2017

REPORT

Individual Claim Development with Machine Learning

Working party members

Authors of the report

Bor Harej (champion)
Roman Gächter
Salma Jamal

Other members

Alexander Schaeper
Alexandre Boumezoued
Andrew McGuinness
Erik Gustafsson
Frank Cuypers
Greg Taylor
Ivan Valdes
Klaus Krøier
Maria Inês Silva
Mario V. Wüthrich
Martin Cairns
Niels Rietdorf
Peter England
Rocco Roberto Cerchiara
Sabine Betz
Stefan Mueck
Tetiana Korzhynska
Tom van den Vorst
Vittorio Magatti

Acknowledgment

Special thanks to Frank Cuypers for mentorship and to Greg Taylor for valuable insights.

Contents

I. Abstract

The report summarizes the work accomplished by a team of people that met within the scope of an ASTIN working party named Individual Claim Development with Machine Learning. The main goal of the working party was to research the field of machine learning in connection with reserving as traditional actuarial work. The report presents some remarkable insights the group managed to discover.

The machine learning technique used, for instance, artificial neural networks. They were implemented in a cascading triangular way similar to triangular reserving methods, and the prediction results were compared with results achieved by classical reserving methods. The findings offer a better understanding of possible complexity of the nature of the claims, point out some weaknesses that traditional methods might have, and indicate a strong potential for machine learning algorithms.

II. Introduction

For more than a century, actuaries have developed non-life losses with the help of aggregate claims triangles, to which they applied techniques such as Chain-Ladder. Undoubtedly this methodology was extremely efficient in a time without computers, but with the availability of today's computing, the loss of information resulting from the aggregation of the individual claims data into accident & development year buckets is barely justifiable.

Modern machine learning techniques (ML) may offer better estimates of provisions, and it is reasonable that our profession should invest more into this research area. In particular, the pattern recognition capabilities of neural networks may provide completely new insights into long tail claims reserving and pricing. This was one of the main reasons to organize a working party to address this issue. So, the main purpose of the working party was to investigate the extent to which machine learning could be useful in reserving and whether it is worthy of additional research. The report shows how the working party approached the challenge, provides a simple explanation of the methodology used, and shows the main findings. Further on, there is a step-by-step overview of the work done.

In order to start solving a simple problem, first we decided to use data with minimal granularity. A good practice in machine learning exercises is to commence with a test against synthetic data, where the correct results are known. Only then should real data be tackled. At the same time, we started the research on machine learning techniques with the available software packages and applicable methodology for individual claim development.

We decided to use programming language R and its freely available neural networks packages. We applied a methodology that closely followed the traditional triangular aggregate data structure approach such as Chain-Ladder approach, but on an individual claim basis[1]. We succeeded in implementing the simple neural network triangular model and calibrating it. For an alternative calculation and better overview over the calculation process, we implemented the same model also in Excel. The

[1] We followed an approach patented by Frank Cuypers [11].

results of predictions on simulated data gave us a strong indication that the neural networks could outperform Chain-Ladder even with the use of data of minimal granularity. At this point we can say we achieved our main goal. The report includes all the steps mentioned above, including the explanation of what neural networks are and how were they applied to individual claim development.

III. Methodology

There is a number of possible methods to approach the task of using machine learning methods to estimate the development of insurance claims on an individual claims basis. Due to their capabilities in recognizing patterns, we have decided to employ artificial neural networks (ANNs); more specifically multilayer perceptrons (MLP).

ANNs belong to a group of ML methods that are called “supervised learning” methods, since they need to be “trained” with a given data set (training data) before they can be used to perform predictive tasks.

1. ANN/MLP Structure

“A multilayer perceptron (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs. An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. MLP utilizes a supervised learning technique for training the network.”[2]

Kaastra and Boyd [1] underlines the fact that “In theory, a (MLP) with one hidden layer with a sufficient number of hidden neurons is capable of approximating any continuous function.”

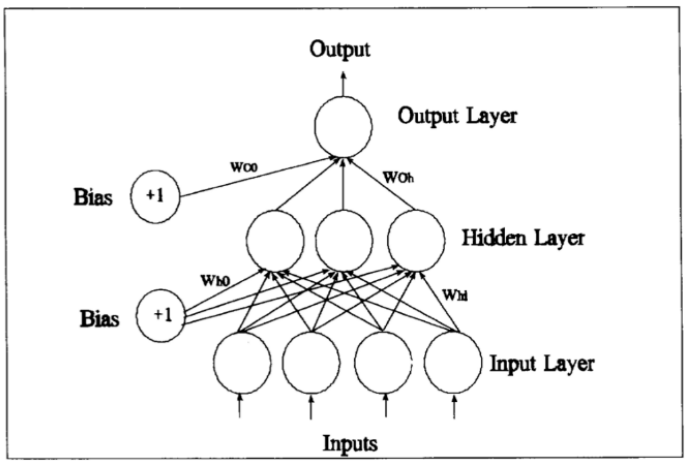

Figure 1 Graphical representation of a MLP with one hidden layer.

Figure 1 [1] shows a graphical representation of such a MLP with one hidden layer. The output layer, consisting of the output node(s) represents the information or data that is to be predicted, based on the available information which is fed into the input nodes.

[2] https://en.wikipedia.org/wiki/Multilayer_perceptron, 27. May 2017

A MLP (and more generally a neural network) is characterized by:

- **The purpose of its building**: supervised/ unsupervised learning, non-learning;

- **Its topology**: feedforward[3] or feedback[4]/recurrent network;

- **Its architecture**: MLP, RBF (Radial Basis Function),… It includes several parameters which do not necessarily have to be initialized: the momentum, the bias, the learning rate and pace, the early stopping criteria,…

- **Its number of hidden neurons/layers**:
  The number of neurons can depend on the architecture of the network, on the learning algorithm,... Several theories exist on the gradual adjustment of this number (Blum [2], Swingler [3], Berry and Linoff [4],...), the most common one consisting of testing many numbers and choosing the one that minimizes the error rate. In this study, the number of hidden layers / neurons was subject to sensitivity tests. The aim was to find the optimal number of neurons / hidden layers optimal for the final model.

- **Its combination and activation functions**:

  - At the level of the input layer, each node is connected to synaptic weights. Weights are then linked to the nodes of the following layer via a **combination** function. This last is generally a weighted sum of the node values by the synaptic weights to which they are linked, and are usually designated by ϑ and called **potential** of the neural network.

  - An **activation** function will, from these sums, assess the activity of each node. Among the most popular activation functions, we find the following ones:

	Activation function
Identity	$g(x) = x$
Linear Threshold function	$g(x) = \begin{cases} 0, & x \leq x_{min} \\ \alpha x + \beta, & x_{min} < x < x_{max} \\ 1, & x \geq x_{max} \end{cases}$
Sigmoid function	$g(x) = \frac{1}{1 + e^{-\alpha x}}$
Hyperbolic tangent	$g(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$
Gaussian function	$g(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

Table 1 List of popular activation functions

Although no formal rule exists to impose the choice of a specific activation function, the sigmoid function and the hyperbolic tangent are

[3] In a feed-forward network, the information moves in one direction: from the input layer to the output layer, via the eventual hidden layers. These kinds of networks are also called acyclic networks as they are static (they do not depend on time).

[4] Recurrent networks use the response given by a node as an input for the nodes of the next layer. Such an operation is possible only if the notion of time is taken into account. In this sense, the neuron output used as an input goes necessarily along with a strictly positive time delay factor. These networks are dynamic.

the most commonly used, particularly in the case of the backpropagation algorithm. This algorithm requires functions whose composition maintains the characteristics of continuity and differentiability.

- The interest of the sigmoid function lies in the fact that its derivative is very easy to calculate:

$$g'(x) = \alpha . \frac{e^{-\alpha x}}{1 + e^{-\alpha x}} = \alpha . g(x) . \big(1 - g(x)\big) \qquad (1)$$

  The sigmoid function gives a response on the space [0, 1],

- The hyperbolic tangent makes it possible to obtain a response on the space [-1, + 1].

The use of a bipolar approach facilitates learning, insofar as, compared to a binary approach, it tends to better "homogenize" the research [5] space.

The sigmoid function and the hyperbolic tangent are perfectly adapted to the gradient descent principle on which backpropagation is based. In either case, the function is strictly monotone: the calculation of the gradient thus makes it possible to obtain a reliable direction.

Furthermore, the activation value of the output neuron constitutes the final result. In the case of regression, for instance, the output neuron is linear.

- **Its learning algorithm**

To avoid the overlearning phenomenon and to optimize the generalization capacities of the model, we have to determine learning rules for it. The learning algorithm formalizes these rules. It minimizes the cost function, which represents the difference between the observed outputs and those predicted by the model. This function gives an idea on the accuracy and the quality of the model.

In the case of regression, the cost function is equal to the mean squared error[5]:

$$C(w) = \sum_{\alpha} (y_\alpha - \hat{y}_\alpha)^2 \qquad (2)$$

[5] Or to, more rarely, the cross entropy:

$$C(w) = -\sum_{\alpha} (y_\alpha \ln(\hat{y}_\alpha) + (1 - y_\alpha)\ln(1 - \hat{y}_\alpha)) \qquad (3)$$

where w is the weights vector, $\hat{y}_\alpha$ are the outputs of the model for the α^{th} neuron and the y_α represent the observed outputs. Outputs are supposed to be Gaussian distributed.

A learning algorithm performs according to its cost function. In the space of the cost function arguments, the algorithm looks for the direction that minimizes this function. It moves in this direction and repeats the same operation until reaching a specific stopping criterion. Each algorithm has its own descent direction. The process of application of a learning algorithm is called backpropagation.

There are several learning algorithms: Levenberg-Marquardt, gradient descent algorithm,…

In this study, we analyzed the performance of two learning algorithms: the backpropagation algorithm (BA) and the scaled conjugate gradient backpropagation (SCG).

- The BA[6] is a first order learning method. It is based on:
  - A propagation step, where the input of each neuron is computed starting from the neurons of the previous layer, using appropriate combination and activation functions;
  - A backpropagation step where a recursive calculation of partial gradients of the cost function is performed. The sum of these partial gradients enables one then to obtain the global cost function of the model.

- The SCG [6] [7] is a second order learning method, in other words: an iterative gradient descent algorithm, where the error function is replaced by its quadratic approximation in the neighbourhood of the point concerned. It is derived from the Newton method, but presents more complexity in its building.

  The SCG algorithm is based on the assumption of a quadratic error and relies on the following observation: in the gradient descent algorithm, the same direction is followed unnecessarily a great number of times. The solution suggested by this learning rule is to reduce the number of iterations by looking for an optimal direction of descent. Each descent is iterated in such a way as to be conjugate to the preceding one.

- **Its weights**

  The link between two neurons is called synaptic weight. This weight relates to the information shared between neurons. At the level of the input layer, each node is connected to synaptic weights. Weights are then linked to the nodes of the following layer via the combination function.

  The weights can be randomly initialized since the backpropagation algorithm will adjust them throughout the learning process.

[6] Please refer to Appendix B for more details

We are aware of MLP limits:

- Algorithmic complexity:
  The backpropagation algorithm is especially known for the slowness of its convergence as soon as the problem becomes a little complex. Hinton [8] has shown that, empirically, the learning time on a sequential machine is $\mathcal{O}(N^3)$, where N is the number of connections in the network.

- Overlearning:
  Overlearning is a drawback of neural networks in general. Most often, this phenomenon occurs when a model scans data a significant number of times until it becomes imbued with all the characteristics of the data, particularly of exceptional features.

  It thus retains these exceptional features, as well as regular behaviours, and generalizes them to the whole database. This phenomenon can be encountered in the case of small datasets, with many explanatory variables, ...

  Cross-validation[7] can reduce the risk of overlearning by finding the network that minimizes the most the generalization error of the model on the test sample. Moreover, model aggregation is also a method which makes it possible to control this risk.

2. Cascading

The first modelling approach requires a cascade of ANNs and may be considered as related to Chain-Ladder (cf. subsection “Approach with one ANN/MLP”).

Figure 2 depicts a claims triangle, containing information about single claims. This example contains both paid amounts (Pd) and reserved amounts (Res) and the indices represent underwriting (UY) or accident year (AY), development year (DY) and claim number.

[7] Please refer to section Cross validation

		DY							
		Pd 1	Res 1	Pd 2	Res 2	Pd 3	Res 3	Pd 4	Res 4
UY	2011	P(1,1,1)	R(1,1,1)	P(1,2,1)	R(1,2,1)	P(1,3,1)	R(1,3,1)	P(1,4,1)	R(1,4,1)
		P(1,1,2)	R(1,1,2)	P(1,2,2)	R(1,2,2)	P(1,3,2)	R(1,3,2)	P(1,4,2)	R(1,4,2)
		P(1,1,3)	R(1,1,3)	P(1,2,3)	R(1,2,3)	P(1,3,3)	R(1,3,3)	P(1,4,3)	R(1,4,3)
		P(1,1,4)	R(1,1,4)	P(1,2,4)	R(1,2,4)	P(1,3,4)	R(1,3,4)	P(1,4,4)	R(1,4,4)
	2012	P(2,1,1)	R(2,1,1)	P(2,2,1)	R(2,2,1)	P(2,3,1)	R(2,3,1)		
		P(2,1,2)	R(2,1,2)	P(2,2,2)	R(2,2,2)	P(2,3,2)	R(2,3,2)		
		P(2,1,3)	R(2,1,3)	P(2,2,3)	R(2,2,3)	P(2,3,3)	R(2,3,3)		
	2013	P(3,1,1)	R(3,1,1)	P(3,2,1)	R(3,2,1)				
		P(3,1,2)	R(3,1,2)	P(3,2,2)	R(3,2,2)				
		P(3,1,3)	R(3,1,3)	P(3,2,3)	R(3,2,3)				
	2014	P(4,1,1)	R(4,1,1)						
		P(4,1,2)	R(4,1,2)						

Figure 2 Claims and reserves triangle with single claims data P(i, j, k) and single reserves data R(I, j, k) where i represents underwriting or accident year, j represents development year and k represents claim number.

To predict the amounts for the lower triangle, we train ANNs (MLPs) with the information available (cf. top row in Figure 3), and use the trained ANNs (MLPs) to predict amounts in the lower triangle (cf. bottom row in Figure 3).

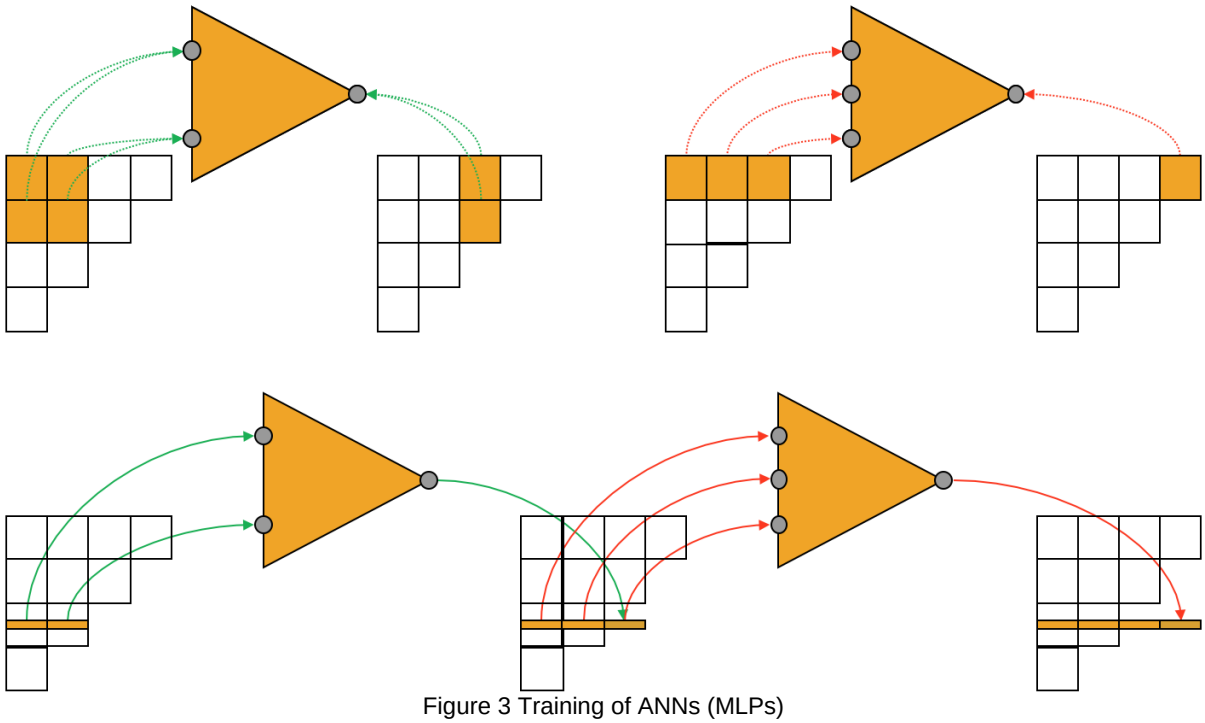

Figure 3 Training of ANNs (MLPs)

In order to predict all of the amounts in the lower triangle, we start with the ANN (MLP) that is trained with the step from DY1 to DY2, using the information from the oldest to

the second-newest UY (and label it ANN1 (MLP1)). This ANN1 (MLP1) is then used to predict the amounts in DY2 for the newest UY.

Then, a second ANN (MLP) is trained with the original information for the step from DY2 to DY3, i.e. using the information from the oldest to the third-newest UY. This second ANN (MLP) will then predict amounts in DY3 for the second-newest and newest UY.

This cascade is repeated until the lower triangle is completed (cf. Figure 4).

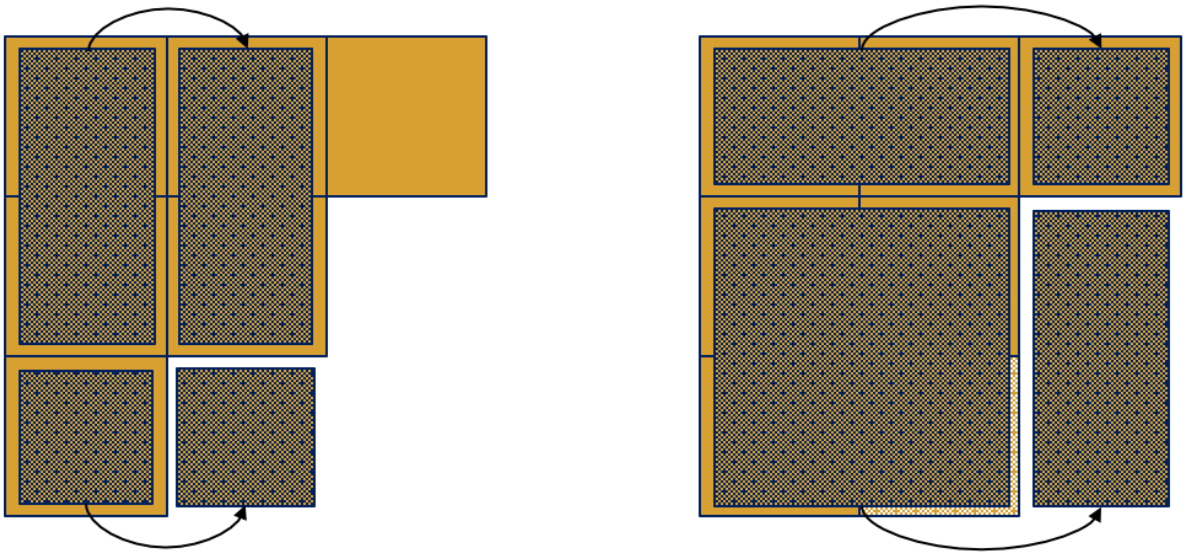

Figure 4 Cascading DY

Since only information is used that was available in the original triangle, the last few ANNs (MLPs), relating to the last UYs, will be trained on a relatively small data set, depending on the number of claims in the earlier UYs.

3. Cross validation

The quality of the predictions of any kind of trained structure in machine learning is (potentially highly) dependent on the available information with which to train the structure.

Indeed, prior to the generalization of the models to the whole data set, one should ensure their validity. The first idea would be to make a second study, independent of that carried out on the learning data set, in order to check if the first results are confirmed. This technique is conceivable in the case of large data sets. However, in reality, whatever the volume of the data set, it can be very time consuming.

There is a process that can overcome this issue by relying only on one data set: self-validation. There are several self-validation methods: simple validation, cross validation, "leave-one-out" validation (or jackknife), bootstrap, etc.

A cross-validation technique consists of:

- separating the available data into a training set and a test set;

- subdividing the first data set into D distinct subsets (typically D = 5);
- selecting one of the D sub-samples as the validation set, and allowing the other D-1, in aggregate, to form the training set;
- performing the learning step on the training set, and generating model, or fitted, values corresponding to the observations in the validation set;
- iterating the last two steps D times, in such a way that each of the D sub-samples is used once and only once as the validation subset;
- calculating the sum of the squares of the errors on the validation set;
- calculating the score of the cross-validation

$$\sqrt{\frac{1}{N}\sum_{i=1}^{D} S_i} = \sqrt{\frac{1}{N}\sum_{i=1}^{D}\sum_{\substack{k \in a\ subset \\ validation\ I}} \left(y_k - g(x_k, w_i)\right)^2} \qquad (2)$$

where

- N represents the size (number of rows) of the data set
- y_k represents the target variable (observation)
- w_i represents the parameters of the model
- x_k represents the vector of the variables of the model
- $g(x_k, w_i)$ is the model's fitted value corresponding to x_k, for the parameters w_i and given the variables x_k

- selecting the model for which the cross validation score is optimal in some sense. If several models of different complexities are candidates for choice because their RMSE (Root Mean Square Error) is small and of the same order of magnitude, choosing the one with the lowest complexity.

Once the optimal model is determined, the learning algorithm is applied to the whole data set (all D subsets in aggregate) used previously for the cross-validation. The performance of the model obtained is estimated on the data kept for the test.

Then, the same operation is repeated $D - 1$ times on each $\frac{D-1}{D}^{\text{th}}$ possible sample, taking each $\frac{1}{D}^{\text{th}}$ remaining as a test sample. Finally, the mean of the D RMSE obtained is computed as to estimate the error rate of the model built on the whole data set.

Application to the study:

In this study, as we relied on synthetic data, the size of the data set has not been an issue. Therefore, we could generate as much data as we needed and directly sample[8] the data set simulated.

4. Parameter optimization and sensitivity analysis

During the project we made several sensitivity analyses. We tested:

[8] Please refer to section Data sampling

- **Different numbers of neurons and hidden layers**. We even constructed a predictor function for a variable number of neurons and examined which choice gave the best prediction. However, at the end, we decided to use the few structures described under “Results” section. Optimal structure remains an open question.
- **Different activation functions**. We tested hyperbolic tangent and sigmoid functions for hidden layers. We didn’t notice much difference between them. For the final results, we used the hyperbolic tangent. We used a linear function on the output layer to obtain results in a form of regression function.
- **Different learning algorithms.** We tried to use two different algorithms: standard BA and SCG. SCG turned out to be more efficient to use, since it didn’t need to adjust learning rate as is the case with standard backpropagation. Some comparisons are presented in appendix A.
- **Stopping criteria.** It turned out that ANNs for early development years needed less learning iterations than ANNs for later development years. Some examples are presented in appendix A. The results below were predicted with ANNs where the learning loop for the first 7 development years’ predictions stopped after 500 iterations, the next 8 after 1000 iterations, and the rest after 5000 iterations.

One weakness of the modelling approach described in the subsection “Cascading” is that – much like classical Chain-Ladder – it is likely to perform well on data that mainly (only) shows development from DY to DY that is consistent over UYs, but also likely to perform poorly on data that shows e.g. varying development factors from UY to UY.

IV. Data

The data issue was crucial in this study. We first simulated synthetic data, but also reflected on the case of real data (See Figure 5).
The reasoning was the following:

- We first simulated as much data as we wanted (section Data simulations) ;
- Then we partitioned the data set into a learning sample and a test sample (section Data sampling).
- Finally, we created samples with individual patterns: paid claims (section Samples with individual patterns), and mixed patterns: cumulative amounts of paid claims and outstanding claims (section Samples with mixed patterns).

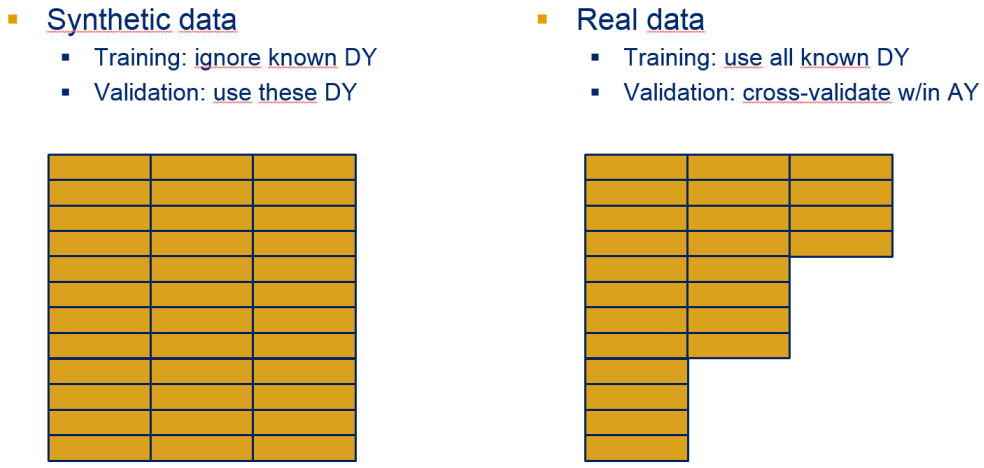

Figure 5 Synthetic Data vs Real Data training and validation steps

We present first the approach followed to generate the artificial data that were the first basis of this study.

1. Data simulations

We relied on numerical data: the amounts of individual claims. Therefore, we generated amounts of paid, outstandings and incurred.

Individual claims obey known distribution laws:

- Paid $P(t) = U \cdot F_P(t)$
- Outstandings $O(t) = U \cdot F_O(t)$
- Incurred $I(t) = P(t) + O(t)$

The variables U and $F(t)$ describe, respectively, the severity and the development patterns of the claims:

- The severity, at ultimate, follows a lognormal law $U \sim LN(\mu, \sigma)$
- The development patterns, age-to-ultimate, follow, at each time t, a lognormal distribution law $F(t) \sim LN(\mu_t, \sigma_t)$

In order to take into account the specifics of paid and outstandings development patterns (see Figure 6), we define the mean of the log-normal distribution in both cases:

$$\begin{cases} F_P(t) \sim LN\left(\left[1 - e^{-\frac{t-\tau}{\lambda}}\right]^{\alpha}, \sigma_t\right), for\ Paid \\ F_O(t) \sim LN\left(\alpha e^{-\left(\frac{t-\tau}{\lambda}\right)^2}, \sigma_t\right), \quad for\ Outstandings \end{cases}$$

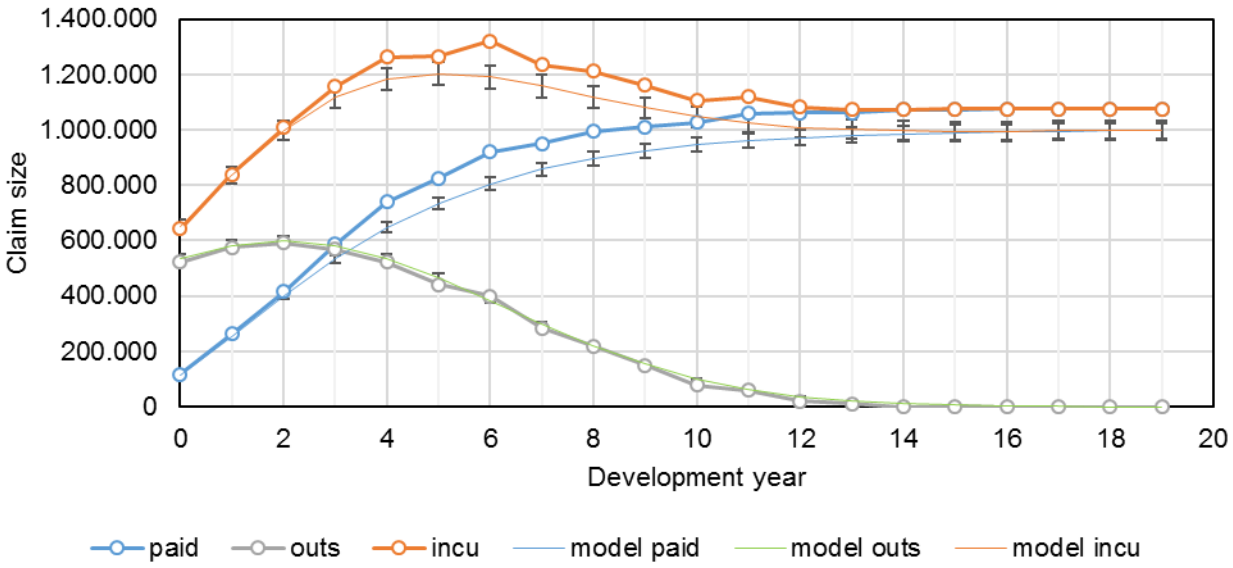

Figure 6 Simulation of the specific development patterns of paid, outstandings and incurred

Finally, in order to induce a dependency between the paid and the outstandings, we link these amounts by a Frank copula $F_P(t) \bowtie F_O(t)$ for each t (see Figure 7). The data model assumes higher dependency in the tail.

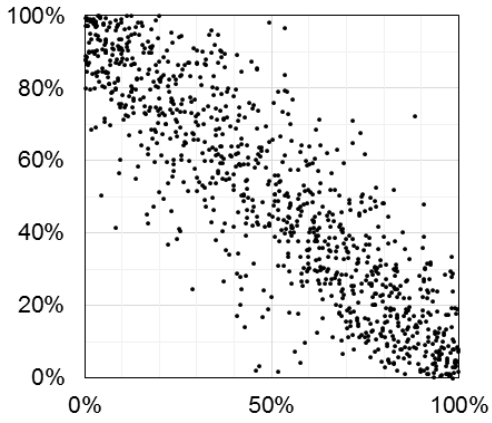

Figure 7 Frank Copula

By this process, we have generated as much data as we wanted (see Figure 8):

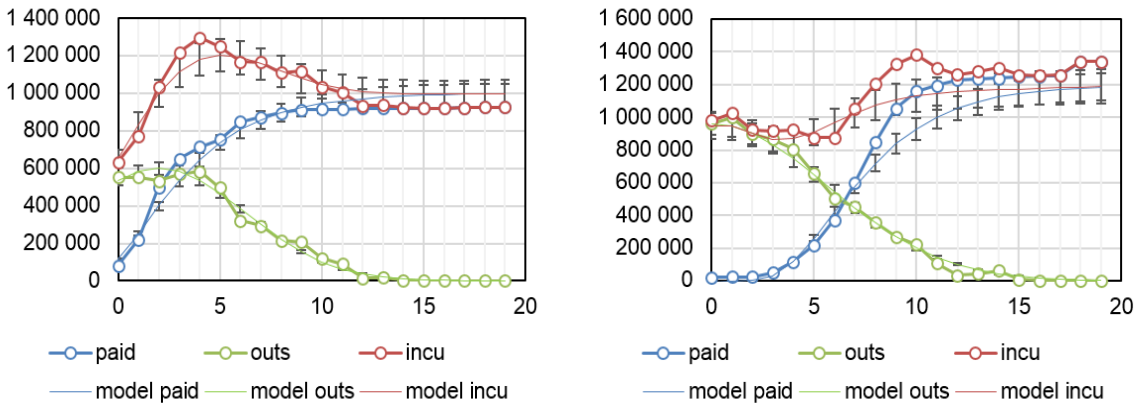

Figure 8 Simulation of different numbers of claims

The data set obtained contained, thus, 20 accident and development years. Each claim has a separate amount of paid and outstandings and each develops with its own distribution law, while maintaining a dependency through Frank's Copula.

2. Data sampling

Sampling is a statistical technique which consists in splitting a database in many samples which will be used for different purposes.

Usually, it is customary to select:

- a **learning sample** to train the model to learn the different structures that characterize the data set;
- a **test sample** to adjust the model parameters;
- a **validation sample** for forecasting purposes: in other words, check the generalization capabilities of the model.

Many sampling approaches exist. The one selected has to be relevant to the purpose of a study. The representativeness of the different populations of the database is crucial. Many learning tests exist to verify this representativeness: Student test, Pearson's Chi-Squared,...

The most basic sampling method consists of randomly drawing a number N of individuals in a database. It is also possible to constitute first partitions of the dataset, in alphabetical order of the family names of the policyholder, for example, and then randomly draw from each of these partitions (stratified sampling). In census studies, the census consists of making a deterministic choice (neighborhoods of a city to be surveyed,...) and then studying all the designated subgroups (cluster sampling). Other sampling techniques exist:

- Random ones: systematic, with a probability proportional to the size,... and
- Non-random ones: sampling by judgment, by quotas,...

In this study, the question of representativeness does not arise insofar as we rely on synthetic data that we generate as much as we need. We perform a random draw of data without replacement: 80% for the learning sample, 20% for the test sample. The test sample was used as both test and validation sample.

3. Synthetic data samples

Data samples were formed according to the method described above. To properly see the effects, we chose two different patterns, one to be more short-tailed, one long-tailed. That could correspond to material and non-material motor claim damages under motor third party liability.

4. Samples with individual patterns

For the short-tailed sample we simulated a sample of 6000 claims (Sample 1), and for a long tail we simulated 4000 claims (Sample 2). The underlying patterns of the two samples are shown in the picture.

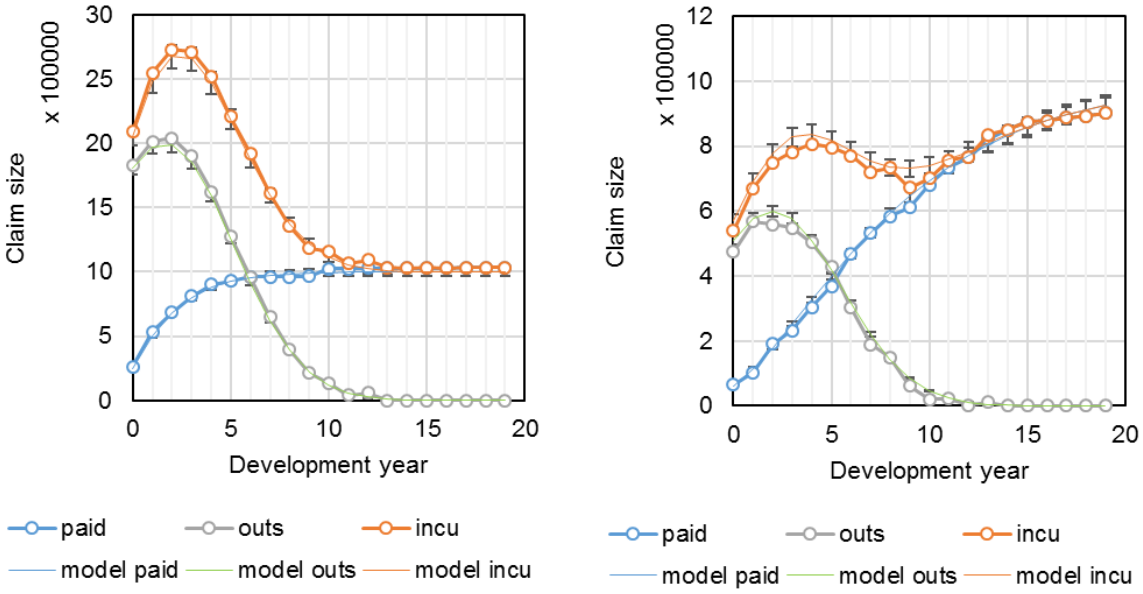

Figure 9 Synthetic data Sample 1 on the left and Sample 2 on the right

Both samples were calibrated to have a mean ultimate amount of paid claims of 1 million. Claim simulations were evenly allocated to 20 accident years. In Sample 1, with 6000 claims, there are 300 claims per accident year, in Sample 2, with 4000 claims, 200 claims per accident year. The next table shows the parameters of calibration of both samples. All standard deviations were set to be 2%.

	Short tail sample		Long tail sample	
	paids	outs	paids	outs
tau	-1,0	1,6	-3,0	2,0
lambda	2,0	5,0	6,0	5,0
alpha	1,5	2,0	3,0	0,6

Table 2 Parameters of calibration of Sample 1 and Sample 2

5. Samples with mixed patterns

We used Sample 1 and Sample 2 to produce three more samples, each with a total number of simulated claims equal to 10.000. Each of these three samples is a simple combination of the two samples above, where a Sample 2 with 4000 claims was modified by dividing each claim amount by 5. The only difference between three new samples is their respective allocations of claims to accident years. The next table presents these allocations.

	Sample 3		**Sample 4**		**Sample 5**	
Accident year	Sample 1 claims	Sample 2 claims	Sample 1 claims	Sample 2 claims	Sample 1 claims	Sample 2 claims
1997	300	200	15	390	280	220
1998	300	200	45	370	280	220
1999	300	200	75	350	280	220
2000	300	200	105	330	280	220
2001	300	200	135	310	280	220
2002	300	200	165	290	280	220
2003	300	200	195	270	280	220
2004	300	200	225	250	280	220
2005	300	200	255	230	280	220
2006	300	200	285	210	280	220
2007	300	200	315	190	280	220
2008	300	200	345	170	280	220
2009	300	200	375	150	280	220
2010	300	200	405	130	280	220
2011	300	200	435	110	280	220
2012	300	200	465	90	280	220
2013	300	200	495	70	280	220
2014	300	200	525	50	280	220
2015	300	200	555	30	460	40
2016	300	200	585	10	500	0

Table 3 Allocation of Sample 1 and Sample 2 to form Sample 3-5

We can describe the first mixed sample (Sample 3) as a sample which has a stable ratio between claims of different patterns. The next has a changing pattern from a high ratio of long tail claims towards high ratio short tail claims (Sample 4). The last sample (Sample 5) presents a sudden change in a ratio at last two AYs.

V. Technique

We used samples of simulated claims of 20 accident years and 20 development years. The starting point comprised only paid claims. The procedure for cascading predictions went in the following steps:

1. Individual claims data were taken as in input in R program,
2. Data were normalized (i.e. we computed data standard score) by each development year,
3. For each development year one ANN was calibrated,
4. Predictions were made,
5. Predictions were de-normalized.

We tested different methods implemented in an R environment ("nnet", "neuralnet" and "mlp"). The structure of an individual ANN had always one output neuron. The number of input neurons of individual ANNs was equal to the number of the development period, simulated with that particular ANN if we number the first development period as 0. For example, if

During the third step, we tested different activation functions, learning functions and we tried many different numbers of hidden layers and number of neurons. We tried the effect of different ANN for different development periods.

The following extension of R code enabled us to use both paid and outstanding amounts of individual claims as input. We modified ANNs in two ways. First, we doubled the number of input and output neurons, so that paid and outstanding amounts were accepted as input and also predicted as an output. Another option was to use, for each development period, two ANNs, one predicting paid claims, another outstanding claims, for use as an inputs to the ANN for the next pair of development periods.

Additional improvement was obtained by manipulation of the input data. Instead of using just cumulative paid and outstanding data, we adopted incremental development factors as the input data for all following development periods other than the first. The inputs were thus

$$\begin{cases} Claims\ paid\ and\ outstanding, for\ first\ development\ period \\ \dfrac{\text{Claims paid/outstanding in current year}}{\text{CLaims paid/outstanding in previous year}} - 1, \qquad for\ all\ other\ development\ periods \end{cases}$$

We could use this approach since all synthetic claims had nonzero payments and outstanding amounts in the first development period.

If the data from the first development year were to contain zero values for paid or outstanding claims, we could adjust all amounts by the addition of some fixed amount, use the same approach as before, and reverse the adjustment in the output predictions.

The above modifications were mostly driven by the effect that random variability had on the result. Taking account of all those modifications, we managed to predict results quite well.

VI. Results

The results section will cover artificial neural network (ANN) predictions, with the results compared with Chain-Ladder predictions. We sampled claims in such a way that all claims have positive payments in the first development year. This sets IBNR to zero and enables us to study only ultimate severity of the claims.

We used the basic triangular method Chain-Ladder to test quality of the prediction made with ANN. We looked at an overall estimation of reserves in comparison with the true value of unpaid claims, the size of the overall error of prediction of individual ultimate paid claims, and how the errors are distributed by accident year.

1. Samples with individual patterns

We started analysis using only paid claims, and later added outstanding claims as an input. The total procedure of claim prediction with the ANN is explained in detail under section Techniques. It turned out that adding outstanding claims as an input didn't always improve predictions.

Sample 1 (short tail sample with 6000 claims)

The next charts present a comparison of sum of errors of individual ultimate claim predictions according to the Chain-Ladder and ANN respectively. The Chain-Ladder calculates only one average pattern, which is applied to all claims, whereas the ANN predicts an individual pattern for each claim. We used original data without any transformation.

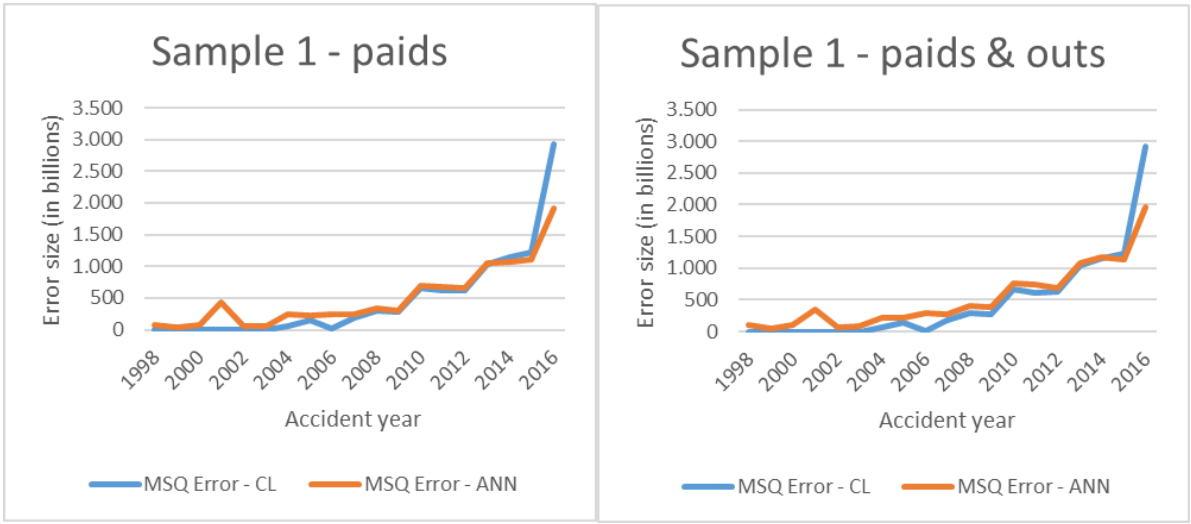

Figure 10 Sum of errors comparisons between CL predictions and ANN predictions by accident year. The first uses paid claims only, and the second both paid and outstanding claims, from Sample 1. All ANNs have one hidden layer with two neurons.

We managed to improve prediction with the data transformation. We transformed into ratios based on cumulative paid and outstanding claim amounts, as described under section Techniques. The next chart shows the comparison between Chain-Ladder predictions and the ANN predictions where paid and outstanding claims were used

and data were transformed into ratios. We used one hidden layer with 5 neurons for first 10 development years and 3 neurons for next 10 development years.

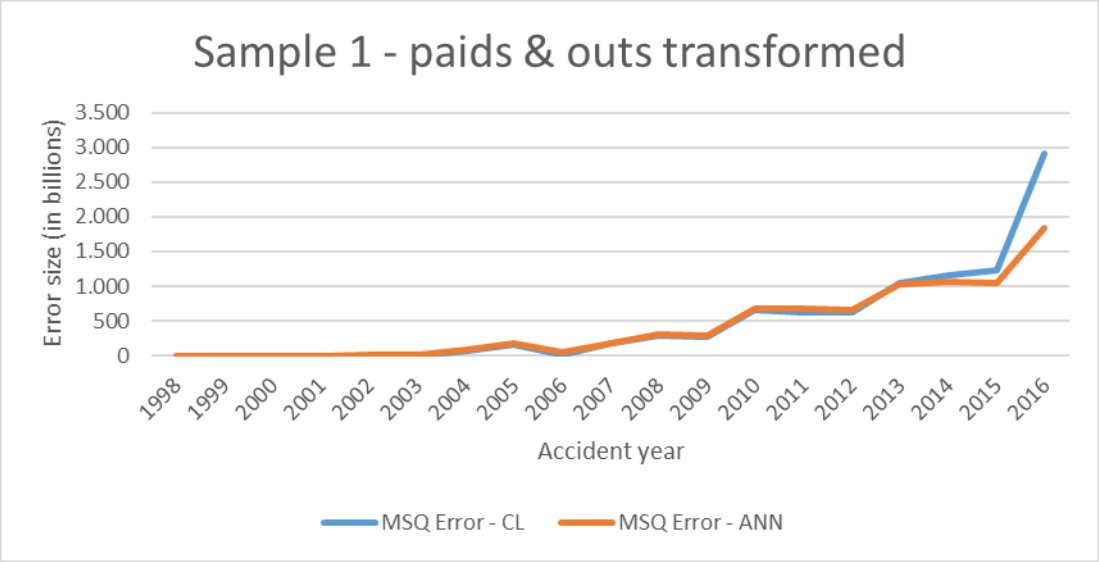

Figure 11 Sum of errors comparison between CL and ANN predictions by accident year. Paid claims and outstanding claims from Sample 1 are used as inputs, and transformed to ratios. All ANNs have one hidden layer with three or more neurons.

The charts show the improvement in prediction if outstanding claims are used and data are transformed. It seems that for the two most recent accident years, which account for the majority of claim reserve, the ANN outperforms the Chain-Ladder in terms of prediction at an individual claim level. However, ANN predictions couldn't outperform in terms of overall reserve prediction, which is shown in next table.

The table presents the quality of overall reserve prediction. In the first column is presented absolute deviation of exact amount of reserves forecast by each method, and in the second column relative deviation. The exact amount of reserves appears in the first row. All results refer to Sample 1. All deviations are written as absolute values (positive value whether lower or higher estimate than target).

Total claim reserve	**618.272.805**	
Chain-Ladder mismatch	684.314	0,1%
ANN paids mismatch	47.823.309	7,7%
ANN paids&outs mismatch	24.521.884	4,0%
ANN paids&outs transformed mismatch	7.730.996	1,3%

Table 4 Reserve estimate results for Sample 1

Sample 2 (long tail sample with 4000 claims)

The long tail sample turned out to be well predicted by the ANN without the use of data transformation. The next charts present a comparison of errors in a same way as before, with just paid claims and both paid and outstanding claims as respective inputs.

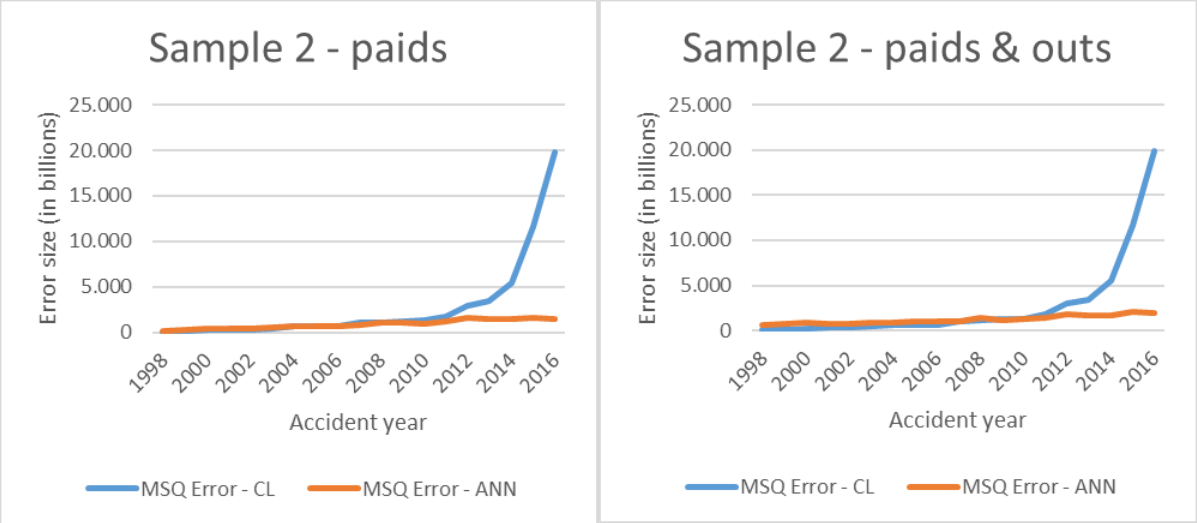

Figure 12 Sum of errors comparisons between CL predictions and ANN predictions by accident year. The first uses paid claims only, and second both paid and outstanding claims, from Sample 2. All ANNs have one hidden layer with two neurons.

We can see a good fit from the ANN without any data transformation. Results with the data transformation were similar as is shown on the next chart.

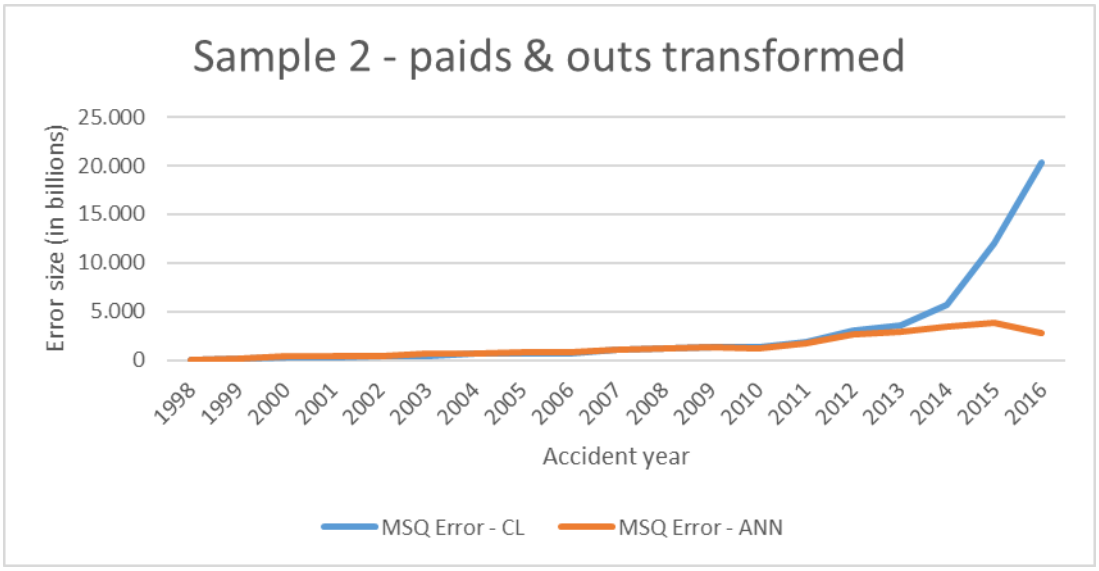

Figure 13 Sum of errors comparisons between CL and ANN predictions by accident year. Paid claims and outstanding claims from Sample 2 are used as inputs. All ANNs have one hidden layer with two neurons.

The next table presents a summary of overall reserve estimation for all methods used. It presents the quality of overall reserve prediction. In the first column is presented absolute deviation of the exact amount of reserves forecast by each method, and in the second column are shown relative deviations. All results refer to Sample 2.

Total claim reserve	**1.325.091.205**	
Chain-Ladder mismatch	9.099.905	0,7%
ANN paids mismatch	36.713.673	2,8%
ANN paids&outs mismatch	3.042.846	0,2%
ANN paids&outs transformed mismatch	9.818.790	0,7%

Table 5 Reserve estimate results for Sample 2

2. Samples with mixed patterns

Sample 3 (evenly distributed claims with two patterns)

A mixture of two underlying patterns turned out to create difficulties at early accident years for an ANN where cumulative amounts of paid claims and outstanding claims were given as inputs. The ANN does quite well on recent years, though less so with transformed data. Results are shown in the, next charts.

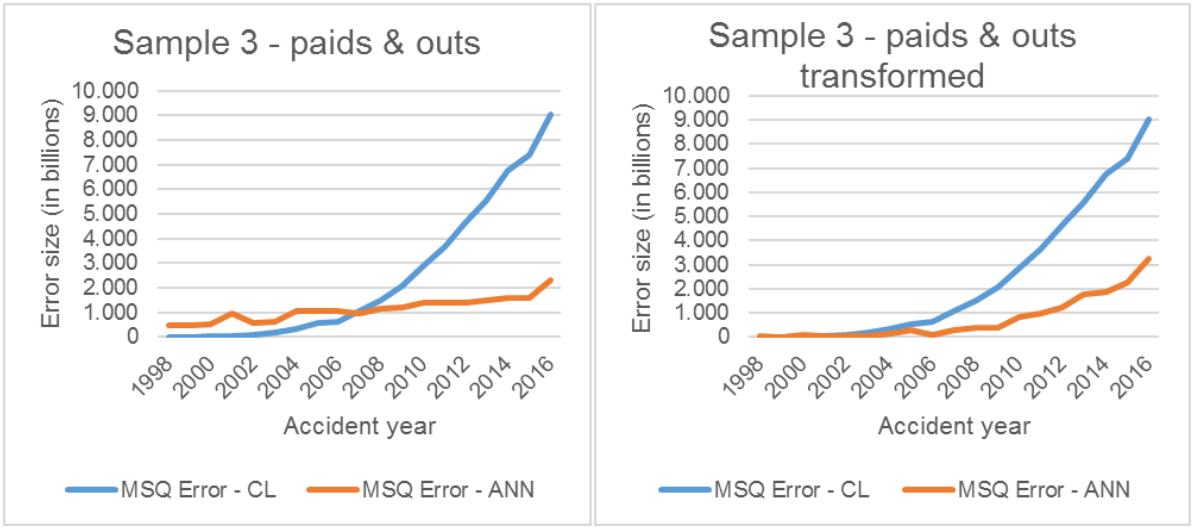

Figure 14 Sum of errors comparisons between CL predictions and ANN predictions by accident year. The first uses cumulative amounts of paid and outstanding claims, and the second transformed amounts, from Sample 3. All ANNs have one hidden layer with two neurons.

After that, we tried to improve our prediction by manipulation of the number of neurons and layers used by an ANN for prediction of individual development years. That could additionally reduce the error in overall reserve estimation. We tried a more complex ANN with two hidden layers and with the number of neurons at the first three development years set to 7 at both layers, and then numbers of neurons gradually diminishing with increasing development year, until the last 8 development years had 2 neurons in each layer. The resulting sum of errors is shown in next table.

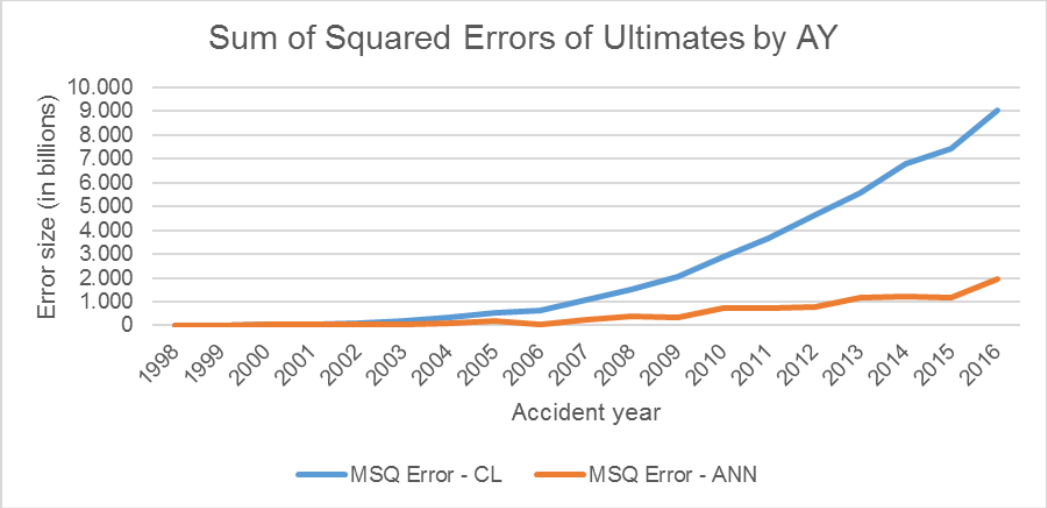

Figure 15 Sum of errors comparisons between CL predictions and ANN predictions by accident year. Paid claims and outstanding claims from Sample 3 are used as an input. All ANNs have more complex structure, as described above.

The next table shows overall reserve results by different methods for Sample 3.

Total claim reserve	**883.291.046**	
Chain-Ladder mismatch	2.672.526	0,3%
ANN paids&outs mismatch	223.688.807	25,3%
ANN paids&outs transformed mismatch	108.412.172	12,3%
ANN paids&outs transformed and complex mismatch	7.451.932	0,8%

Table 6 Reserve estimate results for Sample 3

The Chain-Ladder method estimated reserves very precisely, but on the other hand the method performed badly with the individual claim predictions. The next chart compares individual predictions from Chain-Ladder and ANN with the original claims simulations, using transformed data and the more complex structure of neurons. We can see that part of claims predicted by Chain-Ladder is undervalued, part overvalued. ANNs make better prediction.

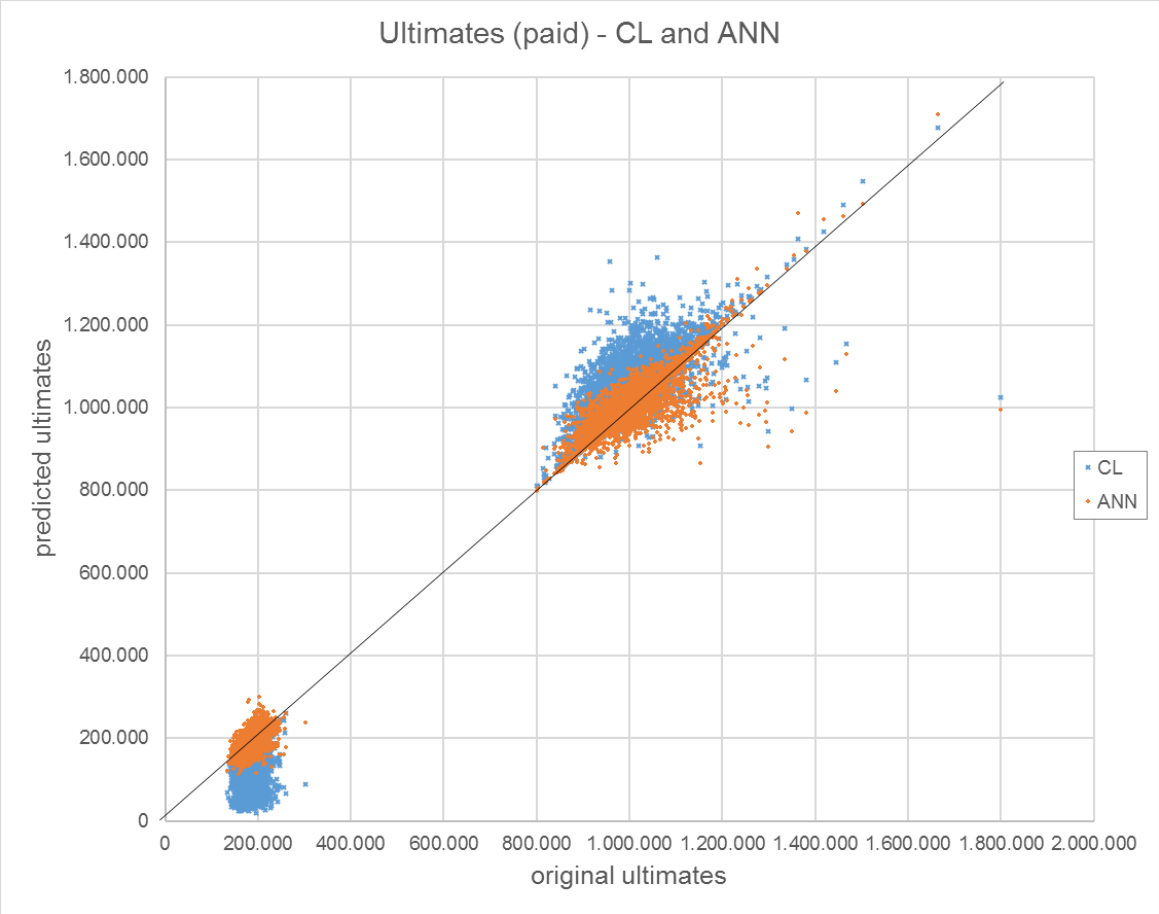

Figure 16 Ultimate claims, as predicted by Chain-Ladder and ANN, in comparison with original simulations.

The next picture presents the development patterns of two typical claims (claim number 4000 and claim number 10000).

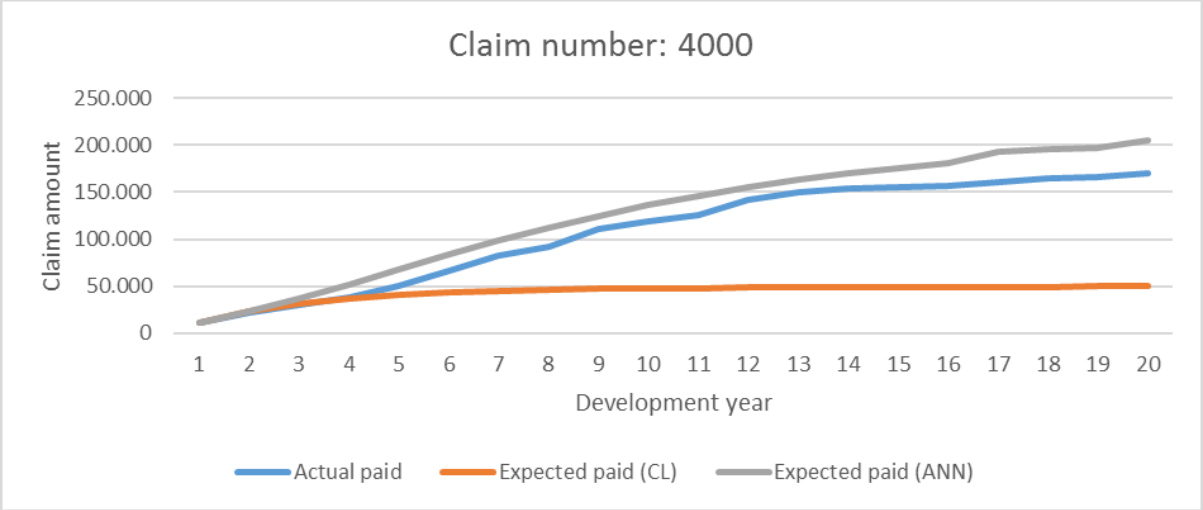

Figure 17 Example of long tail claim from Sample 2

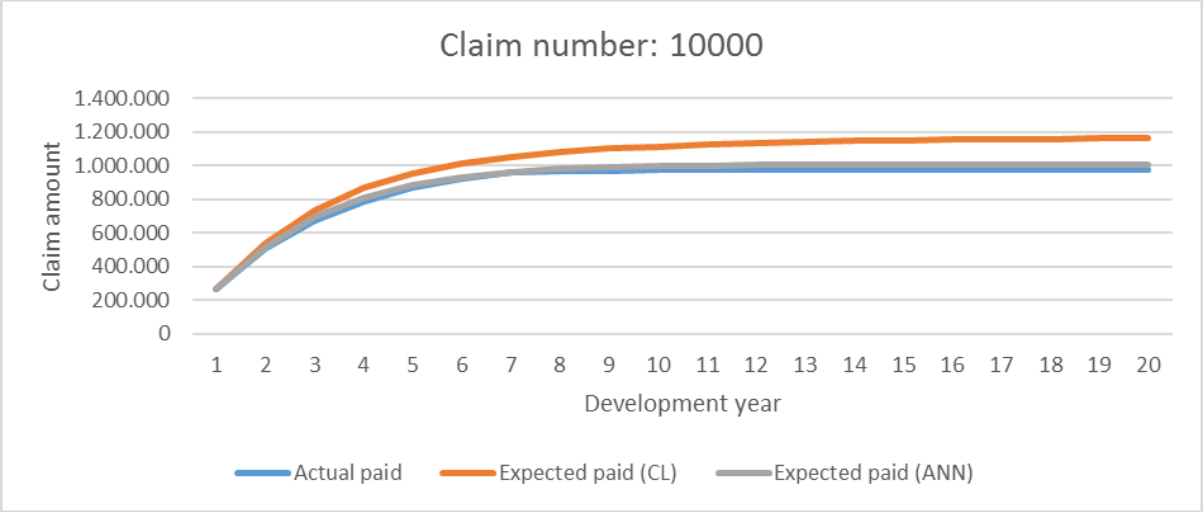

Figure 18 Example of short tail claim from Sample 1

An analytical approach to individual claims is able to differentiate between these two types of claim. Quite exact segmentation can be achieved on the basis of only information on paid and outstanding claims from the first development year. We can notice that short term claims have significantly higher paid claims in the first year than long tail claims. Obviously, more development years add to clarity of the differentiation. From the results above, it seems that ANNs also manage to detect this difference.

Sample 4 (smoothly changing ratio of claims with two patterns)

This sample is a particularly dangerous sample, which may cause serious issues for an insurer. We expect the Chain-Ladder to fail with this sample, but hope that an ANN will work better.

From all methods mentioned so far, we managed to obtain the best results with an ANN on transformed data, where all ANNs have one hidden layer with 2 neurons. The resulting errors by accident year are shown in the next chart.

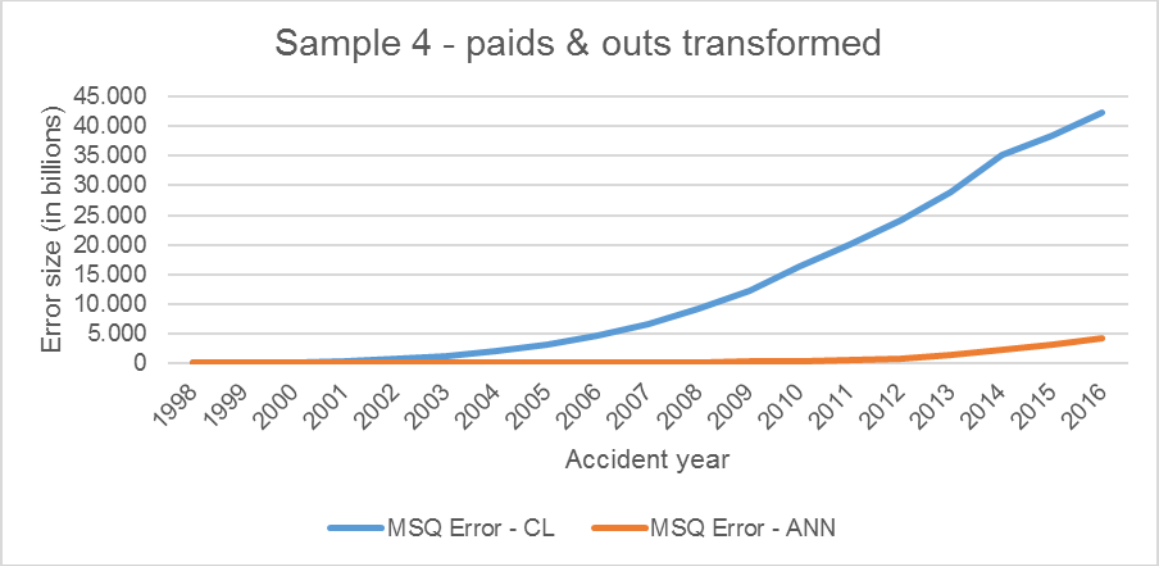

Figure 19 Sum of errors comparisons between CL predictions and ANN predictions by accident year. Paid claims and outstanding claims from Sample 4 are used as input. All ANNs have one hidden layer with two neurons.

As expected, Chain-Ladder significantly overestimated reserves. Reserves estimated with different methods, applied to Sample 4, are shown in the next table. All ANN methods give closer predictions.

Total claim reserve	**1.246.076.425**	
Chain-Ladder mismatch	988.191.744	79,3%
ANN paids&outs mismatch	685.940.226	55,0%
ANN paids&outs transformed mismatch	91.287.288	7,3%
ANN paids&outs transformed and complex mismatch	223.688.807	18,0%
ANN paids&outs complex mismatch	744.521.253	59,7%

Table 7 Reserve estimate results for Sample 4

We could analytically separate claims from Sample 4 into two parts by looking at paid amounts in the first year and then perform the Chain-Ladder estimation separately on both parts of claims. In case we select 100.000 as a separating limit then total Chain-Ladder error from both parts is only 1,3 million. So, with an analytical approach, we manage to achieve better results than ANN, but the ANN performs a differentiation itself.

Sample 5 (sudden change of ratio of claims with two patterns)

With a suddenly changing ratio of claims, we expected similar effects as with sample 4. A double hidden layer ANN structure with more neurons, as described under Sample 3, gave lowest sum of errors.

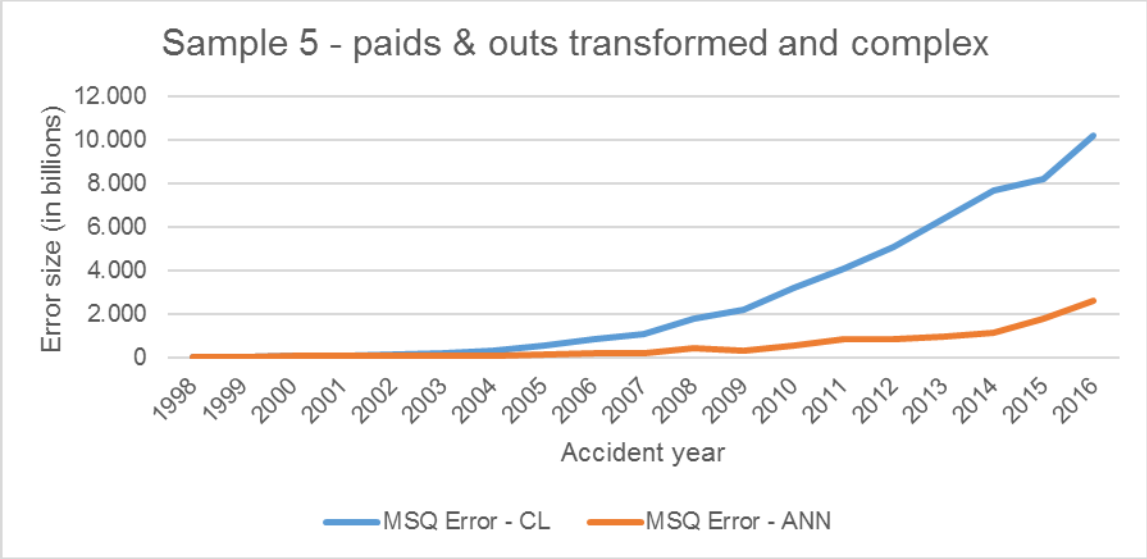

Figure 20 Sum of errors comparisons between CL predictions and ANN predictions by accident year. Paid claims and outstanding claims from Sample 5 are used as inputs. All ANNs have two hidden layers, with the number of neurons at first three development years set to 7 at both layers, and then numbers of neurons gradually diminishing with increasing development year, until the last 8 development years had 2 neurons in each layer.

Similarly, as in the case of Sample 4, all ANN methods gave better estimates of reserves than the Chain-Ladder. Reserve estimations for Sample 5 are shown in next table.

Total claim reserve	**1.055.151.336**	
Chain-Ladder mismatch	99.767.954	9,5%
ANN paids&outs mismatch	83.477.493	7,9%
ANN paids&outs transformed mismatch	91.287.288	8,7%
ANN paids&outs transformed and complex mismatch	20.057.682	1,9%

Table 8 Reserve estimate results for Sample 5

3. Main conclusions

Our main conclusions after analyzing ANN methods in comparison with the commonly used Chain-Ladder method are:

1. Individual development of claims with an ANN cascading method might have higher impact on long tail lines of business.
2. Typically, ANNs predict better if paid and outstanding claims are used as an input. But it is not always the case.
3. Typically, ANNs predict better if input data are modified to ratios, but again it is not always the case.
4. If a line of business is not homogeneous ANN might differentiate claims with statistically different underlying patterns. More generally, CL may under-perform when it is a mis-specified model (data do not behave in accordance with its assumptions).

4. Open issues

During the research, our working party didn't answer several questions and issues. First on the list are the following questions:

- What is an optimal ANN structure?
- What are additional data modifications that could be made?
- How well would perform other machine learning algorithms instead of ANNs?
- What would be a strong combined method, constructed from different ANN methods for different development years?
- Can we replace the cascading method with a method that uses only one ANN?

Connected with the questions above, our working party pointed out few issues to be considered in future:

- Replacement of cascaded development factors by a parametric curve to represent claim payments or incurred amounts over the span of development years. This would reduce the number of parameters (over-fitting) and eliminate the need for a cascaded model.
- Introduction of covariates. These can be powerful predictors, and are not easily accommodated by the CL, nor by many other aggregate models.
- The present paper assumes that claims, or at least subsets of them, conform to the CL assumptions. The synthetic data do so. This is a very strong assumption, often false in practice. One very powerful motivation for the use of an ANN is that it can be formulated in a model-free manner, i.e. no assumption is made about the model form, and the ANN is left to infer this for itself.

5. Next steps

We suggest a few issues that can be considered in the scope of one or several new working parties:

- Further development of suggested cascading ANN method, where the focus of the research would move from synthetic data to real data. Introduction of cross-validation and extension of granularity of input data.
- Implementation of ANNs within a different method, without using triangular cascading architecture of several ANNs. Some ideas are presented above under open issues.
- Introduction of other machine learning algorithms, like random forests or support vector machines.

VII. Summary

The ASTIN working party on Individual Claim Development with Machine Learning managed to test one particular machine learning technique for individual claim development and for reserve estimation.

In order to experiment with artificial neural networks as our favorite machine learning technique, we needed to acquire data. Within the working party we came to a conclusion that the best starting point for consisted of testing artificial neural networks not on real data, but on generated data. This way we could prepare as many different samples as we needed, and add all the features required. We decided to work with data which have information about paid and outstanding amounts by development year for each individual claim. We prepared quite a number of samples, but decided to present only the most illustrative ones. We present one short tail sample, one long tail sample, and three more samples each of which is some sort of a combination of the first and second sample, where the proportions of the first and second samples change over accident years.

The data samples were imported into an R script that performed predictions using a cascading ANN method. We tested several options for importing data. The first was to import only cumulative paid data, the second to add cumulative outstanding data, and the third to transform data into ratios as described under section Technique. We tested different R functions that construct an ANN, tested different activation functions and explored various ANN structures and learning procedures. We decided to present results achieved from some particular ANN structures, which turned out to perform well in many cases.

The resulting predictions using cascading ANNs were compared with the predictions produced by a simple Chain-Ladder technique. Our main goal was to predict individual claims, specifically the way in which they develop through time, and measure the prediction error. Last but not least, we checked how well we managed to predict overall reserves. We found a strong indication that cascading ANNs outperform Chain-Ladder techniques for individual claim development, but Chain-Ladder still perform well for overall reserves estimation in cases where development pattern remains the same accident year by accident year. Prediction results from cascading ANNs, applied to samples where claims structure changed through different accident years, outperformed the Chain-Ladder method both for individual claim development and for overall reserve estimation.

The ICDML working party made only one step towards the research of potential of machine learning techniques to be used in actuarial work. We managed to show some potential of one specific technique of machine learning. However, this research opened more questions than we managed to answer. Maybe the most important one is: are we able to show similar results on real data samples? Future research is needed to clarify this and other issues that we pointed out above.

VIII. Appendices

Appendix A

Sample 4, predicting second development year

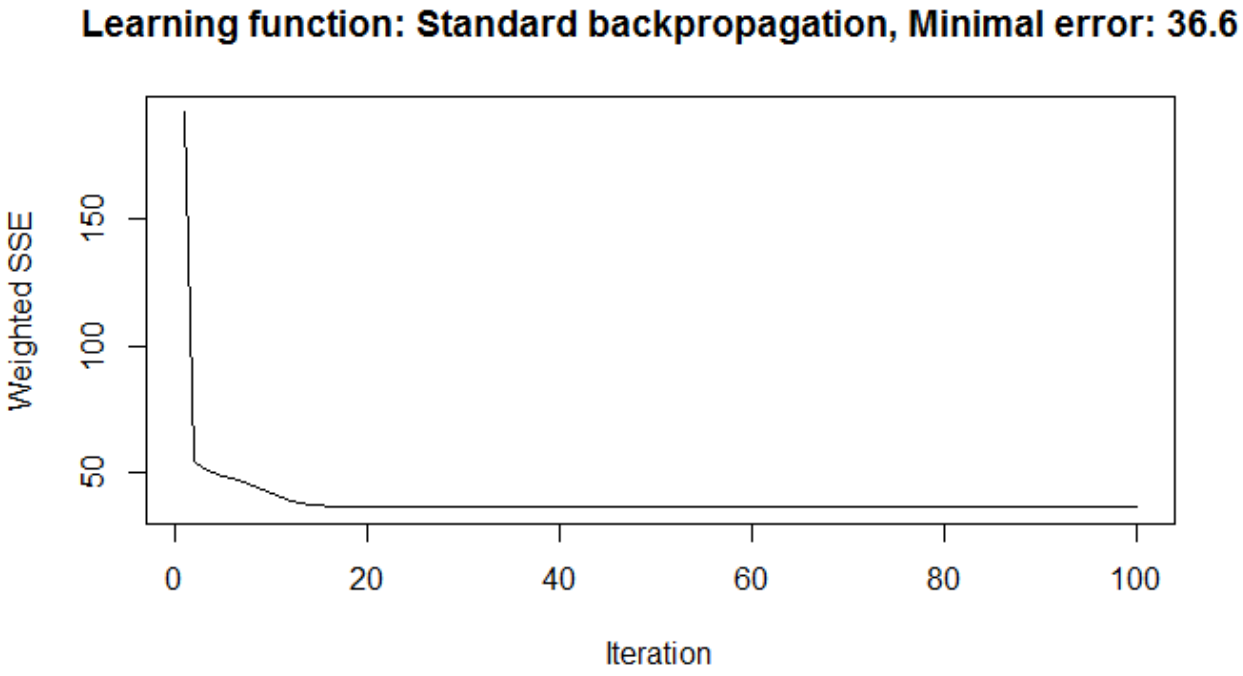

Figure 21 Prediction error of MLP prediction of development year 2 using standard backpropagation. Learning rate is set to 0,007.

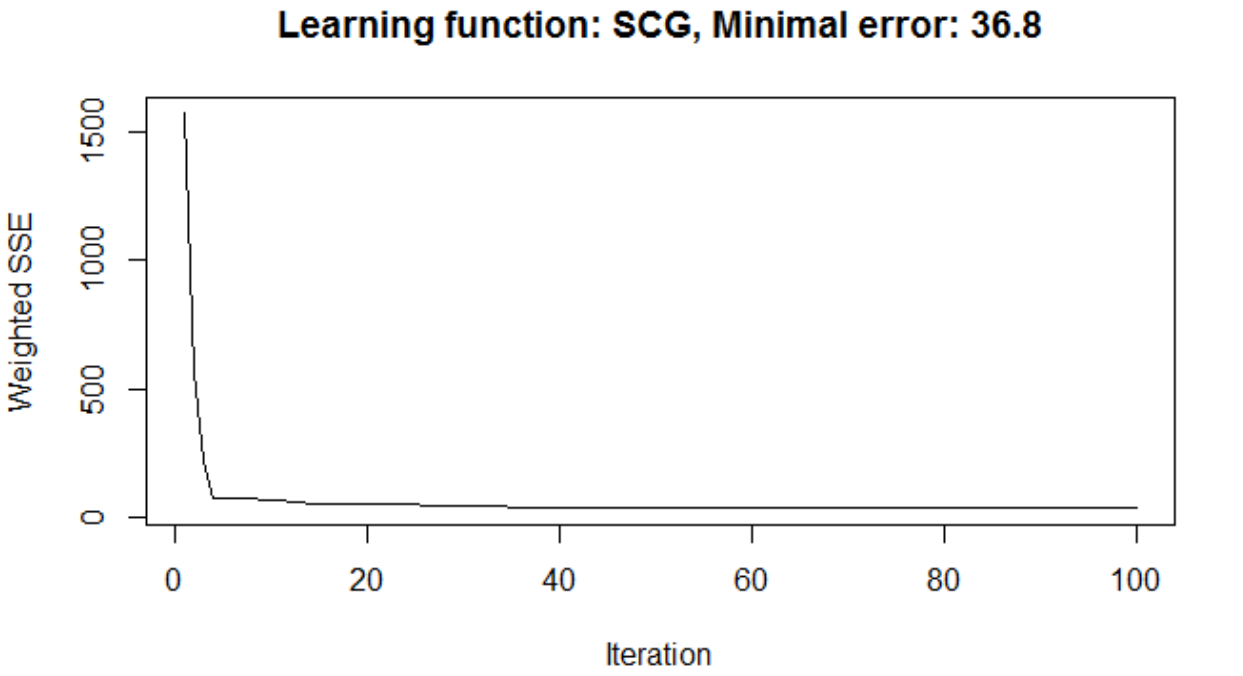

Figure 22 Prediction error of MLP prediction of development year 2 using SCG.

Sample 4, predicting 1th development year

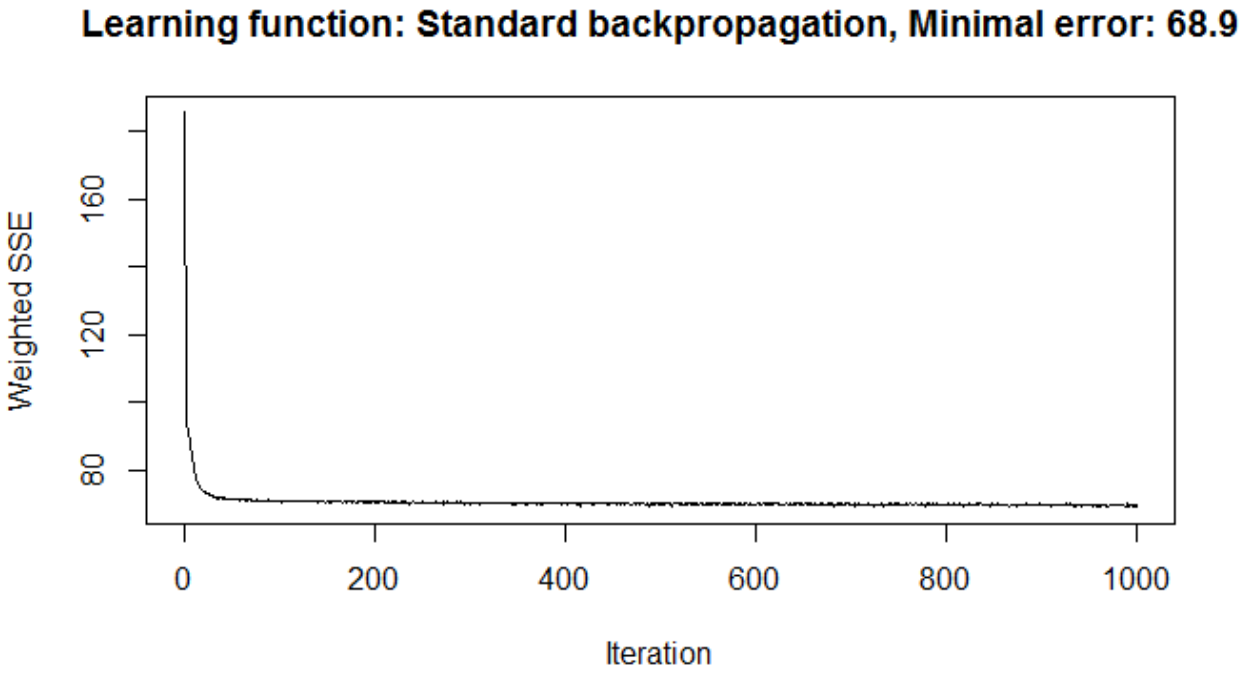

Figure 23 Prediction error of MLP prediction of development year 10 using standard backpropagation. Learning rate is set to 0,007.

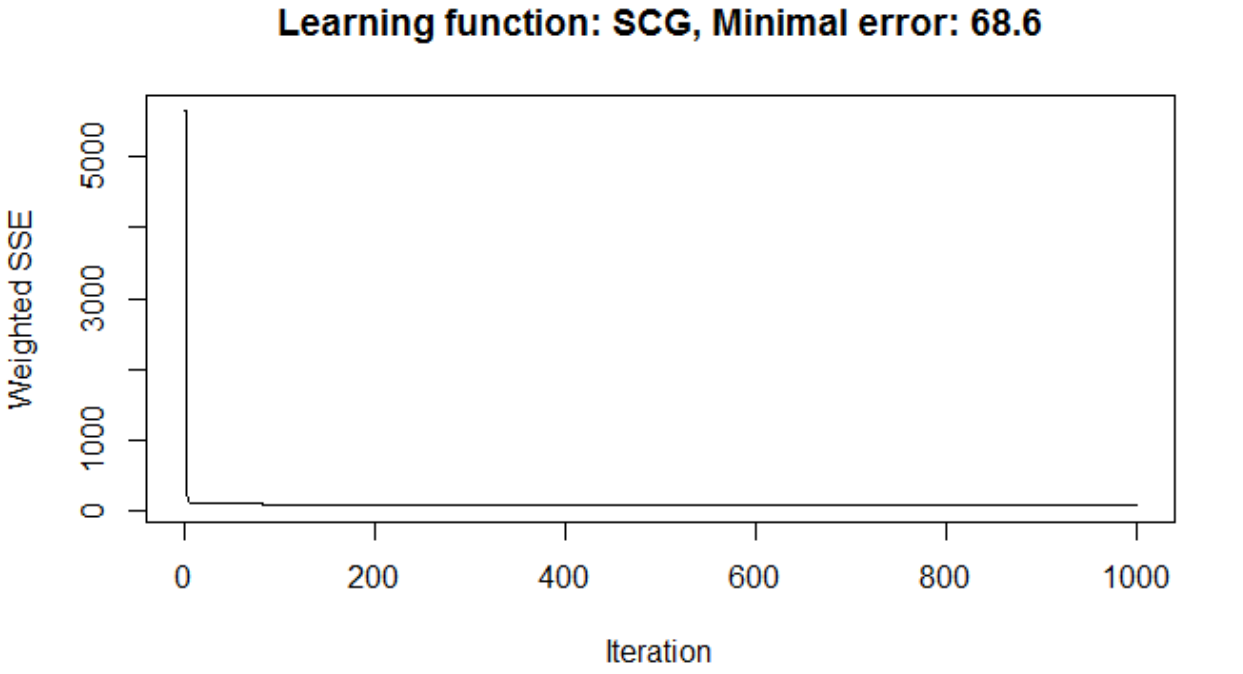

Figure 24 Prediction error of MLP prediction of development year 10 using SCG.

Sample 4, predicting 18th development year

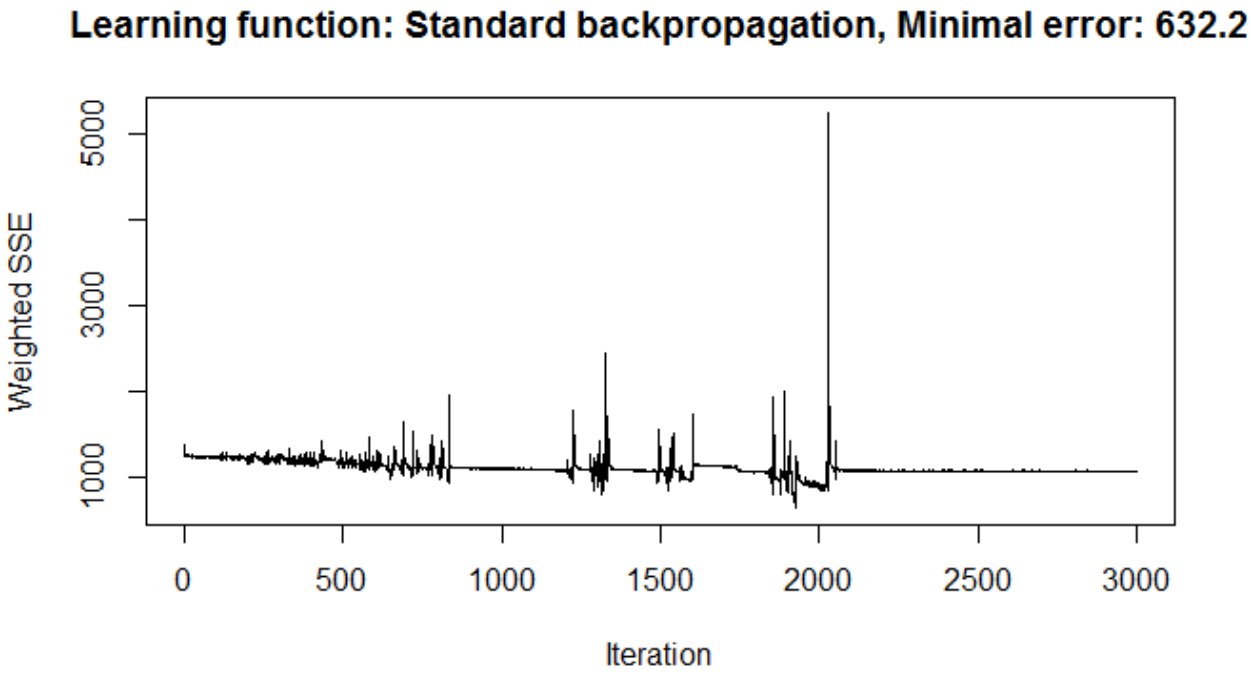

Figure 25 Prediction error of MLP prediction of development year 18 using standard backpropagation. Learning rate is set to 0,007.

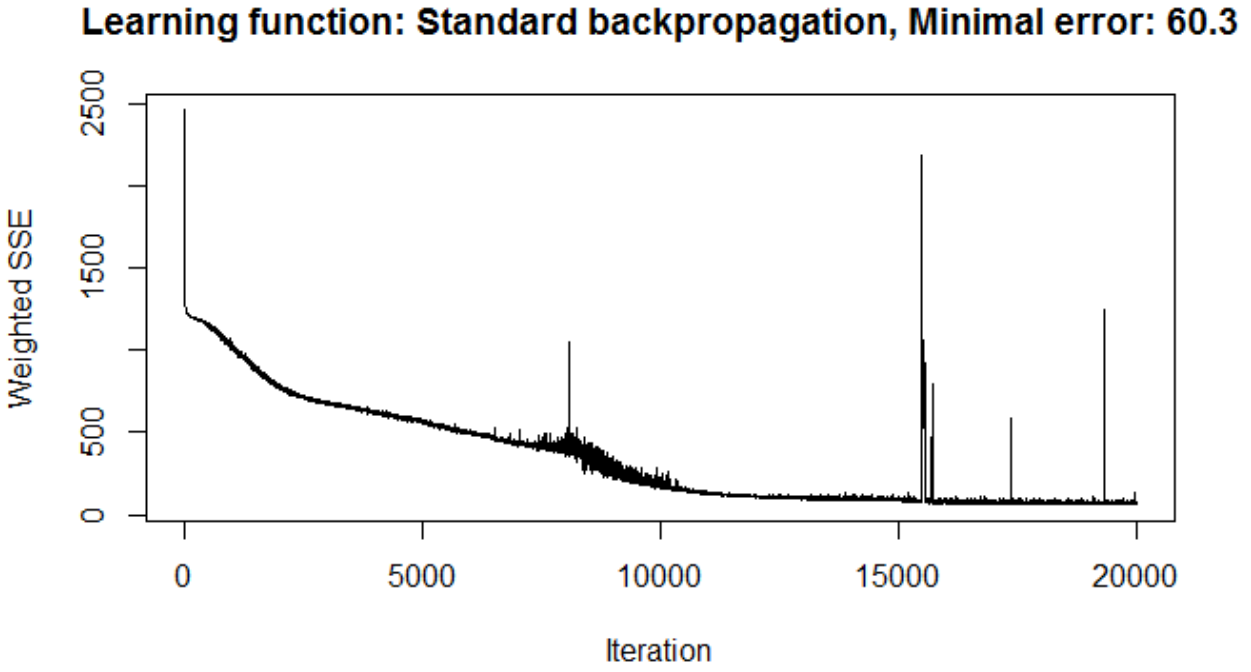

Figure 26 Prediction error of MLP prediction of development year 18 using standard backpropagation. Learning rate is set to 0,0003.

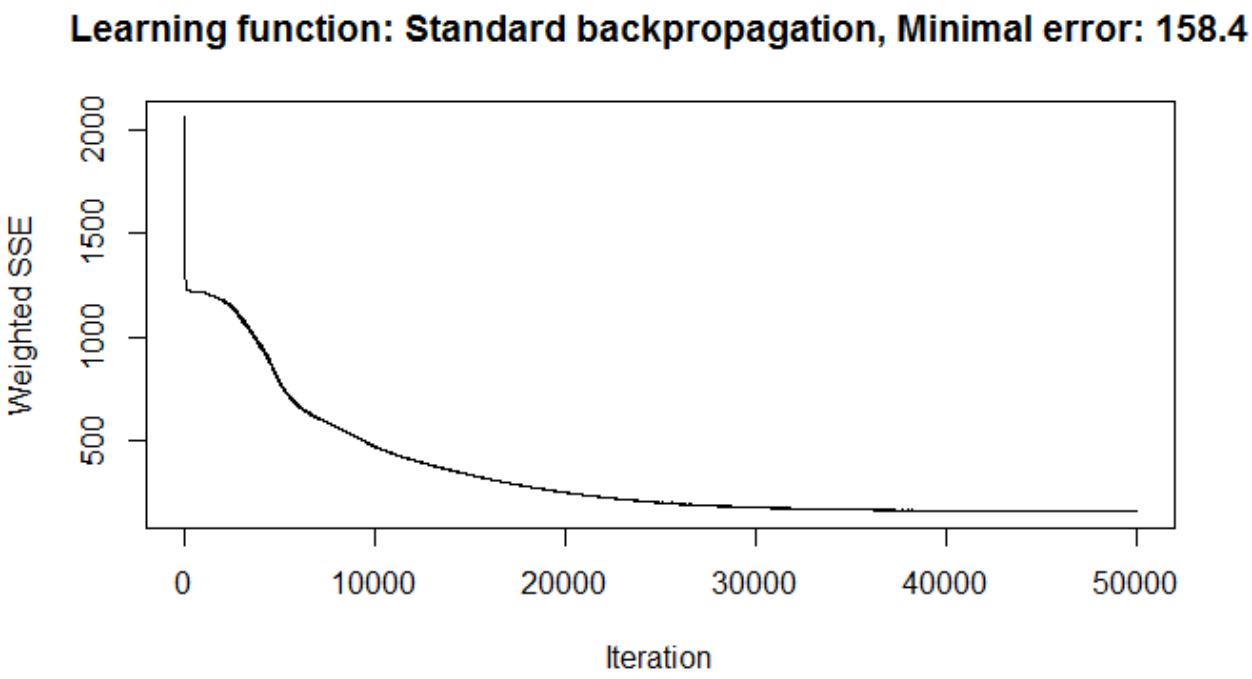

Figure 27 Prediction error of MLP prediction of development year 10 using standard backpropagation. Learning rate is set to 0,0001.

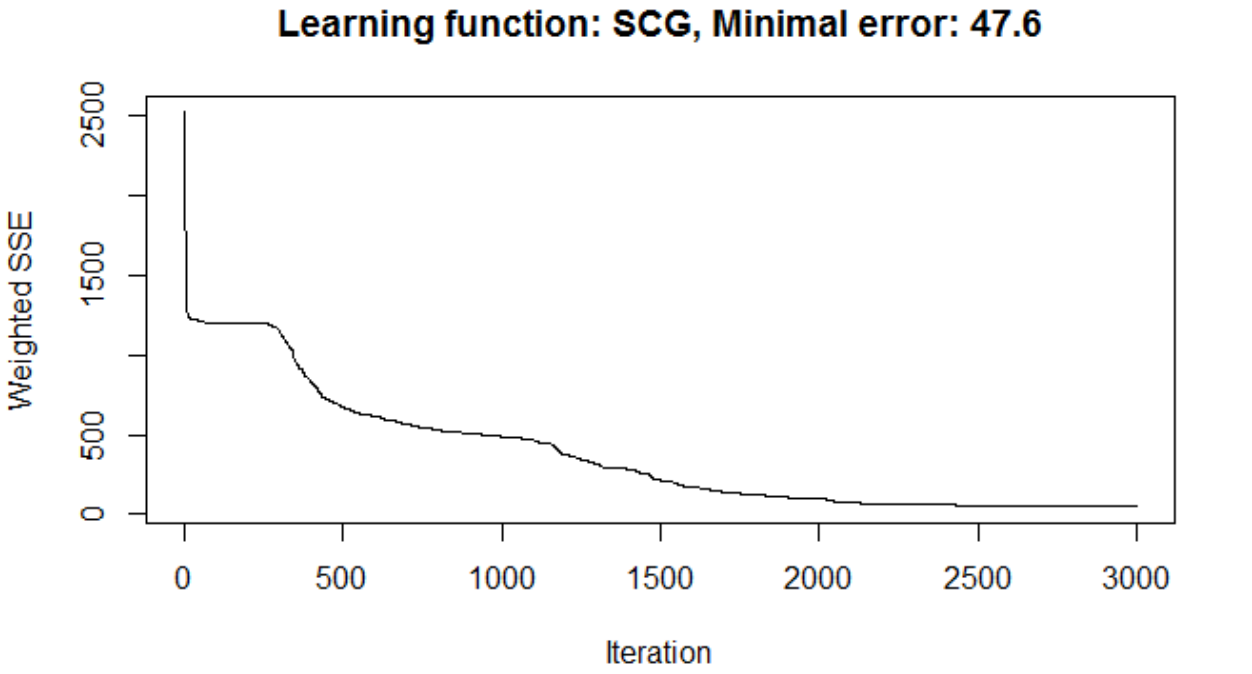

Figure 28 Prediction error of MLP prediction of development year 18 using SCG.

Appendix B

Principle of the Backpropagation algorithm (BA)

At the scale of a neuron, the gradient represents the contribution of this neuron to the global error (or the ***cost function***) of the model. If we denote by C this global error, then the gradient can be expressed as $\frac{\partial C}{\partial \omega_{ij}}$.

For a MLP model, for instance, the global error can be written as:

$$C(\omega) = \frac{1}{2}\sum_{\alpha}\sum_{k}(y_\alpha - \hat{y}_\alpha)^2 \quad (5)$$

And for example:

$$C^\alpha(\omega) = \frac{1}{2}\sum_{k}(y_\alpha - \hat{y}_\alpha)^2 \quad (6)$$

where y_α(respectively $\hat{y}_\alpha$) are the outputs (respectively the expected outputs) of the model, and k the claim number.

Relying only on the gradient to appreciate the error of the model is not sufficient. Indeed, this derivative does not specify the proportion of weights variation.

The backpropagation algorithm developed by Werbos [9], then by Rumelhart et al. [10], computes the cost function gradient value at each iteration. It is based on the fact that the gradient of the cost function is equal to the sum of the partial cost functions $C_\alpha(\omega)$:

$$\nabla C(\omega) = \sum_{\alpha}\nabla C_\alpha(\omega) \quad (7)$$

where ∇ designates the usual gradient operator[9].

The partial derivative is then calculated as :

$$\frac{\partial C^\alpha}{\partial \omega_{ij}} = \left(\frac{\partial C^\alpha}{\partial \vartheta_i}\right)_\alpha \left(\frac{\partial \vartheta_i}{\partial \omega_{ij}}\right)_\alpha = \delta_i^\alpha x_{ij}^\alpha \quad (8)$$

where, for each example :

- ϑ represents the potential of the neural network. In the case of the BA, ϑ is a weighted sum with a constant term : $\vartheta = \omega_0 + \sum_{i>0}\omega_i x_i$;

[9] The gradient operator ∇ of the function f is : $\nabla f(v_1, v_2, \dots, v_n) = \sum_{i=1}^{n}\frac{\partial f}{\partial v_i}e_i$ where e_i is a unit vector pointing in the direction of variable v_i and $\frac{\partial f}{\partial v_i}$ designates a derivative of f with respect to some variable v_i.

- $\left(\frac{\partial C^{\alpha}}{\partial \vartheta_i}\right)_{\alpha} = \delta_i^{\alpha}$ represents the gradient of the partial cost function relatively to the potential linked to the i^{th} neuron;
- $\left(\frac{\partial \vartheta_i}{\partial \omega_{ij}}\right)_{\alpha} = x_{ij}^{\alpha}$ represents the value of the partial derivative of the potential relative to a parameter. It is also the j^{th} input of the i^{th} neuron.

The development of δ_i^{α} differs between the inputs/outputs layer and any hidden layer:

- For the inputs/outputs layer:

$$\begin{aligned}\delta_i^{\alpha} &= \left(\frac{\partial C^{\alpha}}{\partial \vartheta_i}\right)_{\alpha} \\ &= \left(\frac{\partial \frac{1}{2}\sum_k (y_{\alpha} - \hat{y}_{\alpha})^2}{\partial \vartheta_i}\right)_{\alpha} \\ &= -2(y_{\alpha} - \hat{y}_{\alpha})\frac{\partial \hat{y}_{\alpha}}{\partial \vartheta_i} \\ &= {}^{(*)} -2(y_{\alpha} - \hat{y}_{\alpha})\end{aligned}$$

(*) The last equality is due to the fact that, in the case of regression, the output neuron is linear.

- For a hidden layer:

$$\begin{aligned}\delta_i^{\alpha} &= \left(\frac{\partial C^{\alpha}}{\partial \vartheta_i}\right)_{\alpha} \\ &= \left(\frac{\partial C^{\alpha}}{\partial \vartheta_k}\right)_{\alpha}\left(\frac{\partial \vartheta_k}{\partial \vartheta_i}\right)_{\alpha} \\ &= \delta_k^{\alpha}\left(\frac{\partial\left(\omega_0 + \sum_{j>0}\omega_{kj}x_j^{\alpha}\right)}{\partial \vartheta_i}\right)_{\alpha} \\ &= \delta_k^{\alpha}\left(\frac{\partial\left(\omega_0 + \sum_{j>0}\omega_{kj}f(\vartheta_i^{\alpha})\right)}{\partial \vartheta_i}\right)_{\alpha} \\ &= \delta_k^{\alpha} f'(\vartheta_i^{\alpha})\end{aligned}$$

Where f is the actiavation function of the hidden layers.

Finally:

$$\delta_i^{\alpha} = \begin{cases} -2(y_{\alpha} - \hat{y}_{\alpha}), \text{for the inputs/outputs layer} \\ \delta_k^{\alpha} f'(\vartheta_i^{\alpha}), \qquad f \text{or a hidden layer} \end{cases}$$

The BA adjusts the network weights according to sign changes of the gradient rather than the values that weights take. In other words, if we denote the weight linked to the i^{th} individual of the j^{th} class by $w_{i,j}$, then the BA implies that the weight variations can be written as:

$$\Delta w_{i,j} = \epsilon \frac{\partial C}{\partial \omega_{i,j}} x_{i,j} \quad (9)$$

where ϵ is the learning rate linked to the model. This parameter controls the rate of change of the gradient by limiting the weight variations during the learning step.

IX. References

[1] I. Kaastra et M. Boyd, «Designing a neural network for forecasting financial and economic time series,» Neurocomputing 10, pp. 215 - 236, 1996.

[2] A. Blum., Neural networks in C++: an object-oriented framework for building connectionist systems, 1992.

[3] K. Swingler, Applying Neural Networks: A Practical Guide, London: Academic Press, 1996.

[4] M. J. A. Berry et G. Linoff, Data Mining Techniques, NY: JohnWiley Sons, 1997.

[5] R. Rojas, Neural Networks. A Systematic Introduction, Berlin: Springer, 1996.

[6] W. Press, S. Teukolsky, W. Vetterling et B. Flannery, Numerical recipes in C : The Art of Scientific Computing, Cambridge University Press, 1992.

[7] M. F. Moller, «A Scalled Conjugate Gradient Algorithm for Fast Supervised Learning,» Technical Report PB-339 - Compter Science Dept, University of Aarhus, Denmark, 1990.

[8] G. E. Hinton, «Connectionist Learning Procedures,» Artificial Intelligence, 10, pp. 185-234, 1989.

[9] P. Werbos, «Beyond regression : New tools for prediction and analysis in the behavioral sciences,» 1974.

[10] R. J. Williams, D. E. Rumelhart et G. E. Hinton, «Learning internal representations by error backpropagation. Parallel Distributed Processing : Explorations in the Microstructure of Cognition.,» p. 318 – 362, 1986.

[11] F. Cuypers, «System and Method for Automated Establishment of Experience Ratings and/or Risk Reserves,» 2003. https://www.google.com/patents/US20060015373.