Erkenntnisse zu Neuronalen Netzen in der Nicht-Leben Tarifierung

Daten, Zahlen, Algorithmen – Data Science in der Schadenversicherung im Fokus

Cologne, 2nd December 2019

Dr. Jürg Schelldorfer, Actuary SAA

Senior Analytics Professional, Swiss Re

Chair of the «Data Science» working group, Swiss Association of Actuaries

Disclaimer

The opinions expressed in this presentation are those of the author only. They are inspired by the work that the author is doing for both Swiss Re and the SAA, but they do not necessarily reflect any official view of either Swiss Re or the SAA.

Machine Learning in the insurance industry

Dr. Tobias Büttner, Head of Claims, Munich Re, mentioned the following[1]:

Property claims were assessed using images.

But later the reserves had to be increased significantly. **Damages below/hidden in the roofs have not been appropriately estimated.**

Implications of the use of Machine Learning (ML) in insurance:

- **ML can affects operations**, which impact the data actuaries use (i.e. claims, underwritten risks,…)
- **ML can affect** the underyling risks
- ML can be used to strenghten the core skills (extend the actuary's toolbox)
- **Automation** (not necessarily ML) can help to improve efficiency

[1] SZ-Fachkonferenz: KI und Data Analytics in der Versicherungsbranche; Data Analytics im Management von Großschäden, Büttner T. (2019), Munich Re

SAA working party «Data Science»

www.actuarialdatascience.org

articles, data and code of the tutorials,

references to literature

Table of Content

1. Factor embeddings in neural networks[1]

[1] Paper: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3320525

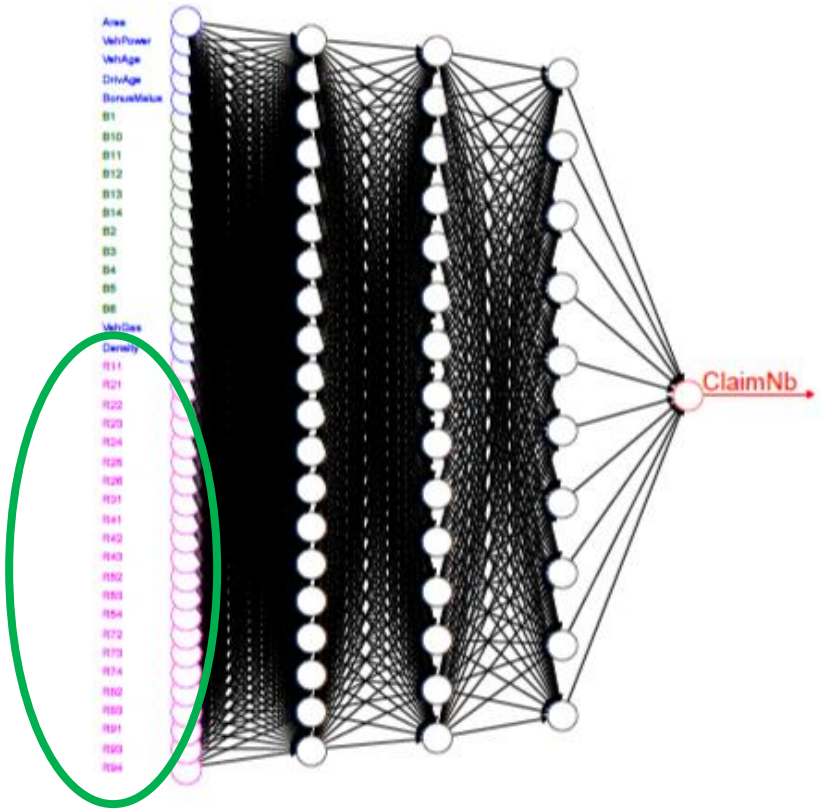
Motivation and data

- In insurance pricing, categorical variables (i.e. vehicle brand, region,…) consist of many levels.
- They are often encoded as dummy variables (or one-hot encoding), i.e. the levels are orthogonal in the feature space.
- Potential improvements:
  i. Grouping of variable levels (by…)
  ii. Use longitude/lattitude for geographical factor variables
  iii. Embedding with Neural Networks

Listing 1: output of command `str(freMTPL2freq)`

```
> str(freMTPL2freq)
'data.frame':    678013 obs. of  12 variables:
 $ IDpol      : num  1 3 5 10 11 13 15 17 18 21 ...
 $ ClaimNb    : num [1:678013(1d)] 1 1 1 1 1 1 1 1 1 1 ...
  ..- attr(*, "dimnames")=List of 1
  .. ..$ : chr  "139" "414" "463" "975" ...
 $ Exposure   : num  0.1 0.77 0.75 0.09 0.84 0.52 0.45 0.27 0.71 0.15 ...
 $ Area       : Factor w/ 6 levels "A","B","C","D",..: 4 4 2 2 2 5 5 3 3 2 ...
 $ VehPower   : int  5 5 6 7 7 6 6 7 7 7 ...
 $ VehAge     : int  0 0 2 0 0 2 2 0 0 0 ...
 $ DrivAge    : int  55 55 52 46 46 38 38 33 33 41 ...
 $ BonusMalus: int  50 50 50 50 50 50 50 68 68 50 ...
 $ VehBrand   : Factor w/ 11 levels "B1","B10","B11",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ VehGas     : Factor w/ 2 levels "Diesel","Regular": 2 2 1 1 1 2 2 1 1 1 ...
 $ Density    : int  1217 1217 54 76 76 3003 3003 137 137 60 ...
 $ Region     : Factor w/ 22 levels "R11","R21","R22",..: 18 18 3 15 15 8 8 20 20 12 ...
```

Neural networks with one-hot encoding

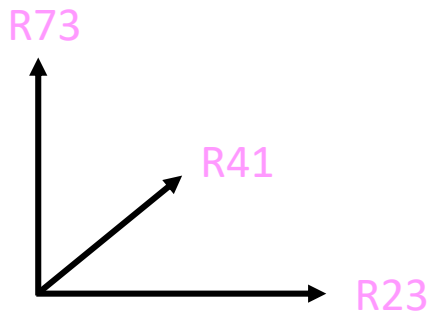


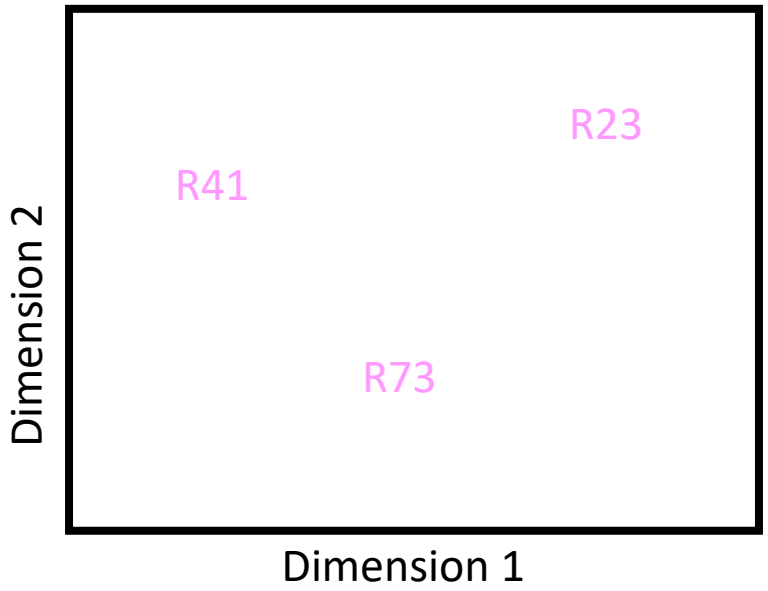
With Neural Networks, there is a challenge when using one-hot encoding:

The number of parameters gets large due to the neural network architecture.

How embeddings work

- Embeddings are very well known in Natural Language Processing (NLP), and significant progress has been made due to embeddings.



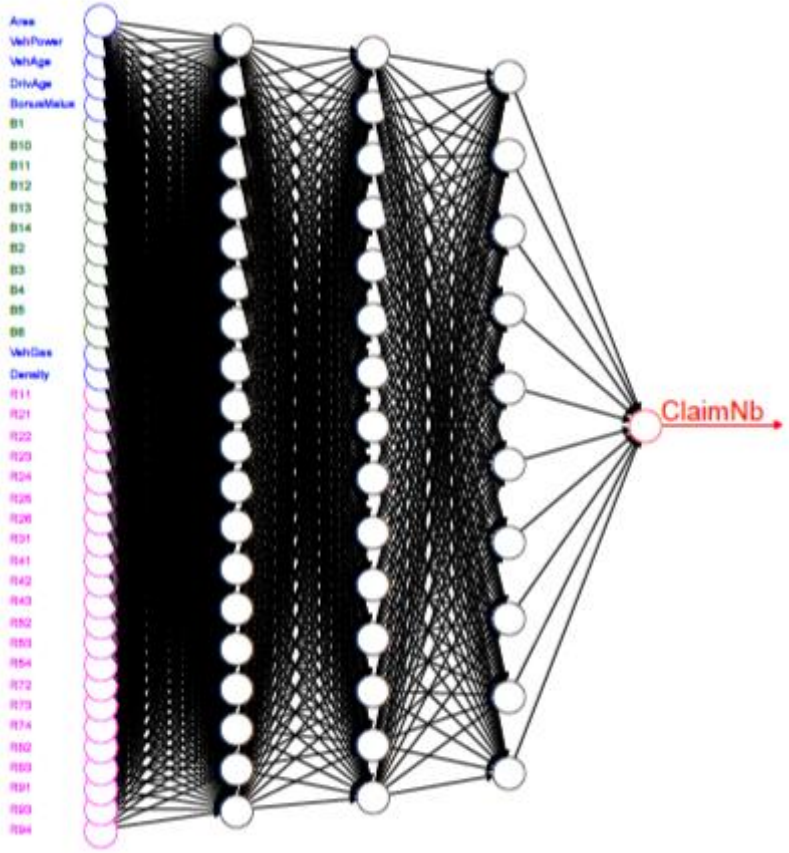


d-dimensional embedding:

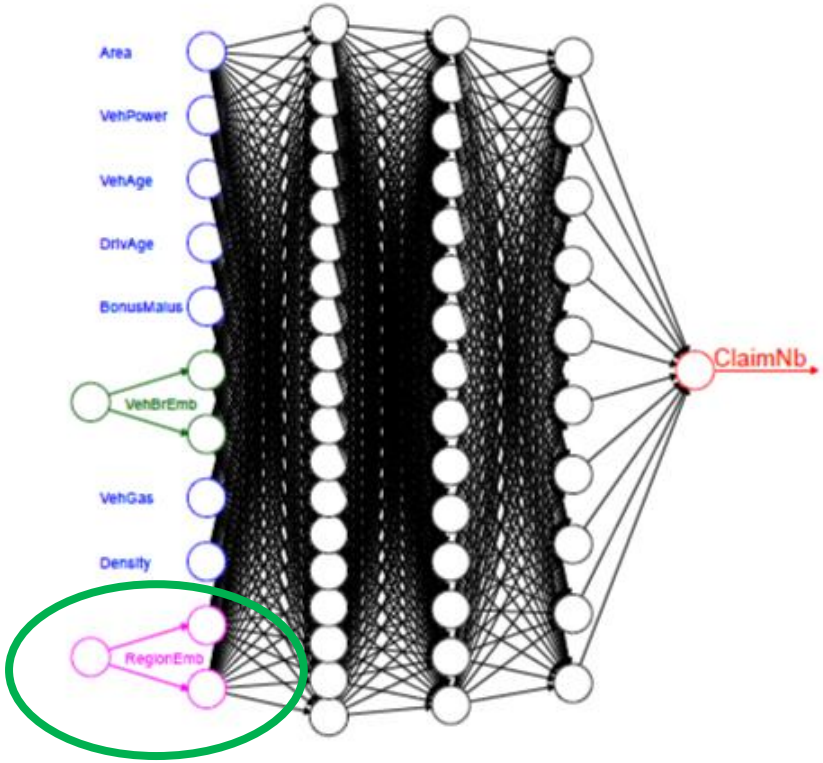
$$e: \{R_1, \dots, R_n\} \rightarrow R^d$$
$$Region \mapsto e(region) = (e_1^{region}, e_2^{region})$$

- The embedding weights e_j^{region} are learned during model training.
- Embeddings can be included as other specialized layers (drop-out, normalization,…)

Neural Network architecture

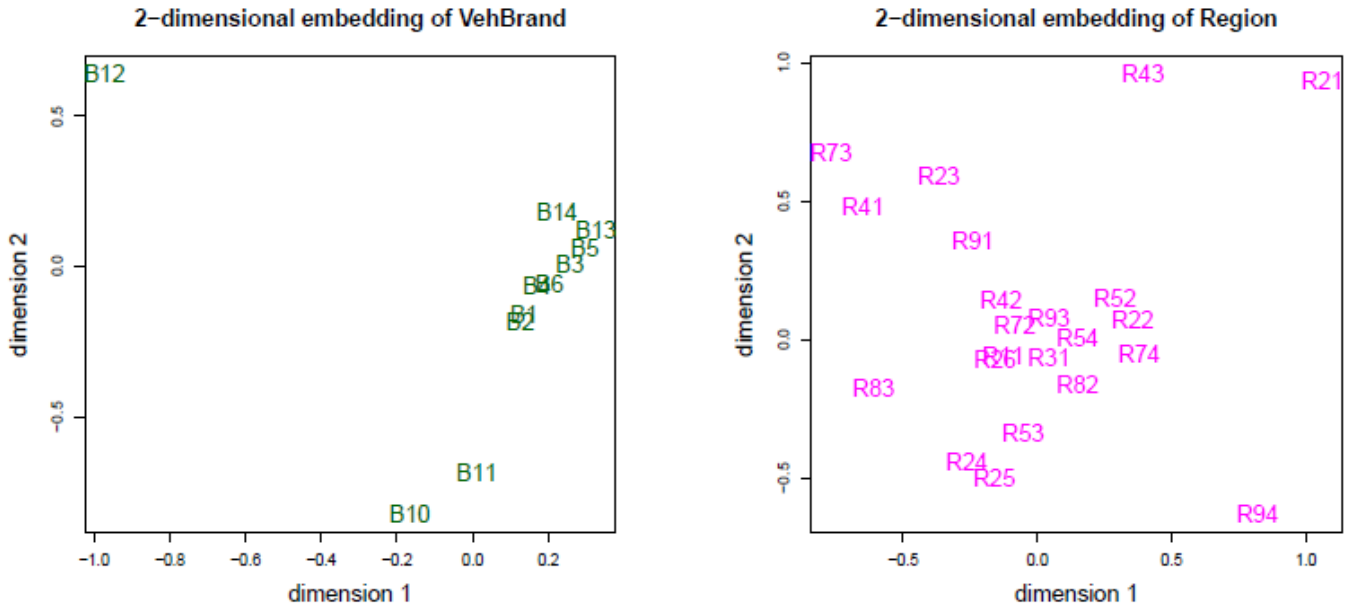


1’301 parameters



792 parameters

Embedding weights



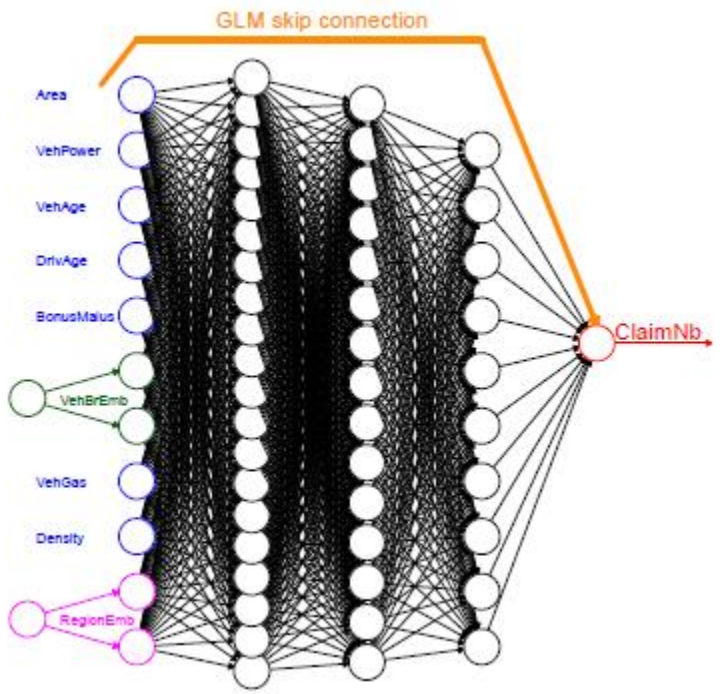
- Using the keras library, there are just a few lines of code to define the embeddings (see the references for further details).
- The embedding can help to group factor levels based on the data (similar levels are close).
- Care is needed as the 2-dimensional embedding visuals are subject to rotation.

2. Combined Actuarial Networks (CANN)[1]

[1] Paper(s): https://doi.org/10.1017/asb.2018.42; https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3320525

Idea to nest a GLM into a neural network

- We usually have already a GLM in production. So we would like to include this knowledge in the neural network.
- Henceforth, the neural network shall identify structure not captured by the current GLM.
- With such an approach, the neural network can be considered as a «challenger model» compared to the «benchmark model».
- Using a so-called «skip connection»:



(Part of) mathematics for CANN

- Linear predictor with regression parameter $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_q)^\top \in \mathbb{R}^{q+1}$

$$\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^q \quad \mapsto \quad \theta^{\mathrm{GLM}}(\boldsymbol{x}; \boldsymbol{\beta}) = \langle \boldsymbol{\beta}, \widetilde{\boldsymbol{x}} \rangle \in \boldsymbol{\Theta}.$$

  ▷ Feature pre-processing is done by the actuary/statistician.

- Choose network of depth $d \in \mathbb{N}$ with network parameter $\boldsymbol{w} = (W_{1:d}, \boldsymbol{w}_{d+1})$

$$\boldsymbol{z} \in \mathbb{R}^{q_d} \quad \mapsto \quad \theta^{\mathrm{NN}}(\boldsymbol{z}; \boldsymbol{w}_{d+1}) = \langle \boldsymbol{w}_{d+1}, \widetilde{\boldsymbol{z}} \rangle \in \boldsymbol{\Theta},$$

  with neural network function (feature pre-processing $\boldsymbol{x} \mapsto \boldsymbol{z}$)

$$\boldsymbol{x} \;\mapsto\; \boldsymbol{z} \;=\; \boldsymbol{z}^{(d:1)}(\boldsymbol{x}) \;=\; \left(\boldsymbol{z}^{(d)} \circ \cdots \circ \boldsymbol{z}^{(1)}\right)(\boldsymbol{x}).$$

  ▷ Feature pre-processing is done by the d hidden network layers.

- Choose linear predictor with parameter $(\boldsymbol{\beta}, \boldsymbol{w}) = (\boldsymbol{\beta}, W_{1:d}, \boldsymbol{w}_{d+1})$

$$\boldsymbol{x} \;\mapsto\; \theta^{\mathrm{CANN}}(\boldsymbol{x}; \boldsymbol{\beta}, \boldsymbol{w}) \;=\; \langle \boldsymbol{\beta}, \widetilde{\boldsymbol{x}} \rangle + \left\langle \boldsymbol{w}_{d+1}, \widetilde{\boldsymbol{z}}^{(d:1)}(\boldsymbol{x}) \right\rangle.$$

Why is this useful?

- Extension of GLM (unfortunately not for GAM).
- One way of including expert knowledge in a Neural Network.
- GLM as good starting value for the optimization.
- Identification of missing interactions in the current GLM.
- Enables uncertainty quantification as good starting value for the optimization.

3. Portfolio bias in neural networks[1]

[1] Paper: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3347177

Unbiased portfolio average

For insurance pricing, we expect that the model provides unbiased estimates on the portfolio level.

- Assume

$$Y_i \overset{\text{ind.}}{\sim} f(y;\theta_i^\star,\phi^\star/w_i;b^\star)=\exp\left\{\frac{y\theta_i^\star-b^\star(\theta_i^\star)}{\phi^\star/w_i}+c^\star(y;\phi^\star/w_i)\right\},$$

- The true portfolio average is given by:

$$\bar{\mu}^\star=\mathbb{E}\left[\frac{\sum_{i=1}^n w_iY_i}{\sum_{i=1}^n w_i}\right]=\frac{1}{\sum_{i=1}^n w_i}\sum_{i=1}^n w_i(b^\star)'(\theta_i^\star),$$

- For the homogenous model, i.e. where all customers have the same frequency, it holds:

Proposition 2.2. *Assume that true model is given by (2.3). The MLE of the homogeneous model provides an unbiased portfolio average, that is,*

$$\mathbb{E}\left[\bar{\mu}^{\text{hom}}\right]=\bar{\mu}^\star, \qquad \textit{with uncertainty} \quad \text{Var}\left(\bar{\mu}^{\text{hom}}\right)=\frac{\phi^\star}{\left(\sum_{i=1}^n w_i\right)^2}\sum_{i=1}^n w_i(b^\star)''(\theta_i^\star).$$

GLM unbiased portfolio average

- For a GLM, assuming

$$Y_i \overset{\text{ind.}}{\sim} f(y;\theta_i,\phi/w_i;b) = \exp\left\{\frac{y\theta_i - b(\theta_i)}{\phi/w_i} + c(y;\phi/w_i)\right\},$$

- holds:

Corollary 2.3. *Assume that true model is given by* (2.3). *The MLE of the GLM provides an unbiased portfolio average, that is,*

$$\mathbb{E}\left[\bar{\mu}^{\text{GLM}}\right] = \bar{\mu}^\star, \qquad \textit{with uncertainty} \quad \text{Var}\left(\bar{\mu}^{\text{GLM}}\right) = \frac{\phi^\star}{\left(\sum_{i=1}^n w_i\right)^2} \sum_{i=1}^n w_i (b^\star)''(\theta_i^\star).$$

This means that the GLM provides the same overall portfolio level as the homogenous model.

Portfolio bias in neural networks

For a neural network:

- The MLE would result in overfitting.
- The gradient descent method (GDM) includes early stopping, hence it stops before reaching a local minimum.
- The solution is not unique, and it even depends on the random seed used! So individual prices depend on the chosen seed.
- For an example, figures are as follows:

Figure 1.2. Remark that the evaluation of (3.5) requires knowledge of the true means $\mu_i^\star$ which are available in our special set-up.

	# param.	in-sample loss $\overline{\mathcal{L}}_{\mathcal{D}}(\boldsymbol{\mu})$	estimation error $\mathcal{E}_{\mu^\star}(\boldsymbol{\mu})$	portfolio average $\bar{\mu}$
(a) true model $\mu_i^\star$		27.7278	0.0000	10.1991%
(b) homogeneous model $\widehat{\mu}_i^{\text{hom}} = \bar{\mu}^{\text{hom}}$	1	29.1065	1.3439	10.2691%
(c) GLM $\widehat{\mu}_i^{\text{GLM}}$	57	28.1282	0.4137	10.2691%
(d1) neural network $\widehat{\mu}_i^{\text{NN}}$ run no. 1	780	27.7204	0.1566	10.2973%
(d2) neural network $\widehat{\mu}_i^{\text{NN}}$ run no. 2	780	27.7484	0.1795	10.0661%
(d3) neural network $\widehat{\mu}_i^{\text{NN}}$ run no. 3	780	27.7621	0.1669	10.0605%

Avoid portfolio bias

The following approaches for overcoming the bias in neural networks are suggested:

- Based on the neurel network estimates, re-fit a glm with the weights from the last hidden layer as new covariates, then it results in an unbiased estimated portfolio average.
- Penalty term on the portfolio average inducing an additional tuning parameter.

Numerical results as follows:

	# param.	in-sample loss $\overline{\mathcal{L}}_{\mathcal{D}}(\widehat{\mu})$	estimation error $\mathcal{E}_{\mu^\star}(\widehat{\mu})$	portfolio average $\bar{\mu}$
(a) true model $\mu_i^\star$		27.7278	0.0000	10.1991%
(b) homogeneous model $\widehat{\mu}_i^{\text{hom}} = \bar{\mu}^{\text{hom}}$	1	29.1065	1.3439	10.2691%
(c) GLM $\widehat{\mu}_i^{\text{GLM}}$	57	28.1282	0.4137	10.2691%
(d1) neural network $\widehat{\mu}_i^{\text{NN}}$ run no. 1	780	27.7204	0.1566	10.2973%
(d2) neural network $\widehat{\mu}_i^{\text{NN}}$ run no. 2	780	27.7484	0.1795	10.0661%
(d3) neural network $\widehat{\mu}_i^{\text{NN}}$ run no. 3	780	27.7621	0.1669	10.0605%
(e1) GLM neural network $\widehat{\mu}_i^{\text{NN+}}$ run no. 1	780	27.7142	0.1605	10.2691%
(e2) GLM neural network $\widehat{\mu}_i^{\text{NN+}}$ run no. 2	780	27.7428	0.1825	10.2691%
(e3) GLM neural network $\widehat{\mu}_i^{\text{NN+}}$ run no. 3	780	27.7555	0.1654	10.2691%

4 – Random Forest[1]

[1] Paper(s): https://arxiv.org/pdf/1904.10890.pdf, https://kuleuvencongres.be/eaj2018/documents/presentations/1-roel-henckaerts.pdf

Random Forest

Motivation:

- Standard random forest implemenations use the mean squared error (L_2) loss.
- L_2 is a priori not the right loss function for claim frequency and severity.

Solutions:

- *distRforest*: decision trees and random forest for frequency (Poisson) and severity distributions (Lognormal, Gamma) (GitHub) filling the gap.

R Universe

Filling the gaps

Model	Poisson	Gamma
Generalized linear model	✓ `stats`	✓ `stats`
Generalized additive model	✓ `mgcv`	✓ `mgcv`
Regression tree	✓ `rpart`	✓ `rpart*`
Random forest	✓ `rpart*`	✓ `rpart*`
Gradient boosting machine	✓ `gbm`	✓ `harrysouthworth/gbm`

*rpart** is the *distRforest* package, which is an extension of *rpart*

5 – Model Risk Management

Literature for Neural Networks

‘Machine Decisions’: Governance of AI and Big Data Analytics; CRO Forum (2019)

- «…that model governance techniques and frameworks that exist today **do not need to be fundamentally altered, but can be enhanced and adjusted** to meet the evolving needs of complex tools and machine learning developments»
- Model Management Framework
- Ethical Framework

Believing the Bot - Model Risk in the Era of Deep Learning

Ronald Richman* Nicolai von Rummell† Mario V. Wüthrich‡

Version of August 29, 2019

Abstract

Deep Learning models are currently being introduced into business processes to support decision-making in insurance companies. At the same time model risk is recognized as an increasingly relevant field within the management of operational risk that tries to mitigate the risk of poor business decisions because of flawed models or inappropriate model use. In this paper we try to determine how Deep Learning models are different from established actuarial models currently in use in insurance companies and how these differences might necessitate changes in the model risk management framework. We analyse operational risk in the development and implementation of Deep Learning models using examples from pricing and mortality forecasting to illustrate specific model risks and controls to mitigate those risks. We discuss changes in model governance and the role that model risk managers could play in providing assurance on the appropriate use of Deep Learning models.

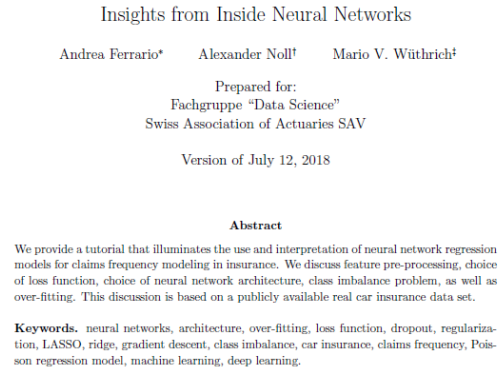
Keywords. Deep learning, Model Risk, Pricing, Mortality Forecasting, Insurance Modelling

1 Introduction

Deep learning refers to a modern approach to designing and fitting neural networks that recently has achieved state of the art results on machine learning problems in computer vision, natural language processing, machine translation and speech recognition, and has become the main avenue for solving unstructured data problems [1, 2]. In addition to unstructured data, deep learning approaches have produced promising results on structured data problems [3], as well as time series forecasting [4]. Modern neural networks are generally characterized by specialized architectures that are adapted to domain-specific problems, as well as by the depth of the networks, meaning to say, that these networks are composed of multiple layers of non-linear functions. Recently, deep learning techniques have been applied to problems within actuarial science such as pricing, reserving, analysis of telematics data and mortality forecasting. For a recent review of these applications, see [5]. The benefits of applying deep learning to actuarial

Model Risk for NN

Guideline for fitting a NN for Pricing



Insights from Inside Neural Networks

Andrea Ferrario* Alexander Noll† Mario V. Wüthrich‡

Prepared for:
Fachgruppe “Data Science”
Swiss Association of Actuaries SAV

Version of July 12, 2018

Abstract

We provide a tutorial that illuminates the use and interpretation of neural network regression models for claims frequency modeling in insurance. We discuss feature pre-processing, choice of loss function, choice of neural network architecture, class imbalance problem, as well as over-fitting. This discussion is based on a publicly available real car insurance data set.

Keywords. neural networks, architecture, over-fitting, loss function, dropout, regularization, LASSO, ridge, gradient descent, class imbalance, car insurance, claims frequency, Poisson regression model, machine learning, deep learning.

Key points[1]

Using neural networks for actuarial modeling, care is needed:

(1) The neural network results depends to some extend on the random seed.

(2) Neural networks may fail to reproduce portfolio averages.

(3) Neural network results may not be stable over time.

For model risk management of deep learning models, the following points are crucial:

(1) Feature engineering is performed by the neural network

(2) A large space of potential models is explored

(3) Focus on predictive accuracy using an explicit loss function

(4) Stochastic training of the models

Carefully check unwanted biases (gender, ethnicity,…) using appropriate techniques

[1] Paper(s): https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3444833

6. Summary

Conclusions

- Statistical learning methods and neural networks allow to fit dependency structures naturally beyond the (currently used) GLM.
- CANN provide the framework for extending the GLM's, allowing to improve the accuracy of the model as well as providing a framework to assess the uncertainties.
- Model risk management needs to be addressed carefully for machine learning models
- Neural networks can be easily implemented using the keras package (in R or Python)

And yet, a very well calibrated GLM may still be as good as an advanced machine learning model in terms of accuracy.

SAA working party «Data Science»

www.actuarialdatascience.org

articles, data and code of the tutorials,

references to literature

Acknowledgements

People:

- All members of the SAA working group
- Dr. Alexander Noll
- Dr. Simon Rentzmann
- Ron Richman

Insititutions:

- Swiss Association of Actuaries (SAA)
- RiskLab at ETH Zurich
- MobiLab for Analytics at ETH Zurich

Companies:

- Swiss Re

References

- www.actuarialdatascience.org
- SZ-Fachkonferenz: KI und Data Analytics in der Versicherungsbranche; Data Analytics im Management von Großschäden, Büttner T. (2019), Munich Re
- Nesting Classical Actuarial Models into Neural Networks, Schelldorfer J. and Wüthrich M.V. (2019), SAA
- Editorial: Yes, we CANN!, Wüthrich, M.V., Merz, M. (2019). ASTIN Bulletin 49/1
- Bias Regularization in Neural Network Models for General Insurance Pricing, Wüthrich M.V. (2019), SSRN
- Boosting insights in insurance tariff plans with tree-based machine learning methods, Henckaerts R. et. Al. (2019), arXiv
- distRforest, Henckaerts R. (2019), GitHub
- 'Machine Decisions': Governance of AI and Big Data Analytics, CRO Forum (2019)
- Believing the Bot – Model Risk in the Era of Deep Learning, Richman R., von Rummell N, Wüthrich M.V. (2019), SSRN
- Insights from Inside Neural Networks, Ferrario A., Noll A., Wüthrich M.V. (2018), SSRN

7. Appendix

(Our) Topics in Actuarial Data Science

We have written the following six tutorials:

1. French Motor Third-Party Liability Claims: Introduction, boosting and neural networks for P&C Pricing
2. Insights from Inside Neural Networks: Guidance how to fit neural networks for insurance data
3. Nesting Classical Actuarial Models into Neural Networks: Embedding of GLM's into neural networks
4. On Boosting: Theory and Applications: Boosting and its variant illustrated with a P&C Kaggle dataset
5. Unsupervised Learning: What is a Sports Car?: Unsupervised learning techniques applied in P&C
6. Lee and Carter go Machine Learning: Recurrent Neural Networks: LSTM NN applied to mortality forecasting

We are working on the following:

- Natural Language Processing and RNN's
- Segmentation using decision trees
- Mortality forecasting and CNN's
- Explainability / Interpretability of machine learning models

Further topics and ideas:

- Missing data and data imputation
- Dissimilarity measures for categorical variables
- Graphical Models / Causality?
- GAN?
- Performance measures and visualizations?
- Spatial modeling and random (Gaussian) fields?

ADS basics: Articles and repositories

The following articles/repositories are fundamental for entering the topic of Actuarial Data Science (ADS):

- Data Analytics for Non-Life Insurance Pricing, ETH Zurich, M.V. Wüthrich and C. Buser
- AI in Actuarial Science, R. Richman, SSRN, 2018
- ADS Tutorials, SAA, 2018-present
- Insurance Analytics – A Primer, International Summer School of the Swiss Association of Actuaries, 2018
- Insurance Data Science: Use and Value of Unusual Data, International Summer School of the Swiss Association of Actuaries, 2019

Model Risk Management:

- 'Machine Decisions': Governance of AI and Big Data Analytics, CRO Forum, 2019
- Believing the Bot – Model Risk in the Era of Deep Learning, R. Richman et. Al, arXiv, 2019

ADS basics: R packages[1,2]

ML meta packages:
- caret
- mlr

Neural Networks:
- keras

Machine/Statistical Learning excl. NN
- rpart
- ranger, randomForest, distRforest
- xgboost, gbm
- cluster, clusterR, tsne, umap, kohonen
- glmnet

data:
- tidyverse
- data.table

Visualisations:
- ggplot2
- DataExplorer
- esquisse

Interpretability:
- iml
- flashlight

Others:
- Rmarkdown
- Rshiny

Insurance data:
- CASdatasets
- simulationengine

[1] R Core Team (2013). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.
[2] CRAN Task View: Machine Learning & Statistical Learning, T. Hothorn, 2019