer = NULL,
  bias_regularizer = NULL,
  activity_regularizer = NULL,
  kernel_constraint = NULL,
  recurrent_constraint = NULL,
  bias_constraint = NULL,
  dropout = 0,
  recurrent_dropout = 0,
  input_shape = NULL,
  batch_input_shape = NULL,
  batch_size = NULL,
  dtype = NULL,
  name = NULL,
  trainable = NULL,
  weights = NULL
)
```

```
Arguments

units: dimensionality of the output space, use 128 in
   the model
activation: activation function to use. Default:
   hyperbolic tangent ("tanh")
use_bias: whether the layer uses a bias vector
dropout: fraction of the units to drop for the linear
   transformation of the inputs, use 0.2 in the model
recurrent_dropout: Fraction of the units to drop for the
    linear transformation of the recurrent state, use
   0.2 in the model

Input shapes

3D tensor with shape (batch_size, timesteps, input_dim)

Output shape
if return_state: a list of tensors. The first tensor is
   the output. The remaining tensors are the last states
   , each with shape (batch_size, units).

if return_sequences: 3D tensor with shape (batch_size,
   timesteps, units).
```

Appendix B

VaR of the silo model

VaR of the silo model

Proof. The variance of the portfolio is

$$\sigma_p^2 = \sigma_1^2 + \sigma_2^2 + 2\rho_{1,2}\sigma_1\sigma_2,$$

where σ_p^2, σ_1^2 and σ_2^2 are the variances of the portfolio, subportfolio 1 and subportfolio 2, respectively and $\rho_{1,2}$ is the correlation coefficient between subportfolio 1 and 2.

The silo model assumes that the correlation coefficient is 1. Then we have

$$\sigma_p^2 = \sigma_1^2 + \sigma_2^2 + 2\sigma_1\sigma_2,$$

which can be simplified as

$$\sigma_p = \sigma_1 + \sigma_2$$

The percentage $\text{VaR}_{95\%,p}$ can be stated as $1.645\sigma_p$. Moreover, the 95% VaR for subportfolio 1 (2) can be denoted as $\text{VaR}_{95\%,1}$ ($\text{VaR}_{95\%,2}$) and can be expressed as $1.645\sigma_1$ ($1.645\sigma_2$). Multiplying both sides of the equation above by 1.645 yields

the portfolio's VaR as follows:

$$\mathrm{VaR}_{95\%,p} = \mathrm{VaR}_{95\%,1} + \mathrm{VaR}_{95\%,2}$$

□

Appendix C

Result from Mack chain-ladder model

We also apply the Mack chain-ladder model on the same data in chapter 5. The predicted full triangle of cumulative paid loss is shown in Table C.1.

Table C.1: Full triangle of cumulative paid loss

	1	2	3	4	5	6	7	8	9	10
1988	54699	108337	143899	164818	179538	185391	188023	189759	190520	193499
1989	60091	119366	151151	174665	185469	192213	196152	198013	199997	203124
1990	65860	130803	172390	197977	210230	219267	222428	224078	225664	229193
1991	61946	121108	158880	182689	195247	201854	204911	206683	208146	211401
1992	65043	128550	164433	187508	199823	208008	211339	213167	214676	218033
1993	72295	144579	185446	208388	219345	227579	231223	233223	234874	238547
1994	81988	151197	189630	212446	226458	234959	238721	240786	242490	246282
1995	83207	152470	190974	217620	231973	240681	244535	246650	248396	252280
1996	79699	143590	184420	210151	224011	232421	236142	238185	239871	243622
1997	75827	145628	187038	213134	227191	235720	239494	241566	243276	247080

The predicted full triangle of cumulative paid loss from Deep Triangle model is shown in Table C.2.

We compared these two triangles and calculated the percentage difference as in Table C.3.

Table C.2: Full triangle of cumulative paid loss

	1	2	3	4	5	6	7	8	9	10
1988	54699	108337	143899	164818	179538	185391	188023	189759	190520	193499
1989	60091	119366	151151	174665	185469	192213	196152	198013	199997	201853
1990	65860	130803	172390	197977	210230	219267	222428	224078	226132	227470
1991	61946	121108	158880	182689	195247	201854	204911	207160	208698	210010
1992	65043	128550	164433	187508	199823	208008	211063	213243	215125	216826
1993	72295	144579	185446	208388	219345	223848	226805	229219	231274	233079
1994	81988	151197	189630	212446	223231	229479	233183	235847	237951	239715
1995	83207	152470	190974	209251	219291	224970	228342	230745	232624	234177
1996	79699	143590	178730	200973	213420	220358	224292	226970	229091	230982
1997	75827	146895	189836	217183	231590	239608	243648	246934	250130	253317

Table C.3: percentage difference

	1	2	3	4	5	6	7	8	9	10
1988	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
1989	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	-0.63%
1990	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.21%	-0.75%
1991	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.23%	0.27%	-0.66%
1992	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	-0.13%	0.04%	0.21%	-0.55%
1993	0.00%	0.00%	0.00%	0.00%	0.00%	-1.64%	-1.91%	-1.72%	-1.53%	-2.29%
1994	0.00%	0.00%	0.00%	0.00%	-1.42%	-2.33%	-2.32%	-2.05%	-1.87%	-2.67%
1995	0.00%	0.00%	0.00%	-3.85%	-5.47%	-6.53%	-6.62%	-6.45%	-6.35%	-7.18%
1996	0.00%	0.00%	-3.09%	-4.37%	-4.73%	-5.19%	-5.02%	-4.71%	-4.49%	-5.19%
1997	0.00%	0.87%	1.50%	1.90%	1.94%	1.65%	1.73%	2.22%	2.82%	2.52%

Bibliography

Abdallah, A., Boucher, J.-P., and Cossette, H. (2015). Modeling dependence between loss triangles with Hierarchical Archimedean copulas. *ASTIN Bulletin*, 45(3):577–599.

Abdallah, A., Boucher, J.-P., Cossette, H., and Trufin, J. (2016). Sarmanov family of bivariate distributions for multivariate loss reserving analysis. *North American Actuarial Journal*, 20:1–17.

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723.

Côté, M.-P., Genest, C., and Abdallah, A. (2016). Rank-based methods for modeling dependence between loss triangles. *European Actuarial Journal*, 6.

De Jong, P. and Heller, G. Z. (2008). *Generalized Linear Models for Insurance Data*. Cambridge University Press.

Efron, B. and Tibshirani, R. J. (1993). An introduction to the bootstrap. *Chapman and Hall*.

England, P. and Verrall, R. (1999). Analytic and bootstrap estimates of prediction errors in claims reserving. *Mathematics and Economics*.

England, P. D. and Verrall, R. J. (2002). Stochastic claims reserving in general insurance. *British Actuarial Journal*, 8/3:443–518.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366.

Kuo, K. (2019). Deeptriangle: A deep learning approach to loss reserving. *Risks*, 7(3):97.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *In Proceedings of the Advances in Neural Information Processing Systems 30, Long Beach, CA, USA, December 4-9*.

Lowe, J. (1994). A practical guide to measuring reserve variability using bootstrapping, operational times and a distribution-free approach. *General Insurance Convention, Institute of Actuaries and Faculty of Actuaries*.

Mack, T. (1993). Distribution-free calculation of the standard error of chain ladder reserve estimates. *ASTIN Bull*, 23(2):213–225.

Mack, T. and Venter, G. (2000). A comparison of stochastic models that reproduce chain ladder reserve estimates. *Mathematics and Economics*.

Marra, G. and Radice, R. (2019). GJRM: generalised joint regression modelling. *R package version 0.2*.

McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

Meyers, G. and Shi, P. (2011). *Loss Reserving Data Pulled from NAIC Schedule P*. `http://www.casact.org/research/index.cfm?fa=loss_reserves_data`.

Nelder, J. A. and Wedderburn, R. W. (1972). Generalized linear models. *Journal of the Royal Statistical Society*, 135:370–384.

Nelsen, R. B. (2006). *An Introduction to Copulas (Springer Series in Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Newman, M. (2018). *Networks*. Oxford University Press.

Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. Determination Press.

Reddi, S. J., Kale, S., and Kumar, S. (2019). On the convergence of adam and beyond.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464.

Shi, P. and Frees, E. W. (2011). Dependent loss reserving using copulas. *ASTIN Bulletin*.

Taylor, G., McGuire, G., and Sullivan, J. (2008). Individual claim loss reserving conditioned by case estimates. *Annals of Actuarial Science*, 3(1-2).

Verrall, R. (2000). An investigation into stochastic claims reserving models and the chain-ladder technique. *Insurance: Mathematics and Economics*.

Verrall, R. and England, P. (2000). Comment on: A comparison of stochastic models that reproduce chain ladder reserve estimates, by Mack and Venter. *Mathematics and Economics*.

Werbos, P. (1975). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78:1550–1560.

Wüthrich, M. and Merz, M. (2008). Stochastic claims reserving methods in insurance. *John Wiley & Sons*.