The use of autoencoders for training neural networks with mixed categorical and numerical features

Łukasz Delong

SGH Warsaw School of Economics
Institute of Econometrics, Division of Probabilistic Methods

Joint work with Anna Kozak, Warsaw University of Technology

Introduction

We investigate three important aspects of training neural networks with mixed categorical and numerical features for supervised learning tasks:

- Numerical representation of categorical features which is used as the input to neural networks,
- Architecture of neural networks, in particular how to concatenate features of different types (numerical and categorical),
- Initialization of parameters (weights and bias terms) of neural networks.

We approach the problems with autoencoders.

Autoencoders

- Let $\boldsymbol{x}$ denote a vector of features of dimension p,
- The vector $\boldsymbol{x}$ consists of mixed numerical and categorical features, where the categorical features are one-hot encoded (by $\boldsymbol{x}_j^{cat}$ we denote the one-hot encoded representation of categorical feature x_j),
- An autoencoder consists of two functions:

$$\varphi : \mathbb{R}^p \mapsto \mathbb{R}^l, \quad and \quad \psi : \mathbb{R}^l \mapsto \mathbb{R}^p,$$

  where the mapping φ is called encoder, and ψ is called decoder,
- We define the reconstruction function:

$$\pi = \psi \circ \varphi : \mathbb{R}^p \mapsto \mathbb{R}^p.$$

- The goal is to find the functions φ and ψ such that the reconstruction error measured with a loss function L:

$$\frac{1}{n}\sum_{i=1}^{n} L(\pi(\boldsymbol{x}_i), \boldsymbol{x}_i),$$

  is minimized in a data set with observations $(\boldsymbol{x}_i)_{i=1}^n$,
- Classical examples of linear autoencoders are based on Principal Component Analysis for numerical data and Multiple Correspondence Analysis for categorical data,
- We build non-linear autoencoders with neural networks.

Denoising autoencoders

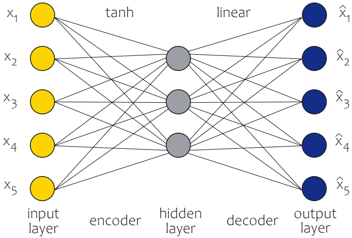
- The mapping $\boldsymbol{x} \mapsto \varphi(\boldsymbol{x})$ from the encoder gives an l-dimensional representation of the p-dimensional vector $\boldsymbol{x}$,
- If $l < p$, we construct under-complete autoencoders,
- If $l = p$, then we can achieve zero reconstruction error by learning the identity mapping,
- If $l > p$, we construct over-complete autoencoders,
- The goal is to find the functions φ and ψ such that the reconstruction error measured with a loss function L:

$$\frac{1}{n}\sum_{i=1}^{n} L(\pi(\tilde{\boldsymbol{x}}_i), \boldsymbol{x}_i),$$

  is minimized in a data set with observations $(\boldsymbol{x}_i)_{i=1}^{n}$,
- This time $\tilde{\boldsymbol{x}}$ denotes a corrupted version of the original input $\boldsymbol{x}$ which is constructed by adding a noise to $\boldsymbol{x}$,
- Such types of autoencoders are called denoising autoencoders - not tested so far in actuarial applications.

Autoencoders for numerical data

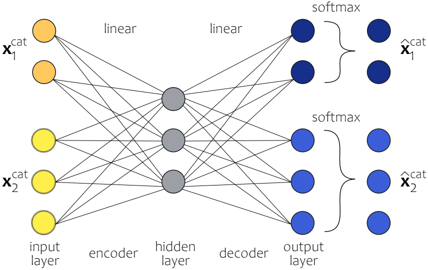
- Autoencoders for numerical data have been introduced to the actuarial literature by Rentzmann and Wüthrich,
- The authors suggest the architecture:



- The mean square error is used to measure the reconstruction error,
- For denoising autoencoders, two types of corruption processes to distort the input:
  - Gaussian disturbance: For each observation, $i = 1, ..., n$, and each numerical feature in the vector $\boldsymbol{x}_i$, the original input is corrupted by the transformation $x_{j,i} \mapsto \tilde{x}_{j,i} \sim N(x_{j,i}, \sigma^2)$,
  - Masking to zero: For each observation, $i = 1, ..., n$, and a fraction v of numerical features in the vector $\boldsymbol{x}_i$ chosen at random, the original input is corrupted by the transformation $x_{j,i} \mapsto \tilde{x}_{j,i} = 0$.

Autoencoders for categorical data - *Joint AE*

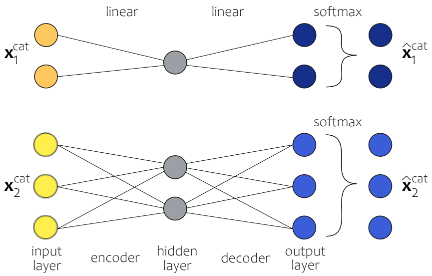
- Autoencoders for categorical data have been less studied in the actuarial literature (e.g. word embeddings),
- We use the architecture:



- The cross-entropy applied across all categories is used to measure the reconstruction error,
- For denoising autoencoders, two types of corruption processes to distort the input:
  - For each observation, $i = 1, ..., n$, and a fraction v of categorical features chosen at random, the original input is corrupted by the transformation:
  - Sampling a new label: $x_{j,i} \mapsto \tilde{x}_{j,i} \sim \hat{F}_{x_j}$ and one-hot encoded with $\tilde{x}_{j,i} \mapsto \tilde{\boldsymbol{x}}^{cat}_{j,i}$, where $\hat{F}_{x_j}$ is the empirical distribution of the feature x_j in the data set,
  - Masking to zero: $\boldsymbol{x}^{cat}_{j,i} \mapsto \tilde{\boldsymbol{x}}^{cat}_{j,i} = \mathbf{0}'$, where $\mathbf{0}$ is a vector of zeros.

Autoencoders for categorical data - *Separate AEs*

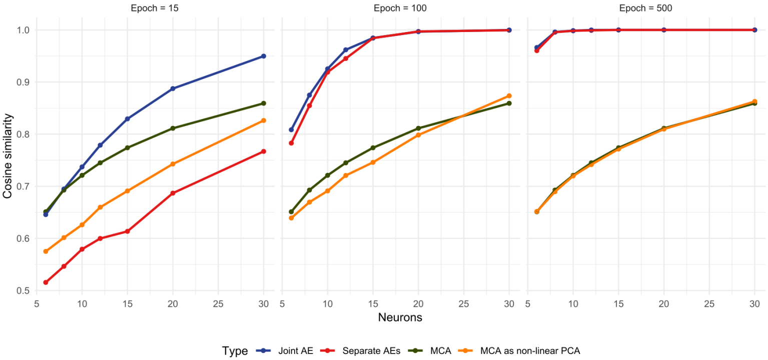
- Since categorical features may have many labels, we could also be interested in using the architecture:



- In this approach, we build separate autoencoders for each categorical feature,
- There are some obvious disadvantages but the approach matches the idea behind pre-processing of categorical features with entity embeddings when we train neural networks for supervised problems.

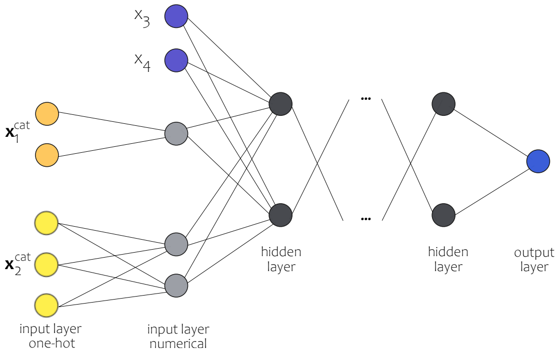
Experiment 1

- We use *freMTPL2freq* data with 6 categorical features,
- We investigate the performance of autoencoders for the categorical features by measuring cosine similarity,
- The results show that our approach is superior to the others in terms of its ability to reconstruct the input from a lower dimensional representation.



Neural networks for supervised problems - A1

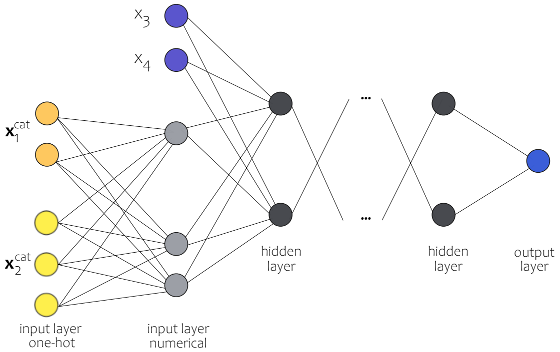
- The dominant approach in the actuarial science for training neural networks with mixed categorical and numerical features for supervised learning problems is to use the architecture (promoted by Richman and Wüthrich):



- We learn entity embedings for a numerical representation of the categorical features and each category has its own seperate embedding,
- All weights of the neural network are initialized by sampling random values from a uniform distribution, Xavier initialization, and the bias terms are initialized with zero.

Neural networks for supervised problems - A2

- Our main goal is to challenge the current approach with the new architecture and the new training algorithm:



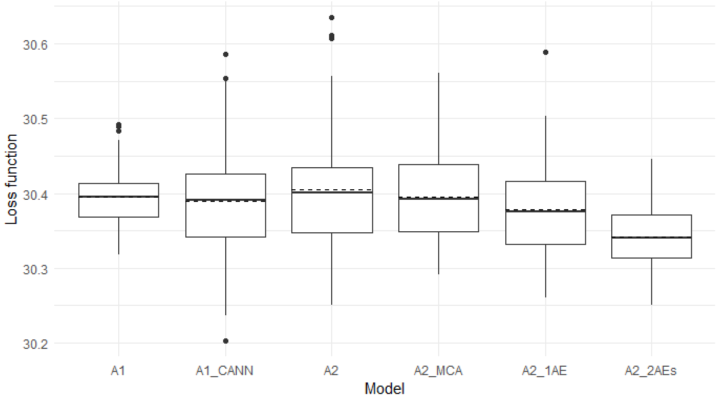
- We learn one embedding jointly for all categorical features,
- We initialize the weights and the bias terms of the neural network with weights and bias terms from denoising autoencoders built for the categorical and numerical features - *stacked autoencoders* in machine learning by Bengio et.al.,
- We have to modify the weights of the representation of the categorical features to have the representation on the same scale where the numerical features live.

Experiment 2

- We use *freMTPL2freq* data with 6 categorical features and 3 continuous features. The response is the number of claims,
- We train neural networks with 3 hidden layers built upon the layer where the categorical features are concatenated with the numerical features,
- We use different architectures and different training algorithms,
- In the first step, we identify the best hyperparameters for our neural networks by investigating a set of possible hyperparameters and evaluating the CV Poisson loss on test sets,
- Next, for the best hyperparameters, we train the neural networks 100 times.

Experiment 2

- Poisson loss functions on a test set:



- The best architectures A1_2AEs use over-complete/denoising autoencoders for pre-training the two layers.

Experiment 2

- Poisson loss functions on a test set:

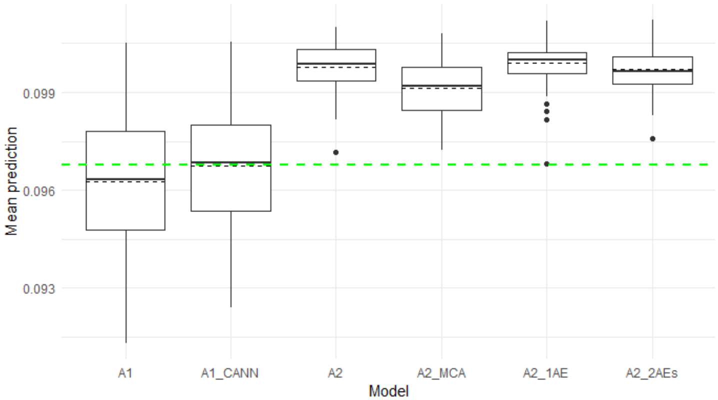
Statistics	A1	A1_CANN	A2	A2_MCA	A2_1AE	A2_2AEs
q0.05	30.3362	30.2875	30.3133	30.3142	30.2863	30.2767
q0.25	30.3684	30.3417	30.3481	30.3495	30.3322	30.3138
avg	**30.3950**	**30.3898**	**30.4045**	**30.3958**	**30.3789**	**30.3425**
q0.75	30.4130	30.4261	30.4338	30.4386	30.4157	30.3719
q0.95	30.4688	30.5101	30.5444	30.5104	30.4833	30.4126
sd	0.0384	0.0708	0.0758	0.0597	0.0624	0.0433

If we compare A2_2AEs to A1:

- We achieve a systematic <u>improvement</u> of the Poisson loss order 0.0525,
- The distribution of the loss is shifted to the left.

Experiment 2

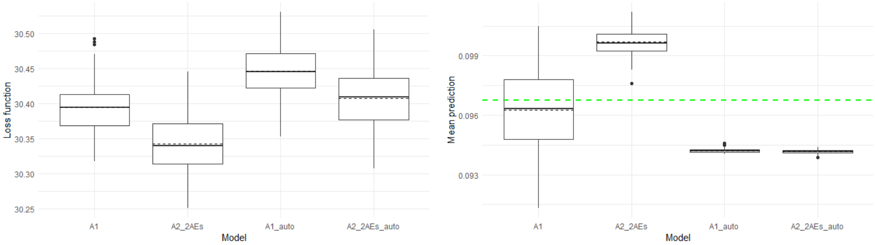
- Global balance property on a test set:



- We observe a slight downward bias for A1 and an upward bias for A2.

Experiment 2

- Poisson loss functions and global balance property for autocalibrated predictors on a test set:



- We can correct the bias and keep a better predictive power of A2_2AEs.

Conclusion

- We present a new approach to modelling categorical and numerical features for neural networks which results in a better predictive power of the trained model,
- We demonstrate that our approach is beneficial for non-life insurance data,
- To use the most important information from multi-dimensional categorial features for efficient pre-processing in hidden layers of a neural network, a new architecture of a neural network is proposed with a joint embedding learned for all categorical features,
- We implement over-complete/denoising autoencoders to derive initial representations of features (neurons) which are used as efficient inputs to hidden layers of a neural network,
- Simple linear autoencoders (PCA, MCA) are not sufficient for pre-training layers and we have to rely on neural networks for building over-complete/denoising autoencoders,
- Optimization of the dimension of the representation of categorical features is simpler,

But

- Bias of the predictions should be controlled,
- We partially lose interpretation of the joint emebedding of categorical features due to higher dimension of the representation.

Conclusion

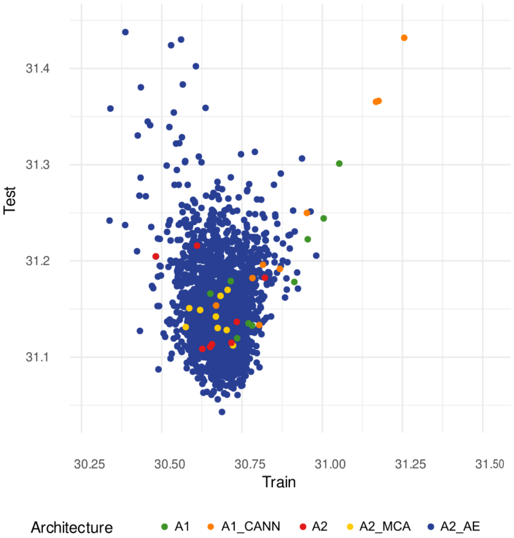
For more details, please see the paper:

- Ł. Delong, A. Kozak, 2023, *The use of autoencoders for training neural networks with mixed categorical and numerical features*, ASTIN Bulletin: The Journal of the IAA, volume 53, issue 2, pp. 213 - 232.

Thank you very much.

Łukasz Delong
SGH Warsaw School of Economics
(from 1st October – University of Warsaw, Faculty of Economic Sciences)
E-mail: ldelong@wp.pl
Homepage: www.lukaszdelong.pl

Experiment 2 - optimization of hyperparameters



Experiment 3

- The optimization of dimensions of entity embeddings for categorical features is simplified as we only have one hyperparameter to fine-tune: