AI in Actuarial Science

By Ronald Richman

Presented at the Actuarial Society of South Africa's 2018 Convention
24–25 October 2018, Cape Town International Convention Centre

ABSTRACT

Rapid advances in Artificial Intelligence and Machine Learning are creating products and services with the potential not only to change the environment in which actuaries operate, but also to provide new opportunities within actuarial science. These advances are based on a modern approach to designing, fitting and applying neural networks, generally referred to as "Deep Learning". This paper investigates how actuarial science may adapt and evolve in the coming years to incorporate these new techniques and methodologies. After providing some background on machine learning and deep learning, and providing a heuristic for where actuaries might benefit from applying these techniques, the paper surveys emerging applications of artificial intelligence in actuarial science, with examples from mortality modelling, claims reserving, non-life pricing and telematics. For some of the examples, code has been provided on GitHub so that the interested reader can experiment with these techniques for themselves. The paper concludes with an outlook on the potential for actuaries to integrate deep learning into their activities.

KEYWORDS

Actuarial science; deep learning; machine learning; insurance; telematics

CONTACT DETAILS

Ronald Richman, Email: ron@ronaldrichman.co.za

1. INTRODUCTION

> The next insurance leaders will use bots, not brokers, and AI, not actuaries
>
> —Schreiber (2017)

Rapid advances in Artificial Intelligence (AI) and Machine Learning are creating products and services with the potential to change the environment in which actuaries operate. Notable among these advances are self-driving cars, which are due to launch in 2018 (drive.ai, 2018), and systems for the automatic diagnosis of disease from images, which gained approval for the automatic diagnosis of diabetic retinopathy in April 2018 (Federal Drug Administration, 2018). Both of these examples rely on deep learning, which is a collection of techniques for the automatic discovery of meaningful features within datasets, through the use of neural networks designed in a hierarchal fashion (LeCun, Bengio & Hinton, 2015). In this paper, the phrase "Deep Learning" will be taken to mean this modern approach to the application of neural networks that has emerged since several ground-breaking publications in 2006 (Hinton, Osindero & Teh, 2006; Hinton & Salakhutdinov, 2006). These products, and similar innovations, will change the insurance landscape by shifting insurance premiums and creating new classes of risks to insure (see, for example Albright, Schneider & Nyce (2017), who discuss the impact of autonomous vehicles on the motor insurance market), potentially creating disruption in one of the main areas in which actuaries currently operate.

As well as the potential disruption in insurance markets, the rapid advances in deep learning have led some to predict that actuaries will no longer be relevant in the insurance industry (Schreiber, 2017). The point of view advanced by this paper is that, contrary to this prediction, the discipline of actuarial science will adapt and evolve to incorporate the new techniques of Deep Learning, and seeks to substantiate this by surveying emerging applications of AI in actuarial science, with examples from mortality modelling, claims reserving, life insurance valuation, telematics analysis and non-life pricing. We thus prefer the view of Wüthrich and Buser (2018) who aim to assist actuaries in addressing the new paradigm of the "high-tech business world" (Wüthrich & Buser, 2018:11). Therefore, the paper has three main goals: firstly, to provide background regarding Deep Learning and how it relates to the more general areas of machine learning and artificial intelligence, secondly to survey recent applications of deep learning within actuarial science and lastly, to provide practical implementations of deep learning methodology using open source software. Thus, code to implement some of the examples is provided in a GitHub repository at the following link: *https://github.com/RonRichman/AI_in_Actuarial_Science/*

The rest of the paper will be organised as follows. Section 2 introduces the main concepts of machine learning, which apply equally to deep learning, discusses how the class of problems addressed by actuaries can often be expressed as regression problems and provides an heuristic as to which of these can be solved using deep learning techniques. Section 3 presents an introduction to deep learning and discusses some of the key recent advances in this field. Section 4 is a survey of recent applications of

deep learning to actuarial problems in mortality forecasting, life insurance valuation, analysis of telematics data and pricing and claims reserving in Short Term insurance (also known as Property and Casualty insurance in the United States and General Insurance in the United Kingdom). The paper concludes by discussing how the actuarial profession can adopt the ideas presented in the paper as standard methods.

2. INTRODUCTION TO MACHINE LEARNING AND DEEP LEARNING

Since some concepts in the machine learning literature may be unfamiliar to actuaries, we begin by providing a brief introduction to key ideas in machine learning, before discussing the applicability of machine learning to problems in actuarial science.

Machine learning is an approach taken within the field of AI whereby AI systems are allowed to build knowledge by extracting patterns from data (Goodfellow, Bengio & Courville, 2016) and has been defined as the field concerned with "the study of algorithms that allow computer programs to automatically improve through experience" (Mitchell, 1997). Machine learning can be divided broadly into three main areas – supervised learning, unsupervised learning and reinforcement learning. Supervised learning is the application of machine learning to datasets that contain both features (predictors in the statistical literature) and outputs of interest (dependent variables in the statistical literature), with the goal of predicting the outputs from the features as accurately as possible (Friedman, Hastie & Tibshirani, 2009). Defining X as a vector (or tensor in the case of image data) of features and y as the output of interest, then supervised learning has as its goal to learn the function $f(X) = y$ in order to make predictions at new vectors that have not yet been seen. Of the three areas of machine learning discussed above, by far the most familiar to actuaries is supervised learning, some examples of which in common actuarial practice are the fitting of generalised linear models (GLMs)[1] to claims datasets to predict the frequency and severity of claims, or to policyholder datasets to predict lapse rates. Models for supervised learning range from simple linear regression models to complex ensembles (meta-models) comprised of decision trees and other functions, fit through techniques such as boosting (Freund & Schapire, 1997; Friedman, 2001).

Unsupervised learning is the application of machine learning to datasets containing only features to find structure within these datasets (Sutton & Barto, 2018). In other words, vectors of features X are available, but there are no corresponding outputs y. The task of unsupervised learning is to find meaningful patterns using only X, which can then be used to further understand the data, or, in some cases, model it. An example of unsupervised learning is Principle Components Analysis (PCA), often applied by actuaries to derive interest rate risk scenarios in the context of Solvency II, see for example Boonen (2017). Other methods for unsupervised learning are k-means, self-organising maps (Kohonen, 1990) and t-distributed stochastic neighbour embedding (t-SNE) (Maaten & Hinton, 2008).

1 A list of all abbreviations/acronyms can be found in Appendix B.

Reinforcement learning is concerned with learning the action to take in situations in order for an agent to maximise a reward signal (Sutton & Barto, 2018). No examples of reinforcement learning applied to actuarial problems are known to the author (although a suggestion for its application in the insurance context was made by Parodi (2016)), and, therefore, reinforcement learning is not discussed further in this paper.

Supervised Learning

Supervised learning is concerned with the task of making accurate predictions from data, and many of the differences between statistical modelling (which is concerned with inference and is probably more familiar to actuaries), and supervised learning, are due to the distinction between the tasks of predicting and explaining. An excellent discussion of these two related but different disciplines is in Shmueli (2010). Some of the key ways in which machine learning is different from statistical modelling are that the machine learning approach favours:

— Building algorithms to predict responses without necessarily specifying a stochastic data generating model or process (Breiman, 2001), leading to models with good predictive performance that are often more difficult to interpret than statistical models.[2]

— Accepting some bias in models if this is expected to reduce the overall prediction error. For example, Shmueli (2010) shows that, in some cases, the "true" data generating model might not be the best choice if prediction is the goal. This is due to the bias-variance[3] trade-off, discussed in detail in Chapter 7 in Friedman, Hastie and Tibshirani (2009). Bias is often introduced (and variance reduced) by adding penalties to regression models, such as in LASSO (Tibshirani, 1996) or elastic-net regression (Zou & Hastie, 2005), or, in the case of mortality graduation, the Whittaker-Henderson smoothing[4] method.

— Quantifying predictive error (i.e. out-of-sample error) by splitting data into training, validation and testing sets, or using by cross-validation, as opposed to fitting models on all available data. The complexity of the fitted model is controlled based on the impact of model complexity on the predictive error, in order to find a model that generalises well to new data, instead of (over-)fitting to the noise in the training dataset.

2 The chain-ladder method was seen as an algorithm until the work of Mack (1993) and Renshaw and Verrall (1998), who provided statistical models underlying the chain-ladder.

3 Variance refers to the variability of the estimated model arising from the fitting of a model to a limited dataset. Actuaries have long known about the dangers of fitting models to and drawing conclusions from limited datasets, and often refer to these types of data and models as lacking "credibility". Describing Arthur Bailey, an actuary who came to embrace credibility methods in the early part of the 20th century, McGrayne (2011) writes that "He (Bailey) wanted to give more weight to a large volume of data than to the frequentists' small sample; doing so felt surprisingly 'logical and reasonable.' He concluded that only a 'suicidal' actuary would use Fisher's method of maximum likelihood, which assigned a zero probability to non-events."

4 As another example in an actuarial context, Mack (1993), after providing bias free estimators for the loss development factors of the chain-ladder model, is concerned with the variability of the chain-ladder estimators and provides his famous formula for its estimation.

Taking the latter points into account, Mullainathan and Spiess (2017) summarise the approach to supervised machine learning in a single expression:

$$\min L\left(f\left(X\right),y\right) \text{ for all } f \in F \text{ subject to } R\left(f\right) < c$$

which states that the goal is to minimise the in-sample error of the model by searching over functions f in a set of functions F, subject to a complexity restriction on the function f, which is calculated using the function $R(\ )$ which is limited to a level of complexity, c. The acceptable level of complexity of the model is a so-called hyper-parameter that is not estimated from the training data, but estimated from the validation dataset or inferred via a method such as cross-validation.

Although the concepts of supervised learning may appear foreign to actuaries trained primarily in statistical modelling and inference, the machine learning approach is currently appearing in recent actuarial research (Noll, Salzmann & Wüthrich, 2018; Parodi, 2014; Wüthrich & Buser, 2018) and has been endorsed at a prominent actuarial convention (Parodi, 2016).

Within supervised learning, problems are commonly grouped into the categories of classification and regression. Classification problems are those in which the output of interest is qualitative, such as the category to which an image belongs (Friedman, Hastie & Tibshirani, 2009). Regression problems are those in which the output is numerical, such as the frequency of claims by a policyholder. Sometimes, the distinction between classification and regression problems is blurred, for example, De Brébisson et al. (2015) use a neural network to predict taxi destinations, as measured by longitude and latitude, which is a regression problem. However, instead of directly predicting longitude and latitude, they approximate this using a set of categories representing areas on a map and predict these using a classification model.

Figure 1 represents schematically the different types of machine learning discussed in this section. Although deep learning has not yet been discussed in detail, to complete this introductory discussion, deep learning is a collection of techniques comprising a modern approach to designing and fitting neural networks that learn a hierarchal, optimised representation of the features, and can be used for supervised (classification and regression), unsupervised and reinforcement learning (i.e. all the categories of machine learning that have been discussed to this point).

Actuarial Modelling and Machine Learning

Actuarial models can often be expressed as regression problems, even if the original problems, at first glance, do not appear to be related to regression. Several obvious and less obvious examples of this are provided in the following points (and in more detail in Table 1):

1. Short-term insurance policies are usually priced with GLMs that model the frequency (Poisson rate regression) and severity (Gamma regression).

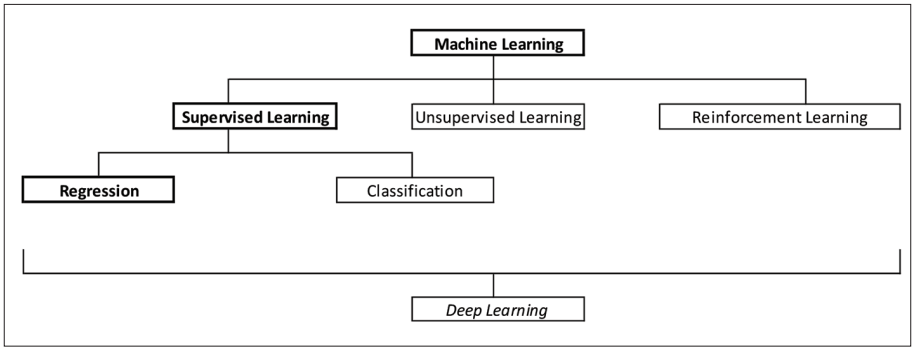

FIGURE 1 Schematic classification of machine learning methods. The branch leading to Regression is in bold since many problems in actuarial science can be expressed as regression problems, as discussed in the next section. Because deep learning models can be used for each of the main branches of machine learning, it spans across the schema.

2. The chain-ladder model can be described as a cross-classified log-linear regression according to the GLM formulation of Renshaw and Verrall (1998).
3. The software implementation of the distribution free chain-ladder model of Mack (1993) model in R (Gesmann et al., 2017) uses several linear regression models to estimate the coefficients, but, notably, the entire set of chain-ladder factors can be estimated using a single weighted linear regression.
4. More advanced Incurred But Not Reported (IBNR) models have been fit using generalised linear mixed models (GLMMs) and Bayesian hierarchal models.
5. The comparison of actual mortality to expected experience (AvE) is often performed manually by life actuaries, but this can be expressed as a Poisson regression model where the output is the number of deaths and the input is the expected mortality rate, with an offset taken as the central measure of exposure to risk.
6. The process of determining a life table based on the mortality experience of a company or portfolio is similarly a regression problem, where the outputs are the number of deaths and the inputs are the genders, ages and other characteristics of the portfolio (the estimated coefficients of the model can then be used to derive the mortality rates). An example of a methodology is given in Tomas and Planchet (2014).
7. Currie (2016) showed that many common mortality forecasting models (the Lee-Carter (Lee & Carter, 1992) and CBD (Cairns, Blake & Dowd, 2006) models among them) can be formulated as regression problems and estimated using GLM and Generalised Non-linear Models (GNMs).
8. Richman (2017) showed that the demographic techniques known as the Near Extinct Generations methods (Thatcher, Kannisto & Andreev, 2002), which

are used to derive mortality rates at the older ages from death counts, can be expressed as a regression model within the GLM framework.

9. Life insurance valuation models perform cashflow projections for life policies (or groups of policies, called model-points), to allow the present value of future cashflows to be calculated for the purpose of calculating reserves or embedded value. Since running these models can take a long time, so-called "lite" valuation models can be calibrated using regression methods.
10. Another regression problem in life valuations concerns attempts to avoid nested stochastic simulations when calculating risk based capital, such as in Solvency II or the Swiss Solvency Test. One option is to calibrate, using regression, a reference set of assets, called the replicating portfolio, to the liability cashflows, and perform the required stresses on the replicating portfolio.

TABLE 1 Feature vectors X and Outputs y for several regression problems in actuarial science

Number	Description	Feature vector	Output
1	Short-term pricing	Policyholder details, such as age, gender, marital status, credit score, postal code of residence, and details of the insured item, for example, on a motor policy, brand, age, power of the engine, overnight parking	Frequency/severity of claim, or pure premium (Tweedie GLM)
2	Over-dispersed Poisson IBNR model	Accident year, reporting year	Incremental claim amount
3	Mack chain-ladder model	Cumulative claims amount at time $t-1$	Cumulative claims amount at time t
4	Hierarchal IBNR models	Refer to 2&3	Cumulative claims amount at time t
5	AvE analysis of mortality	Expected number of deaths	Coefficient of the regression model
6	Portfolio specific mortality model	Age, gender, portfolio characteristics	Mortality rate
7	Mortality forecasting	Year, age	Mortality rate
8	Old-age mortality estimation	Year of birth, age at deaths	Number of deaths
9	Life valuation approximation	Age, gender, portfolio characteristics	Reserve value
10	Life valuation approximation	Value of assets (bonds, equities, options, swaptions) in scenario i	Reserve value

Once an actuarial model is expressed as a regression problem, then solutions to the problem are available in the context of machine learning and deep learning. Thus, taking the above points into consideration, the following (non-exhaustive) heuristic is offered for determining if deep learning solutions (and more broadly, machine learning solutions) are applicable to problems in actuarial science:

> *If an actuarial problem can be expressed as a regression, then machine and deep learning techniques can be applied.*

This heuristic is non-exhaustive, since, as will be shown later, some problems in actuarial science have recently benefitted from the application of unsupervised learning, and, therefore, actuaries faced with a novel problem should also consider unsupervised learning techniques.

RESOURCES

As a brief review section, many aspects of machine learning, especially the numerous techniques used in this field, have not been discussed. For readers looking for more information, a useful resource for actuaries is the lecture notes by Wüthrich and Buser (2018), which provide a mathematical perspective on machine learning with many applications in actuarial science, and the papers by Parodi (2012a; 2012b) which examine machine learning in the context of non-life insurance. Parodi (2014: Chapter 12) discusses some issues of the machine learning process applied in the insurance pricing context. Wüthrich (2018a) applies machine learning to individual claims reserving in and Deprez, Shevchenko and Wüthrich (2017) look at mortality modelling.

Considering more general sources, the book by James et al. (2013) provides an introductory look at many aspects of machine learning, including code in the R language (R Core Team, 2018), and the book by Friedman, Hastie and Tibshirani (2009) provides more theoretical background. Both of these are written from a statistician's perspective, which might be more easily understood by actuaries than similar books written by computer scientists. Kuhn and Johnson (2013) describe the predictive modelling process in the R language and provide case studies on applying many machine learning algorithms.[5] Finally, the book by Efron and Hastie (2016) provides an interesting and up-to-date discussion of the interplay between statistical and machine learning methods.

In terms of software, the caret package (Kuhn, 2008) in R provides a unified interface to machine learning algorithms in other packages, and, in Python, scikit-learn (Pedregosa et al., 2011) is an integrated package containing many machine learning algorithms and utilities.

5 Note, however, that the works just cited provide a somewhat out-of-date view of neural networks, and the resources at the end of the following section provide a helpful counterpoint to critically examine these parts of the books.

3. AN INTRODUCTION TO DEEP LEARNING

Deep learning is the modern approach to designing and fitting neural networks that relies on innovations in neural network methodology, very large datasets and the dramatically increased computing power available via Graphics Processing Units (GPUs). Although currently experiencing a resurgence in popularity, neural networks are not a new concept, with some of the earliest work on neural networks by Rosenblatt (1958). Goodfellow, Bengio and Courville (2016) categorise the development of neural networks into three phases:

— the early models, inspired by theories of biological learning in the period 1940–1960,
— the middle period of 1980–1995, during which the key idea of backpropagation to train neural networks was discovered by Rumelhart, Hinton and Williams (1986), and
— the current resurgent interest in deep neural networks that began in 2006, as documented in LeCun, Bengio and Hinton (2015).

Interest in neural networks declined after the middle period of development for two reasons: firstly, overly ambitious claims[6] about neural networks had been made leading to disappointments from investors in this technology and, secondly, other machine learning techniques began to achieve good results on various tasks (Goodfellow, Bengio & Courville, 2016). The recent wave of research into neural networks began with a paper by Hinton, Osindero and Teh (2006) who provided an unsupervised method of training deep networks that outperformed shallow machine learning approaches (in particular, a radial basis function support vector machine, or RBF-SVM) on an digit classification benchmark (Goodfellow, Bengio & Courville, 2016). Other successes of the deep learning approach soon followed in speech recognition and pedestrian detection (LeCun, Bengio & Hinton, 2015). A breakthrough in computer vision was due to the AlexNet architecture of Krizhevsky, Sutskever and Hinton (2012) who used a combination of improved computing technology (GPUs), recently developed techniques such as rectified linear units (Nair & Hinton, 2010) and dropout (Hinton et al., 2012), which are now used as standard techniques in deep learning and data augmentation to achieve much improved performance on the ImageNet benchmark using a convolutional neural network (or CNN, the architecture of which is described later). Other successes of deep networks are in the field of natural language processing, an example of which is Google's neural translation machine (Wu et al., 2016) and in speech recognition (Hannun et al., 2014). A more recent example is the winning method in the 2018 M4 time series forecasting competition, which used a combination of a deep neural network (a Long Short Term Memory network or LSTM (Hochreiter & Schmidhuber, 1997)) with an exponentially weighted forecasting model (Makridakis, Spiliotis & Assimakopoulos, 2018a). Importantly, although the initial breakthroughs

6 The cynical reader may, at this point, draw a parallel to the present situation.

that sparked the recent interest in deep learning were on training neural networks using unsupervised learning, the focus of research and applications more recently is on supervised learning models trained by back-propagation (Goodfellow, Bengio & Courville, 2016).

This paper asserts that "this time is different" and that the current deep learning research deserves the attention of the actuarial community, not only in order to understand the technologies underlying recent advances in AI but also to expand the capabilities of actuaries and scope of actuarial science. Although actuaries may have heard references to neural networks over the years, and may perhaps have even experimented with these models, the assertion of this paper is that the current period of development of neural networks has produced efficient, practical methods of fitting neural networks, which have emerged as highly flexible models capable of incorporating diverse types of data, structured and unstructured, into the actuarial modelling process. Furthermore, deep learning software packages, such as Keras (Chollet, 2015), make the use of these models practical. This assertion will be expanded upon and discussed in this, and the following, section.

Connection to Artificial Intelligence

Deep learning is a part of the machine learning approach to AI, where systems are trained to recognise patterns within data to acquire knowledge (Goodfellow, Bengio & Courville, 2016). In this way, machine learning represents a different paradigm from earlier attempts to build AI systems (which relied on hard-coding knowledge into knowledge bases) in that the system learns the required knowledge directly from data. Before the advent of deep learning, AI systems had been shown easily to solve problems in formal domains defined by mathematical rules that are difficult for humans, such as the Deep Blue system playing chess. However, highly complex tasks that humans solve intuitively (i.e. without formal rules-based reasoning), such as image recognition, scene understanding and inferring semantic concepts (Bengio, 2009) have been more difficult to solve for AI systems (Goodfellow, Bengio & Courville, 2016). In the insurance domain, for example, consider the prior knowledge gained by an actuary from discussions with underwriters and claims handlers conducted before a reserving exercise. The experienced actuary is able to use this information to modify her approach to reserving, perhaps by choosing one of several reserving methods, or by modifying the application of the chosen method, perhaps by increasing loss ratios or decreasing claims development assumptions. However, automating the process of decoding the knowledge contained within these discussions into a suitable format for a reserving model appears to be a formidable task, mainly because it is often not obvious how to design a suitable representation of this prior knowledge.

In many domains, the traditional approach to designing machine learning systems to tackle these sorts of complex tasks relies on humans to design the feature vector X, or, as to use the terminology of the deep learning literature (Goodfellow, Bengio & Courville, 2016), the representation of the data. This feature vector is then

fed to a "shallow" machine learning algorithm, such as a boosted classifier, to learn from the data. For example, a classical approach to the problem of object localisation in computer vision is given in Viola and Jones (2001), who design a process, called a cascade of classifiers, for extracting rectangular features from images which are then fed into an Adaboost classifier (Freund & Schapire, 1997). An example closer to insurance is within the field of telematics analysis, where Dong et al. (2016) compare and contrast the traditional feature engineering approach with the automated deep learning approach discussed next. Within actuarial science, another example is the hand-engineered features[7] often used by actuaries for IBNR reserving. During the reserving process, an array of Loss Development Factors (LDFs) is calculated and combinations of these factors, calculated according to various statistical measures and heuristics, are 'selected' using the actuary's expert knowledge. Also, loss ratios are set based on different sources of information available to the actuary, some of which are less dependent on the claims data, such as the insurance company business plan, and some of which are calculated directly from the data, as in the Cape Cod (Bühlmann & Straub, 1983) and Generalised Cape Cod methods (Gluck, 1997). These features are then used within a regression model to calculate the IBNR reserves (more discussion of this topic appears in the description of CNNs below).

Designing features is a time-consuming and tedious task, and relies on expert knowledge that may not be transferable to a new domain, making it difficult for computers to learn complex tasks without major human intervention, effort and prior knowledge (Bengio & LeCun, 2007). In contrast to manual feature design, representation learning, which is thoroughly reviewed in Bengio, Courville and Vincent (2013), is a machine learning approach that allows algorithms automatically to design a set of features that are optimal for a particular task. Representation learning can be applied in both an unsupervised and supervised context. Unsupervised representation learning algorithms seek automatically to discover the factors of variation underlying the feature vector X. A commonly used unsupervised representation learning algorithm is PCA which learns a linear decomposition of data into the factors that explain the greatest amount of variance in the feature vector. An early example of supervised representation learning is the Partial Least Squares (PLS) method, described in Geladi and Kowalski (1986). After receiving a feature vector and an output, the PLS method calculates a set of new features that maximise the covariance with the response variable using the NIPALS algorithm (Kuhn & Johnson, 2013),[8] in the hope that these new features will be more predictive of the output than the unsupervised features learned using PCA.

The PCA and PLS methods just described rely on relatively simple linear combinations of the input data. These and similar simpler representation learning methods applied to highly complex tasks, such as speech or image recognition, may often fail

7 See Section 3 where IBNR reserving is cast as a regression problem.

8 The reader may draw an interesting parallel to neural networks from Figure 6.9 of Kuhn and Johnson (2013).

to learn an optimal set of features due to the high complexity of the data that the algorithms are analysing, implying that a different set of representation learning techniques are required for these tasks. Deep learning is a representation learning technique that attempts to solve the problem just described by constructing hierarchies of complex features that are composed of simpler representations learned at a shallow level of the model. Returning to the example from computer vision, instead of hand designing features, a more modern approach is to fit a neural network composed of a hierarchy of feature layers to the image dataset. The shallow layers of the network learn to detect simple features of the input data, such as edges or regions of pixel intensity, while the deeper layers of the network learn more complicated combinations of these features, such as a detector for car wheels or eyes. If performed in an unsupervised context, then the network learns a representation explaining the factors of variation of the input data, while if in a supervised context, the learned representation of the images is optimised for the supervised task at hand, for example, a computer vision task such as image classification.

The usefulness of learned representations often extends beyond the original task that the representations were optimised on. For example, a common approach to image classification problems when only small datasets are available is so-called "transfer learning" which feeds the dataset through a neural network that has been pre-trained on a much larger dataset, thus deriving a feature vector that can then be fed into a second neural network or other classification algorithm, see for example Girshick (2015). A structured data example closer to the problems often solved by actuaries is Guo and Berkhahn (2016) who tried to predict sales volumes based on tabular data containing a categorical feature with high cardinality (i.e. the feature contained entries for each of many different stores, a problem classically solved by actuaries using credibility theory). After training a representation of the categorical data using neural networks, the learned representation was fed into several shallow classifiers, greatly enhancing their performance. This approach to representation learning represents a step towards general AI, in which machines exhibit the ability to learn intelligent behaviours without much human intervention.

Definitions and Notation

This section now provides some mathematical definition of the concepts just discussed. The familiar starting point, in this paper, for defining and explaining neural networks is a simple linear regression model, expressed in matrix form:

$$\hat{y} = f(X) = a + B^t X$$

where the predictions $\hat{y}$ are formed via a linear combination of the features X derived using a vector of regression weights B and an intercept term, a. This simple model is shown in Figure 2 in graphical form. In this model, the features are used without modification directly to calculate the prediction, although, of course, the feature

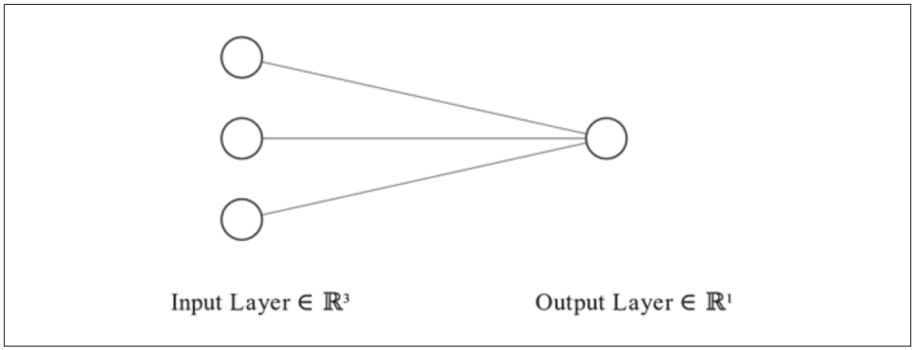

FIGURE 2 Regression model

vector X might have benefited from data transformations or other so-called feature engineering to exploit the data available for the prediction, for example, the addition of interaction effects between some of the variables. An example of an interaction term which might be added to the features in a motor pricing model is the interaction between power-to-weight ratio and gender, which might be predictive of claims. Feature engineering is often performed manually in actuarial modelling, and requires an element of judgement, leading to the potential that useful combinations of features within the data are mistakenly ignored.

Neural networks seek to solve the problem of manual feature engineering by allowing the model itself to design features that are useful in the context of the problem at hand. This is performed by constructing a more complex model that includes non-linear transformations of the features. For example, a simple extension of the linear regression model discussed above can be written as:

$$Z_1 = \sigma_0\left(a_0 + B_0^t X\right)$$
$$\hat{y} = \sigma_1\left(a_1 + B_1^t Z_1\right)$$

which states that an intermediate set of variables, Z^1, is computed from the feature vector X, by applying a non-linear activation function, σ_0 to a linear combination of the features, itself formed by applying weights B_0 and biases a_0 to the input feature vector X (the activation functions σ_0 are also called ridge functions in mathematics). A list of popular activation functions appears in Table 2. Following this first set of transformations of the variables by the non-linear activation function, the predictions of the model are computed as the output of another non-linear function σ_1 applied to a linear combination of the intermediate variables Z^1. The only specifications of the intermediate variables are the number of intermediate variables to calculate and the non-linear function – the data are used to learn a new set of features that is optimally predictive for the problem at hand. This model is shown in Figure 3.

TABLE 2 Popular activation functions

Function name	Definition	Range
Sigmoid	$\sigma(x)=\frac{1}{1+e^{-x}}$	(0,1)
Hyperbolic tangent	$\sigma(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	(-1,1)
Rectified linear unit (ReLu)	$\sigma(x)=\max(0,x)$	$[0,\infty)$
Softmax	$\sigma(x_i)=\frac{e^{x_i}}{\sum_{\forall i} e^{x_i}}$	(0,1)

The last layer of the network is equipped with an activation function designed to reproduce the output of interest; for example, in a regression model, the activation is often the identity function, whereas in a classification model with two classes, the sigmoid activation is used since this provides an output which can be interpreted as a probability (i.e. lies in (0,1)). For multi-class classification, a so-called softmax layer is used, which derives a probability for each class under consideration by converting a vector of real-valued outputs to a vector of probabilities.

To fit the neural network, a loss or objective function, $L(y,\hat{y})$, measuring the quality of (or distance between) the predicted output of the model, $\hat{y}$, compared to the observations y is specified, where the function $L(\ )$ can be chosen flexibly based on the problem at hand. For regression, common examples are the mean squared error (MSE) or the mean absolute error (MAE), while for classification the cross-entropy

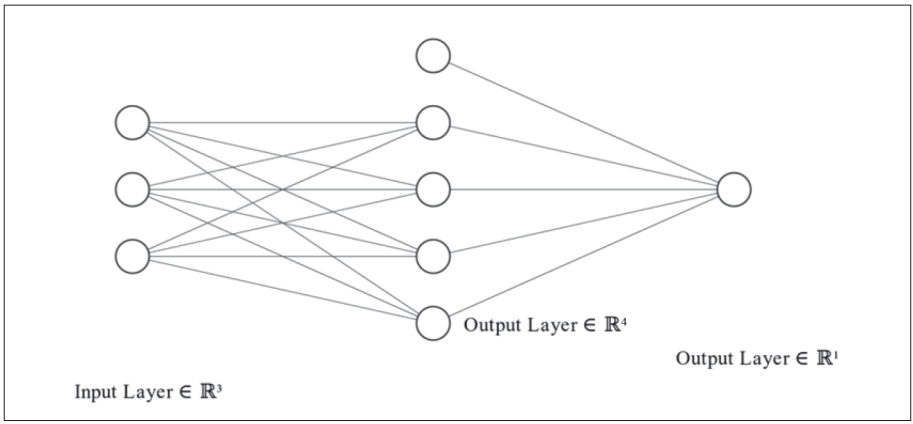

FIGURE 3 Neural network with a single hidden layer, consisting of four neurons plus a bias term

loss is often used. The model is then fit by back-propagation (Rumelhart, Hinton & Williams, 1986) and gradient descent, which are algorithms that adjust the parameters of the model by calculating a gradient vector (i.e. the derivative of the loss with respect to the model parameters, calculated using the chain-rule from calculus) and adjusting the parameters in the direction of the gradient (LeCun, Bengio & Hinton, 2015: Figure 1). In common with other machine learning methods, such as boosting, only a small "step" in the direction of the gradient is taken in parameter space for each round of gradient descent, to help ensure that a robust minimum of the loss function is found. The extent of the "step" is defined by a learning rate chosen by the user. Since the difficulties of performing back-propagation have largely been solved by modern deep learning software, such as the TensorFlow (Abadi et al., 2016) or PyTorch (Paszke et al., 2017) projects, the details are not given but the interested reader can refer to Goodfellow, Bengio and Courville (2016).

The simple model just defined can easily be extended to other more useful models. For example, by replacing the loss function with the Poisson deviance (Wüthrich & Buser, 2018), a Poisson GLM can be fit. More layers can be added to this model to produce a deep network capable of learning complicated and intricate features from the data. Lastly, specialised network layers have been developed to process specific types of data, and these are described later in this section.

Unsupervised learning is also possible, by using a network structure referred to as an auto-encoder (Hinton & Salakhutdinov, 2006). An auto-encoder is a type of non-linear principal components analysis where a (high dimensional) input vector X is fed into a neural network, called the encoder, which ends in a layer with only a few neurons. The encoded vector, consisting of these few neurons, is then fed back into a decoding network which ends on a layer with the same dimension as the original vector X. The loss function that is optimised is $L(X, \hat{X})$ where $\hat{X}$ represents the output of the network that is trained to be similar to the original vector, X. The structure of this model is shown in Figure 4, where a 10-dimensional input is reduced to a single "code" in the middle of the graph.

Deep auto-encoders are often trained using a process called greedy unsupervised learning (Goodfellow, Bengio & Courville, 2016). In this process, a shallow auto-encoder is first trained and, then, the encoded output of the auto-encoder is used as the input into a second shallow auto-encoder network. This process is continued until the required number of layers has been calibrated. A deep auto-encoder is then constructed, with the weights of these layers initialised at the values found for the corresponding shallow auto-encoders. The deep network is then finetuned for the application required.

With these examples in hand, some definitions of common terms in deep learning are now given. The idea of learning new features from the data is called *representation learning*, and is perhaps the key strength of neural networks. Neural networks are often viewed as being composed of layers, with the first layer referred to as *input layer*, the intermediate layers in which new features are computed being referred to as *hidden*

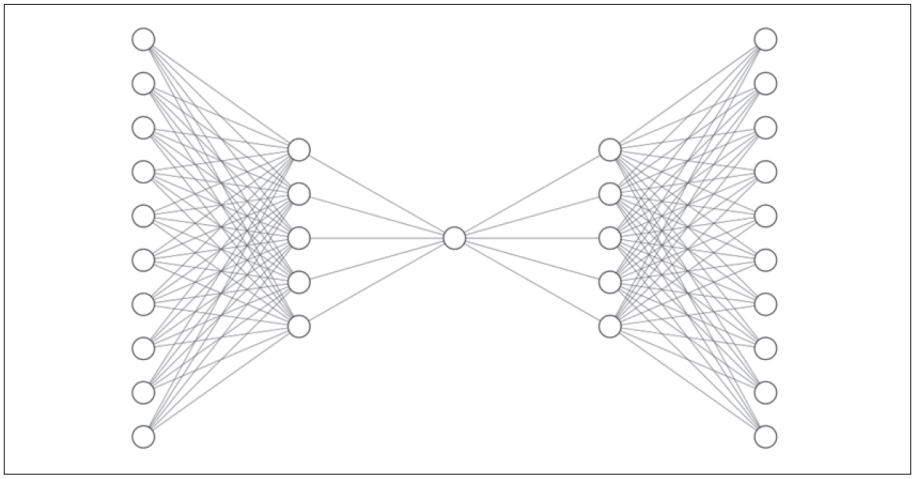

FIGURE 4 An auto-encoder network

layers, and the last calculation based on the learned features referred to as the *output layer*. Within each layer, the regression coefficients forming the linear combination of the features are called *weights*, and the additional intercept terms are called the *biases*. The learned features (i.e. the nodes) in the hidden layers are referred to as *neurons*.

An important intuition regarding deep neural network models is that multiple hidden layers (containing variable numbers of neurons) can be stacked to learn hierarchal representations of the data, with each new hidden layer allowing for more complicated features to be learned from the data (Goodfellow, Bengio & Courville, 2016).

Advanced Neural Network Architectures

Several specialised layers (convolutional, recurrent and embedding layers) have been introduced allowing for representation learning tailored specifically to unstructured data (images and text, as well as time series), or categorical data with a very high cardinality (i.e. with a large number of categories). Deep networks equipped with these specialised layers have been shown to excel at the tasks of image recognition and natural language processing, but also, in some instances, in structured data tasks (De Brébisson et al., 2015; Guo & Berkhahn, 2016). These specialised layers are described in the rest of this section. A principle to consider when reading the following section is that these layers have been designed with certain priors (in the Bayesian sense) or beliefs, in mind. For example, the recurrent layers express the belief that, for the type of data to which they are applied, the information contained in previous data entries has bearing on the current data entry. However, the parameters of these layers are not hand-designed, but learned during the process of fitting the neural network, allowing for optimised representations to be learned from the data.

Note that some advances in neural architectures have not been discussed, mainly because their application is more difficult or less obvious in the domain of actuarial science. For example, Generative Adversarial Nets (GANs) (Goodfellow et al., 2014) which have been used to generate realistic image, video and text are not reviewed. Also not discussed are the energy-based models, such as Restricted Boltzmann Machines (RBMs) which helped to revitalise neural network research since 2006 (Goodfellow, Bengio & Courville, 2016), but which do not appear to be used much in current practice.

CONVOLUTIONAL NEURAL NETWORKS

A convolution is a mathematical operation that blends one function with another (Weisstein, 2003). In neural networks, convolutions are operations performed on data matrices (most often representing images, but also time series or numerical representations of text) by multiplying element-wise part of the data matrix by another matrix, called the filter (or the convolutional kernel) and then adding each element in the resulting matrix. Convolutional operations are performed to derive features from the data, which are then stored as a so-called feature map. Figure 5 is an illustration of convolution applied to a very simple grey-scale image of a digit. To derive the first digit of the feature map shown in Figure 5, the element-wise product of the first 3×3 slice of the data matrix is formed with the filter and the elements are then added.

Many traditional computer vision applications use hand-designed convolutional filters to detect edges. Simple vertical and horizontal edges can be detected with filters comprised of 1, –1 and 0, for example, the filter shown in Figure 5 is a horizontal edge detector. More complicated edge detectors have appeared in the literature, see for example Canny (1987). In deep learning, however, the weights of the convolutional filters are free parameters, with values learned by the network and thus avoiding the tedious process of manually designing feature detectors.

Goodfellow, Bengio and Courville (2016) provide the following intuition for understanding CNNs: suppose that one placed an infinitely strong prior (in the Bayesian sense) on the weights of a neuron that dictated, firstly, that the weights were zero except in a small area and secondly, that the weights of the neuron were the same as the weights of its neighbour, but that its inputs were shifted in space. This would imply that the learned function should only learn local features from the data matrix and that the features detected in an object should be detected regardless of their position within the image. This prior is ideal for detecting features in an image, where specific components will vary in space and the components do not occupy the entire image.

The modern application of CNNs to image data generally involves the repeated application of convolutional layers followed by pooling layers, which replaces the feature map produced by the convolutional layer with a series of summary statistics (Goodfellow, Bengio & Courville, 2016). This induces translation invariance into the learned features, since a small modification of the image will generally not affect the output of the pooling layer. The multiple layers of convolution and pooling are

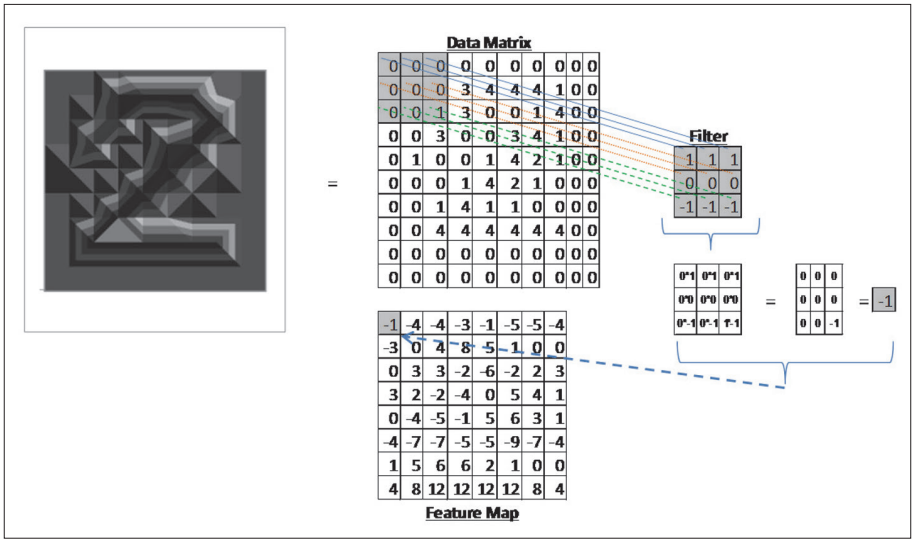

FIGURE 5 Illustration of a convolution applied to a simple grey-scale image of a '2'. The 10×10 image is represented in the data matrix, which consists of numbers representing the pixel intensity of the image. The filter is a 3×3 matrix, which, in this case, has been designed by hand as a horizontal edge detector. The resulting 8×8 feature map is shown as another matrix of numbers. The –1 in the top left corner of the feature map (in grey) is derived by applying the 3×3 filter to the 3×3 region of the data matrix also coloured in grey. The value is negative one, since the only non-zero entry in this part of the data matrix is 1, which is multiplied by the –1 in the bottom right of the filter. These calculations are shown on the right side of the figure. The same filter is applied in a sliding window over the entire data matrix to derive the feature map.

generally followed by several fully connected layers, which are responsible for using the features calculated earlier in the network to classify images or perform other tasks. Figure 6 shows a simple CNN applied to a tensor (matrix with multiple dimensions) of 3×128×128, in which the first dimension is typical of RGB images which have three channels.

As documented in LeCun, Bengio and Hinton (2015), CNNs are not new concepts, with an early example being LeCun et al. (1998). CNNs fell out of favour until the success achieved by the relatively simple modern architecture in Krizhevsky, Sutskever and Hinton (2012) on the ImageNet challenge. Better performing architectures with exotic designs and idiosyncratic names have been developed more recently; an early example of these exotic architectures is the inception model of Szegedy et al. (2015).

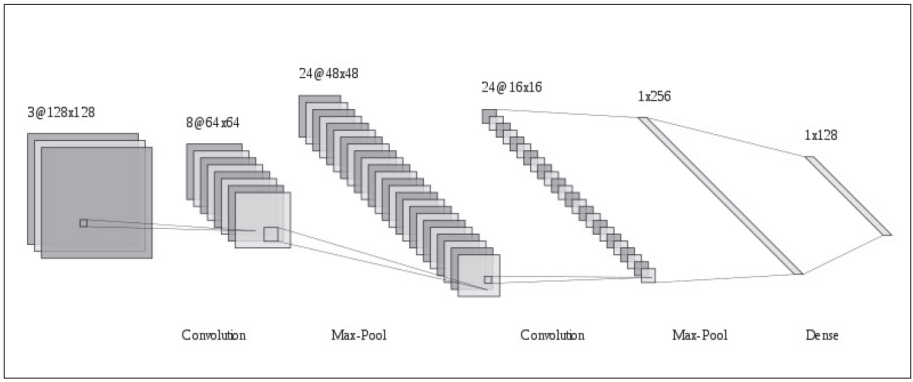

FIGURE 6 A simple convolutional network applied to a 3×128×128 tensor, which is characteristic of images where the pixels are defined on the RGB scale

Parallel to IBNR Reserving

An interesting parallel to the derivation of features for input into IBNR reserving methods (algorithms) can be drawn. The famous triangle of claims appearing in Mack (1993) is shown in Table 3, together with the chain-ladder calculations – performed using the ChainLadder package (Gesmann et al., 2017) in R – as well as earned premium that was estimated to provide a long-run ultimate loss ratio in each year of 65%.

Consider applying a simple 1×2 convolutional filter, consisting of the matrix $\begin{bmatrix}-1 & 1\end{bmatrix}$, to the triangle, after undergoing one of two transformations. In the first, the logarithm of the entries in the triangle not equal to zero is taken and, in the second, the entries are divided by the earned premium, relating to each accident year, to derive the cumulative loss ratios, and then the convolution is applied. The resulting feature matrices are shown in Table 4. The first matrix consists of the logarithm of the individual (i.e. relating to a single accident year) LDFs from the chain-ladder method and the second shows the incremental loss ratios (ILRs) used in the Additive Loss Ratio method described in Mack (2002). That the features underlying some of the most popular algorithms for reserving are relatively simple is not surprising, since these methods are engineered to forecast the total outstanding claims using consecutive single period forecasts. However, the possibility exists that a deep neural network might find more predictive features if trained on enough data.

RECURRENT NEURAL NETWORKS

The architectures discussed to this point have been so-called feed-forward networks, in which the various hidden layers of the network are recalculated for each data point and the hidden "state" of the network is not shared for different data points (i.e. there are no connections between hidden layers computed on different data points). Many data types, though, form sequences and a full understanding of the individual entries

TABLE 3 Claims triangle from Mack (1993) and the chain-ladder calculations

	1	2	3	4	5	6	7	8	9	10
1	357 848	1 124 788	1 735 330	2 218 270	2 745 596	3 319 994	3 466 336	3 606 286	3 833 515	3 901 463
2	352 118	1 236 139	2 170 033	3 353 322	3 799 067	4 120 063	4 647 867	4 914 039	5 339 085	
3	290 507	1 292 306	2 218 525	3 235 179	3 985 995	4 132 918	4 628 910	4 909 315		
4	310 608	1 418 858	2 195 047	3 757 447	4 029 929	4 381 982	4 588 268			
5	443 160	1 136 350	2 128 333	2 897 821	3 402 672	3 873 311				
6	396 132	1 333 217	2 180 715	2 985 752	3 691 712					
7	440 832	1 288 463	2 419 861	3 483 130						
8	359 480	1 421 128	2 864 498							
9	376 686	1 363 294								
10	344 014									

Accident year	Earned premium	Latest	Percentage developed	Ultimate	IBNR	ULR
1	6 002 251	3 901 463	100%	3 901 463	–	65%
2	8 359 568	5 339 085	98%	5 433 719	94 634	65%
3	8 275 117	4 909 315	91%	5 378 826	469 511	65%
4	8 150 625	4 588 268	87%	5 297 906	709 638	65%
5	7 474 154	3 873 311	80%	4 858 200	984 889	65%
6	7 863 340	3 691 712	72%	5 111 171	1 419 459	65%
7	8 708 878	3 483 130	62%	5 660 771	2 177 641	65%
8	10 438 152	2 864 498	42%	6 784 799	3 920 301	65%
9	8 680 409	1 363 294	24%	5 642 266	4 278 972	65%
10	7 645 885	344 014	7%	4 969 825	4 625 811	65%

of the sequences is only possible within the context provided by the other entries, for example, time series, natural language and video data. The problem of maintaining the information gained about the context from previous data points is addressed by recurrent neural networks (RNNs) , which can be imagined as feed-forward networks that share the state of the hidden variables from one data point to another. Whereas feed-forward networks map each input to a single output, RNNs map the entire history of inputs to a single output (Graves, 2012) and thus can build a so-called "memory" of the data points that have been seen.

A common simplifying assumption of actuarial and statistical models is the Markov assumption, which can be stated that once the most recent time-step of the data or parameters is known, the previous data points and parameters have no further bearing

TABLE 4 Feature matrices derived by applying a 1×2 convolution filter to the triangle shown in Table 3, adjusted as described in the text. The first matrix shows the log LDFs and the second shows the ILRs.

Log chain-ladder LDFs									
	1	2	3	4	5	6	7	8	9
1	1.15	0.43	0.25	0.21	0.19	0.04	0.04	0.06	0.02
2	1.26	0.56	0.44	0.12	0.08	0.12	0.06	0.08	
3	1.49	0.54	0.38	0.21	0.04	0.11	0.06		
4	1.52	0.44	0.54	0.07	0.08	0.05			
5	0.94	0.63	0.31	0.16	0.13				
6	1.21	0.49	0.31	0.21					
7	1.07	0.63	0.36						
8	1.37	0.70							
9	1.29								

Incremental Loss Ratios										
	1	2	3	4	5	6	7	8	9	10
1	6%	13%	10%	8%	9%	10%	2%	2%	4%	1%
2	4%	11%	11%	14%	5%	4%	6%	3%	5%	
3	4%	12%	11%	12%	9%	2%	6%	3%		
4	4%	14%	10%	19%	3%	4%	3%			
5	6%	9%	13%	10%	7%	6%				
6	5%	12%	11%	10%	9%					
7	5%	10%	13%	12%						
8	3%	10%	14%							
9	4%	11%								
10	4%									

on the model. RNNs allow this assumption to be relaxed (Goldberg, 2017) by storing the parts of the history, shown to be relevant, within the network's internal memory.

The two common forms of representing an RNN are shown in Figure 7, with the folded representation on the left of the figure, and the unfolded (over time) representation on the right. The figure shows a simple RNN that was proposed by Elman (1990). It should be noted that this figure is a simplification and multi-dimensional inputs into and out of the network are possible, as well as multiple RNN layers stacked on top of each other. Also, nothing precludes using the output of another neural network as the input into an RNN or vice-versa, for example, LeCun, Bengio and Hinton (2015:

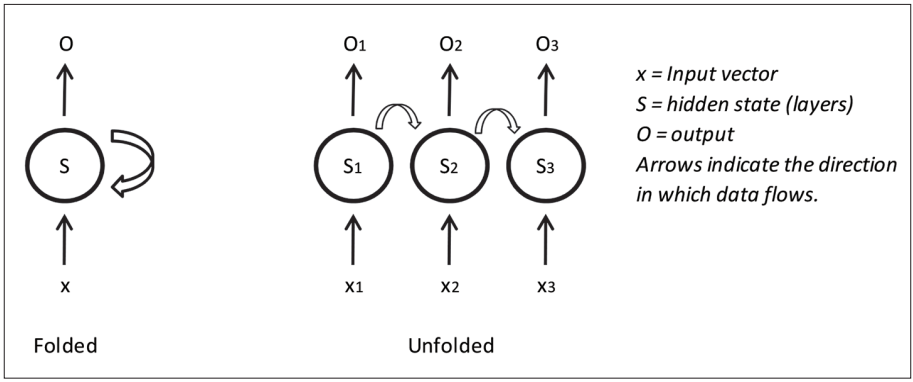

FIGURE 7 Common graphical representations of a Recurrent Neural Network. The graph on the left shows a folded form of the RNN and the graph on the right shows the same RNN unfolded over time. At each step, the parameter values are shared. Note that the inputs to the RNN can be multi-dimensional, the actual RNN can have many layers, and the outputs can be fed into other types of network architectures.

Figure 3) provide an example of an image captioning network which takes, as an input to the RNN, the output of a convolutional neural network and produces, as an output of the RNN, image captions.

Viewed this way, it can be seen that RNNs are a form of deep neural network, where the depth extends over a time dimension, and not just over a processing network (as in the feed-forward networks discussed above) (Goldberg, 2017). Therefore, the back-propagation algorithm can be applied to train the RNN if the gradients are propagated over time (Werbos, 1988).

Although RNNs are conceptually attractive, they proved difficult to train in practice due to the problem of either vanishing, or exploding, gradients (i.e. when applying back-propagation to derive the adjustments to the parameters of the network, the numerical values that are calculated do not provide a suitable basis for training the model) (LeCun, Bengio & Hinton, 2015). Intuitively, back-propagation is similar to the chain rule in calculus, and by the time the chain rule calculations reach the earliest (in time) nodes of the RNN, the weight matrix that defines the feedback loop shown in Figure 7 has been multiplied together so many times that the gradients are no longer accurate.

Other, more complicated, architectures that do not suffer from the vanishing or exploding gradient problem are the LSTM (Hochreiter & Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Chung et al., 2015) network designs. An intuitive explanation of these so-called "gated" designs is that, instead of allowing the hidden state automatically to be updated at each time step, updates of the hidden state should occur only when required (as learned by the gating mechanism of the networks from the data), creating a more stable feedback loop in the hidden state cells than a simple

RNN. The influence of earlier inputs to the network is therefore maintained over time, thus mitigating the training issues encountered with simple RNNs (Goodfellow, Bengio & Courville, 2016; Graves, 2012). For more mathematical details of LSTMs and GRUs and explanation of how these architectures solve the vanishing/exploding gradient problem, the reader is referred to Goodfellow, Bengio and Courville (2016: Section 10.10). These details may appear complicated, and perhaps even somewhat fanciful on a first reading, however, key is that LSTM and GRU cells, which are useful for capturing long-range dependencies in data, can easily be incorporated into a network design using modern software, such as Keras.

RNNs have achieved impressive success in many specialised tasks in Natural Language Processing (NLP) (Goldberg, 2017), machine translation (Sutskever, Vinyals & Le, 2014; Wu et al., 2016), speech recognition (Graves, Mohamed & Hinton, 2013) and, perhaps most interestingly for actuaries, time series forecasting (Makridakis, Spiliotis & Assimakopoulos, 2018a), in combination with exponential smoothing.

EMBEDDINGS

When dealing with categorical data, machine learning algorithms generally rely on the one-hot encoding procedure, which is often called dummy-coding by statisticians. In this encoding scheme, vectors of categorical features are expanded into a set of indicator variables, all of which are set to zero, except for the one indicator variable relating to the category appearing in each row of the feature vector. When applied to many categories, one-hot encoding produces a high-dimensional feature vector that is sparse (i.e. many of the entries are zero), leading to potential difficulties in fitting a model as there might not be enough data to derive credible estimates for each category. Modelling categorical data with high cardinality (i.e. when the number of possible categories is very large) is a task that has been addressed in actuarial science within the framework of credibility theory; see, for example, Bühlmann and Gisler (2006) for a detailed discussion of the Bühlmann-Straub credibility model. Credibility models do not rely exclusively on the observations available for each category, but rather produce pooled estimates that lie between the overall mean and the observation for each category. In statistics, GLMMs are an extension of standard GLMs to include variables that are not modelled solely on the basis of the observed data, but rather using distributional assumptions that lead to similar formulae as credibility theory (for the interested reader, Gelman and Hill (2007) is an easy introduction to GLMMs and Chapter 12 of this book contains discussions that are highly reminiscent of credibility theory).

The problem of modelling high-dimensional and sparse data is particularly acute in NLP (Bengio et al., 2003), where words in a corpus of text are often represented by sparse integer valued features (i.e. one-hot encoding) or more complicated features called n-grams which are essentially small phrases of n words that are grouped together (for example, in the sentence 'The quick brown fox jumps over the lazy dog', the first 3-gram is the phrase 'The quick brown'). N-grams use the context of words

appearing together in sentences to help reduce the dimensionality of the language modelling problem but suffer from two main problems. Firstly, the contexts used are often short (i.e. 1-gram i.e. one-hot encoding or 2-gram schemes are often used) and the similarity between words is not accounted for (i.e. the feature vectors used in the models are orthogonal to each other by design). The insight of Bengio et al. (2003) to solve these problems is that language data can successfully be encoded into low dimensional, dense numerical vectors which can be trained using neural networks. These vectors are trained to be close to each other (in the vector space) if the words are similar and distant if they are not, thus, the model learns to associate correlated words and "gains *statistical strength* by sharing parameters" (Goldberg, 2016:351) (emphasis added to show the connection to credibility methods), which is accomplished by allowing the model to treat words with similar features in a common way (Goodfellow, Bengio & Courville, 2016). In NLP, embeddings are often pre-trained on a large corpus of text and then applied to specific tasks and many researchers have made pre-trained embeddings freely available on the Internet.

Embeddings have been used in wider contexts than NLP, two examples of which are De Brébisson et al. (2015) and Guo and Berkhahn (2016). In the context of predicting taxi destinations based on a number of categorical variables and GPS data, De Brébisson et al. (2015) build a network with separate embeddings for each categorical variable, which was the winning solution in the Kaggle Taxi Service Prediction problem. Guo and Berkhahn (2016) discuss embeddings in the context of the Kaggle Rossmann Sale Prediction competition. Their model consists of several embeddings of categorical variables concatenated into a single layer, on top of which were placed two fully connected layers. The model was trained to predict daily sales for each Rossmann store and the model was tested on two versions of the dataset – one shuffled, and one unshuffled, so that more recent entries appeared towards the end of the dataset. On the shuffled dataset, using embedding layers did not boost the performance of the neural network significantly compared to using one-hot encoding, whereas on the unshuffled dataset, the embeddings provided a relatively small boost in performance. The trained embeddings were then used in other shallow classifiers, and, compared to the one-hot encoded features, provided a major boost in performance i.e. the out-performance of the deep networks was due to the learned feature representation for the categorical data.

Goldberg (2016: Section 4.6) provides some intuition why an embedding layer might not result in better performance in a neural network. When categorical features have been one-hot encoded, the first layer of the neural network will learn a representation of all of the features together that is comparable to an embedding layer, thus, the differences between the approaches are potentially more subtle than would initially appear. The key difference seems to be that embedding layers allow a dense representation to be learned separately for each feature before being combined in the neural network, and, along these lines, Guo and Berkhahn (2016) note that embedding layers allow the network to learn the intrinsic properties of each feature as well as learning about the output distribution.

Embeddings are often visualised using dimensionality reduction algorithms such as PCA or t-SNE (Maaten & Hinton, 2008) and often reveal surprisingly intuitive properties of the learned embeddings. Interesting examples of such a visualisation is Figure 2 in Mikolov et al. (2013), which shows that the embedding layers have learned a semantic relationship relating capital cities to the countries in which they are found and Figure 3 in Guo and Berkhahn (2016) which shows that the embedding of store locations has learned a representation suggestive of the map of Germany (in which the Rossmann stores are located).

Technology and Resources

This section does not provide an exhaustive review of the available software and resources, but discusses examples, from the author's experience, that are particularly useful.

SOFTWARE

Several open-source software packages allowing for the application of deep learning are currently available. Amongst these, the TensorFlow (Abadi, Barham, Chen et al., 2016) and PyTorch (Paszke, Gross, Chintala et al., 2017) packages provide low level interfaces for defining and fitting deep neural networks, both of which are accessible in the Python programming language and provide support for fitting neural networks on central processing units (CPUs) as well as GPUs, the latter being recommended for deep neural networks involving the application of specialised layers. Some of the workflow used when defining TensorFlow models in the form of programming graphs might appear very foreign compared to the same workflow in PyTorch, which, anecdotally, is said to be more intuitive for users of Python.

A notable high level interface is Keras (Chollet, 2015), which provides an intuitive domain-specific language for easily designing and fitting neural networks, which are then run on a low level interface, such as TensorFlow. Keras abstracts the many of the finer details away from the user and requires less technical knowledge than the low level packages mentioned before. An interface between Keras and the R language (R Core Team, 2018) has been made available (Allaire & Chollet, 2018), which is notable since R appears to be more popular among actuaries than Python. For the code examples, this paper uses the R interface to Keras, running on the TensorFlow back-end.

HARDWARE

Modern neural networks require a significant amount of computing power to fit. GPUs are often used to speed up the fitting of deep networks, and the software packages just discussed provide the option to use GPUs. Several web-based platforms offer GPU access in the cloud if local access to a GPU is not available.

RESOURCES

The book by Goodfellow, Bengio and Courville (2016) is currently the most comprehensive resource providing background, theory and perspective on all aspects of

modern neural networks, from the vantage point of computer science. Chollet (2017) provides a practical guide to the Keras package in Python and a similar guide to the Keras package in R is Chollet and Allaire (2018). Both of these resources discuss practical aspects of fitting deep neural networks and provide example code in the respective programming languages. A concise introduction to deep learning from a statistical viewpoint is Section 18 in Efron and Hastie (2016). Beyond books, the online course by Ng (2017) provides theory, practical advice and code examples in Python, and the Deep Learning Nanodegree (Udacity, 2018) combines theory with several projects applying deep learning.

Conclusions

This section has provided a summarised review of representation learning, deep neural networks and their connection to AI, and specialised architectures for dealing with image, text and categorical data. At this point, we refer the reader back to Table 1, to consider what an enhanced feature vector for common actuarial problems, expressed as regression models, could potentially comprise. Some examples of enhanced feature vectors are as follows:

— Using CNNs, image data easily can be incorporated into the actuarial modelling process if this was considered to be predictive.
— Textual data, for example description of risks or claims handlers' notes, could be incorporated into the pricing and reserving process using RNNs and word embeddings.
— Categorical data in actuarial models (for example, brands of cars and product types) can be modelled with embeddings, potentially enhancing predictive power.

The next section explores recent attempts to incorporate neural networks into actuarial models.

4. SURVEY OF DEEP LEARNING IN ACTUARIAL SCIENCE

This section presents a non-exhaustive review of recent applications of neural networks in actuarial science. Since this paper is concerned with the modern application of neural networks, searches within the actuarial literature were confined to articles written after 2006, when the current resurgence of interest in neural networks began (Goodfellow, Bengio & Courville, 2016). The prominent actuarial journals listed in Appendix A were searched for the key phrase "neural networks" and the search was augmented by similar searches on the SSRN and arxiv repositories. Articles making only a passing reference to neural networks were dropped from the list and the resulting list of topics and papers reviewed in this section is:

— Pricing of non-life insurance (Noll, Salzmann & Wüthrich, 2018; Wüthrich & Buser, 2018)
— IBNR reserving (Kuo, 2018b; Wüthrich, 2018b; Zarkadoulas, 2017) and the individual claims simulation machine of Gabrielli and Wüthrich (2018)

— Analysis of telematics data (Gao, Meng & Wüthrich, 2018; Gao & Wüthrich, 2017; Wüthrich & Buser, 2018; Wüthrich, 2017)
— Mortality forecasting (Hainaut, 2018a)
— Approximating nested stochastic simulations (Hejazi & Jackson, 2016; 2017)
— Forecasting financial markets (Smith, Beyers & De Villiers, 2016)

Code for some of the examples in this section and pre-trained models to reproduce the results are provided at: *https://github.com/RonRichman/AI_in_Actuarial_Science/*

Pricing of Non-Life Insurance

The standard method for pricing lines of non-life insurance business with significant volumes of data, such as personal lines, is GLMs (Parodi, 2014), which extend linear regression models in two main ways – firstly, the assumption of a Gaussian error term is replaced with an extended class of error distributions from the exponential family, such as the Poisson, Binomial and Gamma distributions, and, secondly, the models are only assumed to be linear after the application of a link function, canonical examples of which are the log link for Poisson distributions and the logit link for Binomial distributions. Separate models for frequency and severity are often used, but the option of modelling pure premium with Tweedie GLMs also exists. For more details on GLMs, see Ohlsson and Johansson (2010) and Wüthrich and Buser (2018). GLMs have been extended in several notable ways, such to incorporate smooth functions of the covariates in Generalised Additive Models (GAMs) (Wood, 2017) or to apply hierarchal credibility procedures to categorical covariates, as in GLMMs (Gelman & Hill, 2007). Similar to many other supervised machine learning methods, such as boosted trees or LASSO regression, GLMs are usually applied to tabular, or structured, data, in which each row consists of an observation, and the columns consist of explanatory and target variables.

Two recent examinations of the performance of GLMs to model frequency compared to machine learning techniques are Noll, Salzmann and Wüthrich (2018) and Wüthrich and Buser (2018). Noll, Salzmann and Wüthrich (2018) apply GLMs, regression trees, boosting and neural networks to a real Third Party Liability (TPL) dataset consisting of explanatory variables, such as age of the driver and region, and target variables, consisting of the number of claims, which is freely available as an accompaniment to Charpentier (2014). Wüthrich and Buser (2018) apply a similar set of models to a simulated Swiss dataset. Since the study of Noll, Salzmann and Wüthrich (2018) is on real data where the data generating process is unknown (i.e. similar to the situation in practice), we focus on that study, and report supplementary results from Wüthrich and Buser (2018).[9]

9 In passing, we also mention Hainaut (2018b), who applies Self Organising Maps (SOMs) to the problem of variable clustering for non-life modelling, but do not study it further since SOMs are not feed-forward neural networks.

After providing a detailed exploratory data analysis (EDA) of the French TPL dataset, Noll, Salzmann and Wüthrich (2018) apply the fundamental step in the machine learning process of splitting the dataset into training and test subsets, comprising 90% and 10% of the data respectively. Models are fit on the training dataset, but tested on out-of-sample data from the test set, to give an unbiased view of the likely predictive performance if the model were to be implemented in practice.[10]

Since the aim is to compare models of frequency, the loss function (which measures the distance between the observations and the fitted model) used is the Poisson deviance (minimising the Poisson deviance is equivalent to maximising the likelihood of a Poisson model, see Wüthrich and Buser (2018)):

$$L(D,\lambda) = \frac{1}{n}\sum_{i=1}^{n} 2N_i\left(\frac{\lambda_i V_i}{N_i} - 1 - \log\left(\frac{\lambda_i V_i}{N_i}\right)\right)$$

where D represents the dataset with n policy observations having an exposure for policy i of V_i, number of claims N_i and modelled frequency of claim λ_i.

Since the EDA shows that the empirical (marginal) frequencies of claim are not linear for some of the variables, which is the underlying assumption of a log-linear GLM, some of the continuous variables were converted to categorical variables i.e. some manual feature engineering was applied. When fitting the GLM, it was noted that the Area and Vehicle Brand variables appeared not to be significant based on the p-values for these variables, however, including both variables contributes to a better out-of-sample loss, leading the authors to select the full model (see the discussion in Shmueli (2010) and Section 2 above about the difference between explanatory and predictive models). Although extensions of GLMs are not considered in detail this study, Wüthrich and Buser (2018) report good performance of GAMs on the simulated dataset they study (which contains only weak interactions between variables). After fitting the GLM, a regression tree and a Poisson Boosting Machine (PBM) are applied (see Wüthrich & Buser (2018: Section 6) for the derivation of the PBM, which is similar to a Gradient Boosting Machine) to the unprocessed dataset, with both models out-performing the GLM, and the PBM outperforming the regression tree on out-of-sample loss. A shallow neural network was then fit to the data, with a hidden layer of 20 neurons, using the sigmoid activation function and the Poisson deviance loss, with a momentum-enhanced version of gradient descent. The neural network provides an out-of-sample loss between the regression tree and the PBM. Interestingly, the authors note that using the exposure as a feature, instead of a constant multiplied by the claims rate λ_i provides a major performance

10 Often, three splits of the data are recommended – a training split, on which to fit the models, a validation split, on which to compare model performance, and a test split, which is only used at the end of the modelling process to estimate likely performance of the model. Combining the test and validation splits leads to the risk that models will be over-fit to the test set if many iterations of models are used to optimise test set performance.

improvement. The conclusion of the study is that both the PBM and neural network outperform the GLM, which might be because the GLM does not allow for interactions, or because the assumption of a linear functional form is incorrect. Generously, code in R for all of the examples is provided by the authors, and, in the following section, it is shown how the neural network model in this example can be fit and extended using Keras.

ANALYSIS USING KERAS

In this section, we show how Keras can be used to fit several variations of the shallow neural network discussed in Noll, Salzmann and Wüthrich (2018). At the outset, it is worth noting that the signal-to-noise ratio of claims frequency data is often relatively low (Wüthrich & Buser, 2018: Section 2.5.5) and very flexible or high dimensional models might not provide much extra accuracy with this type of data.

As noted above, neural networks learn representations of the features that are optimal for the problem they are optimised on, by stacking layers of neurons on top of each other. Each hidden layer can be thought of as a non-linear regression with a feature representation as the output. A neural network without any hidden layers (i.e. without representation learning) is simply equivalent to a regression model, and, therefore, for ease of exposition we start with a GLM model in Keras, following the data processing steps in Noll, Salzmann and Wüthrich (2018).

Two points should be noted when trying to use Keras to fit a Poisson rate regression. The first step is to define a custom Poisson deviance loss function, which is not a standard loss function within Keras. Secondly, the model required is a Poisson rate regression, where the exposure measure is the time that a policy is in force, and the regression fits the rate of claim based on this exposure measure. To fit this model in Keras, the easiest option is to use the Functional Application Programming Interface to multiply the output of the neural network with the exposure to derive the expected number of claims. Since the rates of claim lie between 0 and 1, the output layer of the neural network was chosen to use a sigmoid activation, and therefore the network learns the rates on a logit scale, which is slightly different from the log scale used by the GLM. This model is denoted *GLM_Keras* in Table 5, whereas the baseline GLM fit using the *glm()* function in R is denoted *GLM*. The code in R to accomplish this is:

```
dense = NumInputs %>%
layer_dense(units = 1, activation = 'sigmoid')

N = layer_multiply(list(dense,Exposure), name = 'N')
```

The out-of-sample Poisson deviance in Table 5 for the GLM fit in Keras is close to that of the baseline GLM. Fitting a shallow neural network requires a single extra line of code:

```
dense = NumInputs %>%
layer_dense(units = 20, activation = 'relu') %>%
layer_dense(units = 1, activation = 'sigmoid')
```

In this case, a hidden layer with 20 neurons was used, with the ReLu activation function. The same dataset used for the GLM that contained the categorical conversion of the numerical inputs was used as the data in this network, which performed on a par with the neural network model in Noll, Salzmann and Wüthrich (2018) and is denoted *NN_shallow* in the table. However, this model is somewhat unsatisfying since it relies on manual feature engineering i.e. converting the numerical inputs to categorical values.

Table 5 Out-of-sample Poisson deviance for the French TPL dataset

Model	Out of sample
GLM	0.3217
GLM_Keras	0.3217
NN_shallow	0.3150
NN_no_FE	0.3258
NN_embed	0.3068
GLM_embed	0.3194
NN_learned_embed	0.2925

Therefore, another network (denoted *NN_no_FE*) was tried, but the out-of-sample performance was worse than the GLM, indicating that the network did not manage to fit continuous functions to the numerical inputs properly. To overcome this deficiency, the specialised embedding layer discussed above was added to the model for all of the numerical and categorical variables, with a hand-picked number of dimensions for each embedding. The intuition behind this is that the network might better be able to learn the highly non-linear functions mapping the numerical variables to frequency by learning a meaningful multi-dimensional vector for each category. To do this, the numerical variables were split into categories by the quantiles of the numerical variable, defined so as to produce a minimum of 10 000 observations per group. The results of this network, denoted *NN_embed*, were significantly better than all of the models considered to this point. The learned embeddings for Driver Age are shown in Figure 8, after applying the t-SNE technique (Maaten & Hinton, 2008) to the reduce the multi-dimensional embedding values down to two dimensions.

The figure shows that the embeddings for the youngest age groups in the model (i.e. 20 and 22) cluster together, with two additional clusters relaying to the young adult and older adult ages apparent. Figure 9 shows the relationship of the embeddings (again in two-dimensions derived using the t-SNE technique) to driver age. The

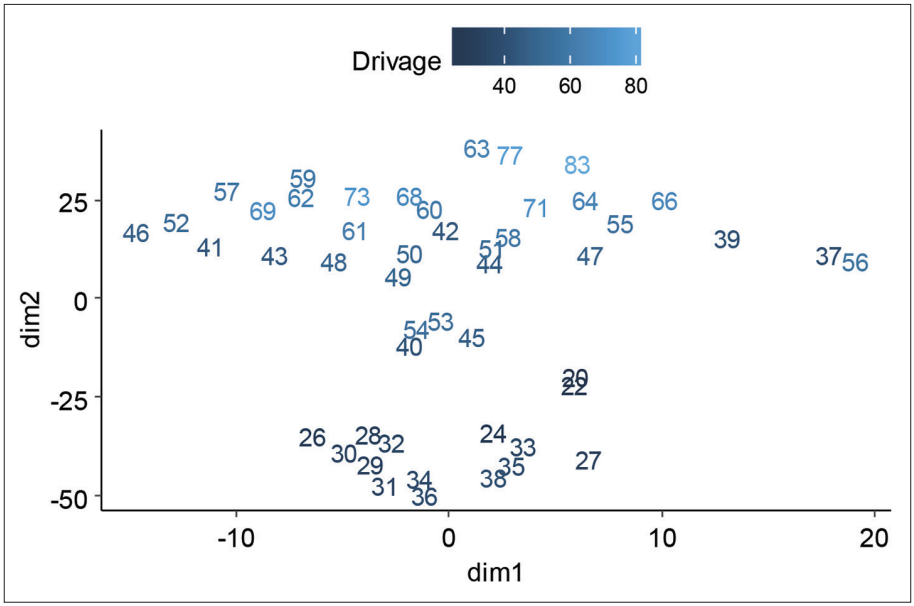

FIGURE 8 Driver age embedding reduced to two dimensions using the t-SNE technique

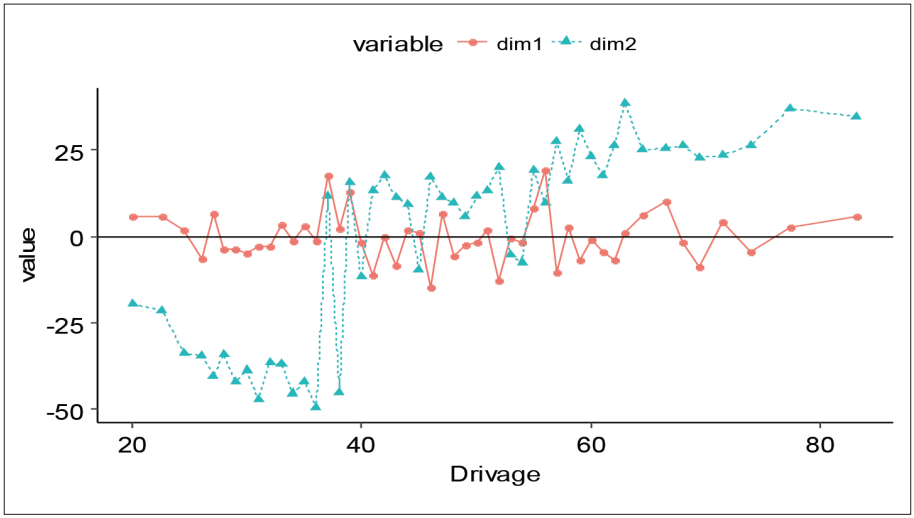

FIGURE 9 Driver age embedding reduced to two dimensions using the t-SNE technique, plotted against the Driver Age variable

interpretation here is that the first dimension represents a schedule of claims rates according to age, with the rates starting off high at the youngest ages (i.e. above zero), dropping down in the middle ages mostly below zero) and then rising at the older ages (mostly above zero). The second dimension shows how the relationship between the claims rates at the younger and older ages might change by "twisting" the schedule of rates (i.e. if the rates at younger ages go down, those at older ages go up and vice-versa). The embedding therefore seems to tie in with the intuition that the youngest and least experienced drivers will have similar high claims frequency experience, that more experienced but nonetheless young drivers will have the lowest claims rates and that the oldest drivers will again experience a rise in frequency.

That the learned embedding has an easy and useful interpretation was then shown by including the embedding in a simple GLM model in place of the categorical Driver Age variable, and this result is referred to in the table as *GLM_embed*. This application of transfer learning (i.e. reuse of a calibrated neural network component in another model) reduced the out-of-sample loss of the GLM relatively significantly, showing that the learned embedding has a meaning outside of the neural network, and, presumably, even better performance could be achieved by replacing all of the variables in the GLM with the corresponding embeddings. Since actuaries are more used to pricing personal lines business using GLMs, pre-training embeddings in a neural network which are then used within GLMs as additional parameter vectors seems like a promising strategy to be explored further.

A last illustration[11] of the flexibility of neural networks builds upon the observation of Noll, Salzmann and Wüthrich (2018) that including exposure as a feature component instead of a multiplicative constant results in a much more predictive model. The network denoted as *NN_learned_embed* implements this idea in two ways – firstly, the exposure is input into the model via an embedding layer, in the same way as the rest of the numerical features, and, secondly, a secondary network is built (within the main network) to apply a multiplicative scaling to the policy exposure. This network is shown in Figure 10.

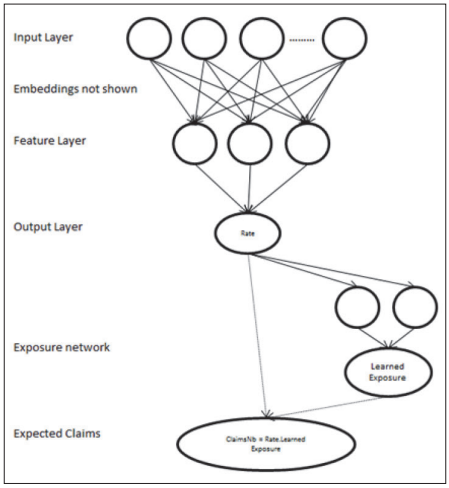

FIGURE 10 Learned exposure network

Of the networks discussed, the learned exposure network by far outperforms the rest (note that including

11 The author wishes to acknowledge and thank Mario Wüthrich for a discussion of the idea of learned exposure.

the exposure embedding but not the secondary network did not produce as optimal a result). To build some intuition, the average learned exposures are shown in Figure 11 plotted against the average exposures in the data, segmented by the number of claims. The figure shows that at the lowest level of exposures, the network has dramatically increased the level of exposure for those policies claiming at least once, as well as for multi-claimants, indicating that the time a policy spends in force is not an optimal exposure measure, and, indeed, telematics data appears to indicate that distance travelled is a more optimal measure (Boucher, Côté & Guillen, 2017).

IBNR Reserving

Reserving for IBNR claims is a fundamental task of actuaries within non-life insurance companies, primarily to ensure the company's current and future ability to meet policy holder claims, but also to allow for the up-to-date tracking of loss ratios on an accident and underwriting year basis, which is essential for the provision of accurate management information. Since Bornhuetter and Ferguson (1973) encouraged actuaries to pay attention to the topic of IBNR reserves, a vast literature on loss reserving

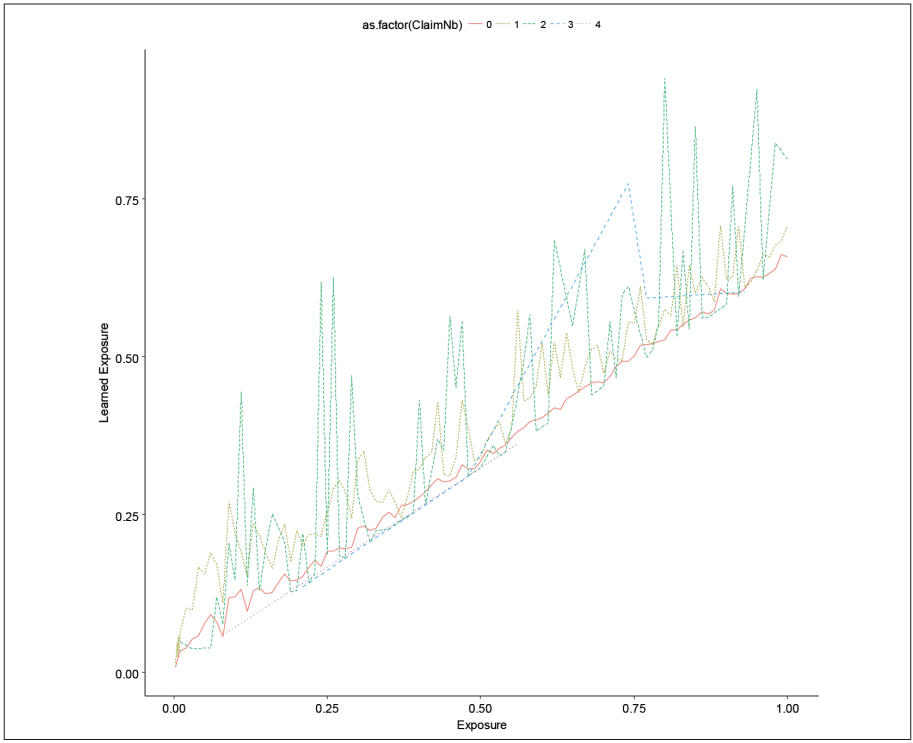

FIGURE 11 Learned exposures, segmented by number of claims made, plotted against exposure.

techniques has emerged; see, for example, Schmidt (2017) and Wüthrich and Merz (2008). Nonetheless, the relatively more simple chain-ladder method remains the most popular choice of method for actuaries reserving for short-term insurance liabilities, globally and in South Africa (Dal Moro, Cuypers & Miehe, 2016).

As mentioned in Sections 2 and 3, the chain-ladder technique can be viewed as a regression, with a specific type of feature engineering used to derive forecasts of the aggregate claims still to be reported, and, therefore, it is not surprising that enhanced approaches based on machine and deep learning are possible. In this section we discuss several recent contributions applying neural networks to the problem of IBNR reserving.

GRANULAR RESERVING USING NEURAL NETWORKS

IBNR reserves are generally calculated at the level of a line of business (LoB), which represents a homogenous portfolio with similar exposure to risk and types of claims. This high-level calculation is generally sufficient for solvency reporting and basic management information. However, a vexing problem in practice is the production of reserves for groupings of business lower than the LoB aggregation, since, on the one hand, these more granular calculations might reveal important information about the business such as particularly poorly performing segments, but on the other hand, the more granular data is often too sparse for the traditional chain-ladder calculations. One potential solution is the various credibility-weighted extensions of the chain-ladder, see for example Bühlmann et al. (2009); Gisler and Wüthrich (2008), however, these have been developed only to the extent of a single level of disaggregation from the usual calculations.

A more recent approach is Wüthrich (2018b) who extends the formulation of the chain-ladder as a regression model to incorporate features such as LoB, type of injury and age of the injured. The basic chain-ladder method uses the aggregated incurred or paid claims (or claim amounts) from accident period *i* developed up to a point in time *j*, $C_{i,j}$, as the sole input feature into a regression model:

$$\hat{C}_{i,j+1} = f_j . C_{i,j}$$

where $\hat{C}_{i,j+1}$ is the estimated claim amount for the next development period and f_j is the so-called LDF (after estimating $\hat{C}_{i,j+1}$, the calculations are continued using the predicted value as the next feature for that accident year). This could be expressed in a functional form $\hat{C}_{i,j+1} = f(X).C_{i,j}$, where X is a feature vector containing only one feature, in this case, the development period *j*. The insight of Wüthrich (2018b) is to extend the feature space X to include other components, namely, LoB, labour sector, accident quarter, age of the injured and body part injured, and then, to fit a neural network model to estimate the expanded function $f(X)$. Once this function is estimated, it is used to derive LDFs at the level of granularity of the expanded feature vector X, and the chain-ladder method can be applied as normal.

One complication is that as the data are disaggregated, the multiplicative structure of the model may fail if $C_{i,j}$ is zero (i.e. no claims have been incurred/paid in accident period i up to a point in time, j). Two approaches are possible. A simpler approach is to calculate a second chain-ladder model, fit for each LoB separately, which projects the total aggregated $C_{i,j}$ to ultimate using modified LDFs that allow only for the development of claims from zero and a more complicated approach would allow for an intercept in the regression model that is only calculated if $C_{i,j}$ is zero, i.e. $\hat{C}_{i,j+1} = f_j.C_{i,j} + I_{\{C_{i,j}=0\}}.g_j$.

The neural network used in Wüthrich (2018b) had a single layer of 20 neurons and used the hyperbolic tangent activation function. Comparing the neural network and chain-ladder reserves to the true underlying dataset, it was found that the neural network reserves were almost perfectly accurate for three of the four LoBs considered, and more accurate than the chain-ladder reserves, while, on the fourth line, both the neural network and chain-ladder reserves were over-stated, the former more than the latter. The results of the neural network approach were also quite accurate at the level of age, injury code, and claims code, which, besides for allowing for a more detailed understanding of the aggregate IBNR requirements, also allows for the production of detailed management information, as well as input into pricing. To explain the out-performance of the neural network model, Wüthrich (2018b) notes that the neural network is inherently a multivariate approach that leverages similar patterns in the data across LoBs, whereas the chain-ladder is applied to each LoB separately.

In summary, the neural network chain-ladder model is a combination of a traditional actuarial method with a machine learning method, and this paradigm of combining traditional techniques with neural networks will be noted throughout this section.

We refer the reader to the appendices in Wüthrich (2018b) for code to fit the model in Keras.

INDIVIDUAL CLAIMS SIMULATION MACHINE

Gabrielli and Wüthrich (2018) make an important contribution by providing software to simulate individual claims for the purpose of testing individual claims reserving methods. Although the methodology used to simulate is not directly applicable to reserving in practice (since it focuses on modelling a claims dataset that has already fully developed), this research is mentioned here because it uses several neural networks to simulate the complicated individual claims data, and, more importantly, because the availability of test datasets appears to be an important step in advancing the state of the art in machine learning. Wüthrich (2018b) contains code to simulate the claims data for the neural network chain-ladder model using the Individual Claims Simulation Machine.

DEEPTRIANGLE

A different approach to IBNR reserving appears in Kuo (2018b), with an associated R package (Kuo, 2018a), who applies many of the concepts mentioned in Section 3

to a large triangle dataset provided by Meyers and Shi (2011). This dataset, which is based on Schedule P submissions of many P&C companies in the United States and is available on the Casualty Actuary Society website, contains paid and incurred claims triangles covering the accident years 1988–1997, the development observed on those triangles in the subsequent years 1998–2006, and earned premiums relating to the accident years. For input to the DeepTriangle model, the incremental paid claims and total outstanding loss reserve triangles were normalised by earned premium, so, effectively, the model is fit to incremental paid and total outstanding loss ratios.

The proposed network combines embedding layers, for the company code categorical feature, and recurrent neural networks (namely a GRU, described in Chung et al. (2015), which is similar to an LSTM unit) for the variable length accident year development up to development period j with fully connected layers and is amongst the most complicated of the architectures surveyed in this section.[12] The network is a so-called multi-task variant on the single output networks seen until now, with the main goal of the network to predict unseen incremental paid claims, but with a secondary goal of predicting the outstanding reserve balances at each point. Multi-task networks often generalise better than single task networks, since the shared parameters learned by the model are constrained towards good (i.e. predictive) values (Goodfellow, Bengio & Courville, 2016). The DeepTriangle model was fit to data from 50 companies in four lines of business, namely, Commercial and Private Auto, Workers Compensation and Other Liability. To fit the network, an initial training run was performed using the data of calendar year 1995 as a training set and 1996–1997 as a validation (out-of-sample) set. The number of epochs to produce the minimal validation set error was recorded, and the model was refit to the data of calendar year 1997 in a second training run. Each extra diagonal predicted by the model was added back to the triangle as input into the next prediction of the model, and the process was followed until the lower triangles were derived. These results show that DeepTriangle network outperforms the traditional Mack and GLM formulations of the chain-ladder on all lines, and is outperformed only on a single line, Commercial Auto, by one of the Bayesian models in Meyers (2015).

The comparison of DeepTriangle to the traditional chain-ladder models is somewhat unfair, for two reasons. Firstly, actuaries in practice would not generally apply the chain-ladder model without carefully considering the potential for the loss development factors to change over the accident years, and secondly, DeepTriangle is a multivariate model fit on 50 triangles of two variables each (paid and outstanding claims), whereas the chain-ladder is fit on a single triangle at a time. A more meaningful comparison would be to a multivariate extension of the chain-ladder, whether a traditional model such as Gisler and Wüthrich (2008) or a neural model such as Wüthrich (2018b). Another practical issue to consider is that the model performance probably depends

12 For another network that is similar in principle to this network, see the architecture used by De Brébisson et al. (2015) to predict taxi destinations.

on the volume of data used (i.e. 50 triangles of claims paid and outstanding) and, in practice, only large multi-line insurers would have sufficient data to calibrate this type of network. However, having noted these concerns, the DeepTriangle network and similar variants, which are possible using different combinations of the neural network building blocks discussed in Section 3, represent an opportunity for greater accuracy in and automation of the loss reserving process.

Whereas DeepTriangle works on the level of aggregate claims triangles, a somewhat similar approach applied to individual claims is Zarkadoulas (2017), who studies the use of neural networks to predict individual claims payments, using the time series of reserves and payments made to date on each individual claim as a feature vector fed into a shallow neural network. Separate networks are calibrated to estimate the $j-1$ development periods through which an individual claim develops in a triangle. Since the feature vector relies on the time series of claims development to date, it would seem that the method can only be applied to claims already reported (i.e. to estimate Incurred But Not enough Reported (IBNeR)) and the details of how this is extended to IBNR claims are unfortunately not clear. In any event, the results of the exercise are very similar to those of the chain-ladder that is also applied.

Code to apply the DeepTriangle model is available in Kuo (2018a).

Analysis of Telematics Data

Motor insurance has traditionally been priced using a number of rating factors which have been found to be predictive of the frequency and severity of claims. These factors (i.e. features) may relate to characteristics of the driver, such as age, gender and credit-score (see Golden et al. (2016) who attempt to explain why credit-score is predictive of insurance claims), or the characteristics of the motor vehicle, such as weight and power. More recently, the opportunity more directly to measure driving habits and style by incorporating telematics data into pricing has arisen (Verbelen, Antonio & Claeskens, 2018). These data are often high dimensional and high frequency feeds containing information on location, speed, acceleration and turning, as well as summarised information about the number, time, distance and road type of trips (Wüthrich & Buser, 2018). The latter data are not as challenging to incorporate into actuarial models and one approach using GAMs and compositional data analysis is in Verbelen, Antonio and Claeskens (2018). More challenging are the former data, primarily because there is no immediately obvious way to incorporate high-frequency sequential data of unknown length into traditional pricing models relying on structured (tabular) data.

Within the actuarial literature, Weidner, Transchel and Weidner (2016b) analyse these high frequency data using the Discrete Fourier Transform and provide examples of how this analysis could be used. Weidner, Transchel and Weidner (2016a) simulate driving profiles using a random waypoint model and match profiles from an empirical dataset back to the simulation model. A different approach that we focus on in the next part of this section is described in a series of papers by Wüthrich (2017), Gao

and Wüthrich (2017) and Gao, Meng and Wüthrich (2018) who discuss the analysis of velocity and acceleration information from a telematics data feed. We follow this discussion with an analysis of the data provided by Wüthrich (2018c) using plain and convolutional auto-encoders. In the last part of the section, we discuss the approach taken outside the actuarial literature.

V-A HEATMAPS

Wüthrich (2017) introduced the concept of so-called velocity-acceleration ("v-a") heat-maps. An empirical GPS location dataset, recorded at a one-second frequency, of 1 753 drivers and 200 recorded trips was used to estimate the velocity and acceleration (i.e. change in average speed) of the driver at each time point. The various speed profiles were distributed into groups depending on the speed, for example, low speeds between 5–20 km/h and highway speeds between 80–130 km/h. Two dimensional heat-maps, with the x axis representing velocity in km/h and the y axis representing acceleration in m/s^2 of the density of the empirical observations were constructed (note that a discrete approach to building the heat-map was taken and another possible approach is two dimensional kernel density estimation). An example heat-map generated using the code provided by Wüthrich (2018c) is shown in Figure 12.

An important practical result is in Gao, Meng and Wüthrich (2018), who provide a statistical analysis showing that around 300 minutes of telematics data is generally enough volume to build a stable v-a heat-map in the 5–20 km/h category.

Wüthrich (2017) and Gao and Wüthrich (2017) then apply several unsupervised learning methods to the v-a heat-maps. In Section 2 above, unsupervised learning was defined as the task of finding meaningful patterns within feature vectors X, which can then be used to further understand the data. In this case, the v-a heat-map is the feature vector comprising the empirical density of observations for each velocity and acceleration group. Wüthrich (2017) applied the k-means algorithm (see James et al. (2013) for details) to derive ten discrete clusters into which the 1 753 v-a heat-maps were classified. Gao and Wüthrich (2017) took a different approach and looked to derive low dimensional continuous summaries of the v-a heatmaps using PCA, and shallow and deep auto-encoder neural networks. PCA was applied to a matrix comprising a row for each driver and a column for each combination of velocity and acceleration, and it was found that relatively good summaries of the heatmaps could be formed using only two of the principal components. The shallow auto-encoder (called a bottleneck network in this study) consisted of a layer of seven neurons, followed by two neurons, followed by seven neurons, with hyperbolic tangent activation functions and ending in a soft-max layer. The deep auto-encoder consisted of an extra layer of neurons before and after the two-neuron layer, which was trained using greedy unsupervised learning (Goodfellow, Bengio & Courville, 2016: Chapter 15) (see Section 3 for more details). Both neural networks were trained using the Kullback-Leibler (K-L) loss function, which, in summary, measures how far apart two distributions are, and amounts to maximising the likelihood of a multinomial distribution on the data. On all the metrics

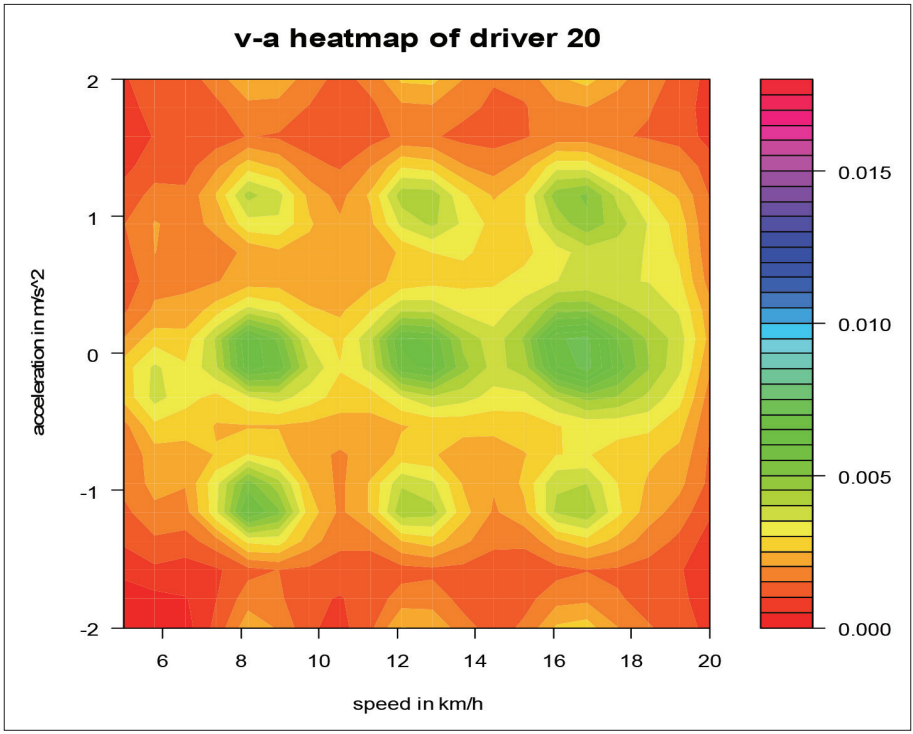

FIGURE 12 v-a heatmap generated using the code provided by Wüthrich (2018c). The x-axis is speed, the y-axis is acceleration and the coloured scale shows the empirical density at each point

considered, the deep auto-encoder network outperformed the shallow auto-encoder, which in turn outperformed the PCA.

Finally, Gao, Meng and Wüthrich (2018) use the features extracted from the v-a heatmaps in a GAM of claim counts. The dataset considered in this study consists of 1 478 divers with 3 332 years of exposure and a relatively high claims frequency of 23.56%. All four methods described above for extracting features from the v-a heatmaps were applied to this new dataset. When applying the k-means algorithm, the number of clusters (i.e. means) were chosen using tenfold cross-validation (see Friedman, Hastie & Tibshirani, 2009: Chapter 7 for more details) to estimate the out-of-sample Poisson deviance. The continuous (PCA and auto-encoder) features were included directly in the GAM. In a stunning result (recall that the features were extracted in an unsupervised manner), the continuous features are not only highly predictive, but, by themselves, produce a better out-of-sample deviance than the models including the traditional features of driver's age and car's age.

While Gao, Meng and Wüthrich (2018) do not attempt to explain why the features

estimated using unsupervised methods are highly predictive in a supervised context, some understanding is possible based on Goodfellow, Bengio and Courville (2016: Chapter 15), who are concerned, in Section 15.1, primarily with greedy unsupervised pre-training, and explain that "the basic idea is that some features that are useful for the unsupervised task may also be useful for the supervised learning task" i.e. the unsupervised representation that is learned may capture information about the problem that the supervised learning algorithm can access.

ANALYSIS USING KERAS

We show in this section how the v-a heatmaps can be analysed using auto-encoders in Keras. For input data, 10 000 heatmaps were generated using the simulation machine of Wüthrich (2018c). Since in any event the data are summarised in the parameters of the simulation machine, we do not seek to add much insight in understanding these data, but rather to understand how auto-encoders can be applied.

Two different types of auto-encoder were applied – a basic fully connected auto-encoder, as well as a convolutional auto-encoder. The convolutional auto-encoder uses several layers of learned convolutional filters to derive the encoded heatmap, instead of fully connected layers, and rebuilds the output from the encoded heatmap using transposed convolutions, which are effectively the reverse operation. Both types of auto-encoders were applied with hyperbolic tangent activations. To train these, a process of greedy unsupervised learning was followed. First, the outer layers of each auto-encoder were trained. Then, the weights derived in the first step were held constant (i.e. frozen) and a new layer was added and trained. This process was followed until the dimensionality of the heatmap had been reduced to a 1×2 vector of codes. The fully connected auto-encoder used two hidden layers between the input and the encoding, whereas the convolutional auto-encoder used three layers of filters. The encoded heatmaps are shown in Figures 13 and 14, where the two-dimensional encoded values have been grouped according the quantiles of the encoding distribution. Whereas the codes produced using the fully connected auto-encoder are relatively highly correlated, with a correlation coefficient of 0.85, the codes produced by the convolutional auto-encoder have a much lower correlation of 0.29, implying that the factors of variation describing the heatmap have been better disentangled by the convolutional auto-encoder. The mean heatmaps in each quantile group are shown in Figures 15 and 16. The interpretation of the learned codes is easier in the latter figure – moving from right to left in the encoded distribution shifts density to a higher velocity, and from bottom to top shifts probability density from a wide area to several key points – whereas in the former figure, it is harder to assign a definite interpretation. We conclude from this exercise that convolutional auto-encoders are a promising technique that should be explored further in the context of telematics data. In particular, it would be interesting to investigate the results of including the codes from the convolutional model in a claims frequency model, as done with the codes from a fully connected auto-encoder in the study of Gao, Meng and Wüthrich (2018).

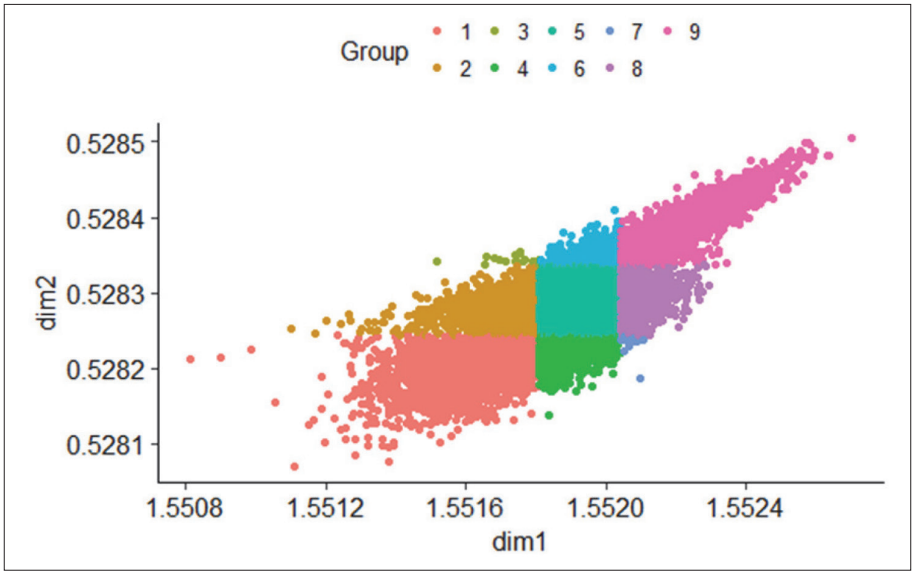

FIGURE 13 Heatmaps encoded using a fully connected autoencoder and grouped by quantile of the encoded distribution

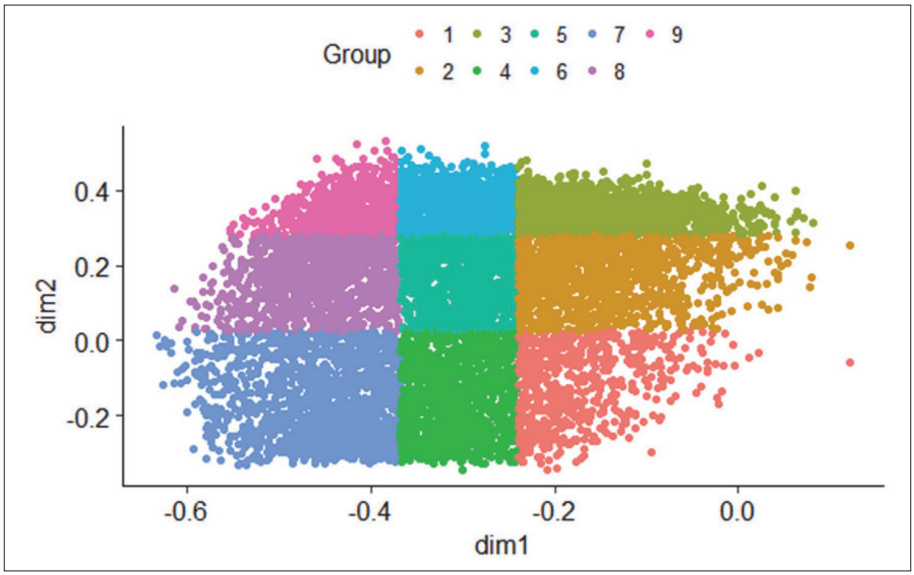

FIGURE 14 Heatmaps encoded using a convolutional autoencoder and grouped by quantile of the encoded distribution

OTHER SOURCES

To summarise the approach discussed to this point, the high-dimensional and high frequency telematics data is first summarised in a v-a heat-map, which, compared to classical features in motor pricing, is itself a relatively high-dimensional feature. The v-a heat-map can be used as a feature in GLM rating models once the data has been summarised using dimensionality reduction techniques. During the (double) summarisation of the data, potentially some information is lost, and, therefore, with enough data, one might expect the application of RNNs to the GPS data, either raw or less summarised than the v-a heat-maps, to outperform approaches that rely on summarised data. Therefore, for completeness, in this section we discuss several recent applications of neural networks to GPS or telematics data from outside the actuarial literature.

De Brébisson et al. (2015) discuss several deep networks applied to the problem of predicting taxi destinations based on variable length GPS data, as well as meta-data describing driver, client and trip features, covering 1.7 million trip records of taxis

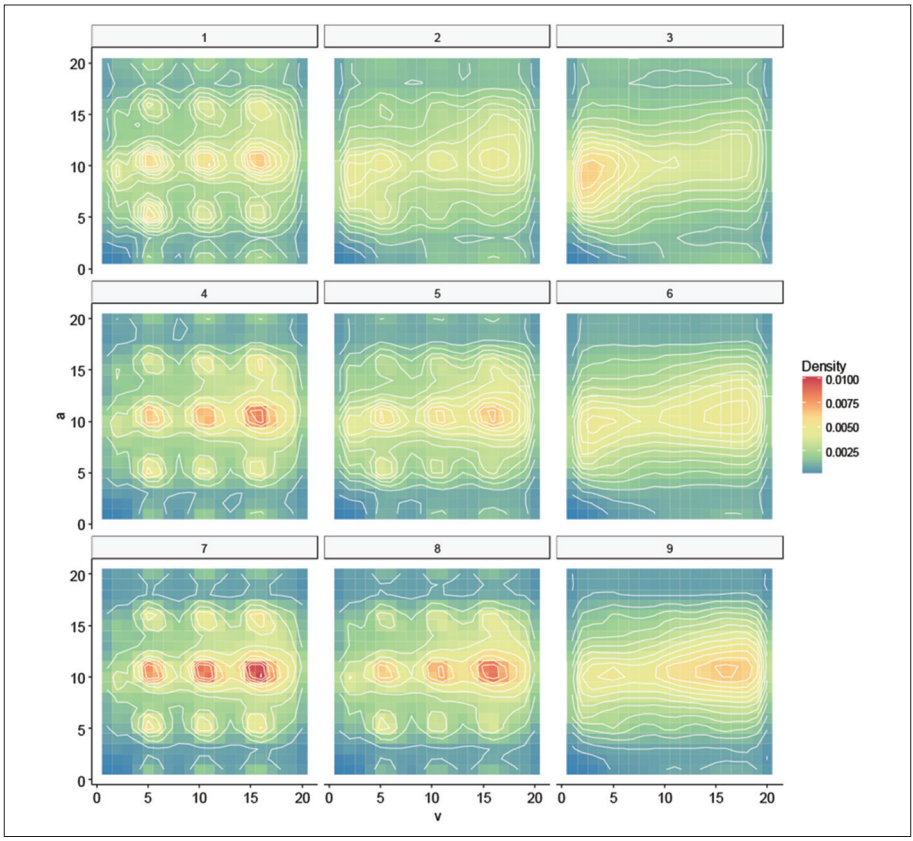

FIGURE 15 Mean heatmap for each group shown in Figure 13

in Portugal. These meta-data were modelled using embedding layers, while several approaches were taken to model the GPS data directly. The simplest approach (which won the competition) was to use a fixed amount of the GPS data as an input directly to a feed-forward network. More complicated models recognised that the variable length GPS data could be modelled using RNNs, and, therefore, RNN and LSTM networks were calibrated on the raw GPS data, with the output of these networks fed into the same network structure just mentioned (i.e. including embedding layers for the meta-data). A variant of these was the highest ranked model within the authors' test set.

Dong et al. (2016) find that deep learning applied directly to GPS data for the task of characterising driving styles does not perform well (whereas in De Brébisson et al. (2015) the goal is to predict the destination i.e. the output is in the same domain as the input data). Thus, they rely on a similar data summary approach to Wüthrich (2017). First, the raw GPS data is segmented into longer trip windows, and shorter measurement windows. Within each measurement window, speed and acceleration,

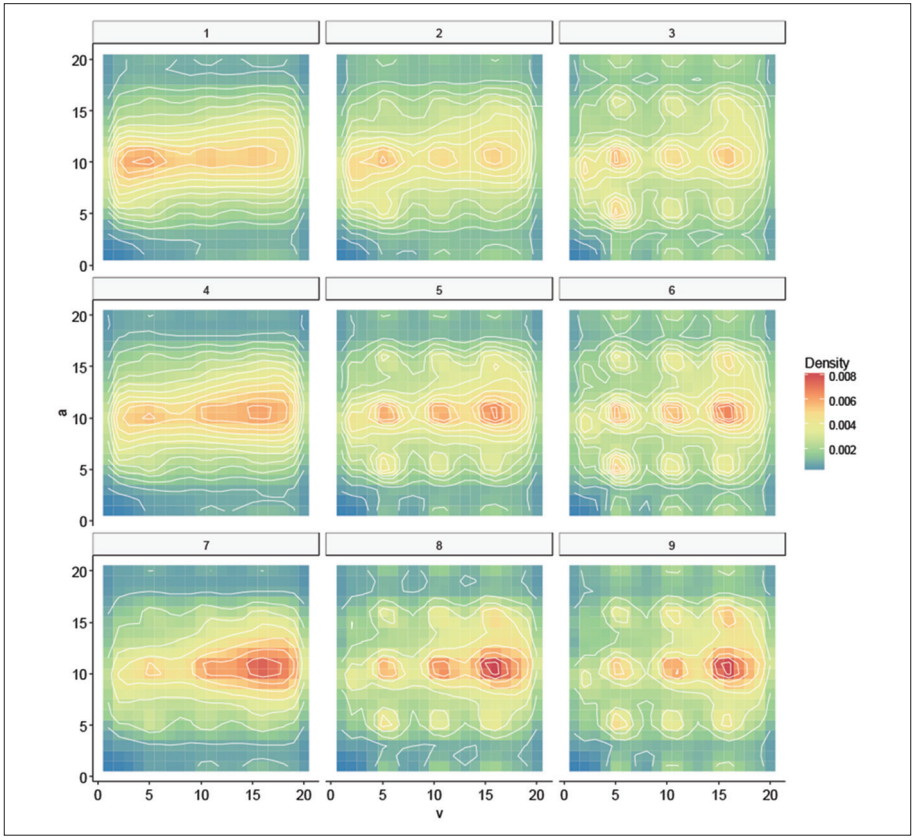

FIGURE 16 Mean heatmap for each group shown in Figure 14

and the change in each of these components over time, and angular momentum is calculated, and then, for each of these basic features, statistics (the mean, standard deviation, minimum, maximum, median, and 25th and 75th percentiles) are calculated, leading to a matrix (or feature-map) of basic movement statistics for each trip window, where each row represents a movement statistic, and each column represents time (i.e. a new observation window). Since driver behaviour manifests itself over time, one would expect that the movement statistics would have some sort of temporal structure, which is analysed using either CNNs or RNNs, with a detailed description of the application of both methods in the paper. Notably cross-correlation between features is not allowed for in either of these models, and, given the performance of the v-a heat-maps discussed above, it would seem that this is an omission. The networks are trained on a dataset consisting of driver trips with the goal of classifying the driver correctly. A variant of the RNN network outperformed all other approaches, including a machine-learning baseline. Investigating the learned features, the study concludes that the network has automatically identified driving behaviour, such as slowing down for sharp turns, or high speed driving on straight roads.

Dong et al. (2017) extend the model of Dong et al. (2016) in several ways, in a new network they call ARNet. Their network design, applied to the same movement statistic feature matrix, relies exclusively on RNNs (specifically, the GRU of Chung et al. (2015)) for driver classification, in a similar setup to Dong et al. (2016), but also includes an auto-encoder branch within the network that aims to reconstruct the (hidden) layer of the network that represents driving style. The idea of adding the auto-encoder to the classification model is explained as follows: the auto-encoder is regularised (i.e. the auto-encoder representation is shrunk towards zero), which should improve the generalisation of the classification branch of the network to unseen data, and the classification branch includes prior information about the driver within the network structure, which should improve the auto-encoder representation of driving style. ARNet outperforms on the two tasks tested in the research – driver classification and driver number estimation, compared to both the shallow and deep machine learning baselines also tested in that research.

Wijnands et al. (2018) use an LSTM network to predict whether a driver's behaviour has changed, based on sequences of telematics data, containing, for each sequence, counts of acceleration, deceleration and speeding events. An LSTM is trained for each driver to classify these sequences of events as belonging to one of three benchmarks, one for safe driving, one for unsafe driving and one for the driver under consideration, where the benchmarks are themselves based on an insurer's proprietary scoring algorithm. If driver behaviour changes, then the LSTM classification is shown to be able to classify the driver's behaviour as arising from the benchmarks, and not as similar to the driver's own previous driving style.

We refer the interested reader to the review in Ezzini, Berrada and Ghogho (2018) for more references of applying machine learning and statistical techniques to these types of data.

Mortality Forecasting

Mortality rates are a fundamental input into actuarial calculations involving the valuation and pricing of life insurance products. Mortality improvement rates are used by actuaries when modelling annuity products and the results of these models are often highly sensitive to this input, while the stochastic variation of the forecasts around the mean is used for regulatory and economic capital modelling. Various methods are used for deriving mortality rate forecasts, with the Lee and Carter (1992) and Cairns, Blake and Dowd (2006) models often used as standard reference points in the actuarial literature. For a benchmark in this section, we focus on the Lee-Carter model, which models log mortality rates (the force of mortality) as a set of baseline average (log) mortality rates for a period which vary non-linearly through time at a rate of change determined for each age multiplied by a time index, as follows:

$$\ln\left(\mu_{x,t}\right) = a_x + \kappa_t . b_x$$

where $\mu_{x,t}$ is the force of mortality at age x in year t, a_x is average log mortality rate during the period at age x, κ_t is the time index in year in year t and b_x is the rate of change of log mortality with respect to the time index at age x. Often, mortality models are first fit to historical mortality data and the coefficients (in the case of the Lee-Carter model, the vector κ) are then forecast using a time series model, in a second step. Many popular mortality forecasting models can be fit using GLMs and GNMs[13] (Currie, 2016) and an R package, StMoMo, automates the model fit and forecasting process (Villegas, Kaishev & Millossovich, 2015).

Hainaut (2018a) is a recent study that uses auto-encoder networks (see Section 3 above) to forecast mortality (these are referred to as "neural network analyzers" in the paper). Mortality rates in France in the period 1946–2014 are used, with the training set being the rates in the period 1946–2000 and the test set covering 2001–2014. The base-line models against which the neural model is compared are the basic Lee-Carter model, fit using Singular Value Decomposition, the Lee-Carter model fit with a GNM and lastly, an enhanced Lee-Carter model with cohort effects, again fit with a GNM.

For the neural model, a series of shallow auto-encoders (similar to those shown in Figure 4) with different numbers of neurons in the hidden layers, using the hyperbolic tangent activation function, are fit. In this study, auto-encoders are viewed through the lens of non-linear PCA, and, for more details on this connection, see Efron and Hastie (2016: Section 18). Before fitting the networks, the mortality rates are standardised by subtracting, for each log mortality rate $\ln\left(\mu_{x,t}\right)$, the average mortality over the period, a_x, and, therefore the aim of the model is to replace the simple linear time-varying

13 Note that the Lee-Carter model cannot be fit with a GLM due to the multiplicative nature of the term $\kappa_t . b_x$, which is comprised of two variables that must each be estimated from the data. In the GLM formulation in R, an interaction term between the variables Year and Age could be fit, but, for this term of the model, this specification would require $t.x$ effects to be fit compared to the $t + x$ effects in the Lee-Carter model.

component of the Lee-Carter model, $\kappa_t.b_x$, with a non-linear time-varying function learned using an auto-encoder. Unlike most modern applications of neural networks, in this study the calibration is performed using genetic evolution algorithms, instead of back-propagation, which is justified in the study since the input to the network is high-dimensional. Another interesting aspect of this network is that the network is not fully connected to the inputs, but rather the neurons in the first hidden layers are assigned exclusively to a set of mortality rates, say those at ages 0–4 for the first neuron, those at ages 5–9 for the second and so on (i.e. the neurons in the first layer are only locally connected). This exploits the fact that mortality rates at nearby ages are similar, and should lead to a more easily trained network. The encoded mortality is then forecast using a random walk model with drift. The study provides several examples that show that the predictive power of the mortality forecasts based on neural models is as good as or better than the best performing of the Lee-Carter models.

ANALYSIS USING KERAS

In this section, we attempt to produce similar results with Keras, noting that the relatively complicated genetic evolution optimisation scheme of Hainaut (2018a) is not supported within the Keras package, but only back-propagation and associated optimisers are supported. We also note that the network in the previous example benefits from substantial manual feature engineering, in that the network is fit only to the time varying component of mortality, and, furthermore, the first hidden layer is only locally connected. The paradigm of representation learning would, however, seem to indicate that the network should be left to figure out these features by itself, and perhaps arrive at a more optimal solution in the process.

Therefore, a first attempt at the problem of mortality forecasting applied greedy unsupervised learning to train fully connected auto-encoders on mortality data (the central rate of mortality, m_x) from England and Wales in the period 1950–2016, covering the ages 0–99, sourced from the Human Mortality Database (HMD) (Wilmoth & Shkolnikov, 2010). The training dataset was taken as mortality in the period 1950–1999 and the test dataset was in the period 2000–2016, and the logarithm of m_x was scaled so as to lie in the interval [0,1]. The neural networks were fit exclusively on this scaled dataset and were compared to a baseline Lee-Carter model fit directly to the raw m_x using the gnm package (Turner & Firth, 2007), and forecast using exponential smoothing as implemented in the forecast package (Hyndman et al., 2015).

The auto-encoders used hyperbolic tangent activations, and each layer was fit for 50 000 epochs using the Adam optimiser (Kingma & Ba, 2014) implemented in Keras, with a learning rate schedule hand designed to minimise the training error. The encoded mortality curves were forecast using a random walk with drift to produce forecasts for each of the years in the period 2000–2016. These results, referred to as *Auto-encoder*, as well as the Lee-Carter baseline are shown in Table 6. The auto-encoder forecasts outperform the baseline Lee-Carter model, showing that a viable auto-encoder model can be fit in Keras without too much manual feature engineering.

Table 6 Mortality forecasting – Out-of-sample MSE

Model	MSE – Out-of-sample
Lee-Carter	0.2624
Auto-encoder	0.1170
Deep_reg	0.0814
Deep_reg_hmd	0.1025

However, it is important to note that fitting the auto-encoders is difficult and computationally expensive, and the results produced on the out-of-sample data can be variable, with worse performance than reported in the table possible. One way of reducing the variability is to average the results of several deep models (Guo & Berkhahn, 2016), but a comparison of the resulting ensemble model to the Lee-Carter model would be unfair. Therefore, we also demonstrate a different approach using deep learning for mortality forecasting and, in the following, show how the Lee-Carter model could be fit and extended using embedding layers.

The Lee-Carter model can be expressed in functional form as

$$\ln\left(\mu_{x,t}\right) = f\left(x,t\right) = a_x + \kappa_t . b_x$$

$$a_x = g\left(x\right)\begin{cases} a_1, & x = 1 \\ a_2, & x = 2 \\ & \cdots \\ a_n, & x = n \end{cases}$$

up to a maximum age n, and similarly for κ_t and b_x. Rather than specify this particular functional form, a neural network can be used to learn the function $f(x,t)$ directly, by using age and calendar year as predictors in a neural network that is then trained to predict mortality rates. This network was fit on the same dataset as the auto-encoders, and consisted of an embedding layer for age and two hidden layers of 32 neurons with the ReLu activation. The year variable was left as a numerical input to the network, and is used to forecast future mortality rates. This network is referred to as *Deep_reg* in Table 6.

A similar network was fit to the entire HMD dataset at once, with an embedding for the Country to which the mortality rates relate. This network is referred to as *Deep_reg_hmd* in Table 6. It can be seen that of all the networks tested, *Deep_reg* outperforms the others, followed closely by *Deep_reg_hmd.* However, as discussed next, on the long-term forecast of rates in 2016 (forecast using data up to 1999), the *Deep_reg_hmd* network outperforms the other networks, followed closely by the auto-encoder.

Figures 17 and 18 show the forecasts of mortality in 2000 (i.e. a one-year horizon) and 2016 (i.e. a 16-year horizon) produced using the models described in this section.

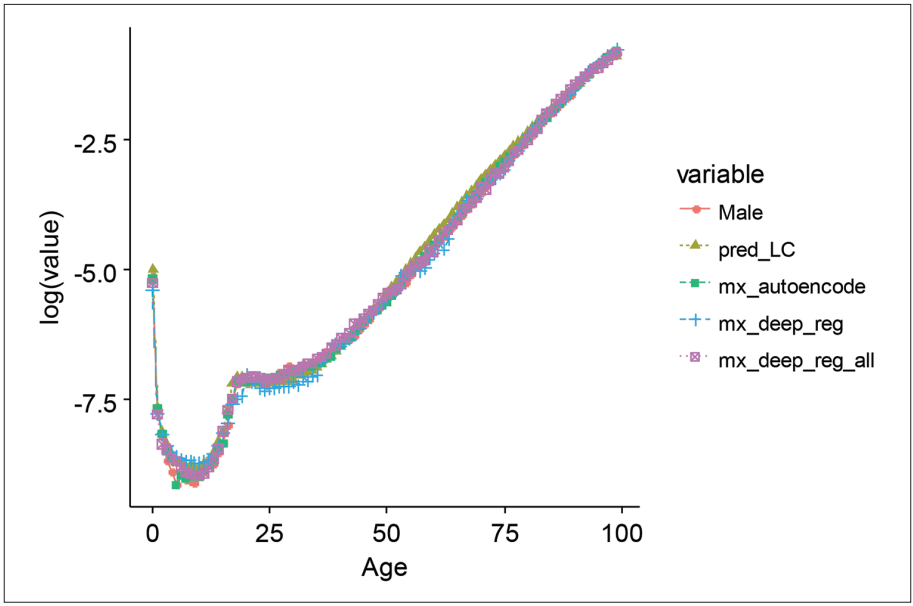

FIGURE 17 Forecasts of mortality in 2000 using the models described in this section, log scale

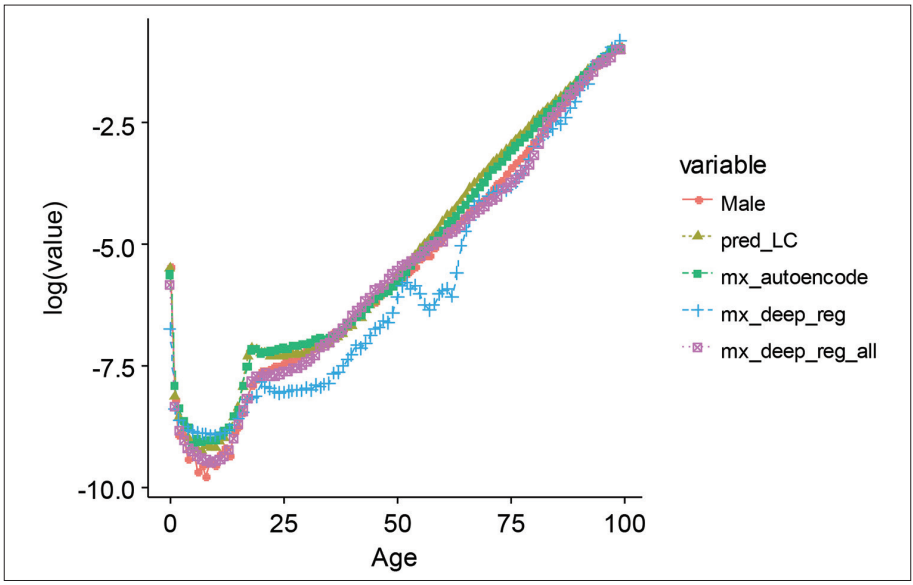

FIGURE 18 Forecasts of mortality in 2016 using the models described in this section, log scale

All of the models are quite close to actual mortality in 2000, but some of the forecasts diverge compared to actual mortality in 2016. In 2016, the *Deep_reg_hmd* forecasts follow the actual mortality curve closely at almost all ages, while the Lee-Carter and auto-encoder forecasts appear too high in middle age. On closer inspection, the *Deep_reg* forecasts in 2000 are quite variable at some ages, and by 2016 have degraded over time and do not appear demographically reasonable, whereas the *Deep_reg_hmd* forecasts have remained demographically reasonable.

The learned embedding for age from the *Deep_reg_hmd* network is shown in Figure 19. The dimensionality of the embedding was reduced to two dimensions using PCA. The first dimension is immediately recognisable as the basic shape of a modern lifetable, which is effectively the function a_x fit by the Lee-Carter model. The second dimension appears to describe the relationship between early childhood, late middle age and old-age mortality, with old-age mortality steepening as early childhood and late middle age mortality declines, and vice-versa.

We conclude from this brief study of mortality that deep neural networks appear to be a promising approach for modelling and forecasting population mortality.

Approximating Nested Stochastic Simulations with Neural Networks

Risk management of life insurance products with guarantees often requires nested stochastic simulations to determine the risk sensitivities (Greeks) and required capital for a portfolio. The first stage of these simulations involves a real-world (i.e. P-measure)

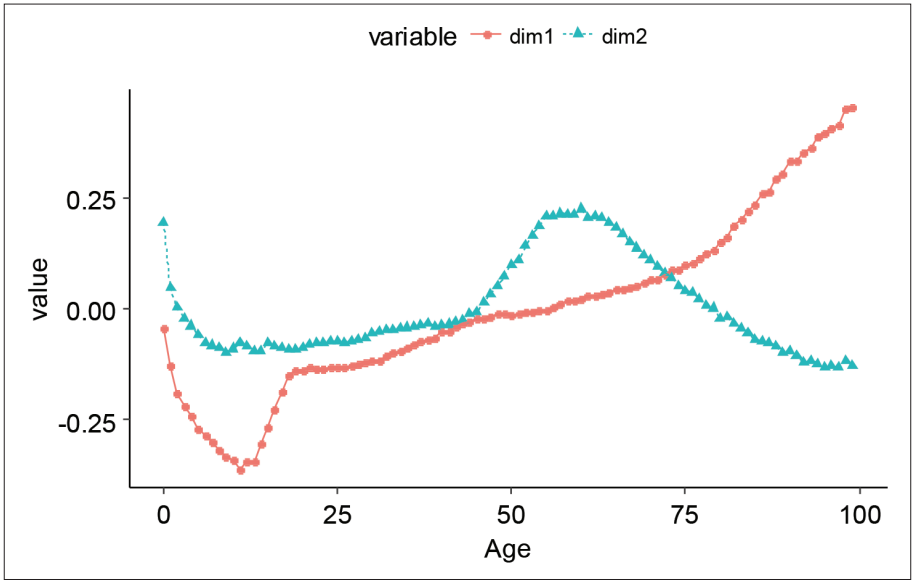

FIGURE 19 Age embedding from the Deep_reg_hmd model, with dimensionality reduced using PCA

simulation of the risk factors in a portfolio, such as the evolution of market factors, mortality and policyholder behaviour. Once the real-world scenario has been run, a risk neutral (i.e. Q-measure) valuation of the contracts is performed, using the real-world baseline established in the first step as an input into the risk neutral valuation. This so-called "inner" step ideally consists of Monte Carlo simulations using a market consistent risk-neutral valuation model. Once the nested simulations have been performed, the risk sensitivities of the portfolio can be calculated and dynamically hedged to reduce the risk of the portfolio, and risk capital can be estimated by calculating risk measures such as value at risk or expected shortfall. A major disadvantage of the nested simulation approach is the computational complexity of the calculations, which may make it impractical for the sometimes intra-day valuations required for the dynamic hedging of life insurance guarantees. As a result, approximation methods such as Least Squares Monte Carlo, replicating portfolios or machine learning methods may be applied to make the calculations more practical. For more detail and an overview of these methods to approximate the inner simulations, the reader is referred to Gan and Lin (2015) and the references therein.

A recent approach uses neural networks to approximate the inner simulations for Variable Annuity[14] (VA) risk management and capital calculations (Hejazi & Jackson, 2016; 2017). In this section we focus on the approximation of the Greeks described in Hejazi and Jackson (2016), since the approximation of the capital requirements in Hejazi and Jackson (2017) follows almost the same process. Instead of running inner simulations for the entire portfolio of the contracts, in this approach, full simulations are performed for only a limited sample of contracts, against which the rest of the contracts in the portfolio are compared using a neural network adaptation of kernel regression, which we describe next based on Hazelton (2014).

Kernel regression is a nonparametric statistical technique that estimates outcomes y with an associated feature vector X using a weighted average of values that are already known, as follows:

$$\hat{y} = \frac{\sum_{i=1}^{N} w_i y_i}{\sum_{i=1}^{N} w_i}$$

where $\hat{y}$ is the estimated value, y_i is a known outcome of example, or point, i and w_i is the kernel function calculated for point i. w_i is often chosen to be the Nadaraya-Watson kernel, which is defined as

$$w_i = \frac{1}{h} K\left(\frac{X - X_i}{h}\right)$$

14 Variable annuities are a North American product similar to unit-linked life insurance with guarantees, which may cover minimum withdrawal and death benefits, amongst others.

where K is a unimodal symmetrical probability distribution, such as a normal distribution, X_i is the feature vector associated with point i and h is the bandwidth, which determines how much weight to give to points which are distant from X. For the easy application of basic kernel regression, we refer the reader to the NP package in R (Racine & Hayfield, 2018).

The insight of Hejazi and Jackson (2016; 2017) is that kernel regression is unlikely to produce an adequate estimate for complex VA products, and therefore they calibrate a kernel function based on a one-layer neural network that is trained to measure the similarity of input VA contracts with those in the training set, as follows:

$$\hat{y} = \frac{\sum_{i=1}^{N} G_{h_i} y_i}{\sum_{i=1}^{N} G_{h_i}}$$

where G_{h_i} is a function calibrated using a neural network that measures how close contract i with feature vector X_i is to the contract with feature vector X. The neural network is set up by choosing a sample of N representative VA contracts, and an input contract against which the representative contracts are compared. For each representative contract, a feature vector describing how similar the input contract is to the representative contract i is calculated (for example, if the contracts share a categorical feature, such as guarantee type, then this component of the vector is set to one, otherwise it is set to zero, and, for quantitative features, such as account value, the relevant component of the feature vector is set to the normalised difference of the features) and the network's output is a vector of weights calibrated using a soft-max layer that describe how similar the input and representative contracts are. For both the input and representative contracts, the Greeks/risk neutral value have already been calculated, and the network is trained to minimise the difference between the Greeks/risk value neutral of the input contract and the weighted average of the Greeks/risk neutral value of the representative contracts, using the vector of weights described in the previous step. The vector of weights required for the adapted kernel regression can be calculated quickly using the trained network, thus dramatically reducing the calculation time for each contract.

Hejazi and Jackson (2016) report the out-performance of the neural network approach compared to traditional statistical approached and Hejazi and Jackson (2017) show that the network estimates the value of the liabilities and solvency capital requirements accurately.

Forecasting Financial Markets

Smith, Beyers and De Villiers (2016) attempt to apply neural networks to the challenging task of forecasting financial markets using lagged data, based on their novel approach to parameterising and training neural networks. In short, instead of

applying the (now) standard back-propagation algorithm (see Section 3 above) they apply the Resilient Backpropagation (Rprop) algorithm (Riedmiller & Braun, 1993) within a systematic framework for choosing the design and hyper-parameters of the neural networks, and use simple time-series models as benchmarks for comparing the performance of the neural networks on South African financial market data. One interesting contribution is a so-called "hybrid model" that first applies time series models and then fits neural networks to the residuals, which amounts to an attempt to apply boosting (see Friedman, Hastie & Tibshirani (2009) for more discussion) in the time-series context. Unfortunately, on this difficult task, the much simpler time-series models perform as well or better than the neural networks.

Similar results of the relatively poor performance of basic machine learning methods, compared to simple time-series models, when applied to time-series forecasting are in Makridakis, Spiliotis and Assimakopoulos (2018b). A counterpoint, though, is the impressive performance of a hybrid exponential smoothing-LSTM network model in the M4 competition (Makridakis, Spiliotis & Assimakopoulos, 2018a), as well as the strong performance of boosting approaches in that competition.

5. DISCUSSION, OUTLOOK AND CONCLUSION

Discussion

Section 4 of this paper has presented recent examples of the application of neural networks in actuarial science. In this section, we seek to distil some of the major themes of these applications, and place them within the wider context of deep learning.

In most of these examples, there is an emphasis on predictive performance and the potential gains of moving from traditional actuarial and statistical methods to machine and deep learning approaches. This is enabled by the measurement framework utilised within machine learning – models are fit on one dataset (the training dataset) and then measured, more realistically, on unseen data (the test dataset). The metrics used for these measurements, such as mean squared error, are generally familiar to actuaries but the focus on measuring predictive performance is perhaps less well considered in the actuarial literature. The focus on predictive performance does not necessarily come at a great cost to understanding the models, and, as shown above, the learned representations from deep neural networks often have a readily interpretable meaning, which is often not the case for shallow machine learning techniques. In the wider deep learning context, the focus on measurable improvements in predictive performance has led to many refinements and enhancements of basic deep learning architectures and this has propelled forward the progress of deep learning research.

Notably, in most of the examples considered above, the deep learning and neural network methods outperformed other statistical and machine learning approaches, thus, to the extent that predictive performance is the key goal for an actuary, it makes sense to consider deep neural networks when choosing a model. In light of this, the earlier assertion of this paper that actuaries should pay attention to deep learning appears to be bolstered by these emerging, successful applications within actuarial

science. Of course, the cost of applying deep learning is that these models are often harder to fit than simpler shallow models and one is often required to resort to technically complicated approaches such as greedy unsupervised learning or choosing a particular optimiser.

A further observation is that some of these examples focus on datasets that are more granular and descriptive than traditional datasets analysed by actuaries. The advantage of these datasets is the potential for greater accuracy in pricing and reserving, and an enhanced understanding of the drivers of business performance. The disadvantage, from the standpoint of traditional actuarial methods, is the increased complexity of the methods and their application.

The public availability of large, real-world datasets for research and benchmarking has been important in the recent progress in deep learning, and, thus, the recent attempts within the actuarial community to provide granular claim (Gabrielli & Wüthrich, 2018), IBNR triangle (Meyers & Shi, 2011) and v-a heatmap (Wüthrich, 2018c) datasets should be recognised and applauded by the actuarial community of practitioners and researchers. As a case in point in the context of telematics data, the work of Dong et al. (2016) and Dong et al. (2017) appears to have been supported by the availability of a large anonymised dataset as part of a Kaggle competition, but, unfortunately at the time of writing, this dataset has been removed from the Kaggle website.

Some of the examples effectively combine deep learning together with traditional statistical models – the IBNR reserving study Wüthrich (2018b) shows how to incorporate a neural network into the traditional chain-ladder algorithm, the non-life pricing example of Gao, Meng and Wüthrich (2018) shows how the outputs of unsupervised learning can be incorporated into a GLM model and Hejazi and Jackson (2016; 2017) show how neural networks can be adapted to enhance kernel regression. The combination of deep learning with statistical methods has shown promising results in the area of time series forecasting (Makridakis, Spiliotis & Assimakopoulos, 2018a) and it seems likely that more hybrid applications of neural networks will emerge in the near future.

A theme running throughout the deep learning literature is that neural networks can be designed to efficiently process and learn from different types of data. Some of the techniques reviewed in Section 4 are relatively straightforward applications of basic designs such as recurrent neural networks, auto-encoders and embedding layers, but new data types require some ingenuity to ensure that the neural networks can be trained, for example the v-a heatmaps of Wüthrich (2017) and the movement domain feature matrices of Dong et al. (2016). As new data sources become available to actuaries, the deep learning architectures described in this paper can be considered, but some element of new design might also be necessary. Furthermore, the application of deep learning to actuarial problems can be assisted with the application of, for example, the heuristic presented in Section 2, but novel approaches, such as unsupervised learning, should not be ignored.

Outlook

The examples of Section 4 show that deep learning is a set of modern machine learning techniques that can enhance the predictive power of models built by actuaries, and provide the means potentially to extend actuarial modelling to new types of data. In fact, moving from traditional methods to those based on deep learning does not require much effort, beyond understanding the basic principles of statistical and machine learning, neural networks and modern machine learning software (where effort is required, though, is in fitting the models and interpreting them). The application of deep learning techniques to actuarial problems seems to be a rapidly emerging field within actuarial science, and it appears reasonable to predict that more advances will occur in the near-term. Furthermore, the comparatively large element of expert judgement involved in designing and fitting deep neural networks fits in well with the frameworks developed for applying controlled expert judgement within the actuarial profession, such as peer review and technical standards. This emerging field of deep learning is, therefore, an opportunity for actuaries to become experts in the application of AI within actuarial science, and for actuarial associations to lead their members with guidance on the application of machine and deep learning techniques.

However, the challenge of non-actuary experts moving into the domain of actuaries should not be ignored, and the examples of the advanced telematics models from outside the actuarial literature discussed in Section 4 should provide a warning that actuaries could become less relevant as subject matter experts within insurance if modern techniques, such as those discussed in this research, are not soon incorporated into the actuarial toolkit.

Among the examples surveyed in Section 4, the high-frequency and high-dimensional telematics data are perhaps the most foreign to actuaries trained in analysing structured data. It could be expected that these types of high-frequency data will become more common and applicable in a number of types of insurance, for example, data from wearable devices, autonomous vehicles or connected homes will likely become important in actuarial work in the future. Already, high-frequency location and other data are available for ships and could be incorporated into pricing models for marine insurance, and similar considerations apply for aviation insurance. Therefore, although the analysis of telematics data from human-driven cars might potentially be somewhat of an evolutionary dead-end within actuarial science if autonomous vehicles replace human-controlled vehicles, nonetheless the methods developed for these data are likely to be useful in other contexts.

Public availability of benchmark datasets and models would encourage more applications of machine and deep learning within actuarial science, and we believe that more actuaries should work to make these available to the actuarial community and that actuarial bodies should communicate these initiatives to their membership. Tutorials, such as Noll, Salzmann and Wüthrich (2018), that benchmark machine and deep learning against traditional actuarial methods contribute greatly to the body of knowledge accessible to the actuarial profession.

Having promoted deep learning in actuarial science, nonetheless, it should be recognised that deep learning is not a panacea for all modelling issues. Applied to the wrong domain, deep learning will not produce better or more useful results than other techniques, and, in this regard, the example of Smith, Beyers and De Villiers (2016) who attempted to predict the South African market with neural networks should be taken as a cautionary tale. Neural networks can be challenging to interpret and explain, but techniques to allow for interpretability, such as showcased in Dong, Yuan, Yang et al. (2017) and implemented in Chollet (2017) and Chollet and Allaire (2018), are increasingly successful at explaining what a neural network has learned. Lastly, deep networks can be challenging to fit and often one needs to resort to complicated technical approaches to achieve optimal results.

As with any technique, whether traditional or based on machine learning, actuaries should apply their professional judgement to consider if the results derived from deep neural networks are fit for purpose and in the public interest.

Conclusion

This research has presented the major ideas of machine and deep learning within the context of actuarial science, and provided examples of applications of deep neural networks to practical problems faced by actuaries in everyday practice. The code examples provided on GitHub accompanying this paper, together with the code in many of the papers cited above, should allow the interested reader to apply deep learning models of their own to traditional actuarial problems.

Several avenues of future research could be pursued. A clear set of benchmark models for actuarial problems could be established, thus making comparison of methods easier and more concrete. The predictive performance of specific neural network architectures on actuarial problems should be investigated in detail and the connections to traditional actuarial methods, such as credibility, could be made. Lastly, the professional implications of these techniques within the context of local regulatory environments should be considered.

ACKNOWLEDGEMENTS

The author wishes to acknowledge the generous contribution into the public domain of code by Noll, Salzmann and Wüthrich (2018) and of previously unavailable data by Wüthrich (2018c). The author wishes to thank Mario Wüthrich for discussion of and insight into several of the papers discussed in Section 4, as well as for helpful comments which improved the quality of this paper.

The neural network diagrams appearing in the paper were created using the NN-SVG software (Lenail, 2018). Data analysis was performed in R (R Core Team, 2018) using the high performance data.table package (Dowle & Srinivasan, 2018), models were fit using Keras (Allaire & Chollet, 2018; Chollet, 2015) with the TensorFlow backend (Abadi et al., 2016) and plots were created using ggplot2 (Wickham, 2016) with themes from the ggpubr package (Kassambara, 2018).

REFERENCES

Abadi, M, P Barham, J Chen, Z Chen et al. (2016). "TensorFlow: A System for Large-Scale Machine Learning," Paper presented at OSDI. Vol. 16:265–283

Albright, J, J Schneider and C Nyce (2017). *The Chaotic Middle*. KPMG. https://assets.kpmg.com/content/dam/kpmg/us/pdf/2017/06/chaotic-middle-autonomous-vehicle-paper.pdf. Accessed: 17 June 2018

Allaire, J and F Chollet (2018). *R interface to Keras*. RStudio, Google. https://cloud.r-project.org/web/packages/keras/index.html

Bengio, Y (2009). "Learning deep architectures for AI", *Foundations and trends® in Machine Learning* 2(1):1–127

Bengio, Y, A Courville and P Vincent (2013). "Representation learning: A review and new perspectives", *IEEE transactions on pattern analysis and machine intelligence* 35(8):1798–1828

Bengio, Y, R Ducharme, P Vincent and C Jauvin (2003). "A neural probabilistic language model", *Journal of Machine Learning Research* 3(Feb):1137–1155

Bengio, Y and Y LeCun (2007). "Scaling learning algorithms towards AI," in Bottou, L, O Chapelle, D DeCoste and J Weston (eds). *Large-Scale Kernel Machines*. MIT Press

Boonen, T (2017). "Solvency II solvency capital requirement for life insurance companies based on expected shortfall", *European Actuarial Journal* 7(2):405–434

Bornhuetter, R and R Ferguson (1973). "The Actuary and IBNR", *Proceedings of the Casualty Actuary Society* Volume LX, Numbers 113 & 114

Boucher, J-P, S Côté and M Guillen (2017). "Exposure as Duration and Distance in Telematics Motor Insurance Using Generalized Additive Models", *Risks* 5(4):54

Breiman, L (2001). "Statistical modeling: The two cultures (with comments and a rejoinder by the author)", *Statistical Science* 16(3):199–231

Bühlmann, H, M De Felice, A Gisler, F Moriconi et al. (2009). "Recursive credibility formula for chain ladder factors and the claims development result", *ASTIN Bulletin: The Journal of the IAA* 39(1):275–306

Bühlmann, H and A Gisler (2006). *A course in credibility theory and its applications*. Springer Science & Business Media

Bühlmann, H and E Straub (1983). "Estimation of IBNR reserves by the methods chain ladder, Cape Cod and complementary loss ratio," Paper presented at International Summer School. Vol. 1983

Cairns, AJG, D Blake and K Dowd (2006). "A Two-Factor Model for Stochastic Mortality with Parameter Uncertainty: Theory and Calibration", *Journal of Risk & Insurance* 73(4):687–718

Canny, J (1987). "A computational approach to edge detection," in *Readings in Computer Vision*. Elsevier, pp. 184–203

Charpentier, A (2014). *Computational actuarial science with R*. CRC Press

Chollet, F (2015). *Keras* keras.io

Chollet, F (2017). *Deep learning with Python*. Manning Publications Co

Chollet, F and J Allaire (2018). *Deep Learning with R*. Manning Publications Co

Chung, J, C Gulcehre, K Cho and Y Bengio (2015). "Gated feedback recurrent neural networks," Paper presented at International Conference on Machine Learning. 2067–2075

Currie, ID (2016). "On fitting generalized linear and non-linear models of mortality", *Scandinavian Actuarial Journal* 2016(4):356–383

Dal Moro, E, F Cuypers and P Miehe (2016). *Non-life Reserving Practices*. ASTIN

De Brébisson, A, É Simon, A Auvolat, P Vincent et al. (2015). "Artificial neural networks applied to taxi destination prediction", *arXiv* arXiv:1508.00021

Deprez, P, P Shevchenko and M Wüthrich (2017). "Machine learning techniques for mortality modeling", *European Actuarial Journal* 7(2):337–352

Dong, W, J Li, R Yao, C Li et al. (2016). "Characterizing driving styles with deep learning", *arXiv* arXiv:1607.03611

Dong, W, T Yuan, K Yang, C Li et al. (2017). "Auto-encoder regularized network for driving style representation learning", *arXiv* arXiv:1701.01272

Dowle, M and A Srinivasan (2018). *data.table*. CRAN: https://cran.r-project.org/web/packages/data.table/index.html

drive.ai (2018). *Drive.ai Announces On-Demand Self-Driving Car Service on Public Roads in Texas*. drive.ai. https://s3.amazonaws.com/www-staging.drive.ai/content/uploads/2018/05/06164346/Press-Release_Drive.ai-Texas-Deployment.pdf. Accessed: 17 June 2018

Efron, B and T Hastie (2016). *Computer Age Statistical Inference.* Cambridge University Press

Elman, J (1990). "Finding Structure in Time", *Cognitive Science* 14(2):179–211

Ezzini, S, I Berrada and M Ghogho (2018). "Who is behind the wheel? Driver identification and fingerprinting", *Journal of Big Data* 5(1):9

Federal Drug Administration (2018). *FDA permits marketing of artificial intelligence-based device to detect certain diabetes-related eye problems*. Federal Drug Administration. https://www.fda.gov/NewsEvents/Newsroom/PressAnnouncements/ucm604357.htm. Accessed: 17 June 2018

Freund, Y and R Schapire (1997). "A decision-theoretic generalization of on-line learning and an application to boosting", *Journal of Computer and System Sciences* 55(1):119–139

Friedman, J (2001). "Greedy function approximation: a gradient boosting machine", *Annals of Statistics*:1189–1232

Friedman, J, T Hastie and R Tibshirani (2009). *The Elements of Statistical Learning : Data Mining, Inference, and Prediction.* New York: Springer-Verlag. http://dx.doi.org/10.1007/978-0-387-84858-7

Gabrielli, A and M Wüthrich (2018). "An individual claims history simulation machine", *Risks* 6(2):29

Gan, G and XS Lin (2015). "Valuation of large variable annuity portfolios under nested simulation: A functional data approach", *Insurance: Mathematics and Economics* 62:138–150

Gao, G, S Meng and M Wüthrich (2018). *Claims Frequency Modeling Using Telematics Car Driving Data*. SSRN. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3102371. Accessed: 29 June 2018

Gao, G and M Wüthrich (2017). *Feature Extraction from Telematics Car Driving Heatmaps.* SSRN. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3070069. Accessed: 29 June 2018

Geladi, P and B Kowalski (1986). "Partial least-squares regression: a tutorial", *Analytica chimica acta* 185:1–17

Gelman, A and J Hill (2007). *Data analysis using regression and multilevelhierarchical models*. Cambridge University Press New York, NY, USA

Gesmann, M, D Murphy, Y Zhang, A Carrato et al. (2017). *ChainLadder: Statistical Methods and Models for Claims Reserving in General Insurance*. https://CRAN.R-project.org/package=ChainLadder

Girshick, R (2015). "Fast R-CNN", *arXiv* arXiv:1504.08083

Gisler, A and M Wüthrich (2008). "Credibility for the chain ladder reserving method", *ASTIN Bulletin: The Journal of the IAA* 38(2):565–600

Gluck, S (1997). "Balancing development and trend in loss reserve analysis," Paper presented at Proceedings of the Casualty Actuarial Society. Citeseer. Vol. 84:482–532

Goldberg, Y (2016). "A primer on neural network models for natural language processing", *Journal of Artificial Intelligence Research* 57:345–420

Goldberg, Y (2017). *Neural network methods for natural language processing*. Morgan & Claypool Publishers

Golden, L, P Brockett, J Ai and B Kellison (2016). "Empirical Evidence on the Use of Credit Scoring for Predicting Insurance Losses with Psycho-social and Biochemical Explanations", *North American Actuarial Journal* 20(3):233–251

Goodfellow, I, Y Bengio and A Courville (2016). *Deep Learning*. MIT Press

Goodfellow, I, J Pouget-Abadie, M Mirza, B Xu et al. (2014). "Generative adversarial nets," Paper presented at Advances in Neural Information Processing Systems. 2672–2680

Graves, A (2012). "Supervised sequence labelling," in *Supervised sequence labelling with recurrent neural networks*. Berlin, Heidelberg: Springer,

Graves, A, A Mohamed and G Hinton (2013). "Speech recognition with deep recurrent neural networks," Paper presented at Acoustics, speech and signal processing (icassp), 2013 ieee international conference on. IEEE. 6645–6649

Guo, C and F Berkhahn (2016). "Entity embeddings of categorical variables", *arXiv* arXiv:1604.06737

Hainaut, D (2018a). "A neural-network analyzer for mortality forecast", *Astin Bulletin* 48(2):481–508

Hainaut, D (2018b). "A self-organizing predictive map for non-life insurance", *SSRN* 2018(29 June)

Hannun, A, C Case, J Casper, B Catanzaro et al. (2014). "Deep speech: Scaling up end-to-end speech recognition", *arXiv* arXiv:1412.5567

Hazelton, M (2014). "Kernel smoothing", *Wiley StatsRef: Statistics Reference Online*

Hejazi, S and K Jackson (2016). "A neural network approach to efficient valuation of large portfolios of variable annuities", *Insurance: Mathematics and Economics* 70:169–181

Hejazi, S and K Jackson (2017). "Efficient valuation of SCR via a neural network approach", *Journal of Computational and Applied Mathematics* 313:427–439

Hinton, G, S Osindero and Y Teh (2006). "A fast learning algorithm for deep belief nets", *Neural Computation* 18(7):1527–1554

Hinton, G and R Salakhutdinov (2006). "Reducing the dimensionality of data with neural networks", *Science* 313(5786):504–507

Hinton, G, N Srivastava, A Krizhevsky, I Sutskever et al. (2012). "Improving neural networks by preventing co-adaptation of feature detectors", *arXiv* (arXiv:1207.0580)

Hochreiter, S and J Schmidhuber (1997). "Long Short-term Memory", *Neural Computation* 9(8):1735–1780

Hyndman, RJ, G Athanasopoulos, S Razbash, D Schmidt et al. (2015). "forecast: Forecasting functions for time series and linear models", *R package version* 6(6):7

James, G, D Witten, T Hastie and R Tibshirani (2013). *An Introduction to Statistical Learning.* Springer

Kassambara , A (2018). *ggpubr: 'ggplot2' Based Publication Ready Plots*. https://cran.r-project.org/web/packages/ggpubr/index.html

Kingma, DP and J Ba (2014). "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*

Kohonen, T (1990). "The self-organizing map", *Proceedings of the IEEE* 78(9):1464–1480

Krizhevsky, A, I Sutskever and G Hinton (2012). "Imagenet classification with deep convolutional neural networks," Paper presented at Advances in Neural Information Processing Systems. 1097–1105

Kuhn, M (2008). "Caret package", *Journal of Statistical Software* 28(5):1–26

Kuhn, M and K Johnson (2013). *Applied Predictive Modeling.* Springer

Kuo, K (2018a). *DeepTriangle*. GitHub: https://github.com/kevinykuo/deeptriangle

Kuo, K (2018b). "DeepTriangle: A Deep Learning Approach to Loss Reserving", *arXiv* arXiv:1804.09253

LeCun, Y, Y Bengio and G Hinton (2015). "Deep Learning", *Nature* 521(7553):436

LeCun, Y, L Bottou, Y Bengio and P Haffner (1998). "Gradient-based learning applied to document recognition", *Proceedings of the IEEE* 86(11):2278–2324

Lee, RD and LR Carter (1992). "Modeling and forecasting US mortality", *Journal of the American statistical association* 87(419):659–671

Lenail, A (2018). NN-SVG. https://github.com/zfrenchee/NN-SVG

Maaten, L and G Hinton (2008). "Visualizing data using t-SNE", *Journal of Machine Learning Research* 9(Nov):2579–2605

Mack, T (1993). "Distribution-free calculation of the standard error of chain ladder reserve estimates", *Astin Bulletin* 23(02):213–225

Mack, T (2002). *Schadenversicherungsmathematik 2. Auflage.* Schriftenreihe Angewandte Versicherungsmathematik, DGVM

Makridakis, S, E Spiliotis and V Assimakopoulos (2018a). "The M4 Competition: Results, findings, conclusion and way forward", *International Journal of Forecasting* 34(4):802–808

Makridakis, S, E Spiliotis and V Assimakopoulos (2018b). "Statistical and Machine Learning forecasting methods: Concerns and ways forward", *PLOS ONE* 13(3):e0194889

McGrayne, S (2011). *The theory that would not die: how Bayes' rule cracked the enigma code, hunted down Russian submarines, & emerged triumphant from two centuries of controversy.* Yale University Press

Meyers, G (2015). *Stochastic loss reserving using Bayesian MCMC models.* Casualty Actuarial Society New York

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

Mikolov, T, I Sutskever, K Chen, G Corrado et al. (2013). "Distributed representations of words and phrases and their compositionality," Paper presented at Advances in neural information processing systems. 3111–3119

Mitchell, T (1997). *Machine learning*. McGraw-Hill Boston, MA

Mullainathan, S and J Spiess (2017). "Machine learning: an applied econometric approach", *Journal of Economic Perspectives* 31(2):87–106

Nair, V and G Hinton (2010). "Rectified linear units improve restricted boltzmann machines," Paper presented at Proceedings of the 27th International Conference on Machine Learning. 807–814

Ng, A (2017). *Deep Learning Specialization*. Coursera. https://www.coursera.org/specializations/deep-learning. Accessed: 24 June 2018

Noll, A, R Salzmann and M Wüthrich (2018). *Case Study: French Motor Third-Party Liability Claims*. SSRN. https://ssrn.com/abstract=3164764 Accessed: 17 June 2018

Ohlsson, E and B Johansson (2010). *Non-life insurance pricing with generalized linear models*. Springer

Parodi, P (2012a). "Computational intelligence with applications to general insurance: a review: I – The role of statistical learning", *Annals of Actuarial Science* 6(2):307–343

Parodi, P (2012b). "Computational intelligence with applications to general insurance: a review: II. Dealing with uncertain knowledge", *Annals of Actuarial Science* 6(2):344–380

Parodi, P (2014). *Pricing in general insurance*. CRC Press

Parodi, P (2016). "Towards machine pricing", Paper presented at GIRO (2016). Dublin

Paszke, A, S Gross, S Chintala, G Chanan et al. (2017). "Automatic differentiation in PyTorch",

Pedregosa, F, G Varoquaux, A Gramfort, V Michel et al. (2011). "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research* 12(Oct):2825–2830

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org

Racine, J and T Hayfield (2018). *np: Nonparametric Kernel Smoothing Methods for Mixed Data Types*. CRAN. https://cran.r-project.org/web/packages/np/index.html

Renshaw, AE and RJ Verrall (1998). "A stochastic model underlying the chain-ladder technique", *British Actuarial Journal* 4(04):903–923

Richman, R (2017). "Old age mortality in South Africa, 1985–2011." Unpublished thesis, Cape Town: University of Cape Town

Riedmiller, M and H Braun (1993). "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," Paper presented at IEEE International Conference on Neural Networks, (1993). IEEE. 586–591

Rosenblatt, F (1958). "The perceptron: a probabilistic model for information storage and organization in the brain", *Psychological Review* 65(6):386

Rumelhart, D, G Hinton and R Williams (1986). "Learning representations by back-propagating errors", *nature* 323(6088):533

Schmidt, K (2017). *A Bibliography on Loss Reserving*. https://www.math.tu-dresden.de/sto/schmidt/dsvm/reserve.pdf. Accessed: 8 July 2018

Schreiber, D (2017). *The Future of Insurance*. DIA Munich 2017: https://www.youtube.com/watch?time_continue=1&v=LDOhFHJqKqI. Accessed: 17 June 2018

Shmueli, G (2010). "To explain or to predict?", *Statistical Science*:289–310

Smith, M, F Beyers and J de Villiers (2016). "A method of parameterising a feed forward multi-layered perceptron artificial neural network, with reference to South African financial markets", *South African Actuarial Journal* 16(1):35–67

Sutskever, I, O Vinyals and Q Le (2014). "Sequence to sequence learning with neural networks," Paper presented at Advances in neural information processing systems. 3104–3112

Sutton, R and A Barto (2018). *Reinforcement learning: An introduction, Second Edition*. MIT Press

Szegedy, C, W Liu, Y Jia, P Sermanet et al. (2015). "Going deeper with convolutions," Paper presented at 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE. 1–9

Thatcher, AR, V Kannisto and K Andreev (2002). "The survivor ratio method for estimating numbers at high ages", *Demographic Research* 6(1):2–15

Tibshirani, R (1996). "Regression shrinkage and selection via the lasso", *Journal of the Royal Statistical Society. Series B (Methodological)*:267–288

Tomas, J and F Planchet (2014). "Prospective mortality tables and portfolio experience," in Charpentier, A (ed). *Computational actuarial science with R*. CRC Press,

Turner, H and D Firth (2007). "Generalized nonlinear models in R: An overview of the gnm package", https://cran.r-project.org/web/packages/gnm/vignettes/gnmOverview.pdf

Udacity (2018). *Deep Learning*. Udacity. https://eu.udacity.com/course/deep-learning-nanodegree--nd101. Accessed: 24 June 2018

Verbelen, R, K Antonio and G Claeskens (2018). "Unravelling the predictive power of telematics data in car insurance pricing", *Journal of the Royal Statistical Society: Series C (Applied Statistics)* (Pre-print)

Villegas, A, V Kaishev and P Millossovich (2015). *StMoMo: An R package for stochastic mortality modelling*. CRAN. https://cran.r-project.org/web/packages/StMoMo/index.html

Viola, P and M Jones (2001). "Rapid object detection using a boosted cascade of simple features," Paper presented at Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001. IEEE. Vol. 1:I-I

Weidner, W, F Transchel and R Weidner (2016a). "Telematic driving profile classification in car insurance pricing", *Annals of Actuarial Science* 11(2):213–236

Weidner, W, FWG Transchel and R Weidner (2016b). "Classification of scale-sensitive telematic observables for risk individual pricing", *European Actuarial Journal* 6(1):3–24

Weisstein, E (2003). *Convolution*. http://mathworld.wolfram.com/Convolution.html. Accessed: 24 June 2018

Werbos, P (1988). "Generalization of backpropagation with application to a recurrent gas market model", *Neural Networks* 1(4):339–356

Wickham, H (2016). *ggplot2: elegant graphics for data analysis*. Springer

Wijnands, J, J Thompson, G Aschwanden and M Stevenson (2018). "Identifying behavioural change among drivers using Long Short-Term Memory recurrent neural networks", *Transportation Research Part F: Traffic Psychology and Behaviour* 53:34–49

Wilmoth, JR and V Shkolnikov (2010). "Human Mortality Database", University of California, Berkeley (USA), and Max Planck Institute for Demographic Research (Germany). Available at www.mortality.org. Accessed: 1 July 2018

Wood, S (2017). *Generalized additive models: an introduction with R.* Chapman and Hall/CRC

Wu, Y, M Schuster, Z Chen, Q Le et al. (2016). "Google's neural machine translation system: Bridging the gap between human and machine translation", *arXiv* arXiv:1609.08144

Wüthrich, M (2018a). "Machine learning in individual claims reserving", *Scandinavian Actuarial Journal*:1–16

Wüthrich, M (2018b). *Neural networks applied to chain-ladder reserving*. SSRN. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2966126. Accessed: 1 July 2018

Wüthrich, M (2018c). *v-a Heatmap Simulation Machine*. https://people.math.ethz.ch/~wueth/simulation.html. Accessed: 1 July 2018

Wüthrich, M and C Buser (2018). *Data analytics for non-life insurance pricing*. Swiss Finance Institute Research Paper. https://ssrn.com/abstract=2870308. Accessed: 17 June 2018

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

Wüthrich, MV (2017). "Covariate selection from telematics car driving data", *European Actuarial Journal* 7(1):89–108

Zarkadoulas, A (2017). "Neural network algorithms for the development of individual losses." Unpublished thesis, Lausanne: University of Lausanne

Zou, H and T Hastie (2005). "Regularization and variable selection via the elastic net", *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67(2):301–320

APPENDIX A
List of Actuarial Journals

Annals of Actuarial science
ASTIN Bulletin
British Actuarial Journal
European Actuarial Journal
Insurance: Mathematics and Economics
Journal of Risk and Insurance
Scandinavian Actuarial Journal

APPENDIX B
Abbreviations/acronyms

AI	Artificial Intelligence
CNN	Convolutional Neural Network
CPU	Central Processing Unit
EDA	Exploratory data analysis
GAM	Generalised Additive Model
GAN	Generative Adversarial Model
GLM	Generalised Linear Model
GLMM	Generalised Linear Mixed Model
GNM	Generalised Non-linear Model
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
IBNeR	Incurred but not Enough Reported
IBNR	Incurred but not Reported
ILR	Incremental Loss Ratio
LASSO	Least absolute shrinkage and selection operator
LDF	Loss Development Factor
LoB	Line of Business
LSTM	Long Short Term Memory
MAE	Mean Absolute Error
MSE	Mean Squared Error
NLP	Natural Language Processing
PBM	Poisson Boosting Machine
PCA	Principal Components Analysis
PLS	Partial Least Squares
RBM	Restricted Boltzmann Machine
RNN	Recurrent Neural Network
SOM	Self Organising Map
TPL	Third Party Liability