Japanese Journal of Statistics and Data Science
https://doi.org/10.1007/s42081-024-00243-4

ORIGINAL PAPER

Risk and Statistics in Actuarial Science

High-cardinality categorical covariates in network regressions

Ronald Richman[1] · **Mario V. Wüthrich**[2]

Received: 23 August 2023 / Revised: 8 December 2023 / Accepted: 12 January 2024

Abstract
High-cardinality (nominal) categorical covariates are challenging in regression modeling, because they lead to high-dimensional models. For example, in generalized linear models (GLMs), categorical covariates can be implemented by dummy coding which results in high-dimensional regression parameters for high-cardinality categorical covariates. It is difficult to find the correct structure of interactions in high-cardinality covariates, and such high-dimensional models are prone to over-fitting. Various regularization strategies can be applied to prevent over-fitting. In neural network regressions, a popular way of dealing with categorical covariates is entity embedding, and, typically, over-fitting is taken care of by exploiting early stopping strategies. In case of high-cardinality categorical covariates, this often leads to a very early stopping, resulting in a poor predictive model. Building on Avanzi et al. (ASTIN Bull, 2024), we introduce new versions of random effects entity embedding of categorical covariates. In particular, having a hierarchical structure in the categorical covariates, we propose a recurrent neural network architecture and a Transformer architecture, respectively, for random-effects entity embedding that give us very accurate regression models.

Keywords Categorical covariates · Categorical features · Nominal features · High-cardinality features · Entity embedding · Embedding layer · Random-effects model · Neural network · Recurrent neural network · Attention layer · Transformer · Regularization · Ridge regularization · Variational inference · Gaussian mean field posterior

✉ Mario V. Wüthrich
mario.wuethrich@math.ethz.ch

Ronald Richman
ronaldrichman@gmail.com

[1] Old Mutual Insure and University of the Witwatersrand, Johannesburg, South Africa

[2] Department of Mathematics, RiskLab, ETH Zurich, Zurich, Switzerland

Published online: 25 February 2024

1 Introduction

Dealing with nominal (unordered) categorical covariates in regression modeling is generally a difficult task, especially, if these nominal categorical covariates have many levels—called high-cardinality covariates—and if some of these levels only have sparse observations. In addition, different categorical covariates can have a hierarchical structure, e.g., in car insurance pricing we may have information about 'vehicle brand'—'vehicle model'—'vehicle detail'. This leads to a natural thinning of observations in each generation of the hierarchy. There is a recent literature on dealing with high-cardinality categorical covariates, potentially having a hierarchical structure. We briefly review this literature and we explain our novel contribution to this literature.

A common way to integrate categorical covariates into neural network regression models is the approach of entity embedding; we refer to Brébisson et al. (2015), Guo and Berkhahn (2016), Richman (2021a, b) and Schelldorfer and Wüthrich (2019). Entity embedding is inspired by natural language processing (NLP) where the corpus of words is embedded into a low-dimensional Euclidean space, so that proximity of words in this low-dimensional Euclidean space reflects similarity in their meanings; see Bengio et al. (2013, 2003, 2006). This entity embedding approach does not take care of sparse levels nor of a hierarchical structure, and the goal of this work is to discuss these two issues. Delong and Kozak (2023) exploit pre-training of entity embeddings using an auto-encoder, and they empirically show that this pre-training leads to better predictive performance. Campo and Antonio (2023) use clustering techniques for pre-processing high-cardinality hierarchical categorical covariates. Both of these two approaches are unsupervised learning methods, because they pre-process the categorical covariates before considering the response variables in a regression model, and the objective function is either a similarity measure (for clustering) or a reconstruction loss (for auto-encoding).

In a supervised learning approach, Campo and Antonio (2023) propose a generalized linear mixed model (GLMM) to implement hierarchical categorical covariates with random effects which are inferred using Bayesian credibility theory; we also refer to Chapter 6 of Bühlmann and Gisler (2005) for Bayesian credibility theory. A random-effects proposal, called GLMMNet, is also considered in Simchoni and Rosset (2022) and Avanzi et al. (2024) for modeling high-cardinality categorical covariates (non-hierarchical) in a neural network regression framework. Since in a non-linear regression model, posterior distributions cannot be calculated explicitly, Avanzi et al. (2024) exploit the method of variational inference for model fitting. The work of Avanzi et al. (2024) builds the starting point of our proposal.

This paper makes the following contributions to the literature. First, the model architecture considered in Simchoni and Rosset (2022) and Avanzi et al. (2024) uses a random-effects entity embedding for high-cardinality categorical covariates. This entity embedding is concatenated with the last hidden layer of a feed-forward neural network that processes the continuous covariates, i.e., the high-cardinality categorical covariates are only integrated into the last hidden layer of the neural network; see Fig. 1 (lhs). The advantage of that proposal is that it maintains interpretability in the categorical covariates (not the continuous ones). The deficiency of that approach is that these categorical covariates cannot interact in a non-trivial way with the other (con-

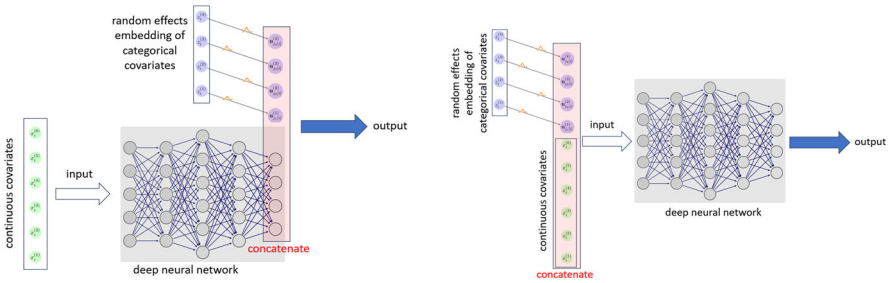

Fig. 1 (lhs) Random-effects entity embedding of Avanzi et al. (2024); (rhs) our proposed random-effects entity embedding architecture

tinuous) covariates (by propagating through the feed-forward neural network layers). We modify this point by changing the network architecture, so that the random-effects embedded categorical covariates can propagate through the network, this is illustrated in Fig. 1 (rhs), and we also change the embeddings from one dimension to higher dimensions allowing for more complex interactions within the network. Our example shows that if we have non-trivial interactions between categorical and continuous covariates, it is necessary to have this bigger modeling complexity to receive good predictive models. Furthermore, we discuss implementation of the random-effects entity embedding, so that it properly scales w.r.t. the observed case weights, and we discuss training of this network architecture. This is done either by weighted L^2-regularization (ridge regularization) or by variational inference of the high-cardinality entity embedding using a Gaussian mean field approximation. We compare the two regularization methods; this is done in Sect. 2. We show that the weighted L^2-regularized version can be seen as a first-order Taylor approximation to the Gaussian mean field variational inference solution; in particular, it involves less hyperparameters and is more easy to train at providing comparably good predictive models. This is verified in Sect. 4 where we study a data example.

Our second contribution extends categorical random-effects entity embedding to a hierarchical structure. The classical hierarchical credibility model has been studied by Jewell (1975) and Bühlmann and Jewell (1987); see also Chapter 6 of Bühlmann and Gisler (2005). This hierarchical credibility model has been extended by Campo and Antonio (2023) to a GLM version having multi-level risk factor random effects, called GLMM. Estimation of this GLMM can be done by the iterative scheme of Ohlsson (2008) within Tweedie's family of distributions using the log-link, or by numerical integration and approximation of intractable likelihoods. We extend the GLMM of Campo and Antonio (2023) by considering multi-dimensional multi-level risk factors random-effects embedding which is regularized according to its hierarchical structure. These multi-level risk factors embeddings then enter a neural network architecture, which extends the model considered in Sect. 2. We relate this architecture to the non-hierarchical one of Sect. 2, and we conclude that in both modeling approaches, we receive the same predictive model, because neural networks can accommodate affine transformations of inputs; this is discussed in Sect. 3.

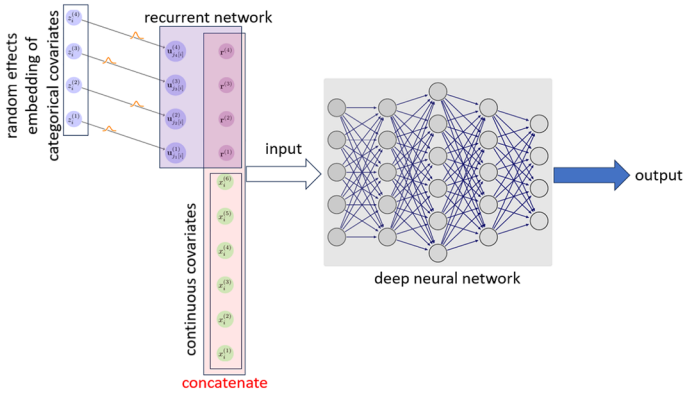

Fig. 2 Hierarchical random-effects entity embedding processed by a recurrent neural network (RNN) layer before concatenating with the continuous covariates

Our third contribution, also presented in Sect. 3, is to interpret hierarchical categorical covariates as a time-series, as hierarchical categorical covariates have a tree structure that is similar to time-series. The main idea then is to understand hierarchical embeddings as step-wise refinements across the generations of the hierarchy. Having this interpretation, it is natural to process hierarchical categorical covariates' embeddings by a recurrent neural network (RNN) layer to learn a new representation before concatenating them with the continuous covariates; this is illustrated in Fig. 2. In our example, we observe that benefiting from the hierarchical structure, and processing this through an RNN layer, improves model accuracy compared to the random-effects entity embedding models presented in Sect. 2. We implement this approach using again weighted L^2-regularization of the high-cardinality hierarchical categorical covariates to prevent over-fitting. Finally, we replace the RNN layer in Fig. 2 by a Transformer layer which nowadays is the most powerful way of dealing with time-series data; see Vaswani et al. (2017). It turns out that this Transformer specification will be the model closest to the true data generating model.

Organization. In the next section, we discuss random-effects entity embedding of high-cardinality categorical covariates, and we show how these models can be fitted to data assuming a neural network regression architecture. In Sect. 3, we study the hierarchical categorical covariates case. In analogy to time-series, we discuss recurrent neural network and Transformer processing, respectively, of these hierarchical categorical covariates. Section 4 presents a data example, where model accuracy of all proposed models is studied. Finally, conclusion is drawn in Sect. 5.

2 Regularization of categorical entity embedding

We introduce entity embedding of high-cardinality (nominal) categorical covariates. Learning these entity embeddings uses regularization with more sparse levels receiving stronger regularization. We show in this section how this intuitive behavior is obtained within a random-effects entity embedding context.

2.1 Random-effects entity embedding

We start by considering one categorical covariate $z \in A = \{a_1, \ldots, a_q\}$ that takes q different levels a_j from set A; the extension to multiple categorical covariates is presented in Sect. 2.7, below. One-hot encoding of this categorical covariate z gives us a q-dimensional representation

$$z \;\mapsto\; \left(\mathbb{1}_{\{z=a_1\}}, \ldots, \mathbb{1}_{\{z=a_q\}}\right)^\top \;\in\; \{0,1\}^q.$$

These are the q basis vectors of the Euclidean space $\mathbb{R}^q$. Compared to Avanzi et al. (2024), we extend the one-hot encoding to a multi-dimensional entity embedding. This requires the choice of an *embedding dimension* $b \in \mathbb{N}$ and of an *embedding matrix* $\mathbf{U} \in \mathbb{R}^{b \times q}$. We then consider the b-dimensional entity embedding map

$$z \in A \;\mapsto\; \boldsymbol{e}_{\mathbf{U}}(z) = \mathbf{U}\left(\mathbb{1}_{\{z=a_1\}}, \ldots, \mathbb{1}_{\{z=a_q\}}\right)^\top \;\in\; \mathbb{R}^b. \tag{2.1}$$

Our main goal is to learn an optimal embedding matrix $\mathbf{U} \in \mathbb{R}^{b \times q}$ for the prediction problem to be solved, in particular, similarity in response behavior of different levels z and $z' \in A$ should be reflected in proximity in embeddings $\boldsymbol{e}_{\mathbf{U}}(z)$ and $\boldsymbol{e}_{\mathbf{U}}(z') \in \mathbb{R}^b$. Note that $b = 1$ reflects the classical encoding in GLMs up to the choice of the reference level (to turn one-hot encoding into dummy coding).

In a next step, we concatenate this embedding $\boldsymbol{e}_{\mathbf{U}}(z) \in \mathbb{R}^b$ of the categorical covariate $z \in A$ with the remaining (real-valued) covariates $\boldsymbol{x} \in \mathbb{R}^{b_0}$; this gives us a feature engineered new real-valued tabular covariate

$$(\boldsymbol{x}, z) \;\mapsto\; (\boldsymbol{x}, \boldsymbol{e}_{\mathbf{U}}(z)) \;\in\; \mathbb{R}^{b_0+b}; \tag{2.2}$$

we also refer to Fig. 1 (rhs). For a given embedding matrix $\mathbf{U} \in \mathbb{R}^{b \times q}$, we receive data sample

$$\mathcal{D}_{\mathbf{U}} = \Big(Y_i, (\boldsymbol{x}_i, \boldsymbol{e}_{\mathbf{U}}(z_i)), v_i\Big)_{i=1}^n;$$

the lower indices $i \in \{1, \ldots, n\}$ denote the different instances, Y_i are the responses of covariates $(\boldsymbol{x}_i, z_i) \in \mathbb{R}^{b_0} \times A$, and $v_i > 0$ are the (given) case weights (exposures) of the instances $i \in \{1, \ldots, n\}$.

We then select a neural network $\mathrm{NN}_{\boldsymbol{\vartheta}}$ of a given architecture and with network parameter $\boldsymbol{\vartheta}$ to model this data $\mathcal{D}_{\mathbf{U}}$; more modeling details of the neural network $\mathrm{NN}_{\boldsymbol{\vartheta}}$ are provided formula (2.23) in Sect. 2.6. This neural network maps inputs to outputs that serve as predictions of the responses Y_i, that is, the predictions are given by the mapping

$$(\boldsymbol{x}_i, \boldsymbol{e}_{\mathbf{U}}(z_i)) \;\mapsto\; \mathrm{NN}_{\boldsymbol{\vartheta}}\left(\boldsymbol{x}_i, \boldsymbol{e}_{\mathbf{U}}(z_i)\right);$$

this is illustrated in Fig. 1 (rhs). The optimal neural network $\mathrm{NN}_{\boldsymbol{\vartheta}}$ of a given architecture is found by minimizing a pre-selected loss function L over the network parameter $\boldsymbol{\vartheta}$,

i.e., we aim at solving

$$\arg\min_{\vartheta} \sum_{i=1}^{n} v_i \, L\Big(Y_i, \mathrm{NN}_{\vartheta}\left(\boldsymbol{x}_i, \boldsymbol{e}_{\mathbf{U}}(z_i)\right)\Big).$$

This optimization assumes that we know the embedding matrix $\mathbf{U} \in \mathbb{R}^{b \times q}$. A full optimal parameter search optimizes over the embedding matrix, too, solving

$$\arg\min_{\vartheta, \mathbf{U}} \sum_{i=1}^{n} v_i \, L\Big(Y_i, \mathrm{NN}_{\vartheta}\left(\boldsymbol{x}_i, \boldsymbol{e}_{\mathbf{U}}(z_i)\right)\Big). \tag{2.3}$$

This optimization (2.3) is called no pooling in Antonio and Zhang (2014) and Avanzi et al. (2024), because it does not impose any restrictions in parameter estimations of categorical covariates. The other extreme case is to set $\mathbf{U} = \mathbf{0}$, called complete pooling, which does not consider the categorical covariates z_i in the regression model at all. Random-effects entity embedding is between these two extreme cases choosing a prior distribution π on $\mathbf{U}$ that regularizes parameter estimation of categorical covariates.

Note that this framework (2.3) can easily be extended to multiple categorical covariates. All categorical covariates are embedded as in (2.1), but accounting for their numbers of levels and they may also have different embedding dimensions. These embeddings are then concatenated as in (2.2) to receive a new tabular covariate that collects the real-valued covariates $\boldsymbol{x} \in \mathbb{R}^{b_0}$ and all entity embeddings of the categorical covariates, more details are given in Sect. 2.7.

2.2 Random-effects embedding within the exponential dispersion family

In all what follows, we assume that Y_i follows a member of the exponential dispersion family (EDF) with cumulant function κ, canonical link $h = (\kappa')^{-1}$ and constant dispersion parameter $\varphi > 0$. This implies that response Y_i has EDF density

$$Y_i \;\sim\; f(y) = \exp\left\{\frac{y h(\mu_i) - \kappa(h(\mu_i))}{\varphi / v_i} + a(y, \varphi, v_i)\right\},$$

for μ_i being the expected value, and $a(\cdot)$ a normalizing function, so that the density $f(y)$ integrates to 1 w.r.t. the given σ-finite measure $\nu(y)$ on $\mathbb{R}$; we refer to Chapter 2 of Wüthrich and Merz (2023). In that case, it is natural to choose the deviance loss function for L. Optimization (2.3) under this deviance loss function choice is equivalent to maximizing the corresponding log-likelihood function, for independent EDF responses $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^\top$ given by

$$\ell_{\boldsymbol{Y}}(\vartheta \,|\, \mathbf{U}) = \log f_{\vartheta}\left(\boldsymbol{Y} \,|\, \mathbf{U}\right) \;\propto\; \sum_{i=1}^{n} \frac{Y_i \, h\left(\mathrm{NN}_{\vartheta}(\boldsymbol{x}_i, \boldsymbol{e}_{\mathbf{U}}(z_i))\right) - \kappa\left(h\left(\mathrm{NN}_{\vartheta}(\boldsymbol{x}_i, \boldsymbol{e}_{\mathbf{U}}(z_i))\right)\right)}{\varphi / v_i},$$

where the proportionality sign $\propto$ indicates that we have dropped all terms that do not depend on $\boldsymbol{\vartheta}$ and $\mathbf{U}$. The density $f_{\boldsymbol{\vartheta}}(\boldsymbol{Y}|\mathbf{U})$ considers the distribution of the responses $\boldsymbol{Y}$ for given random effects $\mathbf{U}$ and given network parameter $\boldsymbol{\vartheta}$, which determine the means by $\mu_i = \mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, \boldsymbol{e}_{\mathbf{U}}(z_i))$. For random-effects modeling, we now need to choose a prior density π for $\mathbf{U}$. This then gives the joint log-likelihood function of $(\boldsymbol{Y}, \mathbf{U})$

$$\begin{aligned}\log f_{\boldsymbol{\vartheta}}(\boldsymbol{Y}, \mathbf{U}) &= \ell_{\boldsymbol{Y}}(\boldsymbol{\vartheta}|\mathbf{U}) + \log \pi(\mathbf{U}) \\ &\propto \left[\sum_{i=1}^{n} \frac{Y_i\, h\left(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, \boldsymbol{e}_{\mathbf{U}}(z_i))\right) - \kappa\left(h\left(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, \boldsymbol{e}_{\mathbf{U}}(z_i))\right)\right)}{\varphi/v_i} \right] + \log \pi(\mathbf{U}).\end{aligned} \tag{2.4}$$

In notation (2.4), we omit that π may involve more parameters that need to be determined; we come back to this below. Log-likelihood function (2.4) is also called *complete log-likelihood*, because it assumes that we have observed the responses $\boldsymbol{Y}$ *and* the (latent) random effects $\mathbf{U}$. The general problem in this field now is to solve the estimation problem under unobserved random effects $\mathbf{U}$, and estimate those. We are going to discuss different estimation methods in Sects. 2.3–2.5.

2.3 Maximal a posterior estimator

Technically, the most basic way to solve the above estimation problem related to the joint log-likelihood (2.4) is to determine the maximal a posteriori (MAP) estimator of $\mathbf{U}$, jointly with the maximum-likelihood estimator (MLE) of the network parameter $\boldsymbol{\vartheta}$, given $\mathbf{U}$. This is obtained by maximizing (2.4) jointly in $\boldsymbol{\vartheta}$ and $\mathbf{U}$, for given $\boldsymbol{Y}$. We describe this under a very specific prior density π choice, because this leads to the classical ridge regularization of Tikhonov (1943); we also refer to Hastie et al. (2015).

The matrix $\mathbf{U} = \left[\mathbf{u}_1, \ldots, \mathbf{u}_q\right] \in \mathbb{R}^{b \times q}$ is a random matrix under π with column vectors $\mathbf{u}_j \in \mathbb{R}^b$. For categorical covariates z_i of instances $1 \le i \le n$, we are going to change notation (2.1), because this is going to be more convenient in the sequel. Each categorical covariate z_i selects exactly one of these column vectors of $\mathbf{U}$, that is

$$\mathbf{u}_{j[i]} = \boldsymbol{e}_{\mathbf{U}}(z_i) = \mathbf{U}\left(\mathbb{1}_{\{z_i = a_1\}}, \ldots, \mathbb{1}_{\{z_i = a_q\}}\right)^\top \in \mathbb{R}^b, \tag{2.5}$$

where we adopt the notation $j[i]$ from Avanzi et al. (2024) saying that covariate z_i takes level $a_{j[i]}$

$$j[i] = \left\{j' \in \{1, \ldots, q\} \text{ with } z_i = a_{j'}\right\} = \left(\mathbb{1}_{\{z_i = a_1\}}, \ldots, \mathbb{1}_{\{z_i = a_q\}}\right)(1, \ldots, q)^\top. \tag{2.6}$$

Next, we assume that the column vectors $(\mathbf{u}_j)_{j=1}^q$ are i.i.d. with centered Gaussian prior distributions having i.i.d. components with variance τ^2. Thus, all elements of $\mathbf{U}$

are i.i.d. centered Gaussian with identical variance $\tau^2 > 0$. This assumption allows us to rewrite the joint log-likelihood of $(\boldsymbol{Y}, \mathbf{U})$, given in (2.4), as follows:

$$\begin{aligned}
&\log f_{\boldsymbol{\vartheta}}(\boldsymbol{Y}, \mathbf{U}) \\
&\quad \propto \sum_{j'=1}^{q} \left(\left[\sum_{i=1}^{n} \mathbb{1}_{\{j[i]=j'\}} \frac{Y_i\, h\left(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, \mathbf{u}_{j'})\right) - \kappa\left(h\left(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, \mathbf{u}_{j'})\right)\right)}{\varphi / v_i} \right] - \frac{1}{2\tau^2} \left\| \mathbf{u}_{j'} \right\|^2 \right) \\
&\quad = \frac{1}{\varphi} \sum_{j'=1}^{q} \sum_{i=1}^{n} \mathbb{1}_{\{j[i]=j'\}}\, v_i \left[Y_i\, h\left(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, \mathbf{u}_{j'})\right) - \kappa\left(h\left(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, \mathbf{u}_{j'})\right)\right) - \frac{1}{w_{j'}} \frac{\varphi}{2\tau^2} \left\| \mathbf{u}_{j'} \right\|^2 \right] \\
&\quad = \frac{1}{\varphi} \sum_{i=1}^{n} v_i \left[Y_i\, h\left(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, \mathbf{u}_{j[i]})\right) - \kappa\left(h\left(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, \mathbf{u}_{j[i]})\right)\right) - \frac{1}{w_{j[i]}} \frac{\varphi}{2\tau^2} \left\| \mathbf{u}_{j[i]} \right\|^2 \right], \qquad (2.7)
\end{aligned}$$

with aggregated case weights for $1 \le j' \le q$

$$w_{j'} = \sum_{i=1}^{n} \mathbb{1}_{\{j[i]=j'\}}\, v_i. \qquad (2.8)$$

The crucial observation from (2.7) is that the prior terms $-\|\mathbf{u}_{j'}\|^2/(2\tau^2)$ scale inversely proportionally to the aggregate case weights $w_{j'}$. This fact provides an instance adapted regularization of the random effects $(\mathbf{u}_j)_{j=1}^{q}$ accounting for the multiplicity of a certain level $a_{j'}$ in the entire data

$$\mathcal{D} = \Big(Y_i, (\boldsymbol{x}_i, z_i), v_i \Big)_{i=1}^{n}.$$

Thus, levels $a_{j'}$ with many observations $z_i = a_{j'}$, i.e., $j[i] = j'$, receive only a small influence from the prior distribution, whereas sparse levels are regularized more strongly by the prior Gaussian distribution. The MAP is then given by

$$\left(\widehat{\boldsymbol{\vartheta}}^{\mathrm{MAP}}, \widehat{\mathbf{U}}^{\mathrm{MAP}} \right) = \underset{\boldsymbol{\vartheta}, \mathbf{U}}{\arg\max}\ \log f_{\boldsymbol{\vartheta}}(\boldsymbol{Y}, \mathbf{U}). \qquad (2.9)$$

From (2.7), we also observe that the prior π regularizes with a regularization parameter $\lambda = \varphi/(2\tau^2) > 0$. In MAP estimation, this regularization parameter is considered as a hyperparameter which is either given a priori or which is determined using cross-validation. Under a neural network function $\mathrm{NN}_{\boldsymbol{\vartheta}}$ for the regression model, the specific choice of $\lambda > 0$ is less relevant; this is going to be discussed in detail in Sect. 2.6.

2.4 Variational Bayesian estimation

We exploit a Bayesian posterior expectation approach in this section. The natural candidate for MLE is to consider the marginal log-likelihood of the observations $\boldsymbol{Y}$

$$\ell_{\boldsymbol{Y}}(\boldsymbol{\vartheta}) = \log f_{\boldsymbol{\vartheta}}(\boldsymbol{Y}) = \log \int f_{\boldsymbol{\vartheta}}(\boldsymbol{Y}, \mathbf{U})\, \mathrm{d}\mathbf{U} = \log \int \exp\left\{ \ell_{\boldsymbol{Y}}(\boldsymbol{\vartheta}|\mathbf{U}) + \log \pi(\mathbf{U}) \right\} \mathrm{d}\mathbf{U}. \qquad (2.10)$$

This integration makes the individual instances $1 \leq i \leq n$ dependent. Typically, the integral in (2.10) cannot be calculated explicitly which makes it infeasible to go along this direction. The idea is to maximize a lower bound of (2.10) to receive an approximate solution to the MLE of $\boldsymbol{\vartheta}$.

We introduce the posterior density of $\mathbf{U}$, given observations $\boldsymbol{Y}$. Using Bayes' rule, we have

$$\pi_{\boldsymbol{\vartheta}}(\mathbf{U}|\boldsymbol{Y}) = \frac{f_{\boldsymbol{\vartheta}}(\boldsymbol{Y},\mathbf{U})}{f_{\boldsymbol{\vartheta}}(\boldsymbol{Y})} = \frac{f_{\boldsymbol{\vartheta}}(\boldsymbol{Y}|\mathbf{U})\,\pi(\mathbf{U})}{\int f_{\boldsymbol{\vartheta}}(\boldsymbol{Y},\mathbf{U})\,\mathrm{d}\mathbf{U}}.$$

This posterior density is also intractable as it involves the same integral as in (2.10). In variational Bayesian (VB) estimation, we approximate this posterior density. The approximating density is called *variational density*. Assume $p_\psi(\mathbf{U})$ are candidate densities to approximate the posterior density $\pi_{\boldsymbol{\vartheta}}(\mathbf{U}|\boldsymbol{Y})$. These candidates are parametrized by ψ. The accuracy of the approximation is measured by the Kullback–Leibler (KL) divergence, denoted by $D_{\mathrm{KL}}(\cdot\|\cdot)$. The optimal approximation within the candidate densities is obtained by

$$\begin{aligned}\psi^* &= \arg\min_{\psi} D_{\mathrm{KL}}\left(p_\psi(\mathbf{U})\,\|\,\pi_{\boldsymbol{\vartheta}}(\mathbf{U}|\boldsymbol{Y})\right)\\ &= \arg\min_{\psi} \int p_\psi(\mathbf{U}) \log\left(\frac{p_\psi(\mathbf{U})}{\pi_{\boldsymbol{\vartheta}}(\mathbf{U}|\boldsymbol{Y})}\right)\mathrm{d}\mathbf{U}. \end{aligned} \tag{2.11}$$

Naturally, this optimal parameter $\psi^* = \psi^*(\boldsymbol{Y},\boldsymbol{\vartheta})$ is a function of the observations $\boldsymbol{Y}$ and the network parameter $\boldsymbol{\vartheta}$. Since we do not know the network parameter $\boldsymbol{\vartheta}$, we need to jointly estimate $\boldsymbol{\vartheta}$ and ψ to get a good approximation to the optimal true model.

Using any variational density $p_\psi(\mathbf{U})$, we can rewrite the marginal log-likelihood (2.10) as follows; see, e.g., Wüthrich and Merz (2023, Lemma 11.19):

$$\ell_{\boldsymbol{Y}}(\boldsymbol{\vartheta}) = \mathcal{E}(\psi|\boldsymbol{Y},\boldsymbol{\vartheta}) + D_{\mathrm{KL}}\left(p_\psi(\mathbf{U})\,\|\,\pi_{\boldsymbol{\vartheta}}(\mathbf{U}|\boldsymbol{Y})\right), \tag{2.12}$$

with the evidence lower bound (ELBO) defined by

$$\mathcal{E}(\psi|\boldsymbol{Y},\boldsymbol{\vartheta}) = \int p_\psi(\mathbf{U}) \log\left(\frac{f_{\boldsymbol{\vartheta}}(\boldsymbol{Y},\mathbf{U})}{p_\psi(\mathbf{U})}\right)\mathrm{d}\mathbf{U} = \mathbb{E}_{\mathbf{U}\sim p_\psi}\left[\log\left(\frac{f_{\boldsymbol{\vartheta}}(\boldsymbol{Y},\mathbf{U})}{p_\psi(\mathbf{U})}\right)\right].$$

Using (2.12) and the positivity of the KL divergence, we have lower bound

$$\ell_{\boldsymbol{Y}}(\boldsymbol{\vartheta}) \geq \max_{\psi} \mathcal{E}(\psi|\boldsymbol{Y},\boldsymbol{\vartheta}), \tag{2.13}$$

and, maximizing this ELBO in ψ is equivalent to minimizing the KL divergence given in (2.11) in ψ, because the left-hand side of (2.12) is independent of ψ.

In view of (2.13), the VB inference approach now seems clear. Namely, we aim at maximizing the ELBO for receiving an approximate solution to the MLE $\widehat{\boldsymbol{\vartheta}}^{\mathrm{MLE}}$ of the network parameter $\boldsymbol{\vartheta}$. The lower bound (2.13) suggests that we can alternate

maximizations of ψ (for given $\boldsymbol{\vartheta}$) and $\boldsymbol{\vartheta}$ (for given ψ), this will approximate the maximal log-likelihood $\ell_{\boldsymbol{Y}}(\widehat{\boldsymbol{\vartheta}}^{\mathrm{MLE}})$ from below, or, more precisely, we find (at least) a local maximum of the lower bound to the objective function. Usually, one is satisfied by such an approximate solution, as any better solution is intractable. For this reason, we further exploit the ELBO. It satisfies

$$\mathcal{E}(\psi|\boldsymbol{Y},\boldsymbol{\vartheta}) \;=\; -D_{\mathrm{KL}}\left(p_\psi(\mathbf{U})\,\|\pi(\mathbf{U})\right) + \mathbb{E}_{\mathbf{U}\sim p_\psi}\Big[\log f_{\boldsymbol{\vartheta}}(\boldsymbol{Y}|\mathbf{U})\Big]. \tag{2.14}$$

Maximizing the ELBO in (2.13) means that we maximize the expected data log-likelihood, last term in (2.14), under an approximate candidate posterior $\mathbf{U}\sim p_\psi$ of the true posterior distribution, this is also called reconstruction term, see, e.g., Odaibo (2019). The negative KL divergence in (2.14) then acts as a regularizer that ensures that the approximate candidate posterior $\mathbf{U}\sim p_\psi$ reflects the prior assumptions π on the latent random effects $\mathbf{U}$.

For the prior density π, we choose i.i.d. centered Gaussians with variance $\tau^2>0$ for all components of the embedding matrix $\mathbf{U}$, see also (2.7), and for the variational densities $p_\psi(\mathbf{U})$ we choose the Gaussian mean field family, meaning that all components of $\mathbf{U}$ are independent and Gaussian with mean parameters $(\nu_{k,j})_{1\le k\le b,1\le j\le q}$ and variance parameters $(\sigma^2_{k,j})_{1\le k\le b,1\le j\le q}$. The advantage of this choice is that the KL divergence in (2.14) takes a very simple form, the disadvantage of this choice is that it cannot capture any posterior dependence in $\mathbf{U}$. This Gaussian mean field choice gives us KL divergence, see, e.g., Wüthrich and Merz (2023, Example 11.20)

$$D_{\mathrm{KL}}\left(p_\psi(\mathbf{U})\,\|\pi(\mathbf{U})\right) = \sum_{k=1}^{b}\sum_{j=1}^{q}\frac{1}{2}\left(-\log\left(\frac{\sigma^2_{k,j}}{\tau^2}\right)-1+\frac{\sigma^2_{k,j}+\nu^2_{k,j}}{\tau^2}\right), \tag{2.15}$$

with $2bq$-dimensional variational density parameter

$$\psi = (\nu_{k,j},\sigma^2_{k,j})_{1\le k\le b,1\le j\le q}. \tag{2.16}$$

Using a Gaussian mean field posterior VB approximation also allows us to apply the reparametrization trick of Kingma and Welling (2019), saying that the column vectors $\mathbf{u}_j\sim\mathcal{N}(\nu_j=(\nu_{k,j})_{1\le k\le b},\Sigma_j=\mathrm{diag}(\sigma^2_{k,j})_{1\le k\le b})$ of $\mathbf{U}$ have the same distributions as

$$\mathbf{u}_j \;\overset{(\mathrm{d})}{=}\; \nu_j+\Sigma_j^{1/2}\boldsymbol{\varepsilon}_j, \tag{2.17}$$

for a b-dimensional standard Gaussian $\boldsymbol{\varepsilon}_j\sim\mathcal{N}(\mathbf{0},\mathrm{diag}(1))$.

Collecting all these assumptions and derivations, and assuming an EDF (2.4) for the responses $\boldsymbol{Y}$, we obtain from (2.14) the ELBO

$$\mathcal{E}(\psi|\boldsymbol{Y},\boldsymbol{\vartheta}) = \sum_{k=1}^{b}\sum_{j=1}^{q}\frac{1}{2}\left(\log\left(\frac{\sigma^2_{k,j}}{\tau^2}\right)+1-\frac{\sigma^2_{k,j}+\nu^2_{k,j}}{\tau^2}\right)$$

$$
\begin{aligned}
&+ \sum_{j'=1}^{q} \mathbb{E}_{\boldsymbol{\varepsilon}_{j'}} \left[\sum_{i=1}^{n} \mathbb{1}_{\{j[i]=j'\}} \right. \\
&\left. \frac{Y_i \, h\left(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, v_{j'} + \Sigma_{j'}^{1/2} \boldsymbol{\varepsilon}_{j'})\right) - \kappa\left(h\left(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, v_{j'} + \Sigma_{j'}^{1/2} \boldsymbol{\varepsilon}_{j'})\right)\right)}{\varphi / v_i} \right] \\
&= \frac{1}{\varphi} \sum_{j'=1}^{q} \sum_{i=1}^{n} \mathbb{1}_{\{j[i]=j'\}} \, v_i \left\{ \frac{1}{w_{j'}} \sum_{k=1}^{b} \frac{\varphi}{2} \left(\log\left(\frac{\sigma_{k,j'}^2}{\tau^2}\right) + 1 - \frac{\sigma_{k,j'}^2 + v_{k,j'}^2}{\tau^2} \right) \right. \\
&\left. + \mathbb{E}_{\boldsymbol{\varepsilon}_{j'}} \left[Y_i \, h\left(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, v_{j'} + \Sigma_{j'}^{1/2} \boldsymbol{\varepsilon}_{j'})\right) - \kappa\left(h\left(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, v_{j'} + \Sigma_{j'}^{1/2} \boldsymbol{\varepsilon}_{j'})\right)\right) \right] \right\},
\end{aligned}
$$

where we recall definition (2.8) of the aggregated case weights $w_{j'}$. The latter expected value is usually replaced by an empirical version. Having a large sample size n, it suffices to simulate one independent Gaussian $\boldsymbol{\varepsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathrm{diag}(1))$ for each instance $1 \le i \le n$. This provides us with an empirical version of the ELBO

$$
\begin{aligned}
\widehat{\mathcal{E}}(\psi \,|\, \boldsymbol{Y}, \boldsymbol{\vartheta}) = \frac{1}{\varphi} \sum_{i=1}^{n} v_i \left\{ \frac{1}{w_{j[i]}} \sum_{k=1}^{b} \frac{\varphi}{2} \left(\log\left(\frac{\sigma_{k,j[i]}^2}{\tau^2}\right) + 1 - \frac{\sigma_{k,j[i]}^2 + v_{k,j[i]}^2}{\tau^2} \right) \right. \\
\left. + Y_i \, h\left(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, v_{j[i]} + \Sigma_{j[i]}^{1/2} \boldsymbol{\varepsilon}_i)\right) - \kappa\left(h\left(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, v_{j[i]} + \Sigma_{j[i]}^{1/2} \boldsymbol{\varepsilon}_i)\right)\right) \right\}.
\end{aligned} \tag{2.18}
$$

Using this empirical ELBO, we solve with gradient descent

$$
\left(\widehat{\boldsymbol{\vartheta}}^{\mathrm{VB}}, \widehat{\psi}^{\mathrm{VB}}\right) = \underset{\boldsymbol{\vartheta}, \psi}{\arg\max} \, \widehat{\mathcal{E}}(\psi \,|\, \boldsymbol{Y}, \boldsymbol{\vartheta}), \tag{2.19}
$$

where $\boldsymbol{\vartheta}$ denotes the network parameter and ψ denotes the variational density parameter of the Gaussian mean field approximation.

We compare the MAP approach (2.7) and the VB inference approach (2.18). These two approaches have in common that the regularization term scales inversely proportionally to the case weights $w_{j'}$, defined in (2.8). This highlights that the influence of the prior density π vanishes if we have many observations for a certain level $a_{j'} \in A$ of the categorical covariate $(z_i)_{i=1}^n$. The MAP approach (2.7) regularizes the latent random effects $\mathbf{u}_{j'}$ directly and regularization scales as $\varphi/(2\tau^2 w_{j'}) = \lambda / w_{j'}$. From this, we see that the specific choices of the dispersion parameter $\varphi > 0$ and the prior uncertainty $\tau^2 > 0$ do not matter, but only their ratio $\lambda = \varphi/(2\tau^2)$ seems important; we come back to this in (2.22) below. This behavior is different in the VB inference approach (2.18). It regularizes the parameters of the random effects (not the random effects themselves) and the specific choices of φ and τ^2 matter, not only their ratio. Finally, in the MAP approach, the random effects $\mathbf{u}_j$ are determined by maximizing (2.7) yielding MAP $\widehat{\mathbf{u}}_j^{\mathrm{MAP}}$; see (2.9). In the VB inference approach, we can estimate these random effects by the approximate posterior means $\widehat{\mathbf{u}}_j^{\mathrm{post}} = \widehat{v}_j^{\mathrm{VB}}$, where these approximate posterior means are obtained from maximization (2.19).

2.5 Ad-hoc random-effects estimation

Comparing the MAP approach (2.7) and the VB inference approach (2.18), there are two essential differences, namely, regularization is different and the random effects are considered differently in the data log-likelihood $\ell_{\boldsymbol{Y}}(\boldsymbol{\vartheta}|\mathbf{U})$. The VB inference approach (2.18) has been received by an empirical version of the ELBO. Using a Taylor expansion $\log(x) \approx (x-1) - (x-1)^2/2$, we can approximate the empirical version of the ELBO as follows:

$$\begin{aligned}\frac{1}{w_{j'}}\sum_{k=1}^{b}\frac{\varphi}{2}\left(\log\left(\frac{\sigma_{k,j'}^2}{\tau^2}\right)+1-\frac{\sigma_{k,j'}^2+\nu_{k,j'}^2}{\tau^2}\right)\\ \approx\ -\frac{1}{w_{j'}}\frac{\varphi}{2\tau^2}\left\|\nu_{j'}\right\|^2\ -\frac{1}{w_{j'}}\frac{\varphi}{4}\sum_{k=1}^{b}\left(\frac{\sigma_{k,j'}^2}{\tau^2}-1\right)^2.\end{aligned}$$

If we plug this Taylor approximation into (2.18), we have

$$\begin{aligned}\widetilde{\mathcal{E}}(\psi|\boldsymbol{Y},\boldsymbol{\vartheta}) = \frac{1}{\varphi}\sum_{i=1}^{n}v_i\Bigg\{-\frac{1}{w_{j[i]}}\frac{\varphi}{2\tau^2}\left\|\nu_{j[i]}\right\|^2-\frac{1}{w_{j[i]}}\frac{\varphi}{4}\sum_{k=1}^{b}\left(\frac{\sigma_{k,j[i]}^2}{\tau^2}-1\right)^2\\ +Y_i\,h\left(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i,\nu_{j[i]}+\Sigma_{j[i]}^{1/2}\boldsymbol{\varepsilon}_i)\right)-\kappa\left(h\left(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i,\nu_{j[i]}+\Sigma_{j[i]}^{1/2}\boldsymbol{\varepsilon}_i)\right)\right)\Bigg\}.\end{aligned} \tag{2.20}$$

For $\sigma_{k,j[i]}^2 \equiv 0$, this precisely gives the MAP approach (2.7) with $\mathbf{u}_{j[i]}$ replaced by $\nu_{j[i]}$. Formula (2.20) randomizes the random-effects case individually by adding noise $\Sigma_{j[i]}\boldsymbol{\varepsilon}_i$, and at the same time, this additional noise term gets regularized through $\Sigma_{j[i]}$ on the first line of (2.20). In this sense, we can interpret the empirical ELBO estimation approach as a double Bayesian model.

In our examples, we also consider optimization of (2.20), providing the ad-hoc estimators

$$\left(\widehat{\boldsymbol{\vartheta}}^{\text{ad-hoc}},\widehat{\psi}^{\text{ad-hoc}}\right) = \underset{\boldsymbol{\vartheta},\psi}{\arg\max}\,\widetilde{\mathcal{E}}(\psi|\boldsymbol{Y},\boldsymbol{\vartheta}). \tag{2.21}$$

2.6 Hyperparameter selection for regularization

We consider the above three regularization methods of the MAP (2.9), VB inference estimator (2.19), and the ad-hoc method (2.21). These involve the dispersion parameter $\varphi > 0$ and the prior Gaussian uncertainty $\tau^2 > 0$. We discuss the selection of these two parameters in this section. For this, we first need to understand how neural networks $\mathrm{NN}_{\boldsymbol{\vartheta}}$ work.

Fully connected feed-forward neural network. We briefly discuss the structure of a neural network $\mathrm{NN}_{\boldsymbol{\vartheta}}$, and for a detailed description, we refer to Wüthrich and

Merz (2023, Chapter 7). A neural network is a mapping

$$\mathrm{NN}_{\boldsymbol{\vartheta}} : \mathbb{R}^{b_0+b} \to \mathbb{R}, \qquad (\boldsymbol{x}, \boldsymbol{e}_{\mathbf{U}}(z)) \mapsto \mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}, \boldsymbol{e}_{\mathbf{U}}(z)),$$

for given embedding matrix $\mathbf{U}$ and embedding dimension $b \in \mathbb{N}$; see (2.2). For the neural network architecture, we choose a fixed depth $d \in \mathbb{N}$, and then we compose d hidden neural network layers $\boldsymbol{\ell}^{(l)} : \mathbb{R}^{r_{l-1}} \to \mathbb{R}^{r_l}$, $1 \le l \le d$, to a mapping

$$(\boldsymbol{x}, \boldsymbol{e}_{\mathbf{U}}(z)) \;\mapsto\; \boldsymbol{\ell}^{(d:1)}(\boldsymbol{x}, \boldsymbol{e}_{\mathbf{U}}(z)) = \left(\boldsymbol{\ell}^{(d)} \circ \cdots \circ \boldsymbol{\ell}^{(1)}\right)(\boldsymbol{x}, \boldsymbol{e}_{\mathbf{U}}(z)) \;\in\; \mathbb{R}^{r_d},$$

for given dimensions $r_l \in \mathbb{N}$, $1 \le l \le d$. The lth hidden layer with $r_l \in \mathbb{N}$ neurons and hyperbolic tangent activation function reads as, $\boldsymbol{x} = (x_1, \ldots, x_{r_{l-1}})^\top \in \mathbb{R}^{r_{l-1}}$

$$\boldsymbol{\ell}^{(l)}(\boldsymbol{x}) = \left(\tanh\left(\vartheta_{1,0}^{(l)} + \sum_{k=1}^{r_{l-1}} \vartheta_{1,k}^{(l)} x_k\right), \ldots, \tanh\left(\vartheta_{r_l,0}^{(l)} + \sum_{k=1}^{r_{l-1}} \vartheta_{r_l,k}^{(l)} x_k\right)\right)^\top \in \mathbb{R}^{r_l}, \tag{2.22}$$

for network weights $\boldsymbol{\vartheta}^{(l)} = (\vartheta_{j,k}^{(l)})_{1\le j\le r_l; 0\le k\le r_{l-1}} \in \mathbb{R}^{r_l(r_{l-1}+1)}$. We initialize $r_0 = b_0 + b$. Finally, because our responses will be positive, we choose the exponential output function, giving us

$$(\boldsymbol{x}, \boldsymbol{e}_{\mathbf{U}}(z)) \;\mapsto\; \mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}, \boldsymbol{e}_{\mathbf{U}}(z)) = \exp\left\{\vartheta_0^{(d+1)} + \sum_{k=1}^{r_d} \vartheta_k^{(d+1)} \ell_k^{(d:1)}(\boldsymbol{x}, \boldsymbol{e}_{\mathbf{U}}(z))\right\}; \tag{2.23}$$

this provides us with additional network weights $\boldsymbol{\vartheta}^{(d+1)} = (\vartheta_k^{(d+1)})_{0\le k\le r_d} \in \mathbb{R}^{r_d+1}$. Collecting all these network weights gives us the network parameter $\boldsymbol{\vartheta} = (\boldsymbol{\vartheta}^{(1)}, \ldots, \boldsymbol{\vartheta}^{(d+1)})$ of $\mathrm{NN}_{\boldsymbol{\vartheta}}$. The reason for explicitly recalling this structure (and not citing to the literature) will become clear in the next paragraph.

Hyperparameter selection. To implement the random-effects estimations (2.9), (2.19) and (2.21), we need to select the hyperparameters $\varphi > 0$ (dispersion parameter) and $\tau^2 > 0$ (degree of information in Gaussian prior π). We are going to argue that we can set $\varphi = 1$, and $\tau^2 > 0$ only needs a specific choice in the VB inference case (2.19) and the ad-hoc case (2.21) for a neural network $\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, \boldsymbol{e}_{\mathbf{U}}(z_i)) = \mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, \mathbf{u}_{j[i]})$, but not in the MAP case (2.9). To see this, we come back to the specific structure of the hidden neural network layers given in (2.22).

▷ *MAP case* (2.9). Consider the neurons in the first hidden layer $\boldsymbol{\ell}^{(1)}$. The s-th neuron in this first hidden layer, $1 \le s \le r_1$, has network weights $(\vartheta_{s,k}^{(1)})_{k=0}^{r_0=b_0+b} \in \mathbb{R}^{r_0+1}$, where the last b components refer to the embedding $\boldsymbol{e}_{\mathbf{U}}(z_i) = \mathbf{u}_{j[i]} \in \mathbb{R}^b$. From (2.22), we observe that we can scale the embeddings $\mathbf{u}_{j[i]} \in \mathbb{R}^b$ with any positive constant $c > 0$ (not depending on the level index $j[i]$), and we obtain the same value in the sth neuron (2.22), if we divide the network weights of that neuron

by the same constant, i.e., if we set

$$(\vartheta^{(1)}_{s,b_0+k}/c)_{k=1}^{b} \in \mathbb{R}^b. \tag{2.24}$$

In view of the regularization term in (2.7), given by

$$\frac{1}{w_{j[i]}} \frac{\varphi}{2\tau^2} \left\|\mathbf{u}_{j[i]}\right\|^2 = \frac{1}{w_{j[i]}} \left\| \frac{\sqrt{\varphi}}{\sqrt{2}\tau} \mathbf{u}_{j[i]} \right\|^2 = \frac{1}{w_{j[i]}} \left\| \sqrt{\lambda} \mathbf{u}_{j[i]} \right\|^2,$$

we observe that the specific value of $\lambda = \varphi/(2\tau^2) > 0$ does not matter, because the network weights $(\vartheta^{(1)}_{s,b_0+k})_{k=1}^{b} \in \mathbb{R}^b$ will accommodate any positive value by rescaling the networks weights correspondingly as in (2.24). Thus, it only matters whether we have some regularization $\tau^2 \in (0, \infty)$ or whether we do not have any regularization $\tau^2 = \infty$ in (2.7). For this reason, we could set $\varphi = 2$ and $\tau^2 = 1$ in the MAP case (2.7), yielding $\lambda = \varphi/(2\tau^2) = 1$. In the second item of the following remarks, we explain why in practice a different initialization still matters.

Remarks 2.1 • If we choose to have regularization, we note that it is not the size of the regularization that matters but the relative inverse case weights $1/w_j$ across all levels $a_j \in A$ of the categorical covariates. If we also want to impose a constraint on the sizes of the embeddings $\mathbf{u}_j$, then we also need to regularize the network weights $(\vartheta^{(1)}_{s,b_0+k})_{k=1}^{b} \in \mathbb{R}^b$ in the first hidden layer, so that a compensation (2.24) is penalized, e.g., we can require that the norms of these weights in the MAP case are of a similar magnitude as the ones in the non-regularized case.

- In our examples, we will select $\lambda = \varphi/(2\tau^2) > 0$ differently. Note that the choice of λ directly relates to the sizes of the network weights (2.24). For efficient gradient descent training of neural networks all weights, i.e., components of the network parameter $\boldsymbol{\vartheta}$, should live on a similar scale; otherwise, gradient descent fitting is not efficient. That is, for gradient descent training, all components of the network parameter $\boldsymbol{\vartheta}$ are initialized randomly following a certain distribution having the same scale for all components of $\boldsymbol{\vartheta}$, and, typically, it is hard for the gradient descent algorithm to fully adapt to rescale the weights in one component, if the scale is misspecified in that component. For this reason, the explicit choice of $\lambda > 0$ will impact gradient descent training, and we are going to choose $\lambda > 0$ different from 1, so that we receive good fitting results with gradient descent, i.e., such that the entity embedded categorical covariates have a similar range as the pre-processed continuous covariates.

▷ *Variational Bayesian inference case* (2.19). We can do a similar observation for the VB inference case (2.18). We first rewrite the regularization term using the notation $\lambda = \varphi/(2\tau^2)$

$$-\frac{1}{w_{j[i]}} \sum_{k=1}^{b} \frac{\varphi}{2} \left(\log\left(\frac{\sigma^2_{k,j[i]}}{\tau^2} \right) + 1 - \frac{\sigma^2_{k,j[i]} + \nu^2_{k,j[i]}}{\tau^2} \right)$$

$$= \frac{\lambda}{w_{j[i]}} \left[\|\nu_{j[i]}\|^2 + \sum_{k=1}^{b} \sigma_{k,j[i]}^2 - \tau^2 - \tau^2 \log\left(\frac{\sigma_{k,j[i]}^2}{\tau^2} \right) \right]. \tag{2.25}$$

From this, we see that $\lambda > 0$ can again be scaled through the network weights similar to (2.24), and τ^2 needs to accommodate correspondingly. However, there remains the hyperparameter τ^2 that needs to be chosen optimally to balance the influence of the Gaussian noise terms in the VB inference. In our examples, we choose for $\lambda > 0$ the same value as for the MAP case and we select the optimal τ^2 by a grid search. For more interpretation, we refer to the next case.

▷ *Ad-hoc case* (2.21). In this case, we have regularization term

$$\frac{\lambda}{w_{j[i]}} \left[\left\|\nu_{j[i]}\right\|^2 + \frac{\tau^2}{2} \sum_{k=1}^{b} \left(\frac{\sigma_{k,j[i]}^2}{\tau^2} - 1 \right)^2 \right].$$

This term gives a nice interpretation to τ^2, namely, it gives the scale to the randomness $\sigma_{k,j}$ coming from the Gaussian noise terms in the random effects, see (2.17). In fact, this can also be seen from (2.25), because the terms under the summation have the same form as Poisson deviance losses which are zero if and only if $\sigma_{k,j[i]}^2 = \tau^2$.

2.7 Extension to the multiple categorical covariate case

We extend to multiple categorical covariates. Assume that in total, we have $T \geq 1$ categorical covariates, and for each $1 \leq t \leq T$, we have q_t levels $A^{(t)} = \{a_1^{(t)}, \ldots, a_{q_t}^{(t)}\}$. The categorical covariate of instance i then takes the values

$$\boldsymbol{z}_i = \left(z_i^{(1)}, \ldots, z_i^{(T)} \right)^\top \in A^{(1)} \times \cdots \times A^{(T)}.$$

We apply the framework of one-hot encoding and embedding (2.1) to each categorical component $z_i^{(t)} \in A^{(t)}$ individually, and we choose an identical embedding dimension $b \in \mathbb{N}$ for all $1 \leq t \leq T$. This provides us with embeddings

$$\boldsymbol{z}_i \;\mapsto\; \left(\boldsymbol{e}_{\mathbf{U}^{(1)}}(z_i^{(1)}), \ldots, \boldsymbol{e}_{\mathbf{U}^{(T)}}(z_i^{(T)}) \right) \in \mathbb{R}^{b \times T}, \tag{2.26}$$

for embedding matrices of the categorical covariates $z_i^{(t)}$, $1 \leq t \leq T$

$$\mathbf{U}^{(t)} = \left[\mathbf{u}_1^{(t)}, \ldots, \mathbf{u}_{q_t}^{(t)} \right] \in \mathbb{R}^{b \times q_t}. \tag{2.27}$$

Adopting notation (2.6), we use $j_t[i] \in \{1, \ldots, q_t\}$ for saying that instance i with categorical covariate $\boldsymbol{z}_i$ takes level $z_i^{(t)} = a_{j_t[i]}^{(t)}$ for index t. This gives us for (2.26) the equivalent formulation

$$\boldsymbol{z}_i \;\mapsto\; \left(\mathbf{u}_{j_1[i]}^{(1)}, \ldots, \mathbf{u}_{j_T[i]}^{(T)} \right) \in \mathbb{R}^{b \times T}. \tag{2.28}$$

This is a multi-categorical version of (2.5). We concatenate all covariates for given embedding matrices $(\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(T)})$

$$(\boldsymbol{x}_i, \boldsymbol{z}_i) \;\mapsto\; \left(\boldsymbol{x}_i, \mathbf{u}^{(1)}_{j_1[i]}, \ldots, \mathbf{u}^{(T)}_{j_T[i]}\right) \;\in\; \mathbb{R}^{b_0+Tb}. \tag{2.29}$$

This is the multiple categorical covariate extension of (2.2). Finally, we extend the input dimension of the neural network correspondingly to receive the network mapping

$$(\boldsymbol{x}_i, \boldsymbol{z}_i) \;\mapsto\; \mathrm{NN}_{\vartheta}\left(\boldsymbol{x}_i, \mathbf{u}^{(1)}_{j_1[i]}, \mathbf{u}^{(2)}_{j_2[i]}, \ldots, \mathbf{u}^{(T)}_{j_T[i]}\right); \tag{2.30}$$

this architecture is illustrated in Fig. 1 (rhs). Model fitting and regularization is done completely analogously to above. In the MAP (2.9), VB inference (2.19), and the ad-hoc (2.21) cases, respectively, we have regularization terms

$$\begin{aligned}
&\sum_{t=1}^{T} \frac{\lambda_t}{w_{j_t[i]}} \left\| \mathbf{u}^{(t)}_{j_t[i]} \right\|^2, \\
&\sum_{t=1}^{T} \frac{\lambda_t}{w_{j_t[i]}} \left[\|\nu^{(t)}_{j_t[i]}\|^2 + \sum_{k=1}^{b} (\sigma^{(t)}_{k,j_t[i]})^2 - \tau_t^2 - \tau_t^2 \log\left(\frac{(\sigma^{(t)}_{k,j_t[i]})^2}{\tau_t^2} \right) \right], \\
&\sum_{t=1}^{T} \frac{\lambda_t}{w_{j_t[i]}} \left[\left\| \nu^{(t)}_{j_t[i]} \right\|^2 + \frac{\tau_t^2}{2} \sum_{k=1}^{b} \left(\frac{(\sigma^{(t)}_{k,j_t[i]})^2}{\tau_t^2} - 1 \right)^2 \right],
\end{aligned} \tag{2.31}$$

with regularization parameters $\lambda_t \geq 0$, prior uncertainty parameters $\tau_t^2 > 0$, with aggregated case weights for $j'_t \in \{1, \ldots, q_t\}$

$$w_{j'_t} = \sum_{i=1}^{n} \mathbb{1}_{\{j_t[i]=j'_t\}}\, v_i, \tag{2.32}$$

and with Gaussian mean field posterior approximation. Note that for the regularization terms (2.31), we assume independent Gaussian priors across the different categorical covariates.

3 The hierarchical random-effects case

In Sect. 2.7, we have presented the case of multiple categorical covariates $\boldsymbol{z}_i = (z_i^{(1)}, \ldots, z_i^{(T)})^\top$. These covariates have not been related to each other. In the present section, we impose more structure on these categorical covariates, namely, we assume that they have a hierarchical structure. This equips this set-up with additional information on the categorical covariates. There are different ways of modeling this case. Before discussing these different ways, we introduce a tree structure notation that is useful in this context.

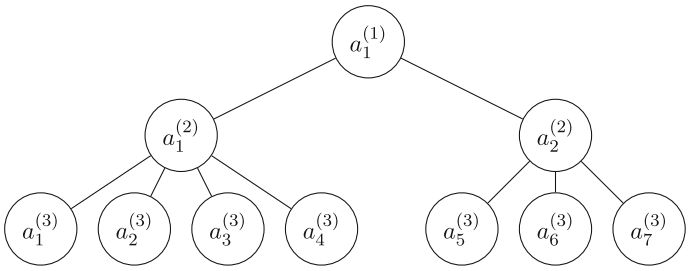

Fig. 3 Descendants of $a_1^{(1)} \in A^{(1)}$ with $T = 3$ generations

3.1 Tree structure and notation

As an example for random-effects modeling in the hierarchical categorical covariates case we can think of having information about 'vehicle brand'—'vehicle model'—'vehicle detail'. We assume that these covariates have a tree structure, and we use index $t \in \{1, \ldots, T\}$ to label the different generations in this tree. In the above example, generation $t = 1$ corresponds to 'vehicle brand', generation $t = 2$ to 'vehicle model' and generation $t = 3$ to 'vehicle detail'. We assume that the categorical levels in each generation uniquely determine the membership in the previous generations, e.g., a certain 'vehicle detail' level can only belong to one 'vehicle model' and one 'vehicle brand', respectively. This guarantees identifiability in the tree structure, so that all ancestors of a certain level in generation $t \geq 2$ can uniquely be determined.

It will be useful to extend the previously introduced labeling (2.6) to the different generations. Choose an index $j_t' \in \{1, \ldots, q_t\}$ in generation $2 \leq t \leq T$. We define its direct ancestor (using a slight abuse of notation) by $j_{t-1}[j_t'] \in \{1, \ldots, q_{t-1}\}$, this is the direct ancestor in generation $t-1$ of level $a_{j_t'}^{(t)} \in A^{(t)}$ in generation t. Similarly, we define the descendants of a given index $j_t' \in \{1, \ldots, q_t\}$ of a given generation $1 \leq t \leq T-1$ as follows:

$$\mathcal{I}_{j_t'} = \left\{ j \in \{1, \ldots, q_{t+1}\} \text{ with } j_t[j] = j_t' \right\}.$$

Figure 3 gives an example.

Choosing an identical embedding dimension b for all generations in (2.28) has the advantage that we can measure the distance of a certain level embedding $\mathbf{u}_{j_t'}^{(t)}$ in generation $2 \leq t \leq T$ to the one of its ancestor $\mathbf{u}_{j_{t-1}[j_t']}^{(t-1)}$. We define the corresponding differences (increments)

$$\Delta_{j_t'}^{(t)} = \mathbf{u}_{j_t'}^{(t)} - \mathbf{u}_{j_{t-1}[j_t']}^{(t-1)}, \tag{3.1}$$

and we initialize for $t = 1$ with $\Delta_{j_1'}^{(1)} = \mathbf{u}_{j_1'}^{(1)}$. This provides us with the Euclidean distance

$$\left\| \Delta_{j_t'}^{(t)} \right\| = \left\| \mathbf{u}_{j_t'}^{(t)} - \mathbf{u}_{j_{t-1}[j_t']}^{(t-1)} \right\|; \tag{3.2}$$

this is the distance between a random effect $\mathbf{u}_{j'_t}^{(t)} \in \mathbb{R}^b$ in generation t and the one of its direct ancestor $\mathbf{u}_{j_{t-1}[j'_t]}^{(t-1)} \in \mathbb{R}^b$ in generation $t-1$. Typically, we assume that these hierarchical embeddings provide a refinement of the regression function. In this case, descendants of a given level in generation $t-1$ will fluctuate around its ancestor, meaning that we receive a clustering around the ancestor resulting in small distances (3.2). Exactly, this intuition is going to be implemented (recursively) in the sequel by considering the random-effects increments $\Delta_{j'_t}^{(t)} \in \mathbb{R}^b$.

3.2 Gaussian hierarchical entity embedding

We extend the Gaussian categorical entity embedding approach of Sect. 2.7 to the hierarchical case. We are going to present four different modeling proposals called H0, H1, RNN and Transformer. The first proposal H0 is the canonical modeling set-up resulting from the discussion of (3.2). The second proposal H1 can be seen as a reduced form approach of H0 to make the model smaller. The third proposal, RNN, stems from 'recurrent neural network'. Formula (3.1) proposes a linear aggregation of increments, and the RNN architecture in Sect. 3.5 will modify this to a non-linear version. Finally, in Sect. 3.6, we process the hierarchical categorical entity embeddings with a Transformer layer that is another popular method of dealing with time-series data.

Hierarchical random-effects modeling. For random-effects modeling, we need to choose a prior distribution π on the embedding matrices (2.27). We define the sequence of embedding matrices

$$\mathbf{U}^{(1:T)} = \left(\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(T)}\right).$$

For interpretation, the generation index $1 \le t \le T$ plays a similar role as the time index in a time-series. We assume that $\mathbf{U}^{(1:T)}$ is a Markovian process in t, and conditionally, given $\mathbf{U}^{(t-1)}$, all components of $\mathbf{U}^{(t)} = (U_{k,j}^{(t)})_{1\le k\le b, 1\le j\le q_t}$ are independent with

$$U_{k,j}^{(t)}\Big|_{\mathbf{U}^{(t-1)}} \;=\; U_{k,j}^{(t)}\Big|_{U_{k,j_{t-1}[j]}^{(t-1)}} \;\sim\; \mathcal{N}\left(U_{k,j_{t-1}[j]}^{(t-1)}, \tau_t^2\right), \tag{3.3}$$

for $2 \le t \le T$. Basically, this is a Gaussian version of Jewell's credibility model (Jewell, 1975). In the sequel, it is more convenient to express (3.3) in the increments (3.1). We choose conditionally independent multivariate Gaussian increments

$$\Delta_j^{(t)}\Big|_{\mathbf{U}^{(t-1)}} \;\sim\; \mathcal{N}\left(\mathbf{0}, \operatorname{diag}(\tau_t^2)\right), \tag{3.4}$$

where this is a b-dimensional multivariate Gaussian distribution with independent components. Aggregation across generations gives us for a sequence $j_1, \ldots, j_T$ of

level indices with $j_t \in \mathcal{I}_{j_{t-1}}$ for all t

$$\mathbf{u}_{j_t}^{(t)} \;=\; \mathbf{u}_{j_{t-1}}^{(t-1)} + \Delta_{j_t}^{(t)} \;=\; \sum_{s=1}^{t} \Delta_{j_s}^{(s)}. \tag{3.5}$$

We collect all increments in the following random (time-series) vector:

$$\Delta^{(1:T)} = \left((\Delta_{j_1}^{(1)})_{j_1=1}^{q_1}, \ldots, (\Delta_{j_T}^{(T)})_{j_T=1}^{q_T} \right).$$

Hierarchical Model H0. For given embedding matrices $\mathbf{U}^{(1:T)}$ and increments $\Delta^{(1:T)}$, respectively, and adapting the neural network NN_{ϑ} to the input dimension $b_0 + Tb$, see (2.30), this gives us the data log-likelihood

$$\ell_{\boldsymbol{Y}}(\vartheta | \Delta^{(1:T)}) \\ \propto \sum_{i=1}^{n} \frac{Y_i\, h\left(\mathrm{NN}_{\vartheta}(\boldsymbol{x}_i, \mathbf{u}_{j_1[i]}^{(1)}, \ldots, \mathbf{u}_{j_T[i]}^{(T)}) \right) - \kappa\left(h\left(\mathrm{NN}_{\vartheta}(\boldsymbol{x}_i, \mathbf{u}_{j_1[i]}^{(1)}, \ldots, \mathbf{u}_{j_T[i]}^{(T)}) \right) \right)}{\varphi / v_i}.$$

This neural network has the same structure as (2.30), and it uses input (2.29), but in fact, we model its increments by (3.4). Therefore, we reformulate

$$\mathrm{NN}_{\vartheta}\left(\boldsymbol{x}_i, \mathbf{u}_{j_1[i]}^{(1)}, \mathbf{u}_{j_2[i]}^{(2)}, \ldots, \mathbf{u}_{j_T[i]}^{(T)} \right) = \mathrm{NN}_{\vartheta}\left(\boldsymbol{x}_i, \Delta_{j_1[i]}^{(1)}, \sum_{s=1}^{2} \Delta_{j_s[i]}^{(s)}, \ldots, \sum_{s=1}^{T} \Delta_{j_s[i]}^{(s)} \right), \tag{3.6}$$

being expressed in the random-effects increments $\Delta^{(1:T)}$. The notation on the right-hand side of (3.6) is more convenient when it comes to discuss regularization; we refer to Sects. 3.3–3.4.

Hierarchical Model H1. One could be concerned by the fact that the network input turns out to be very high-dimensional if there are many generations in the hierarchical categorical covariates; see (2.29). An alternative modeling approach is to only consider the last generation's embedding, resulting in a neural network

$$\mathrm{NN}_{\vartheta}\left(\boldsymbol{x}_i, \mathbf{u}_{j_T[i]}^{(T)} \right) = \mathrm{NN}_{\vartheta}\left(\boldsymbol{x}_i, \sum_{s=1}^{T} \Delta_{j_s[i]}^{(s)} \right). \tag{3.7}$$

Here, the network input has dimension $b_0 + b$, compared to $b_0 + Tb$ in (3.6). Nevertheless, the random effects $\mathbf{u}_{j_T[i]}^{(T)}$ consider all involved levels of all generations through the sum over their increments $\Delta_{j_s[i]}^{(s)}$, and these increments are regularized by a Gaussian prior π; see (3.4).

3.3 Regularization and random-effects estimation

We start with the MAP method. Using one of the two neural network approaches (3.6) or (3.7) for $\bullet$, gives us the joint log-likelihood of $(\boldsymbol{Y}, \Delta^{(1:T)})$

$$\begin{aligned}\log f_{\boldsymbol{\vartheta}}(\boldsymbol{Y}, \Delta^{(1:T)}) &\propto \left[\sum_{i=1}^{n} \frac{Y_i\, h\left(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, \bullet)\right) - \kappa\left(h\left(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, \bullet)\right)\right)}{\varphi / v_i}\right] + \log \pi(\Delta^{(1:T)}) \\ &\propto \frac{1}{\varphi} \sum_{i=1}^{n} v_i \Bigg[Y_i\, h\left(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, \bullet)\right) - \kappa\left(h\left(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, \bullet)\right)\right) \\ &\quad - \sum_{t=1}^{T} \frac{\lambda_t}{w_{j_t[i]}} \left\| \Delta^{(t)}_{j_t[i]} \right\|^2 \Bigg], \end{aligned} \tag{3.8}$$

For formula (3.8) to be correct, we assume that for every level in $A^{(T)}$ (last generation), we have at least one observation. This implies that also all ancestors have positive aggregated case weights. Based on this, we can find the MAP of $\Delta^{(1:T)}$ and the network parameter $\boldsymbol{\vartheta}$ completely analogously to Sect. 2.3 using gradient descent. Again, this is just a regularized gradient descent optimization where we regularize the random-effects embedding inversely proportionally to the case weights, and using a ridge regularization on the increments $\Delta^{(1:T)}$.

We also highlight the similarity of this last expression to (2.31), i.e., we only exchange the random effects $\mathbf{u}^{(t)}_{j_t[i]}$ by the corresponding increments $\Delta^{(t)}_{j_t[i]}$. The VB inference case and the ad-hoc case are completely analogous; see also (2.31). Therefore, we refrain from stating them explicitly.

3.4 Individual vs. hierarchical regularization

In this section, we argue that fitting Hierarchical Model H0 using, e.g., regularization (3.8) is equivalent to fitting the non-hierarchical model (2.30) using regularization (2.31). Therefore, Hierarchical Model H0 is superfluous and will not be further studied. The reason is again that using affine transformations, we can simply redefine the network weights to go from one to the other model. We give the technical argument. Choose a sequence $j_1, \ldots, j_T$ of level indices with $j_t \in \mathcal{I}_{j_{t-1}}$ for all t. The first hidden layer of neural network (3.6) has in the s-th neuron the following structure:

$$\begin{aligned}\tanh\Bigg(&\vartheta^{(1)}_{s,0} + \sum_{k=1}^{b_0} \vartheta^{(1)}_{s,k} x_k + \sum_{k=b_0+1}^{b_0+b} \vartheta^{(1)}_{s,k} \Delta^{(1)}_{j_1,k-b_0} + \ldots \\ &+ \sum_{k=b_0+(T-1)b+1}^{b_0+Tb} \vartheta^{(1)}_{s,k} \sum_{l=1}^{T} \Delta^{(l)}_{j_l,k-b_0+(T-1)b} \Bigg)\end{aligned}$$

$$
\begin{aligned}
&= \tanh\left(\vartheta^{(1)}_{s,0} + \sum_{k=1}^{b_0} \vartheta^{(1)}_{s,k} x_k + \sum_{k=1}^{b} \vartheta^{(1)}_{s,k+b_0} \Delta^{(1)}_{j_1,k} + \dots \right. \\
&\quad \left. + \sum_{k=1}^{b} \vartheta^{(1)}_{s,k+b_0+(T-1)b} \sum_{l=1}^{T} \Delta^{(l)}_{j_l,k}\right) \\
&= \tanh\left(\vartheta^{(1)}_{s,0} + \sum_{k=1}^{b_0} \vartheta^{(1)}_{s,k} x_k + \sum_{k=1}^{b} \widetilde{\vartheta}^{(1)}_{s,k+b_0} \Delta^{(1)}_{j_1,k} + \dots \right. \\
&\quad \left. + \sum_{k=1}^{b} \widetilde{\vartheta}^{(1)}_{s,k+b_0+(T-1)b} \Delta^{(T)}_{j_T,k}\right),
\end{aligned}
$$

where we have restructured the network weights

$$
\widetilde{\vartheta}^{(1)}_{s,k+b_0+ub} = \sum_{l=u}^{T-1} \vartheta^{(1)}_{s,k+b_0+lb}, \tag{3.9}
$$

for $0 \le u \le T-1$. We observe that the network NN_{ϑ} can cope with this transformation, by just re-defining the network weights in the first hidden layer for the hierarchical categorical covariates, and then, the approach is identical to the multiple categorical covariate case of Sect. 2.7, just having a slightly different interpretation for $\mathbf{u}^{(t)}_{j_t[i]}$ and $\Delta^{(t)}_{j_t[i]}$, respectively. However, the network and its fitting procedure do not see this different interpretation, and, in fact, we obtain the same predictive model when trained on data. Therefore, we do not further consider Hierarchical Model H0.

In contrast, Hierarchical Model H1 is different from the models in Sect. 2.7. It has a lower complexity, as it implicitly assumes that all parameters (3.9) are identical for $u \in \{0, \dots, T-1\}$. In our example below, we will only study this Hierarchical Model H1. In the next section, we will see that the Hierarchical Model H0 can nicely serve as a motivation to more complex models based on RNN layers.

3.5 Recurrent network hierarchical random effects

The issue why the hierarchical structure H0 is not directly useful in view of the network presented in Sect. 2.7 is that the neural network NN_{ϑ} can cope with affine transformations. If we want to insist on the exploration of the hierarchical structure in a family of categorical covariates, we need to transform the increments $\Delta^{(t)}_{j_t[i]}$ non-linearly, i.e., we should not simply add them as in (3.6). This motivates the idea to exploit a RNN layer $\boldsymbol{\ell}^{\mathrm{RNN}}$ after entity embedding and before imputing to NN_{ϑ}. An RNN layer $\boldsymbol{\ell}^{\mathrm{RNN}}$ has precisely the same structure as the layer in (2.22), we only change the input dimension, such that we can consider recursively for $t \ge 1$

$$
\boldsymbol{\ell}^{\mathrm{RNN}} : \mathbb{R}^{2b} \to \mathbb{R}^{b}, \qquad \left(\Delta^{(t)}_{j_t}, \boldsymbol{r}^{(t-1)}\right) \mapsto \boldsymbol{r}^{(t)} = \boldsymbol{\ell}^{\mathrm{RNN}}\left(\Delta^{(t)}_{j_t}, \boldsymbol{r}^{(t-1)}\right), \tag{3.10}
$$

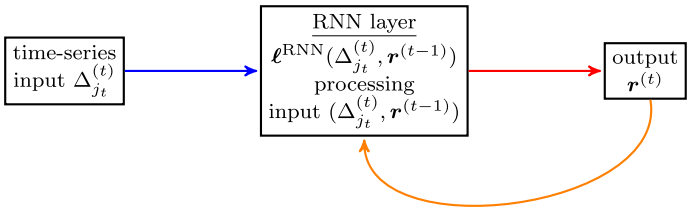

Fig. 4 RNN layer ℓ^{RNN} recursively processing the input $(\Delta_{j_t}^{(t)}, r^{(t-1)})$

with initialization $r^{(0)} = \mathbf{0} \in \mathbb{R}^b$. That is, the RNN layer ℓ^{RNN} recursively processes the increments of the time-series $\Delta^{(1:T)}$, providing an encoding of ancestors $r^{(t)} = r^{(t)}(\Delta_{j_1}^{(1)}, \ldots, \Delta_{j_t}^{(t)})$ at time t. This recursive encoding always uses the same network parameter in ℓ^{RNN}. In contrast to (3.5), we do not aggregate linearly, but we let the network ℓ^{RNN} specify the (non-linear) aggregation. Figure 4 illustrates an RNN layer.

The remainder is as in (3.6), i.e., after concatenating, we consider a neural network

$$\text{NN}_\vartheta \left(x_i, r^{(1)}, r^{(2)}, \ldots, r^{(T)} \right). \tag{3.11}$$

This architecture is illustrated in Fig. 2. Fitting then works as, e.g., in (3.8). However, it also includes the parameters from the RNN layer ℓ^{RNN}.

Remark 3.1 The neural network (3.11) considers the entire RNN processed time-series $r^{(1:T)} = (r^{(1)}, r^{(2)}, \ldots, r^{(T)})$. In network implementations, this usually requires to set a parameter called "return sequence" to true. Alternatively, the RNN layer could only output the last encoding $r^{(T)} = r^{(T)}(\Delta_{j_1}^{(1)}, \ldots, \Delta_{j_T}^{(T)})$, which collects the whole time-series in one variable. In fact, this then precisely corresponds to a non-linear version of Hierarchical Model H1 (3.7).

3.6 Transformer processing of hierarchical random effects

The RNN layer recursively processes the time-series components of $\Delta^{(1:T)}$. A popular alternative that processes the whole time-series $(\Delta_{j_1}^{(1)}, \ldots, \Delta_{j_T}^{(T)})$ at once is a Transformer layer. Transformer layers have been invented by Vaswani et al. (2017), and they are the most powerful tools these days to deal with natural language processing (NLP). Instead of recursively processing a time-series, Transformers assign attention weights to the elements of the time-series data, emphasizing importance of individual parts of the time-series. We briefly sketch an attention layer, and for more details, we refer to Vaswani et al. (2017), in particular, Fig. 1 of that reference. An attention layer consists of queries q_t, keys k_t and values v_t, $1 \le t \le T$, given by

$$\begin{aligned} k_t &= \tanh\left(b_K + W_K \Delta_{j_t}^{(t)}\right) \in \mathbb{R}^b, \\ q_t &= \tanh\left(b_Q + W_Q \Delta_{j_t}^{(t)}\right) \in \mathbb{R}^b, \end{aligned}$$

$$\boldsymbol{v}_t = \tanh\left(\boldsymbol{b}_V + W_V \Delta_{j_t}^{(t)}\right) \in \mathbb{R}^b,$$

for weight matrices $W_K, W_Q, W_V \in \mathbb{R}^{b\times b}$, biases $\boldsymbol{b}_K, \boldsymbol{b}_Q, \boldsymbol{b}_V \in \mathbb{R}^b$, and where the hyperbolic tangent function is applied element-wise. The idea behind this terminology is the following. The key $\boldsymbol{k}_t \in \mathbb{R}^b$ of $\Delta_{j_t}^{(t)}$ tries to find a query $\boldsymbol{q}_s \in \mathbb{R}^b$ of a component $\Delta_{j_s}^{(s)}$ that matches. The components $\Delta_{j_t}^{(t)}$ and $\Delta_{j_s}^{(s)}$ then start to communicate to see whether their keys and queries match. For this, we stack the keys, queries, and values in matrices

$$\begin{aligned} K &= K(\Delta_{j_1}^{(1)}, \ldots, \Delta_{j_T}^{(T)}) = [\boldsymbol{k}_1, \ldots, \boldsymbol{k}_T]^\top \in \mathbb{R}^{T\times b}, \\ Q &= Q(\Delta_{j_1}^{(1)}, \ldots, \Delta_{j_T}^{(T)}) = \left[\boldsymbol{q}_1, \ldots, \boldsymbol{q}_T\right]^\top \in \mathbb{R}^{T\times b}, \\ V &= V(\Delta_{j_1}^{(1)}, \ldots, \Delta_{j_T}^{(T)}) = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_T]^\top \in \mathbb{R}^{T\times b}. \end{aligned}$$

The matching problem is now computed by applying the softmax function to the rows of the following matrix providing the attention weights:

$$A = A(\Delta_{j_1}^{(1)}, \ldots, \Delta_{j_T}^{(T)}) = \mathsf{softmax}\left(\frac{QK^\top}{\sqrt{b}}\right) \in \mathbb{R}^{T\times T}.$$

This has the following interpretation. If the key $\boldsymbol{k}_t$ of component $\Delta_{j_t}^{(t)}$ matches the query $\boldsymbol{q}_s$ of component $\Delta_{j_s}^{(s)}$, their scalar product is large

$$\langle \boldsymbol{q}_s, \boldsymbol{k}_t \rangle = \boldsymbol{q}_s^\top \boldsymbol{k}_t = (QK^\top)_{s,t},$$

and the attention weight $A_{s,t} \in [0, 1]$ is close to 1. An attention layer, also called attention head, is then defined by

$$H = H(\Delta_{j_1}^{(1)}, \ldots, \Delta_{j_T}^{(T)}) = A\,V \in \mathbb{R}^{T\times b}.$$

This encodes the time-series $(\Delta_{j_1}^{(1)}, \ldots, \Delta_{j_T}^{(T)}) \in \mathbb{R}^{b\times T}$ by an attention head. A Transformer layer is based on this attention head. First, we aggregate the two time-series to a new time-series

$$\left(\boldsymbol{h}^{(1)}, \ldots, \boldsymbol{h}^{(T)}\right) = (\Delta_{j_1}^{(1)}, \ldots, \Delta_{j_T}^{(T)}) + H^\top \in \mathbb{R}^{b\times T}.$$

Each of these components $\boldsymbol{h}^{(t)} \in \mathbb{R}^b$ is then processed through an auto-encoder consisting of two neural network layers $\boldsymbol{\ell}^{(2:1)} = \boldsymbol{\ell}^{(2)} \circ \boldsymbol{\ell}^{(1)}$ with input dimension being equal to the output dimension (auto-encoder), i.e., we process

$$\boldsymbol{h}^{(t)} \in \mathbb{R}^b \mapsto \boldsymbol{\ell}^{(2:1)}(\boldsymbol{h}^{(t)}) \in \mathbb{R}^b.$$

In particular, all components $\boldsymbol{h}^{(t)}$, $1 \leq t \leq T$, share the same network parameters in this auto-encoder which is called a time-distributed layer in network jargon. We aggregate the resulting time-series once more with the previous time-series providing us the Transformer

$$\begin{aligned}\left(\widetilde{\boldsymbol{r}}^{(1)}, \ldots, \widetilde{\boldsymbol{r}}^{(T)}\right) &= \texttt{Transformer}(\Delta_{j_1}^{(1)}, \ldots, \Delta_{j_T}^{(T)}) \\ &= \left(\boldsymbol{h}^{(1)} + \boldsymbol{\ell}^{(2:1)}(\boldsymbol{h}^{(1)}), \ldots, \boldsymbol{h}^{(T)} + \boldsymbol{\ell}^{(2:1)}(\boldsymbol{h}^{(T)})\right) \in \mathbb{R}^{b \times T}.\end{aligned} \tag{3.12}$$

The remainder is now completely analogous to (3.11), namely, input the Transformer layer output together with the continuous covariates $\boldsymbol{x}$ to a deep feed-forward neural network

$$\mathrm{NN}_{\vartheta}\left(\boldsymbol{x}, \widetilde{\boldsymbol{r}}^{(1)}, \ldots, \widetilde{\boldsymbol{r}}^{(T)}\right). \tag{3.13}$$

Remarks 3.2

- There is a major difference though between the RNN case (3.11) and the Transformer case (3.13), and Remark 3.1 does not apply in the latter case. The RNN model is time-causal, meaning that $\boldsymbol{r}^{(t)} = \boldsymbol{r}^{(t)}(\Delta_{j_1}^{(1)}, \ldots, \Delta_{j_t}^{(t)})$ only considers the first t components of the time-series, whereas the Transformer layer considers (in our case) the entire information in every component t, i.e., $\widetilde{\boldsymbol{r}}^{(t)} = \widetilde{\boldsymbol{r}}^{(t)}(\Delta_{j_1}^{(1)}, \ldots, \Delta_{j_T}^{(T)})$. One could make the Transformer layer time-causal too, but this is not necessary in our modeling problem.
- In relation to the previous item: actually we do not need a hierarchical structure to apply this Transformer layer approach, and this proposal works on any high-cardinality covariate situation.

4 Example: regularization of categorical entity embedding

4.1 Description of the data

We consider a synthetic insurance claim frequency example.[1] We use a synthetic dataset, because this has the advantage that we know the ground truth, and the quality of any of the studied regression models can be compared to the true model.

In a first step, we need to construct an insurance portfolio $(\boldsymbol{x}_i, z_i)_{i=1}^n$. Our insurance portfolio has $n = 199,971$ insurance policies, and for these policies, we have continuous and binary covariates

$$\boldsymbol{x} = (\texttt{VehUse}, \texttt{Town}, \texttt{DrivAge}, \texttt{VehWeight}, \texttt{VehPower}, \texttt{VehAge})^\top \in \mathbb{R}^6.$$

Furthermore, we have hierarchical categorical covariates `VehBrand`, `VehModel` and `VehDetail`. For the simulation of the continuous covariates $\boldsymbol{x}$, we consider the

[1] The data and code can be downloaded from:
https://github.com/wueth/High-Cardinality-Covariates-Regularization

Listing 1 Excerpt of simulated dataset D.

```
'data.frame':   199971 obs. of  11 variables:
 $ VehUse    : num  0 1 0 0 0 0 1 0 0 1 ...
 $ Town      : num  1 1 1 0 0 1 0 0 1 1 ...
 $ DrivAge   : num  51 41 25 40 43 45 72 60 30 19 ...
 $ VehWeight: num  1730 1760 1230 1010 2150 1160 1050 1040
     1370 1300 ...
 $ VehPower : num  169 249 109 84 166 144 61 65 113 77 ...
 $ VehAge    : num  3 2 2 9 5 2 2 2 0 0 ...
 $ VehBrand : Factor w/ 20 levels "A","B","C","D",..: 10 11 11
     1 13 11 16 10 11 10 ...
 $ VehModel : Factor w/ 100 levels "Aa","Ab","Ac",..: 47 54 54
     4 63 54 80 46 54 47 ...
 $ VehDetail: Factor w/ 470 levels "Aa1","Aa2","Aa3",..: 218
     250 250 18 297 249 376 ...
 $ True      : num  0.1382 0.1127 0.1573 0.0696 0.1188 ...
 $ ClaimNb   : int  0 0 0 0 0 1 0 0 0 0 ...
```

same algorithm as for the generation of the synthetic data in Mayer et al. (2023). These continuous covariates are then extended by the categorical ones, which are simulated from a categorical GLM using the continuous covariates as independent variables. Listing 1 gives an excerpt of the resulting simulated data. We have 20 `VehBrands`; these have 100 different `VehModels`, which in turn have 470 different `VehDetails`.

Figure 13 in the appendix illustrates the marginal distributions of the insurance policies for all covariates. We observe the typical shapes for `DrivAge`, `VehWeight`, `VehPower` and `VehAge`, and their dependence structure has also been chosen, such that it reflects a real insurance portfolio. The categorical covariates are shown on the last row of Fig. 13 and, in particular, `VehDetail` has some levels with only very sparse observations. This is also verified by Fig. 5 that shows the case weights $w_{j_3}^{(3)}$ of `VehDetail` across all levels having indices $j_3 \in \{1, \ldots, q_3 = 470\}$; note that we set $v_i \equiv 1$. The smallest level has three observations and the most common one 24,171. For `VehBrand`, these numbers are 1,363 and 58,766, and for `VehModel`, we have 55 and 36,112.

Based on this portfolio $(\boldsymbol{x}_i, z_i)_{i=1}^n$, we construct the true regression function μ^*. Our choice of μ^* is as follows, we also refer to Listing 2 in the appendix. First, we define a non-linear transformation of the driver age variable. Set $\mathtt{DA}_1 = (\mathtt{DrivAge} - 66)/60$ and define

$$\mathtt{DA}_2(\mathtt{DrivAge}) = 0.05 + \mathtt{DA}_1^8 + 0.4\,\mathtt{DA}_1^3 + 0.3\,\mathtt{DA}_1^2 + 0.06\,\mathtt{DA}_1.$$

The true regression function is chosen by

$$\begin{aligned}\log \mu^*(\boldsymbol{x}, z) = {} & 0.15\,\mathtt{Town} + \log\left(\mathtt{DA}_2(\mathtt{DrivAge})\right) + (0.3 + 0.15\,\mathtt{Town})\,\mathtt{VehPower}/100 \\ & + 0.1\,\mathtt{VehPower}/\left(\mathtt{VehWeight}/100\right)^2 + 0.2 \sum_{j_1=1}^{q_1} \beta_{j_1}^{(1)} \mathbb{1}_{\{\mathtt{VehBrand}=a_{j_1}^{\mathtt{VehBrand}}\}} \\ & + \sum_{j_2=1}^{q_2} \left(0.2\,(2\,\mathtt{Town} - 1) + 0.1\,\mathtt{VehUse}\right) \beta_{j_2}^{(2)} \mathbb{1}_{\{\mathtt{VehModel}=a_{j_2}^{\mathtt{VehModel}}\}}\end{aligned}$$

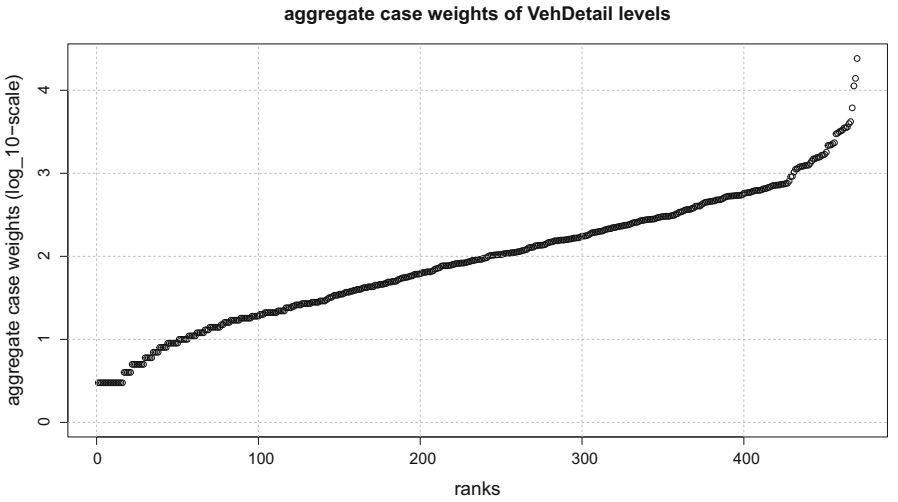

Fig. 5 Aggregate case weights $w_{j_3}^{(3)}$ of all levels $a_{j_3}^{(3)} \in A^{(3)}$ of the categorical covariate `VehDetail` having $q_3 = 470$ different levels; the x-axis is ordered w.r.t. the ranks

$$+ 0.3\,(2\,\texttt{VehUse} - 1) \sum_{j_3=1}^{q_3} \beta_{j_3}^{(3)} \mathbb{1}_{\{\texttt{VehDetail}=a_{j_3}^{\texttt{VehDetail}}\}} + 0.03\,\texttt{VehAge}, \tag{4.1}$$

with parameters $(\beta_{j_1}^{(1)})_{j_1=1}^{q_1}$, $(\beta_{j_2}^{(2)})_{j_2=1}^{q_2}$ and $(\beta_{j_3}^{(3)})_{j_3=1}^{q_3}$ taking values in $(-1, 1)$. We remark that the categorical covariates interact with the continuous ones in a non-linear way on the log-scale, in particular, the terms $(2\,\texttt{Town} - 1) \in \{-1, +1\}$ and $(2\,\texttt{VehUse}-1) \in \{-1, +1\}$ may lead to sign switches in the parameters $\beta_{j_2}^{(2)}$ and $\beta_{j_3}^{(3)}$, respectively. Line 11 of Listing 1 called `True` gives these true expected frequencies $\mu^*(\boldsymbol{x}_i, z_i)$ over the entire portfolio.

Finally, we simulate independent Poisson random variables $Y_i \sim \text{Poi}(\mu^*(\boldsymbol{x}_i, z_i))$ which provides us with the data

$$\mathcal{D} = \Big(Y_i, (\boldsymbol{x}_i, z_i), v_i \equiv 1\Big)_{i=1}^{n}.$$

Figure 14 in the appendix gives the marginal observed and true frequencies. These are supported by empirical two standard deviations confidence bounds, estimated for each level individually and assuming a Poisson distribution, i.e., these bounds are obtained by

$$\widehat{\mu}_{j_t} \;\pm\; 2 \cdot \sqrt{\frac{\widehat{\mu}_{j_t}}{w_{j_t}}},$$

where $\widehat{\mu}_{j_t}$ is the empirical frequency over all observations Y_i that have covariate level $z_i^{(t)} = a_{j_t}^{(t)}$ in generation t. For sparse levels, these confidence bounds become very

wide and empirical frequency estimations carry a lot of uncertainty as can be seen from Fig. 14.

4.2 Plain-vanilla benchmark models

We start by considering benchmark models. These benchmarks do not use any regularization. The first benchmark model is a plain-vanilla neural network using embedding layers for categorical covariates, the second benchmark model is the GLMMNet of Avanzi et al. (2024) not using any regularization, and the third benchmark model is a LightGBM regression model. We fit these three regression approaches to different inputs, considering all continuous covariates $\boldsymbol{x}$ and less or more of the categorical covariates

$$\boldsymbol{z} = (\texttt{VehBrand}, \texttt{VehModel}, \texttt{VehDetail}). \tag{4.2}$$

We start by describing implementation of these three benchmark models.

Plain-vanilla neural network with entity embedding without regularization. We proceed as follows. We choose the Poisson model for the responses Y_i. The Poisson model belongs to the EDF with cumulant function $\kappa(\theta) = \exp(\theta)$ and canonical link $h(m) = \log(m)$ for $\theta \in \mathbb{R}$ and $m > 0$; see Wüthrich and Merz (2023, Section 2.2.2). We start by fitting a plain-vanilla feed-forward neural network to these data using the covariates $\boldsymbol{x}_i \in \mathbb{R}^6$ as continuous inputs and subsets of the categorical covariates $\boldsymbol{z}_i$ are inputted by $b = 2$ dimensional entity embeddings without regularization. This corresponds to setting $\lambda_t = 0$ in (2.31). We have log-likelihood in this Poisson case (without regularization)

$$\begin{aligned}\ell_{\boldsymbol{Y}}(\boldsymbol{\vartheta}|\mathbf{U}^{(1:T)}) \;\propto\; \sum_{i=1}^{n} Y_i \,\log\Big(\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, \mathbf{u}^{(1)}_{j_1[i]}, \ldots, \mathbf{u}^{(T)}_{j_T[i]})\Big) \\ -\mathrm{NN}_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, \mathbf{u}^{(1)}_{j_1[i]}, \ldots, \mathbf{u}^{(T)}_{j_T[i]}),\end{aligned}$$

with dispersion $\varphi = 1$ and exposures $v_i \equiv 1$. We use a neural network of depth $d = 3$ with numbers of neurons $(r_1, r_2, r_3) = (20, 15, 10)$ in the three hidden layers, hyperbolic tangent activation function, the log-link (canonical link) for the output. The loss function chosen is the Poisson deviance loss. The resulting neural network is very similar to the one in Wüthrich and Merz (2023, Listing 7.4).

We fit this neural network on the dataset $\mathcal{D}$ using the `nadam` version of stochastic gradient descent (SGD) and using early stopping on a 20% validation set $\mathcal{V}$ of $\mathcal{D}$. This selection needs some care for the resulting training set $\mathcal{T} = \mathcal{D}\backslash\mathcal{V}$ which serves at calculating the SGD steps. Since we have some scarce levels $a^{(T)}_{j_T} \in A^{(T)}$, we need to ensure that every level $a^{(T)}_{j_T}$ appears at least once in the training data $\mathcal{T}$; otherwise, the corresponding embedding $\mathbf{u}^{(T)}_{j_T} \in \mathbb{R}^b$ of that level $a^{(T)}_{j_T}$ remains untrained. Since neural network fitting involves several elements of randomness (like initialization of the algorithm), see Wüthrich and Merz (2023, Figures 7.16−7.17), we average all network predictors over an ensemble of ten individual network fittings; for more

details, we refer to Wüthrich and Merz (2023, Section 7.4.4), in particular, to formula (7.44) of that reference.

GLMMNet with entity embedding without regularization. For the GLMMNet of Avanzi et al. (2024), we need to modify the network architecture, and we also refer to Fig. 1. Namely, the concatenation of the embeddings of the categorical covariates is applied to the neuron activations of the last hidden layer. Thus, in view of (2.23), we choose a fully connected feed-forward neural network that only processes the continuous covariates

$$\boldsymbol{x} \;\mapsto\; \boldsymbol{\ell}^{(d:1)}(\boldsymbol{x}) = \left(\boldsymbol{\ell}^{(d)} \circ \cdots \circ \boldsymbol{\ell}^{(1)}\right)(\boldsymbol{x}) \;\in\; \mathbb{R}^{r_d}.$$

These transformed continuous covariates are then concatenated with the embeddings to provide the output

$$\text{GLMMNet}_{\boldsymbol{\vartheta}}(\boldsymbol{x}, z) = \exp\left\{\vartheta_0^{(d+1)} + \sum_{k=1}^{r_d} \vartheta_k^{(d+1)} \boldsymbol{\ell}_k^{(d:1)}(\boldsymbol{x}) + \sum_{t=1}^{T} \sum_{l=k}^{b} \vartheta_{r_d+(t-1)b+k}^{(d+1)} e_{k,\mathbf{U}^{(t)}}(z^{(t)})\right\}, \tag{4.3}$$

where $e_{k,\mathbf{U}^{(t)}}(z^{(t)})$ is the kth component of embedding $\boldsymbol{e}_{\mathbf{U}^{(t)}}(z^{(t)}) \in \mathbb{R}^b$.

In our applications, we are going to consider the same network architecture for $\boldsymbol{\ell}^{(d:1)}$ of depth $d = 3$ as in the plain-vanilla neural network case, except that we adjust the input dimension to the continuous covariates $\boldsymbol{x}$, only. The training is done as described above, we use the same training-validation split, and we apply ensembling over ten different network fittings.

LightGBM. The third benchmark model that we consider is the LightGBM regression tree boosting of Ke et al. (2017). For training and prediction, we use the R package `lightgbm`, and we use the hyperparameters as in Mayer et al. (2023),[2] the minimal number of instances in each leaf is set to 50. To have comparability with the networks, we exercise early stopping on exactly the same training and validation split $\mathcal{T}$ and $\mathcal{V}$ of the entire data $\mathcal{D}$ as above.

For implementation, we use the `Keras` library (Chollet et al., 2017) in R for the networks and the `lightgbm` package (Ke et al., 2017) for LightGBM. The fitted models $\widehat{\mu}$ are then compared to the true model μ^* using the average KL divergence, in the Poisson case given by

$$\overline{D}_{\text{KL}}\left(\mu^* \| \widehat{\mu}\right) = \frac{1}{n} \sum_{i=1}^{n} \widehat{\mu}(\boldsymbol{x}_i) - \mu^*(\boldsymbol{x}_i) - \mu^*(\boldsymbol{x}_i) \log\left(\frac{\widehat{\mu}(\boldsymbol{x}_i)}{\mu^*(\boldsymbol{x}_i)}\right);$$

see Wüthrich and Merz (2023, Example 2.24). Table 1 presents these average KL divergences of the three benchmark models and for different selections of the categorical covariates. Additionally, we provide in the appendix the corresponding rooted mean squared errors (RMSEs) and the mean absolute errors (MAEs); see Tables 5 and 7. Not considering any categorical covariates (complete pooling) provides an average

[2] https://github.com/JSchelldorfer/ActuarialDataScience/tree/master/14-SHAP

Table 1 Benchmark models with high-cardinality categorical covariates using $b = 2$ dimensional embedding layers for the networks (2.30) and (4.3); numbers in round brackets $(\cdot)$ in the 1st column indicate the number of considered categorical covariate components; figures are in 10^{-2}

		Average KL divergence		
		Network (2.30)	GLMMNet (4.3)	LightGBM
(0)	Null model (empirical mean)	1.0342	1.0342	1.0342
(0)	w/o categorical covariates	0.3947	0.3970	0.3958
(1)	With `VehBrand`	0.2622	0.3076	0.2763
(1)	With `VehModel`	0.2188	0.2961	0.2499
(1)	With `VehDetail`	0.2615	0.3292	0.2240
(2)	With `VehBrand`, `VehModel`	0.2312	0.2958	0.2618
(3)	With `VehBrand`, `VehModel`, `VehDetail`	0.2694	0.3305	0.2191

KL divergence of 0.3947 and 0.3958, respectively. The LightGBM has a decreasing average KL divergence in the granularity of the categorical covariates, `VehBrand` – `VehModel` – `VehDetail` are hierarchical, and the best model is received by including all of them giving us an average KL divergence of 0.2191. The networks do not have this monotonicity, because early stopping implies in the case of high-cardinality categorical features a very early stopping time to prevent from over-fitting. In our case, the plain-vanilla network with a 2-dimensional entity embedding leads to the best result of 0.2188 if we only include `VehModel`. If we include `VehDetail`, we stop the SGD algorithm very early resulting in under-fitting on the remaining covariates. Therefore, it is necessary to apply regularization to these high-cardinality categorical covariates. Moreover, we observe that the GLMMNet is not fully competitive. This is because we have non-linear interactions between categorical and continuous covariates that cannot be captured by architecture (4.3), e.g., `VehUse` interacts with `VehDetail`, see (4.1). On the other hand, we should mention that the GLMMNet has the advantage of better interpretability.

4.3 High-cardinality entity embedding regularization

Next, we consider regularization of entity embeddings. In this section we do not consider the hierarchical structure, but we just explore regularizations (2.31) in the network NN_{ϑ} given in (2.30) and the GLMMNet (4.3) for the embedding matrices $\mathbf{U}^{(t)}$.

Since regularization requires selection of the hyperparameters λ_t and τ_t^2, we run a preliminary fit with cross-validation to select optimal hyperparameters. This is done as follows. For `VehBrand` we set $\lambda_1 = 0$, i.e., we do not regularize the `VehBrand` entity embedding, because each level $a_{j_1}^{(1)} \in A^{(1)}$ in the first generation has many observations. Then, we do a hyperparameter grid search for $\lambda_2 > 0$ using MAP regularization using the model considering `VehBrand` and `VehModel`. This optimal λ_2 is kept fixed for the rest of the models. Afterward, we include `VehDetail` to the MAP regularization to exploit the optimal $\lambda_3 > 0$. We proceed analogously for τ_t^2 in the ad-hoc and the VB regularization cases (using the MAP optimal values for λ_t). Table 4 in the appendix reports the hyperparameters used.

Table 2 gives the KL divergence results, and in the appendix, we provide the corresponding RMSE and MAE results; see Tables 6 and 8. We observe that regularization of high-cardinality categorical covariates is highly beneficial to get better predictive models. Considering all categorical covariates allows us to reduce the average KL divergence from 0.2188 (best model in Table 1) by roughly 1/3 to 0.1410 (last line of Table 2). This is also verified by the RMSE results, see Tables 5 and 6, and by the MAE results, see Tables 7 and 8. Another interesting observation is that the type of regularization only has a marginal influence on the results.

Figure 6 shows the resulting embeddings $\widehat{\mathbf{u}}_{j_1}^{(1)}, \widehat{\mathbf{u}}_{j_2}^{(2)}, \widehat{\mathbf{u}}_{j_3}^{(3)} \in \mathbb{R}^2$ in the MAP regularized case including all three categorical covariates (4.2) (last line of Table 2). Because we choose embedding dimension $b = 2$, we can nicely illustrate these embeddings. The color scale is chosen, such that red color refers to small case weights $w_{j_t}^{(t)}$ and

Table 2 Regularization of high-cardinality categorical covariates `VehModel` and `VehDetail`; numbers in round brackets $(\cdot)$ in the 1st column indicate the number of considered categorical covariate components; the selected hyperparameters are given in Table 4; figures are in 10^{-2}

		Average KL divergence	
		Network (2.30)	GLMMNet (4.3)
(2)	No regularization: with `VehBrand`, `VehModel`	0.2312	0.2958
(2)	MAP regularization: with `VehBrand`, `VehModel`	0.2212	0.2799
(2)	Ad-hoc regularization: with `VehBrand`, `VehModel`	0.2260	0.2794
(2)	VB regularization: with `VehBrand`, `VehModel`	0.2331	0.2800
(3)	No regularization: with `VehBrand`, `VehModel`, `VehDetail`	0.2694	0.3305
(3)	MAP regularization: with `VehBrand`, `VehModel`, `VehDetail`	0.1446	0.2584
(3)	Ad-hoc regularization: with `VehBrand`, `VehModel`, `VehDetail`	0.1449	0.2596
(3)	VB regularization: with `VehBrand`, `VehModel`, `VehDetail`	0.1410	0.2589

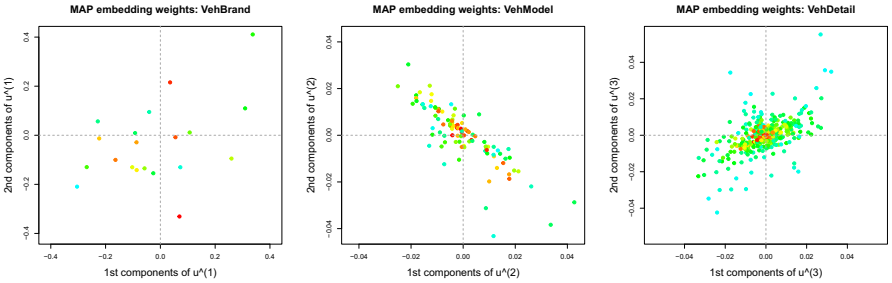

Fig. 6 Resulting embeddings $\widehat{\mathbf{u}}_{j_1}^{(1)}, \widehat{\mathbf{u}}_{j_2}^{(2)}, \widehat{\mathbf{u}}_{j_3}^{(3)} \in \mathbb{R}^2$ in the MAP regularized case including all three categorical covariates (4.2); red color shows small case weights and cyan color shows high case weights

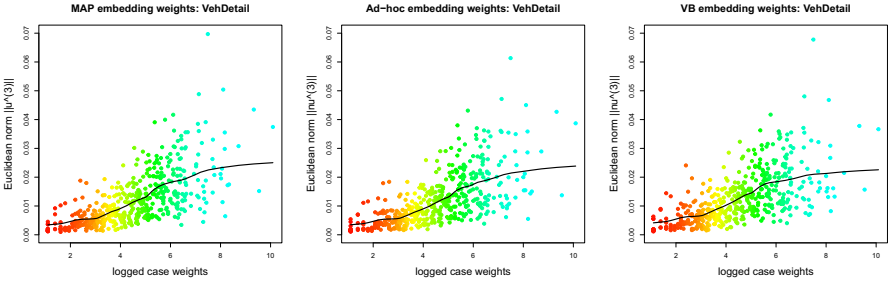

Fig. 7 Euclidean norms $\|\widehat{\mathbf{u}}_{j_3}^{(3)}\|$ and $\|\widehat{\nu}_{j_3}^{(3)}\|$ of the embeddings of the categorical variable `VehDetail` plotted against the logged case weights $\log(w_{j_3}^{(3)})$: (lhs) MAP regularized, (middle) ad-hoc regularized, and (rhs) VB regularized; colors coincide with the ones of Fig. 6 (rhs)

cyan color refers to high case weights $w_{j_t}^{(t)}$. Figure 6 illustrates the results: `VehBrand` has not been regularized ($\lambda_1 = 0$), and the colors of the dots do not have any structure in Fig. 6 (lhs). `VehModel` and `VehDetail` are regularized with $\lambda_2 = \lambda_2 = 10^3$, and the levels $a_{j_t}^{(t)} \in A^{(t)}$ with small case weights $w_{j_t}^{(t)}$ are more concentrated around the origin than the other levels; see Fig. 6 (middle, rhs). Remark that the other two cases of the ad-hoc regularization and the VB regularization look similar to Fig. 6.

The last statement of above is verified in Fig. 7 where we show the Euclidean norms $\|\widehat{\mathbf{u}}_{j_3}^{(3)}\|$ and $\|\widehat{\nu}_{j_3}^{(3)}\|$ of the embeddings of the categorical variable `VehDetail` plotted against the logged case weights $\log(w_{j_3}^{(3)})$ for all three considered regularization methods; the black line gives a spline fit to these Euclidean norms. We observe that on average these Euclidean norms are increasing in increasing case weights. This precisely reflects less strong regularization in (2.31) of levels $a_{j_3}^{(3)}$ that have more frequent observations $j_3[i]$, $1 \le i \le n$.

The ad-hoc regularization method and the VB inference regularization have the advantage that we also estimate posterior standard deviations $\widehat{\sigma}_{j_t}^{(t)} = (\widehat{\sigma}_{k,j_t}^{(t)})_{1 \le k \le b} \in \mathbb{R}^b$ with the estimated embedding means $\widehat{\nu}_{j_t}^{(t)} \in \mathbb{R}^b$, see (2.16) and (2.19). We remark that these standard deviation estimates are rather sensitive in the choice of the prior uncertainties $\tau_t > 0$, we have used the choices given in Table 4. We calculate for the

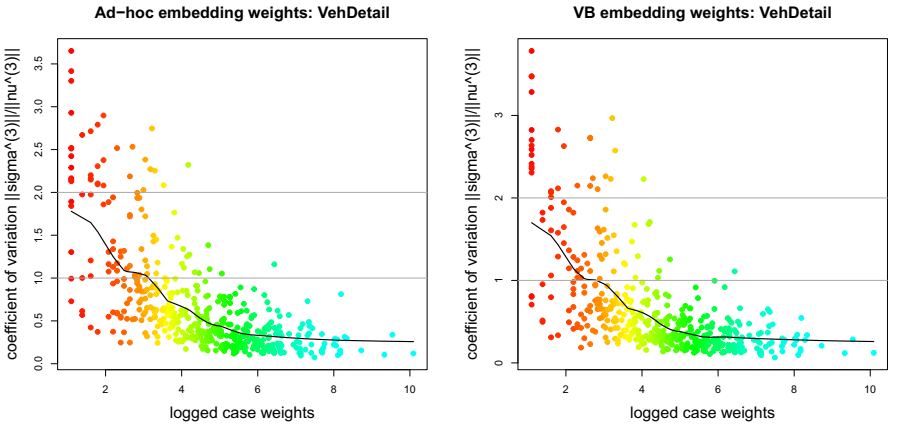

Fig. 8 Coefficients of variations $\|\widehat{\sigma}^{(3)}_{j_3}\|/\|\widehat{v}^{(3)}_{j_3}\|$ of the embeddings of the categorical variable `VehDetail` plotted against the logged case weights $\log(w^{(3)}_{j_3})$: (lhs) ad-hoc regularized, and (rhs) VB regularized; colors coincide with the ones of Fig. 6 (rhs)

last generation $t = T = 3$, `VehDetail`, the coefficients of variations $\|\widehat{\sigma}^{(3)}_{j_3}\|/\|\widehat{v}^{(3)}_{j_3}\|$, and this reflects uncertainty divided by the mean estimates, i.e., a relative uncertainty in the embeddings. These coefficients of variations are shown in Fig. 8 with horizontal lines at levels 1 and 2. We observe that for bigger case weights $w^{(3)}_{j_3}$, these coefficients of variations are well bounded below 2, which means that it is unlikely that $\mathbf{u}^{(3)}_{j_3}$ defined by (2.17) takes a value close to zero. This rejects the null hypothesis of level $a^{(3)}_{j_3}$ not being significant for prediction. Only for some levels that have scarce observations (low case weights), we cannot reject the null hypothesis.

4.4 Hierarchical categorical entity embeddings

In this section, we present the hierarchical regularization approaches introduced in Sect. 3. These are the Hierarchical Model H1 (3.7), the RNN layer encoding of the entity embedding given in (3.11), and the Transformer processed entity embeddings given in (3.13). In the previous section, we have seen that the type of regularization only has a small influence on the results. For this reason, we only consider the MAP regularization in the sequel, as it only involves the hyperparameters λ_t. A preliminary hyperparameter search has provided that we can use the same regularization parameters λ_t as in the non-hierarchical MAP case.

Table 3 presents the results, and we conclude that recognizing the hierarchical structure in our categorical covariates can further improve the models. In fact, these last three approaches provide the highest accuracy of all models considered here.

In Hierarchical Model H1, only the aggregated estimated increments $\widehat{\mathbf{u}}^{(3)}_{j_3} = \widehat{\Delta}^{(1)}_{j_1[j_3]} + \widehat{\Delta}^{(2)}_{j_2[j_2]} + \widehat{\Delta}^{(3)}_{j_3}$ enter the neural network; see (3.7). Figure 9 shows this aggregation recursively over the generations $1 \le t \le T = 3$ from left to right; the coloring

Table 3 Regularization of high-cardinality categorical covariates using the hierarchical structure with regularization parameters $\lambda_2 = \lambda_3 = 10^3$; figures are in 10^{-2}

		MAP case		
		KL div.	RMSE	MAE
(2)	Non-hierarchical: with `VehBrand`, `VehModel`	0.2212	2.4583	1.7592
(2)	Hierarchical H1: with `VehBrand`, `VehModel`	0.2144	2.4258	1.7415
(2)	RNN layer: with `VehBrand`, `VehModel`	0.2222	2.4695	1.7621
(2)	Transformer layer: with `VehBrand`, `VehModel`	0.2041	2.3759	1.7039
(3)	Non-hierarchical: with `VehBrand`, `VehModel`, `VehDetail`	0.1446	2.0501	1.3653
(3)	Hierarchical H1: with `VehBrand`, `VehModel`, `VehDetail`	0.1351	1.9941	1.3066
(3)	RNN layer: with `VehBrand`, `VehModel`, `VehDetail`	0.1354	2.0013	1.3020
(3)	Transformer layer: with `VehBrand`, `VehModel`, `VehDetail`	0.1294	1.9463	1.2802

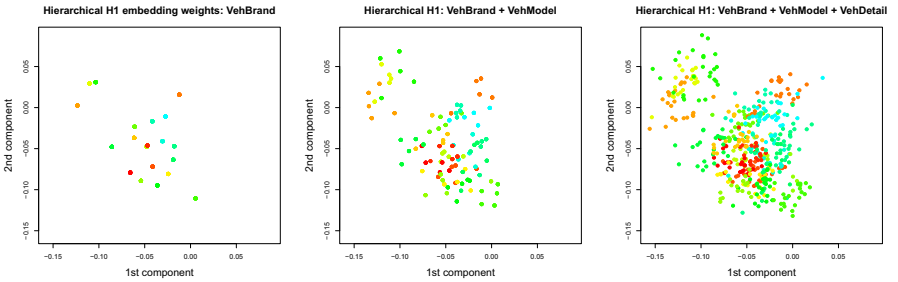

Fig. 9 Hierarchical Model H1: (lhs) estimates $\widehat{\mathbf{u}}^{(1)}_{j_1} = \widehat{\Delta}^{(1)}_{j_1}$, (middle) estimates $\widehat{\mathbf{u}}^{(2)}_{j_2} = \widehat{\Delta}^{(1)}_{j_1[j_2]} + \widehat{\Delta}^{(2)}_{j_2}$, and (rhs) estimates $\widehat{\mathbf{u}}^{(3)}_{j_3} = \widehat{\Delta}^{(1)}_{j_1[j_3]} + \widehat{\Delta}^{(2)}_{j_2[j_2]} + \widehat{\Delta}^{(3)}_{j_3}$; the coloring is w.r.t. `VehBrand`

is the same in all figures and it has been taken according to the case weights $w^{(1)}_{j_1}$ in the first generation `VehBrand`. We observe a clustering and refinement of the embeddings across the generations, which precisely reflects the motivation discussed in Sect. 3 of considering a hierarchical clustering across the generations. Interestingly, this aggregated estimate $\widehat{\mathbf{u}}^{(3)}_{j_3} \in \mathbb{R}^b$ carries sufficient information, so that it outperforms the non-hierarchical versions of Table 2. At the first sight, this seems surprising, because we lose information by aggregation. However, the crucial point is that we have a multi-dimensional embedding $\widehat{\mathbf{u}}^{(3)}_{j_3} \in \mathbb{R}^b$, $b > 1$, and the different dimensions may well play different roles in the subsequent regression model of the neural network $\mathrm{NN}_{\vartheta}(\boldsymbol{x}_i, \mathbf{u}^{(T)}_{j_T[i]})$. In our case, embedding dimension $b = 2$ seems sufficient, but more complex problems may require bigger embedding dimensions.

Figure 10 shows on the first row the Euclidean norms of the embeddings $\Delta^{(1)}_{j_1}$, $\Delta^{(2)}_{j_2}$ and $\Delta^{(3)}_{j_3}$, and on the second row the Euclidean norms of the RNN layer outputs $\boldsymbol{r}^{(1)}_{j_1}$, $\boldsymbol{r}^{(2)}_{j_2}$ and $\boldsymbol{r}^{(3)}_{j_3}$, for all levels in $A^{(1)}$, $A^{(2)}$ and $A^{(3)}$. The first row shows the regularization in the `VehModel` embeddings (middle) and the `VehDetail` embeddings (rhs), with smaller Euclidean norms for smaller case weights $w^{(t)}_{j_t}$. The lower row shows the RNN layer outputs given by see (3.10)

$$\begin{aligned}
\boldsymbol{r}^{(1)}_{j_1} &= \boldsymbol{\ell}^{\mathrm{RNN}}\left(\Delta^{(1)}_{j_1}, \mathbf{0}\right), \\
\boldsymbol{r}^{(2)}_{j_2} &= \boldsymbol{\ell}^{\mathrm{RNN}}\left(\Delta^{(2)}_{j_2}, \boldsymbol{r}^{(1)}_{j_1[j_2]}\right), \\
\boldsymbol{r}^{(3)}_{j_3} &= \boldsymbol{\ell}^{\mathrm{RNN}}\left(\Delta^{(3)}_{j_3}, \boldsymbol{r}^{(2)}_{j_2[j_3]}\right).
\end{aligned}$$

Figure 11 illustrates the RNN layer outputs, and similar to Fig. 9, we observe a clustering w.r.t. `VehBrand`. However, since the RNN layer performs non-linear transformations, we cannot simply aggregate from left to right in Fig. 11, but we have non-linear transformations and there also seems to be a rotation (clockwise by $\pi/2$) going from the inclusion of `VehBrand` in Fig. 11 (lhs) to the inclusion of `VehDetail` Fig. 11 (rhs).

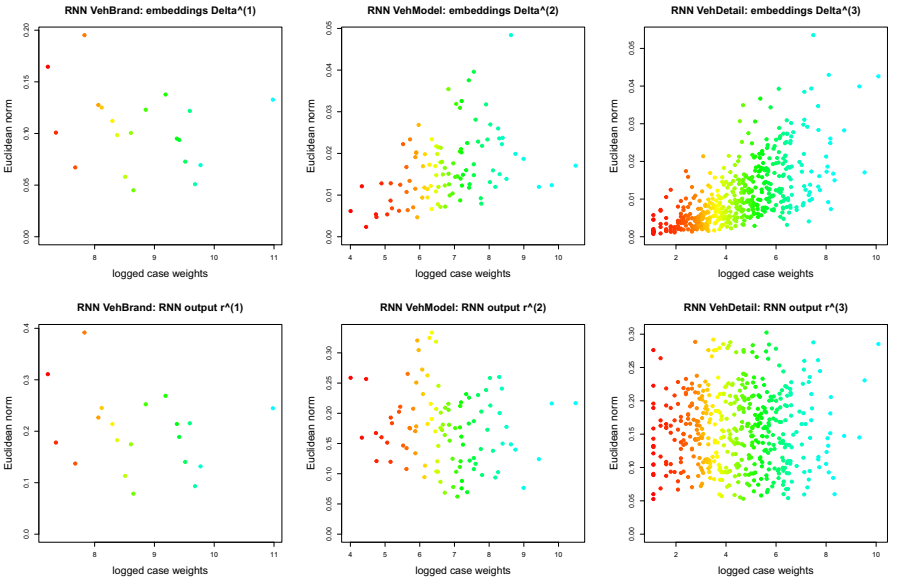

Fig. 10 RNN layer: (upper row) Euclidean norms of embeddings $\Delta_{j_1}^{(1)}$, $\Delta_{j_2}^{(2)}$ and $\Delta_{j_3}^{(3)}$; (lower row) Euclidean norms of RNN layer outputs $\boldsymbol{r}_{j_1}^{(1)}$, $\boldsymbol{r}_{j_2}^{(2)}$ and $\boldsymbol{r}_{j_3}^{(3)}$

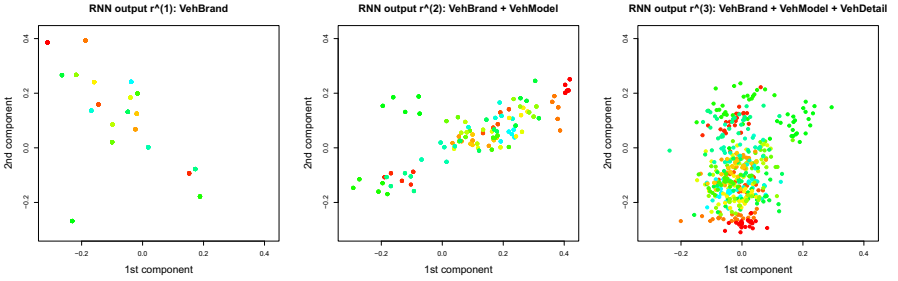

Fig. 11 RNN layer outputs: (lhs) $\boldsymbol{r}_{j_1}^{(1)}$, (middle) $\boldsymbol{r}_{j_2}^{(2)}$, and (rhs) $\boldsymbol{r}_{j_3}^{(3)}$; the coloring is w.r.t. `VehBrand`

Figure 12 shows the analogous plots to Fig. 10 but for the Transformer layer entity embedding processing. The upper row shows the Euclidean norms of the entity embeddings $\Delta_{j_1}^{(1)}$, $\Delta_{j_2}^{(2)}$ and $\Delta_{j_3}^{(3)}$. Again, we can clearly see stronger regularization in `VehModel` and `VehDetail` for levels with smaller case weights, middle and right plots on the upper row of Fig. 12. The lower row in Fig. 12 shows the Transformer layer outputs $\widetilde{\boldsymbol{r}}^{(1)}$, $\widetilde{\boldsymbol{r}}^{(2)}$ and $\widetilde{\boldsymbol{r}}^{(3)}$ of the entity embedding inputs $(\Delta_{j_1}^{(1)}, \Delta_{j_2}^{(2)}, \Delta_{j_3}^{(3)})$. Since we no longer have time-causality in the Transformer outputs, see Remarks 3.2, every output component has maximal cardinality being equal to the number of levels of the last generation `VehDetail`, i.e., this differs from the RNN plot in Fig. 10. Also because the Transformer output (3.12) involves several non-linear transformations, we sacrifice part of the interpretability for having a better predictive model. This completes our example.

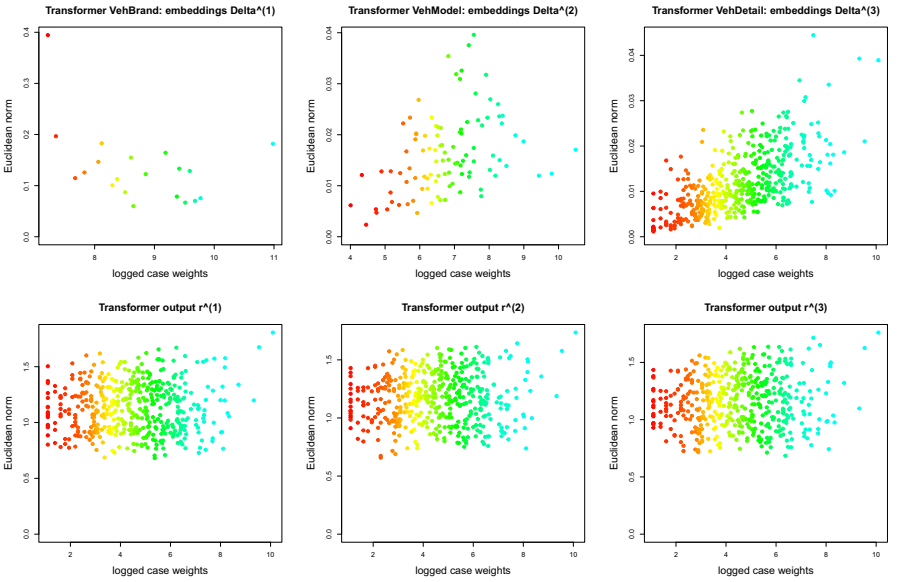

Fig. 12 Transformer layer: (upper row) Euclidean norms of embeddings $\Delta_{j_1}^{(1)}$, $\Delta_{j_2}^{(2)}$ and $\Delta_{j_3}^{(3)}$; (lower row) Euclidean norms of Transformer layer outputs $\widetilde{\boldsymbol{r}}^{(1)}$, $\widetilde{\boldsymbol{r}}^{(2)}$ and $\widetilde{\boldsymbol{r}}^{(3)}$

5 Conclusions

The research question studied in this paper is motivated by practical needs in insurance pricing, where one typically faces the problem of having high-cardinality categorical covariates, potentially having a hierarchical structure. If we do not consider regularization of such high-cardinality categorical covariates, we often receive poor predictive models. In this paper, we discuss regularization of high-cardinality categorical covariates, either using the maximal a posteriori (MAP) estimator which is equivalent to ridge regularization, or using variational Bayesian inference in a random-effects model for categorical covariates. Both approach provide comparable predictive performance. The former can be seen as a first-order Taylor approximation to the latter, and it seems that the first-order terms are sufficient (which also requires less hyperparameters). Our second contribution is that we show that predictive performance can be further improved, if the high-cardinality categorical covariates encoding considers the hierarchical structure, if there is any. The hierarchical structure can be visualized with trees and interpreted as time-series. This motivates to apply a recurrent neural network layer or a Transformer layer to process the hierarchical categorical random effects. We analyze these proposals on data which verify the improved predictive performance of these latter two proposals.

Acknowledgements The authors kindly thank Michael Mayer for posing us this interesting question of integrating high-cardinality categorical covariates into neural network regression models.

Funding Open access funding provided by Swiss Federal Institute of Technology Zurich

Data Availability The example is based on synthetic data which can be downloaded from: https://github.com/wueth/High-Cardinality-Covariates-Regularization

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Appendix A: An Empirical analysis of synthetic data and hyperparameter selection

This appendix presents an empirical analysis of the synthetic data used, the chosen regression function, and the hyperparameters for network fitting.

Listing 2 Choice of the true expected frequency $\mu^*(\boldsymbol{x}, \boldsymbol{z})$.

```
# transform the DrivAge variable
age_effect <- function(DrivAge) {
                    x <- (DrivAge - 66) / 60
                    0.05 + x^8 + 0.4*x^3 + 0.3*x^2 + 0.06*x
                        }

# effect of categorical variables
cat_effect <- function(cat1, seed) {
                    cat1 <- model.matrix(~cat1)[,-1]
                    set.seed(seed)
                    cat1 %*% (2*runif(ncol(cat1))-1)
                    }

#
true_model <- function(data) {
  log_lambda <- with(
    data,
      0.15 * Town +
      log(age_effect(DrivAge)) +
      (0.3 + 0.15 * Town) * VehPower / 100 +
      0.1 * VehPower / (VehWeight / 100)^2 +
      0.2 * cat_effect(VehBrand, 100) +
      0.2 * (2*Town-1) * cat_effect(VehModel, 100) +
      0.1 * VehUse * cat_effect(VehModel, 100) +
      0.3 * (2*VehUse-1) *cat_effect(VehDetail, 300) +
      0.03 * VehAge
  )
  exp(log_lambda)
}
```

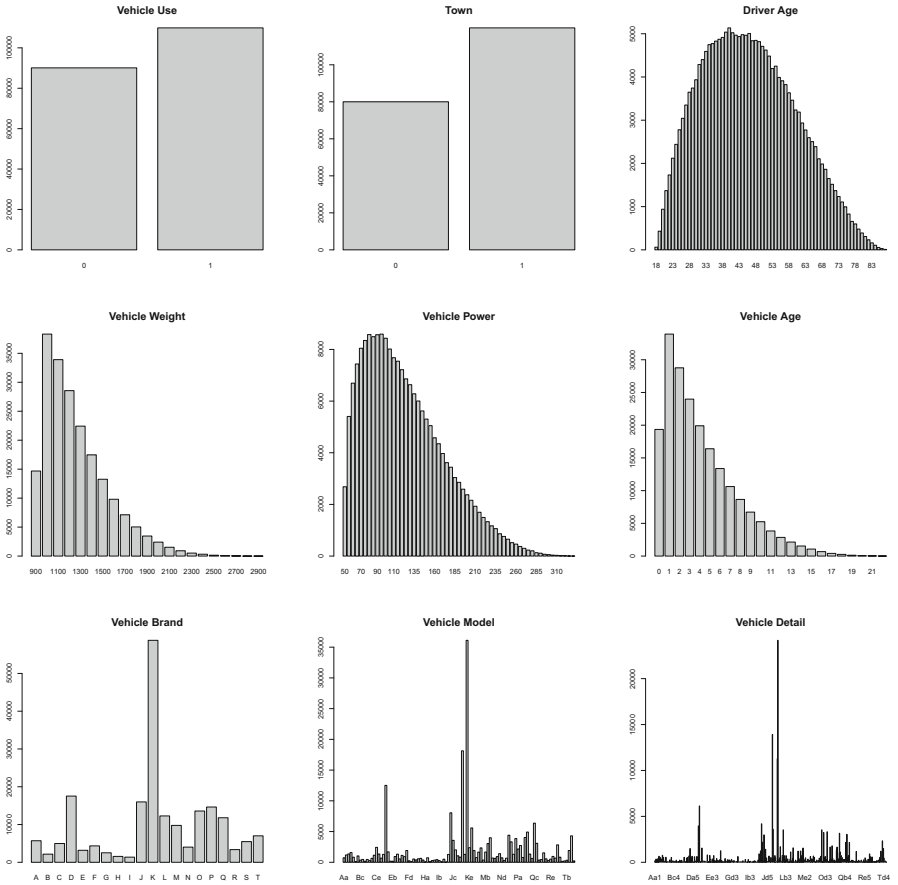

Fig. 13 Numbers of policies of each covariate level; in total we have $n = 199,971$ insurance policies

Table 4 Hyperparameters used to receive Table 2

		Network (2.30)					GLMMNet (4.3)				
		λ_1	λ_2	τ_2^2	λ_3	τ_3^2	λ_1	λ_2	τ_2^2	λ_3	τ_3^2
(2)	MAP regularization	0	10^3	–	–	–	0	10^3	–	–	–
(2)	Ad-hoc regularization	0	10^3	10^{-6}	–	–	0	10^3	10^{-4}	–	–
(2)	VB regularization	0	10^3	10^{-4}	–	–	0	10^3	10^{-5}	–	–
(3)	MAP regularization	0	10^3	–	10^3	–	0	10^3	–	10^3	–
(3)	Ad-hoc regularization	0	10^3	10^{-6}	10^3	10^{-5}	0	10^3	10^{-6}	10^3	10^{-5}
(3)	VB regularization	0	10^3	10^{-4}	10^3	10^{-3}	0	10^3	10^{-5}	10^3	10^{-5}

Table 5 RMSE values: benchmark models with high-cardinality categorical covariates using $b = 2$ dimensional embedding layers for the networks (2.30) and (4.3); numbers in round brackets $(\cdot)$ in the 1st column indicate the number of considered categorical covariate components; figures are in 10^{-2}

		Rooted mean squared error (RMSE)		
		Network (2.30)	GLMMNet (4.3)	LightGBM
(0)	Null model (empirical mean)	5.2571	5.2571	5.2571
(0)	w/o categorical covariates	3.3196	3.3335	3.3058
(1)	With `VehBrand`	2.7201	2.9170	2.7644
(1)	With `VehModel`	2.4597	2.8094	2.6210
(1)	With `VehDetail`	2.6130	2.9929	2.5035
(2)	With `VehBrand`, `VehModel`	2.5245	2.8053	2.6870
(3)	With `VehBrand`, `VehModel`, `VehDetail`	2.7011	3.0008	2.4688

Table 6 RMSE values: regularization of high-cardinality categorical covariates `VehModel` and `VehDetail`; numbers in round brackets $(\cdot)$ in the 1st column indicate the number of considered categorical covariate components; the selected hyperparameters are given in Table 4; figures are in 10^{-2}

		Rooted mean squared error (RMSE)	
		Network (2.30)	GLMMNet (4.3)
(2)	No regularization: with `VehBrand`, `VehModel`	2.5245	2.8053
(2)	MAP regularization: with `VehBrand`, `VehModel`	2.4583	2.7537
(2)	Ad-hoc regularization: with `VehBrand`, `VehModel`	2.4876	2.7525
(2)	VB regularization: with `VehBrand`, `VehModel`	2.5211	2.7519
(3)	No regularization: with `VehBrand`, `VehModel`, `VehDetail`	2.7011	3.0008
(3)	MAP regularization: with `VehBrand`, `VehModel`, `VehDetail`	2.0501	2.6571
(3)	Ad-hoc regularization: with `VehBrand`, `VehModel`, `VehDetail`	2.0663	2.6651
(3)	VB regularization: with `VehBrand`, `VehModel`, `VehDetail`	2.0381	2.6608

Table 7 MAE values: benchmark models with high-cardinality categorical covariates using $b = 2$ dimensional embedding layers for the networks (2.30) and (4.3); numbers in round brackets $(\cdot)$ in the 1st column indicate the number of considered categorical covariate components; figures are in 10^{-2}

		Mean absolute error (MAE)		
		Network (2.30)	GLMMNet (4.3)	LightGBM
(0)	Null model (empirical mean)	3.7977	3.7977	3.7977
(0)	w/o categorical covariates	2.3904	2.3992	2.3867
(1)	With `VehBrand`	1.9438	2.2137	2.0754
(1)	With `VehModel`	1.7401	2.0831	1.9624
(1)	With `VehDetail`	1.7202	2.1241	1.8385
(2)	With `VehBrand`, `VehModel`	1.7869	2.0839	2.0207
(3)	With `VehBrand`, `VehModel`, `VehDetail`	1.7671	2.1231	1.8239

Table 8 MAE values: regularization of high-cardinality categorical covariates `VehModel` and `VehDetail`; numbers in round brackets $(\cdot)$ in the 1st column indicate the number of considered categorical covariate components; the selected hyperparameters are given in Table 4; figures are in 10^{-2}

		Mean absolute error (MAE)	
		Network (2.30)	GLMMNet (4.3)
(2)	No regularization: with `VehBrand`, `VehModel`	1.7869	2.0839
(2)	MAP regularization: with `VehBrand`, `VehModel`	1.7592	2.0524
(2)	Ad-hoc regularization: with `VehBrand`, `VehModel`	1.7848	2.0482
(2)	VB regularization: with `VehBrand`, `VehModel`	1.8143	2.0530
(3)	No regularization: with `VehBrand`, `VehModel`, `VehDetail`	1.7671	2.1231
(3)	MAP regularization: with `VehBrand`, `VehModel`, `VehDetail`	1.3653	1.9264
(3)	Ad-hoc regularization: with `VehBrand`, `VehModel`, `VehDetail`	1.3597	1.9285
(3)	VB regularization: with `VehBrand`, `VehModel`, `VehDetail`	1.3438	1.9248

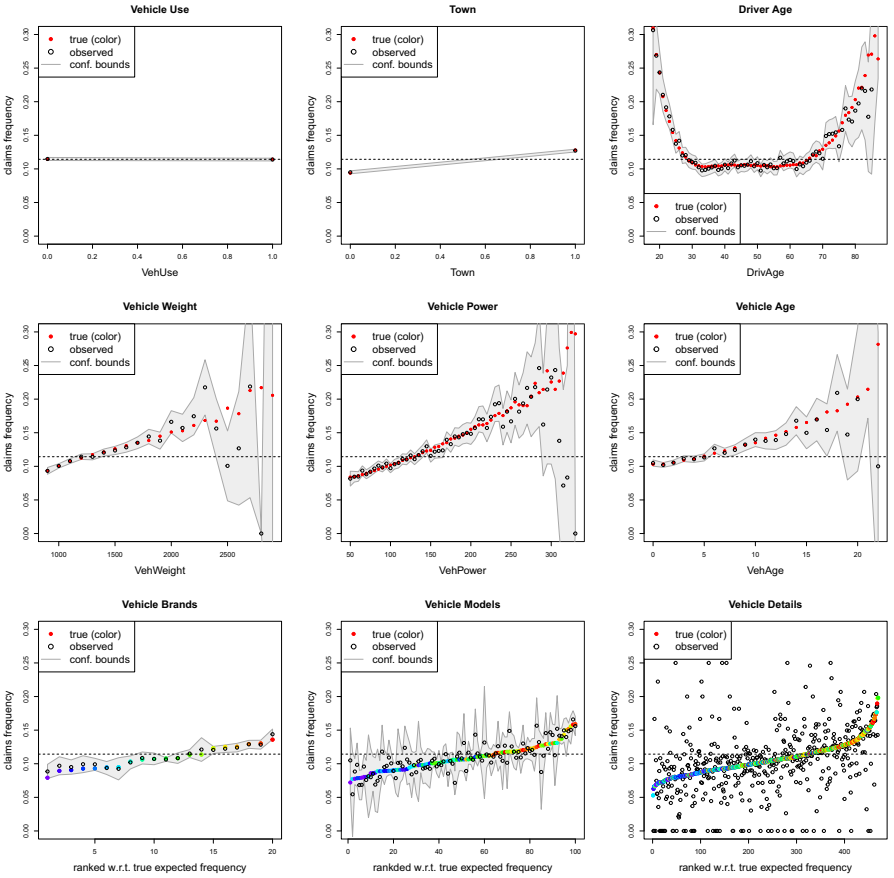

Fig. 14 Marginal frequencies (true and empirical) of all covariates. The y-scale is identical in all plots; the gray area computes two standard deviations empirical confidence bounds for all levels using a Poisson assumption; the coloring in the graphs `VehBrand`, `VehModel`, `VehDetail` (last row) is identical indicating the `VehBrand` by different colors

References

Antonio, K., & Zhang, Y. (2014). Linear mixed models. In E. W. Frees, G. Meyers, & R. A. Derrig (Eds.), *Predictive modeling applications in actuarial science* (Vol. I, pp. 182–216). Cambridge University Press.

Avanzi, B., Taylor, G., Wang, M., Wong, B. (2024). Machine learning with high-cardinality categorical features in actuarial applications. *ASTIN Bulletin 54*(2) (in press)

Brébisson, de A., Simon, É., Auvolat, A., Vincent, P., & Bengio, Y. (2015). Artificial neural networks applied to taxi destination prediction. arXiv:1508.00021

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Learning Intelligence, 35*(8), 1798–1828.

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

Bengio, Y., Schwenk, H., Senécal, J.-S., Morin, F., & Gauvain, J.-L. (2006). Neural probabilistic language models. In D. E. Holmes & L. C. Jain (Eds.), *Innovations in machine learning, Studies in fuzziness and soft computing* (Vol. 194, pp. 137–186). Springer.

Bühlmann, H., & Gisler, A. (2005). *A course in credibility theory and its applications*. Springer.
Bühlmann, H., & Jewell, W. S. (1987). Hierarchical credibility revisited. *Bulletin of the Swiss Association of Actuaries, 1987*(1), 35–54.
Campo, B. D. C., & Antonio, K. (2023). Insurance pricing with hierarchically structured data an illustration with a workers' compensation insurance portfolio. *Scandinavian Actuarial Journal*. (**in press**).
Campo, B. D. C., & Antonio, K. (2023). On clustering levels of a hierarchical categorical risk factor. arXiv:2304.09046
Chollet, F., & Allaire, J. J., et al. (2017). R interface to Keras. https://github.com/rstudio/keras
Delong, Ł., & Kozak, A. (2023). The use of autoencoders for training neural networks with mixed categorical and numerical features. *ASTIN Bulletin, 53*(2), 213–232.
Guo, C., Berkhahn, F. (2016). Entity embeddings of categorical variables. arXiv:1604.06737
Hastie, T., Tibshirani, R., & Wainwright, M. (2015). *Statistical learning with sparsity: The Lasso and generalizations*. CRC Press.
Jewell, W. S. (1975). The use of collateral data in credibility theory: A hierarchical model. *Giornale dell'Instituto Italiano degli Attuari, 38*, 1–16.
Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). `LightGBM`: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems, 30*, 3146–3154.
Kingma, D. P., & Welling, M. (2019). An introduction to variational autoencoders. *Foundations and Trends in Machine Learning, 12*(4), 307–392.
Mayer, M., Meier, D., & Wüthrich, M. V. (2023). SHAP for actuaries: Explain any model. SSRN Manuscript ID 4389797.
Odaibo, S. G. (2019). Tutorial: Deriving the standard variational autoencoder (VAE) loss function. arXiv:1907.08956
Ohlsson, E. (2008). Combining generalized linear models and credibility models in practice. *Scandinavian Actuarial Journal, 2008*(4), 301–314.
Richman, R. (2021). AI in actuarial science—A review of recent advances—Part 1. *Annals of Actuarial Science, 15*(2), 207–229.
Richman, R. (2021). AI in actuarial science—A review of recent advances—Part 2. *Annals of Actuarial Science, 15*(2), 230–258.
Schelldorfer, J., Wüthrich, M. (2019). Nesting classical actuarial models into neural networks. SSRN Manuscript ID 3320525.
Simchoni, G., & Rosset, S. (2022). Integrating random effects in deep neural networks. arXiv:2206.03314.
Tikhonov, A. N. (1943). On the stability of inverse problems. *Doklady Akademii Nauk SSSR, 39*(5), 195–198.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. arXiv:1706.03762v5
Wüthrich, M. V., & Merz, M. (2023). *Statistical foundations of actuarial learning and its applications*. Springer Actuarial. https://link.springer.com/book/10.1007/978-3-031-12409-9