Temporal Dynamics in Generative Models

Masterarbeit im Studiengang Master of Science in Mathematik

verfasst von

Maren Hackenberg

am Mathematischen Institut der Fakultät für Mathematik und Physik

an der Albert-Ludwigs-Universität Freiburg

unter Anleitung von

Prof. Dr. Harald Binder

Acknowledgement

First and foremost, I thank Harald Binder for his invaluable support and encouragement, for introducing me to a fascinating field of research and giving me the opportunity to write this thesis, and for never running out of good ideas.

Also, I am very grateful to Peter Pfaffelhuber for co-supervising this thesis.

I thoroughly appreciate the positive and supportive atmosphere at the IMBI and feel honoured to have met and worked with such fantastic colleagues. In particular, I would like to thank the members of the AG Machine Learning: Caroline Broichhagen, Daniela Zöller, Göran Köber, Kiana Farhadyar, Martin Treppner, Moritz Hess and Stefan Lenz.

Considering myself extremely lucky to have friends both very supportive and skilled in mathematics and statistics, I am deeply grateful to Anselm Hudde, Maja von Cube, Nora Scherer and Pascal Schlosser for the time they took to discuss and proofread this thesis and for all their helpful comments.

My final thanks go to my brother Paul, who so generously helped me out with the necessary hardware, to my parents Anke and Thomas and to Anselm for always standing behind me and for unconditionally and unfailingly believing in me.

Contents

CHAPTER 1

Introduction

When considering biomedical time-series data from individuals, understanding and capturing the underlying development patterns on an individual level is the first and essential step towards personalised medicine. Yet, such data are often characterised by a sparse, highly irregular time grid of measurements and individual-specific development patterns, which complicates the corresponding modelling task.

In this thesis, I develop a generative deep learning model that infers individual-specific dynamics in a low-dimensional latent representation as solutions of ordinary differential equations (ODEs) from sparse and irregularly observed time-series data. The method is inspired by recent advances on combining black-box deep learning with explicit mechanistic modelling by differential equations and motivated by a scenario from the German National Cohort (NAKO) epidemiologic study.

Specifically, I simulate a data scenario based on a NAKO sub study: For each individual, an extensive characterisation with measurements of many variables is made at a baseline time point. To assess the level of measurement error in the data, a smaller subset of these variables is measured again at a second time point that is different for all individuals, i.e., the time spans between first and second measurement vary. Consequently, the data reflects individual-specific dynamic processes that are observed sparsely and irregularly with measurement noise. My aim is to develop a model that extracts underlying individual development patterns in such a setting despite the measurement uncertainty and the sparsity (only two time points) and irregularity of the time grid. Here, this irregular grid can potentially enable modelling of more complex dynamics by combining measurements of similar individuals.

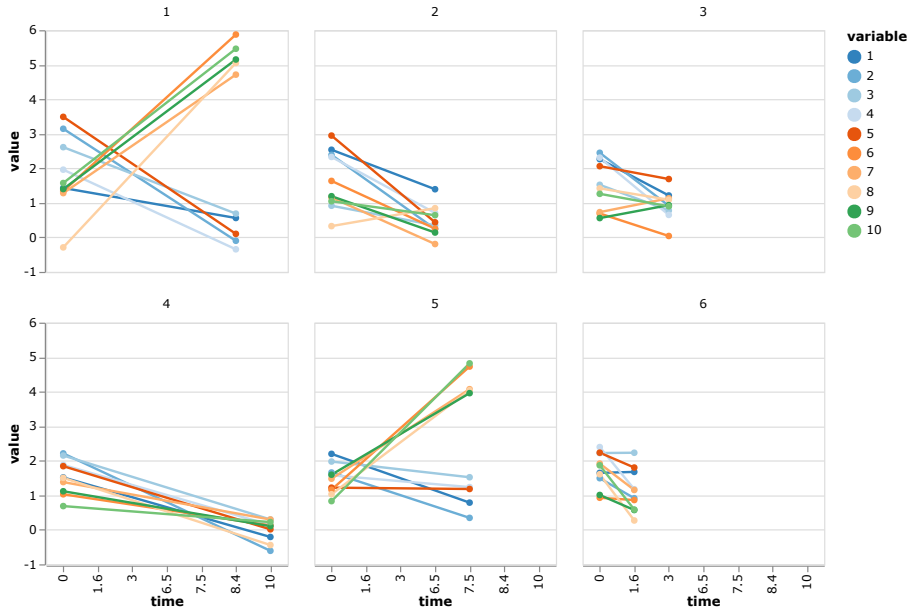


FIGURE 1. Exemplary data structure from simulated data. Each panel represents one individual, dots represent individual measurements and colors indicate different variables. The x-axis denotes the time of measurements and the y-axis their values.

Figure 1 illustrates the structure of the simulated data setting: The time point of the second measurement varies as contrasted by individuals 1 and 6. Different development patterns are reflected by two groups of individuals, one with an upward trend in some variables and a downward trend in others (e.g., individuals 1 and 5), and the other one with a downward trend in all the variables (e.g., individuals 2 and 4). Furthermore, the observations are characterised by groups of variables that share common trends, suggesting an underlying lower-dimensional structure.

To build a representation of this lower-dimensional structure of the development patterns and in this way recover the central factors of variation underlying the data in an unsupervised way, I employ deep learning, specifically a variational autoencoder (VAE). I adapt the VAE and constrain its representation of the data in a latent space to model smooth trajectories by imposing an ODE system on that space. Furthermore, I assume inter-individual differences in development patterns

to be grounded in differences in the variables measured only at baseline and use them to infer an individual-specific set of ODE parameters. To be able to model more complex underlying dynamics, I extend the model and enrich every individual's information by assigning to it a batch of individuals with similar underlying development patterns. The combination of all second time point measurements in the batch then serves as proxy information on the common dynamics at multiple time points. By training the model on these batches, I thus exploit the irregularity of second measurement time points to address the problem of time sparsity.

Using the simulated data described above, I investigate the capability of the model to recover individual trajectories from both linear and non-linear two-dimensional ODE systems with two or four unknown parameters and to infer for a given observation a group of individuals with similar trajectories.

The proposed method is based on the central idea of bringing together deep learning and differential equations. This can be seen as an approach that bridges the gap between what has been referred to as the "two cultures in the use of statistical modeling to reach conclusions from data" ([**1**, p. 1]): While inferring a low-dimensional latent representation in an unsupervised way directly from the data without specific assumptions on its structure with a deep neural network belongs to the "algorithmic modelling culture", the assumption of an explicit data model in the form of an underlying system of differential equations from which observations are generated with some random measurement noise represents the "data modelling culture".

In general, formulating explicit scientific models allows to explicitly encode known structures of the data-generating forces such as physical laws or constraints into the model and treat other factors as noise, while the black-box nature of deep neural networks generally lacks interpretability. However, explicit stochastic or mechanistic data models make simplifying assumptions on the generating process that often cannot be validated. Algorithmic approaches such as neural networks have been successfully applied in many domains to approximate complex functions or learn

patterns from data in an unsupervised way without explicit assumptions on the underlying data-generating process, but require large amounts of training data.

Instead of viewing these two approaches to statistical modelling as opposing each other, recent lines of research attempt to reconcile them and combine their respective advantages: The methodology of universal differential equations (UDEs) developed in [**2**] aims at augmenting scientific models with machine-learnable structures as a step towards "data-efficient machine learning". With the UDEs, the authors propose differential equation models defined by a mechanistic and deterministic model that incorporates prior scientific knowledge, while a part of the equation contains an embedded black-box algorithm like a neural network that can account for missing or unobserved parts of the process. The authors employ their approach in several modelling tasks across scientific disciplines to show how UDEs can provide both interpretable and flexible models while requiring less data than pure machine learning based approaches and being efficiently trained within modern differential programming frameworks.

My method is inspired by the recently developed neural ODEs ([**3**]). Here, the authors introduce continuous-depth neural networks that parameterise the derivative of the hidden state and apply their idea to develop a latent continuous time-series model. While this model is also based on a VAE architecture with an ODE system governing the latent space, it assumes a fixed time grid with dense, regularly spaced observations. To overcome this limitation, the authors proposed an extension for irregularly sampled time series data in a follow-up work ([**4**]): Originally, a recurrent neural network (RNN) with a discrete sequence of hidden states is used to encode the observed time series into a latent representation, which assumes sequences with equidistant time steps. In [**4**], this is generalised to an ODE-RNN encoder with continuously changing hidden states and an ODE specifying their dynamics.

A conceptually similar method to model second-order ODEs within a VAE framework is presented by the ODE^{2}VAE model ([**5**]). Here, the latent representation consists of a position and a velocity vector inferred separately from the input time

series with an additional Bayesian network employed to handle uncertainty in the dynamics. The model does not, however, explicitly account for any uncertainty in the observation or measurement of these dynamics.

The GRU-ODE-Bayes approach of [**6**] also integrates ODE modelling with Bayesian inference, but is based on a different neural network architecture. The authors aim at continuous modelling of sporadically observed time series data with stochastic differential equations (SDEs). To this end, they combine a smooth version of the gated recurrent unit (GRU), basically a RNN with continuous instead of discrete hidden state updates similar to the ODE-RNN in [**4**], with a Bayesian update. The GRU propagates the latent state forward and allows for an update in a Bayesian fashion whenever a new observation becomes available. While the idea of alternating between a prediction and filtering step is similar to the Kalman filter, the model does not rely on the linearity or linearisation of the hidden state update and its formulation of the dynamics as ODE is better suited to model long-term dependencies and capture more complex dynamics. Since the model is not built on a VAE architecture, it neither infers a latent representation that compresses the main factors of variation in lower dimensions nor permits to generate new data samples. Although it solves a SDE and hence allows to generate trajectories by sampling paths of the SDE solution, the distribution of these trajectories is determined by the stochastic process that solves the SDE and does not necessarily account for noise factors in the original data observations.

More generally, several approaches integrate modelling of dynamic processes into a VAE architecture without using ODEs. The authors of [**7**] propose a Gaussian Process-VAE (GP-VAE) that models temporal dynamics as Gaussian processes in latent space. Here, again, many time points are needed to accurately infer the Gaussian process capturing the dynamics and the time series in latent space is represented by a discrete series of steps rather than a smooth function. Consequently,

the structure of the latent space does not immediately allow for inter- or extrapolation of the time series, i.e, no data can be generated for time points not observed in the training data.

The temporal difference VAE (TD-VAE) developed in [**8**] is build on the idea of forming a hidden belief state that encodes a deterministic representation of the filtering posterior of the hidden state given all observations up to a specific time. This belief state is computed based on a sufficient statistic of the future given past observations and is propagated forward from one time step to the next. The model is trained by comparing observations from two (not necessarily subsequent) time points and assumes a densely and regularly sampled time grid. While the TD-VAE infers an abstract representation of the data in the hidden belief state and is able to learn from separated time points without backpropagating through the entire time interval and predict several time steps in the future, it does not learn to represent the dynamics as an explicit continuous trajectory.

Compared to the approaches mentioned above, the model proposed in this thesis is the only one inferring individual-specific dynamics by using additional baseline information and modelling individual development patterns as smooth ODE solutions based on an extremely sparse and irregular time grid of noisy measurements. The VAE architecture allows for the generation of new samples from the data distribution, while the representation of individual temporal patterns as ODE solutions permits straightforward inter- and extrapolation of the learned time series by solving the ODE at different time points than the ones originally observed.

In this thesis, I present the approach in detail. Chapter 2 establishes the theoretical background. In Chapter 3, I present the methodology of my approach including a derivation of the training objective and a method to realise the model extension to batches of similar individuals. Chapter 4 is devoted to applications of the model in various simulated data settings based on different characteristics of temporal development patterns. Finally, I discuss my results, the limitations of the proposed approach and possible directions for future research in Chapter 5.

CHAPTER 2

Background

1. Generative deep learning

In this chapter, I outline the central concepts and objects that form the basis for my method. First, I define (deep) neural networks and state their central property of being general function approximators. Next, I explain the training procedure of neural networks and finish by giving a brief introduction to generative models as a special class of deep learning algorithms. The following sections are loosely based on [**9**, Chapter 6, pp. 164-194 and 200-209; Chapter 20, pp. 651-662].

1.1. Definitions.

DEFINITION 1. For $n, k_1, \dots, k_{n+1} \in \mathbb{N}$ and distinct continuous functions $g_1, \dots, g_n$, where $g_i : \mathbb{R}^{k_i} \to \mathbb{R}^{k_{i+1}}$ for all $i = 1, \dots, n$ a **feedforward neural network (FNN)** f is the function composition

$$\begin{aligned} f : \mathbb{R}^{k_1} &\to \mathbb{R}^{k_{n+1}}, \\ x &\mapsto (g_n \circ g_{n-1} \circ \cdots \circ g_2 \circ g_1)(x). \end{aligned} \tag{2.1}$$

The term "neural network" stems from a loose analogy between the mathematical models and a biological model of the way information is processed through a network of brain cells or neurons. The term "feedforward" refers to the fact that such a neural network maps an input value x through the functions that define f to an output value $f(x)$ without allowing for any feedback connections or recurrence.

A function $g_i : \mathbb{R}^{k_i} \to \mathbb{R}^{k_{i+1}}, i \in \{1, \dots, n\}$ of the composition f is called a **layer** of the neural network. Additionally, the input vector $x \in \mathbb{R}^{k_1}$ is often represented as its own layer, called the **input layer** of the neural network. The last layer, where the transformation g_n is applied to produce the final output $f(x)$ of the neural network,

is called **output layer**, i.e., a neural network satisfying Defitinion 1 always consists of at least one input and one output layer. For $n > 1$, the layers between the input and output layer are called **hidden layers**, and if there is more than one hidden layer, i.e., if $n > 2$, the neural network is called **deep**.

Typically, a $g_i : \mathbb{R}^{k_i} \to \mathbb{R}^{k_{i+1}}, i \in \{1, \dots, n\}$ is represented as an affine linear transformation followed by a typically non-linear **activation function**: For some $x \in \mathbb{R}^{k_i}$, g_i would be of the form

$$g_i(x) = \phi_i.(W_i x + b_i), \tag{2.2}$$

for a continuous function $\phi_i : \mathbb{R} \to \mathbb{R}$, a matrix $W_i \in \mathbb{R}^{k_{i+1} \times k_i}$ and a vector $b_i \in \mathbb{R}^{k_{i+1}}$, where $\phi.$ denotes the element-wise application of the function ϕ to each component of the vector $W_i x + b_i$.

In the following, I assume that each function $g_i : \mathbb{R}^{k_i} \to \mathbb{R}^{k_{i+1}}, i \in \{1, \dots, n\}$ defining an FNN as in (2.1) is of the form (2.2). The matrices $W_i, i \in \{1, \dots, n\}$ are called **weights** of f and the vectors $b_i, i \in \{1, \dots, n\}$ are called **biases** of the FNN f. The set $\{W_1, \dots, W_n, b_1, \dots, b_n\}$ of all weights and biases is also called the set of **parameters** of the FNN.

Commonly used activation functions include
the **logistic** function $S : \mathbb{R} \to \mathbb{R}, x \mapsto \frac{\exp(x)}{\exp(x)+1}$,
the **hyperbolic tangent** function $\tanh : \mathbb{R} \to \mathbb{R}, x \mapsto \frac{\exp(2x)-1}{\exp(2x)+1}$
and the **rectified linear unit (ReLU)** function $\mathrm{ReLU} : \mathbb{R} \to \mathbb{R}, x \mapsto \max\{0, x\}$.

From Definition 1 we see that a neural network is a function composed of a finite number of linear and non-linear transformations. They are also referred to as **multilayer perceptrons** and form the basic building blocks of deep learning models.

Due to the structure of a feedforward flow of information through a composition of functions not involving recurrence, the mathematical FNNs can be represented as directed acyclic graphs: For a layer $i \in \{1, \dots, n\}$, let $o_i := (g_{i-1} \circ \cdots \circ g_1)(x) \in \mathbb{R}^{k_i}$ be the output of the previous layer $i - 1$. Typically, the individual dimensions of the output of the current layer $g_i(o_i)^{(1)}, \dots, g_i(o_i)^{(k_{i+1})}$ correspond to the nodes of

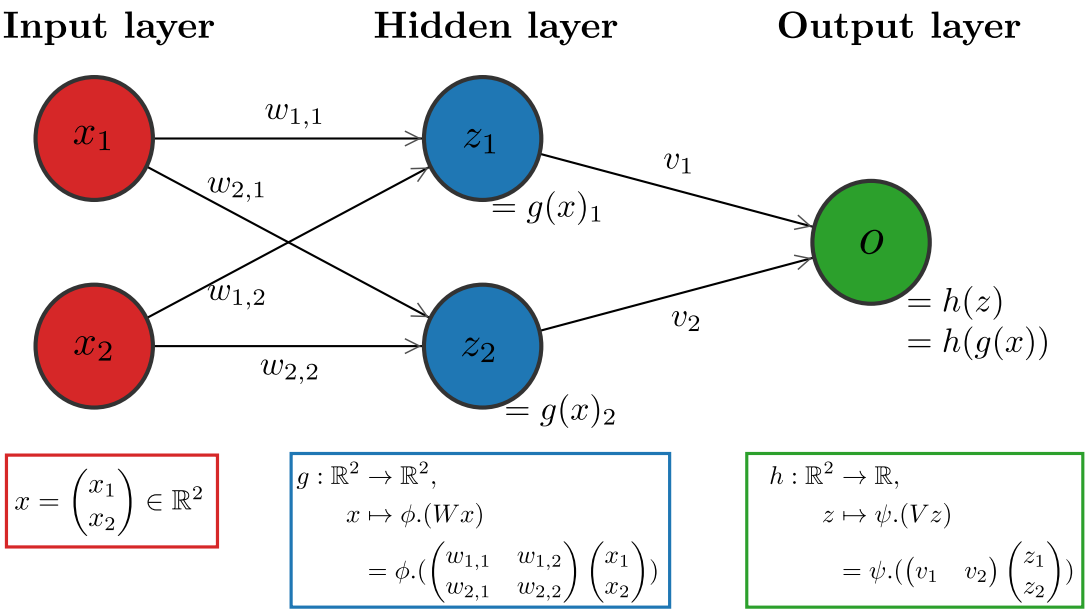


FIGURE 2. Simple FNN example.

the graph in layer i, that are also called **units**. The edges of the graph denote the mutual dependencies of the layers as defined by the function compositions.

Figure 2 illustrates the representation of a simple FNN as a directed acyclic graph. Here, the network consists of three layers, an input layer with two nodes, one hidden layer with two nodes and an output layer with one node. The network thus defines a function

$$f : \mathbb{R}^2 \to \mathbb{R},$$
$$x \mapsto f(x) = h(g(x)),$$

where the functions $g : \mathbb{R}^2 \to \mathbb{R}^2, x \mapsto g(x) = \phi.(Wx)$, $h : \mathbb{R}^2 \to \mathbb{R}, z \mapsto h(z) = \psi.(Vz)$ defining the hidden and output layer, respectively, each consist of a linear transformation with a weight matrix $W \in \mathbb{R}^{2\times 2}$ resp. $V \in \mathbb{R}^2$ followed by a non-linear activation function $\phi : \mathbb{R} \to \mathbb{R}$ resp. $\psi : \mathbb{R} \to \mathbb{R}$.

1.2. Neural networks as universal function approximators. Generally, FNNs are applied to approximate functions, that, e.g., map some input data x to a label y and thus serve as a classifier. Prior to their successful application in many areas such as pattern and sequence recognition, medical diagnosis, finance or

game playing ([**9**, pp. 22-26]), the basic theoretical property of feedforward neural networks as universal function approximators has been investigated and proved in several versions.

The **universal approximation theorem** states that a feed-forward neural network with a single layer in addition to the input layer containing a finite number of nodes can approximate any Borel measurable function on a compact subset of a finite-dimensional $\mathbb{R}$-vector space under certain assumptions on the activation function ([**9**, pp. 194f.]). Simple FNNs can thus in theory represent a wide variety of interesting functions when given appropriate parameters; however, the theorem does not say anything about whether those parameters can be actually found by an explicit algorithm.

One of the first versions of the theorem was formulated and proved by George Cybenko in 1989 ([**10**] and is presented following [**10**, pp. 305-307].

DEFINITION 2. A function $\sigma : \mathbb{R} \to \mathbb{R}$ is called **sigmoidal**, if

$$\sigma(x) \to \begin{cases} 1 & \text{as} \quad x \to +\infty \\ 0 & \text{as} \quad x \to -\infty. \end{cases}$$

THEOREM 3 (Universal approximation theorem). *Let σ be a continuous sigmoidal function, let $I_n = [0,1]^n \subset \mathbb{R}^n$ denote the n-dimensional unit cube in $\mathbb{R}^n$. Then, functions $G : I_n \to \mathbb{R}$ consisting of finite sums of the form*

$$G(x) = \sum_{i=1}^{N} v_i \sigma(w_i^\top x + b_i), \tag{2.3}$$

where $x \in I_n, v_i, b_i \in \mathbb{R}, w_i \in \mathbb{R}^n$ for all $i = 1, \ldots, N$ are dense in $\mathcal{C}(I_n)$ with respect to the supremum norm, i.e., for any continuous function $f \in \mathcal{C}(I_n)$ and any $\varepsilon > 0$, there exist constants $v_i, b_i \in \mathbb{R}$ and vectors $w_i \in \mathbb{R}^n$ for all $i = 1, \ldots, N$, such that for $G(x)$ as in (2.3)

$$| G(x) - f(x) | < \varepsilon \quad \textit{for all } x \in I_n.$$

The proof is given for a slightly different scenario of a **discriminatory** function σ and is based on an application of the Hahn-Banach theorem and the Riesz representation theorem. The fact that the Hahn-Banach theorem essentially follows from the axiom of choice already hints at the nature of the universal approximation theorem as a theoretical result without a constructive proof that does not provide us with any practical instruction of how to actually construct these approximating networks for a given function.

To prove the theorem, we first need to define discriminatory functions:

DEFINITION 4. A function $\sigma : \mathbb{R} \to \mathbb{R}$ is called **discriminatory**, if for a measure μ on I_n, it follows from

$$\int_{I_n} \sigma(v^\top x + b) d\mu(x) = 0 \quad \text{for all } v \in \mathbb{R}^n, b \in \mathbb{R}$$

that $\mu \equiv 0$.

With that definition, we can reformulate the universal approximation theorem as follows:

THEOREM 5. *Let σ be a continuous discriminatory function. Then, finite sums of the form*

$$G(x) = \sum_{i=1}^{N} v_i \sigma(w_i^\top x + b_i), \tag{2.4}$$

where $x \in I_n, v_i, b_i \in \mathbb{R}, w_i \in \mathbb{R}^n$ for all $i = 1, \ldots, N$ are dense in $\mathcal{C}(I_n)$ with respect to the supremum norm.

PROOF. The following proof is a slightly more detailed version of the proof in [**10**, p. 306]. Let $\mathcal{G} \subset \mathcal{C}(I_n)$ be the set of functions of the form $G(x)$ as in (2.4). It is clear that $\mathcal{G}$ is a linear subspace of $\mathcal{C}(I_n)$. We prove by contradiction that the closure of $\mathcal{G}$ with respect to the supremum norm is all of $\mathcal{C}(I_n)$.

Assume for contradiction that the closure $\bar{\mathcal{G}}$ is a proper subspace of $\mathcal{C}(I_n)$. The space $\mathcal{C}(I_n)$ is complete with respect to the supremum norm. Then, it follows from

the Hahn-Banach theorem that there is a bounded linear functional L on $\mathcal{C}(I_n)$ with the property that $L \neq 0$ but L vanishes on $\bar{\mathcal{G}}$ and thus also on $\mathcal{G}$.

$\mathcal{C}(I_n)$ is a Hilbert space with respect to the L^2 inner product. Thus, by the Riesz representation theorem, there is a function $l \in \mathcal{C}(I_n)$ such that

$$L(f) = \langle l, f \rangle_{L^2} = \int_{I_n} f(x) l(x) d\lambda(x)$$

for all functions $f \in \mathcal{C}(I_n)$, where λ denotes the Lebesgue measure.

Then, the function

$$\mu : \mathcal{B}(I_n) \to \mathbb{R}, \mu(A) := \int_A l(x) d\lambda(x)$$

defines a measure μ on I_n and L can be written as

$$L(f) = \int_{I_n} f(x) d\mu(x) \tag{2.5}$$

for all $f \in \mathcal{C}(I_n)$. In particular, since obviously $\sigma(v^\top(\cdot)+b) \in \bar{\mathcal{G}}$ for all $v \in \mathbb{R}^n, b \in \mathbb{R}$, it follows that

$$L(\sigma(v^\top(\cdot) + b)) = \int_{I_n} \sigma(v^\top x + b) d\mu(x) = 0$$

for all $v \in \mathbb{R}^n$ and $b \in \mathbb{R}$. Since σ was assumed to be discriminatory, it follows that $\mu \equiv 0$. We can see from (2.5) that this implies $L = 0$ on all $\mathcal{C}(I_n)$, contradicting our assumption. Hence, the subspace $\mathcal{G}$ must be dense in $\mathcal{C}(I_n)$. □

The universal approximation theorem, Theorem 3, now follows from Theorem 5 and the following lemma:

LEMMA 6. *Any bounded, measurable sigmoidal function is discriminatory. In particular, any continuous sigmoidal function is discriminatory.*

PROOF. The (rather technical) proof can be found in [**10**, pp. 307f.]. □

It was later shown by Leshno et al. ([**11**]) that FNNs satisfying the assumptions of the universal approximation theorem (Theorem 3) are a universal approximator if and only if the activation function is not polynomial.

In order for FNNs with a single hidden layer to be universal approximators, their layer width can be exponentially large with respect to the desired accuracy. In

2017, Lu et al. ([**12**]) proved a universal approximation theorem for deep FNNs with bounded width. In particular, they showed that any Lebesgue integrable function on an n-dimensional input space can be approximated with respect to L^1 distance by an FNN of width $n + 4$ with ReLU activation functions, if the network depth is allowed to grow.

1.3. Training neural networks with backpropagation. The process of finding a parameter set, i.e., determining weights and biases such that a FNN approximates a specific input-output-mapping is called **learning** or **training** of the FNN. The term **deep learning** thus refers to the process of approximating certain input-output-mappings with deep neural networks.

More precisely, this is realised by specifying an objective in terms of a **cost function** that, for example, often denotes a measure of distance between each output value of the FNN and the corresponding training target. If a FNN and the objective are designed such that the cost function is differentiable with respect to the network parameters, we can optimise it by taking its gradient with respect to the parameters and set it to zero, thus "learning" parameters that best approximate the desired mapping, as quantified by the defined objective.

In this light, nearly all deep learning algorithms can be described as examples of a standard procedure from classical statistical learning: Given a dataset, we define a model, a cost function and an optimisation procedure, and aim to optimise the model parameters with respect to the objective specified via the cost function.

If a model includes nonlinearities, such as non-linear activation functions of FNNs, most standard cost functions can no longer be optimised in closed form ([**9**, Chapter 5, p. 151]). This requires an iterative numerical optimisation procedure, such as the gradient descent algorithm, to approximately minimise the cost function. Specifically, FNNs are typically trained with the **backpropagation** algorithm, first developed in [**13**] and described by the original authors as follows: "The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired

output vector." ([**13**, p. 533]) Backpropagation is employed to learn parameters of a neural network such that an arbitrary cost function of the FNN output that is differentiable with respect to the parameters is minimised.

In the forward pass of one iteration, an input value is passed through all the layers one by one, by applying the function defining the layer with the current set of parameters. Having applied all layer functions, the loss for the obtained output value can be calculated. If we are interested in learning a classifier, i.e., a mapping assigning labels to each input, we also regard the loss as a function of the true label to compare the output to. Since the output has been derived from applying the FNN function to the input and that function depends on the current parameter values, we can also for a fixed input regard the loss as a function of the network parameters. Assuming it is differentiable with respect to the parameters, we can calculate the partial derivatives of the loss function with respect to each parameter and set the derivatives to zero. Due to the structure of the FNN as a function composition, the chain rule has to be applied multiple times to obtain the gradients: The loss directly depends on the parameters of the last layer, which depend on those of the second last layer, etc. In that way, the gradients are propagated backwards through the network to obtain the partial derivative with respect to each parameter – hence the term backpropagation. After obtaining the partial derivatives with respect to all the parameters, the current parameter values can be updated according to the gradient descent algorithm. Proceeding in this way, the cost function is iteratively optimised with respect to the network parameters and the network is thereby trained to ultimately approximate the desired input-output mapping. Neccessary conditions for this procedure are a scalar loss value for each input and a differentiable FNN structure.

Figure 3 illustrates the backpropagation algorithm with the simple FNN example from Figure 2. The loss function is chosen to be the squared difference between an FNN output and the corresponding true label.

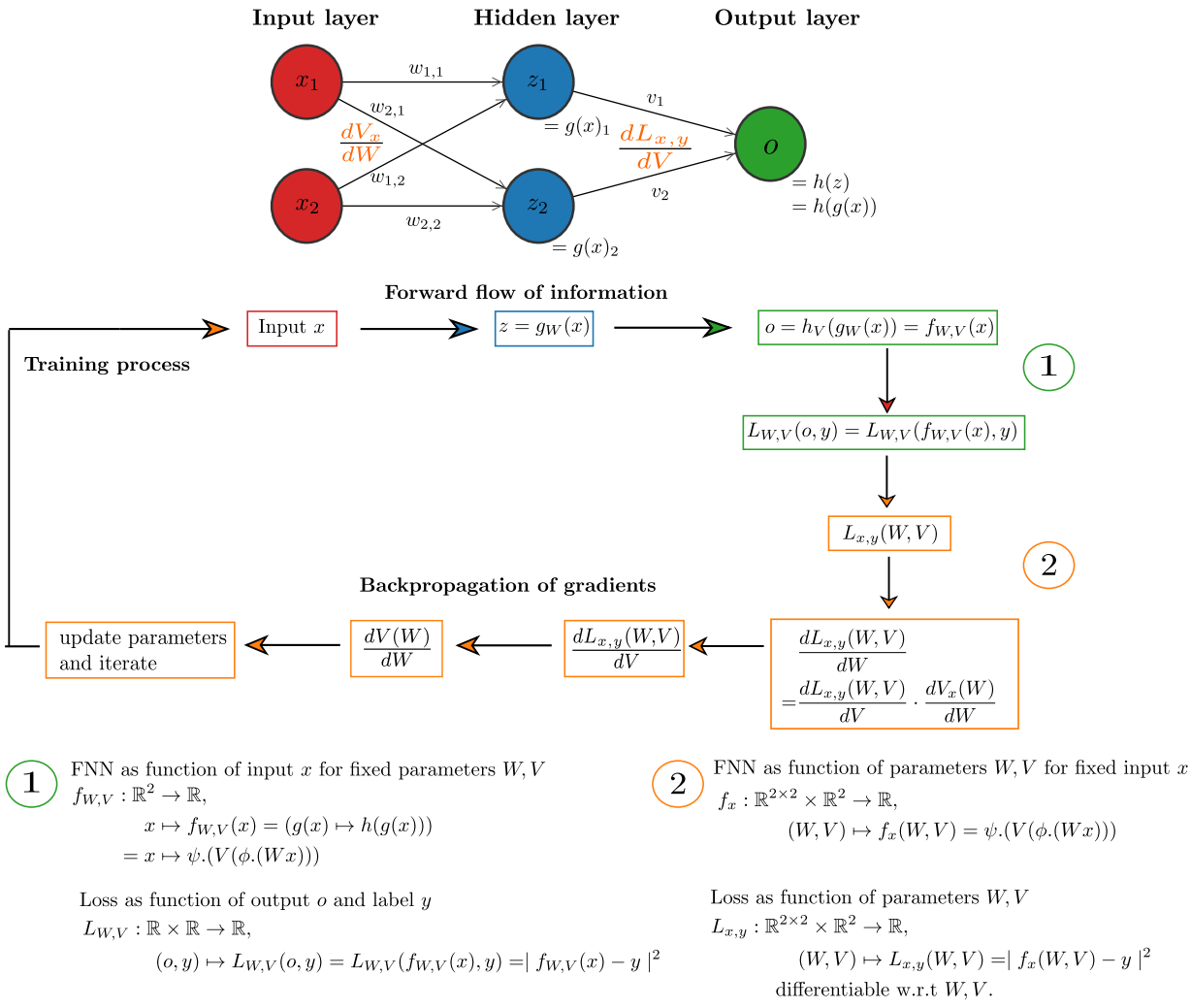


FIGURE 3. Training an FNN with backpropagation.

Often the **stochastic gradient descent (SGD)** algorithm is employed for training, which uses randomly drawn minibatches of data $\mathcal{M} \subset \mathcal{D}$ to obtain an unbiased estimate $\hat{L}(\theta, \varepsilon)$ of the objective $L(\theta)$, i.e., $\mathbb{E}_{P\mathcal{E}}[\hat{L}(\theta, \cdot)] = L(\theta)$ for FNN parameters θ, where the random variable $\mathcal{E}$ describes the sampling of the minibatch ([**14**, p. 72]). In the k-th step of the SGD algorithm, the current parameter value θ_k is updated to θ_{k+1} according to

$$\theta_{k+1} \leftarrow \theta_k + \alpha_k \cdot \nabla_\theta \hat{L}(\theta, \varepsilon_k),$$

where α_k denotes the stepsize and is also called the **learning rate** and ε_k represents a realisation of $\mathcal{E}$ corresponding to the sampling of the current minibatch. Training a FNN can now be achieved by repeating this updating step which is also called a training **epoch** until a pre-defined convergence criterion of the loss function is reached. From now on, all FNNs are assumed to be differentiable with respect to the parameters, thus allowing for an iterative optimisation using (stochastic) gradient descent.

1.4. Generative models. Neural networks can be trained to approximate different types of input-output relations and have been successfully applied in various classification tasks. Having learned a classifier and thus being able to discriminate and label new observations does not, however, yield any insight into the data-generating process itself. Additionally, this form of supervised learning typically requires large amounts of labelled training data. After training, the deterministic function represented by such a discriminative neural network will always produce the exact same label for a specific observation and thus does not provide any information about uncertainty on the label assignment ([**14**, pp. 2f.]).

To understand the data-generating mechanisms and characterise their variability as a result of random forces governing the process, a **generative** deep learning model (as opposed to a discriminative one) aims at modelling a joint probability distribution over all the variables ([**9**, Chapter 20, p. 651]). It thus simulates how the data is generated in the real world and accounts for the stochastic nature of that process. A further advantage is that a generative deep learning model can directly learn from data in an unsupervised manner and does not require labelled training examples.

Having successfully trained a generative model in the form of a neural network then means having captured the process underlying these original observations and formulated it as a probability distribution. As a direct consequence, the model can generate new samples that exhibit the central patterns and structure of the original data. This artificial data can be useful for, e.g., sample size calculations for the

planning of future experiments or for settings where due to data protection rules to guarantee privacy it is not possible to share individual-level data.

While all generative deep learning models share the characteristic of representing a multivariate probability distribution, they differ in the way that distribution is specified. In the following, three commonly used examples of deep generative models are briefly reviewed.

Restricted and **deep Boltzmann machines (RBMs/DBMs)** are energy-based models, i.e., they define the joint probability distribution of a multi-dimensional binary random vector using an energy function. Originally, they were introduced as a general approach to learning arbitrary probability distributions over binary vectors ([**15**]). The probability distribution is parameterised by a neural network with one (in RBMs) or more (in DBMs) hidden layers. To approximate it, a partition function has to be learned that is generally computationally intractable. However, the specific structure of the neural network allows for efficient Markov Chain Monte Carlo (MCMC) approximation of the desired distributions by using Gibbs Sampling ([**9**, Chapter 20, pp. 651-662]).

Training **generative adversarial networks (GANs)** ([**16**]) can be described as a two-player minimax game of a generator and a discriminator that are both specified by neural networks: Given a dataset and a pre-defined distribution of an input noise random variable, the generative network represents a mapping from the initial distribution to the data distribution the GAN aims to model. The discriminator, a second neural network, outputs the probability that a given input came from the true data distribution rather than from the output distribution of the generator. The discriminator is then trained to maximise the probability of correctly distinguishing between samples from the two, while the generator is simultaneously trained to increase the error rate of the discriminator's classification, i.e., to fool the discriminator. The probability distribution function of the original data is thus only available implictly through the samples produced by the generative network.

Finally, a **variational autoencoder (VAE)**([**17**]) aims at learning a low-dimensional representation of the input data given by a latent random variable. The transformations from the original data space to the latent space and back are given by independently parameterised, but jointly optimised neural networks: The encoder learns a transformation from the input data to the typically lower-dimensional latent space. It outputs the parameters of an approximation to the posterior distribution of the latent variable given the input data. The generative network is trained to learn the conditional distribution of the input data given a sample of the posterior distribution of the latent variables from the encoder network, such that it approximates the true underlying data distribution. Thus, the encoder is the approximate inverse of the generative model and the two networks are trained jointly, such that the encoder learns a meaningful representation of the input data that can in turn be successfully decoded to samples from the data distribution by the generative network.

In comparison to GANs, VAEs are reported to have complimentary properties ([**14**, p. 5]): While GANS can generate subjectively realistic samples, they often only cover a part of the true data distribution, whereas VAEs tend to generate "blurrier", more dispersed samples, but better capture the underlying density.

Further advantages of the VAE are that it explicitly parameterises the probability distributions involved and that it learns to compress the data into a low-dimensional representation that reflects the central factors of variation governing the dataset. These properties allow us to integrate smoothness constraints and explicit modelling of temporal dynamics into the model and learn a more structured, interpretable low-dimensional representation that provides insight into the main underlying trends driving the temporal patterns in the original data. For these reasons, my own model is based on the framework of a variational autoencoder. In the following sections,I outline the theoretical foundation on which the VAE is build and explain the model architecture and training procedure in detail.

2. Variational inference

2.1. Prerequisites and notation. The mathematical framework for the following sections is as follows: Let $\Omega := \mathbb{R}^p$, $\mathcal{F} := \mathcal{B}(\mathbb{R}^p)$ and $(\Omega, \mathcal{F}, (P_\theta)_{\theta\in\Theta})$ be a parametric statistical model with parameter space $\Theta \subset \mathbb{R}^d$ for some $d \in \mathbb{N}$. Let P be a probability measure on $(\Omega, \mathcal{F})$, let X be a real-valued p-dimensional random variable on $(\Omega, \mathcal{F}, P)$ with probability distribution P^X, let $\mathcal{D} = \{x_1, \dots, x_n\}$ be a dataset of n independent samples of X. We assume that the true distribution P^X is unknown and attempt to approximate it with a member of the family $(P_\theta)_{\theta\in\Theta}$. Let Z be a second lower-dimensional real-valued variable on $(\Omega, \mathcal{F}, P)$ with probability distribution P^Z which we do not observe but assume to govern the underlying mechanism according to which observations of X are generated (see Section 2.2). We assume our parametric family to be flexible enough to include or approximate the true distributions of X and Z and their joint distribution.

Additionally, we assume that for every $\theta \in \Theta$, P_θ is absolutely continuous with respect to the Lebesgue measure λ. Then, it follows from the Radon-Nikodym theorem that for every $\theta \in \Theta$, P_θ admits a density with respect to λ that we denote by $p_\theta = \frac{dP_\theta}{d\lambda}$. We assume the Lebesgue measure to be an equivalent measure for all probability measures P_θ, $\theta \in \Theta$, implying that all densities with respect to the Lebesgue measure are strictly positive.

In general, I denote random variables with capital letters and their realisations with small letters. To denote densities of random variables (provided they exist), I use small letters with the random variable as a subscript, i.e., $p_Z(\cdot)$ is the density of Z. For ease of notation, I abbreviate conditional distributions and densities by writing, e.g., $p_{X|z}(x)$ for $p_{X|Z=z}(x)$. I write $\mathbb{E}_p[q]$ for the integral $\int_{-\infty}^{\infty} q(x) \cdot p(x)\, dx$, where q and p are densities with respect to the Lebesgue measure λ, such that $\mathbb{E}_p[q] = \mathbb{E}[q(X)]$ for a random variable X on $(\Omega, \mathcal{F}, P)$ with $\frac{dP^X}{d\lambda} = p$, and therefore $\mathbb{E}[q(X)] = \int_\Omega q(X)\, dP = \int_{-\infty}^{\infty} q(x) \cdot p(x)\, dx$.

I denote with $p_X(x, \theta)$ the likelihood function that parameterises the family of potential densities of X as a function of the parameter θ, i.e., evaluates the likelihood of a particular realisation x of X under the density defined by a specific θ. Analoguously, I denote with $p_{X,Z} : \mathbb{R}^2 \times \Theta, (x, z, \theta) \mapsto p_{X,Z}(x, z, \theta)$ for realisations x and z of X and Z, respectively, the function parameterising the potential joint densities of X and Z as a function of θ and also use this notation to parameterise families of potential densities of a single random variable or of conditional probability distributions. For simplicity, I refer to these potential "candidate" densities of distributions or random variables simply as densities.

2.2. Bayesian inference. The data are assumed to be generated in a two-step process ([**17**, p. 2]): First, a sample z of a the lower-dimensional random variable Z on $(\Omega, \mathcal{F}, P)$ is drawn according to its distribution $z \sim P^Z$, called the **prior distribution**. We assume that $P^Z = P_{\theta^Z}$ for a $\theta^Z \in \Theta$. Secondly, a sample $x = x_i$ for some $i \in \{1, \dots, n\}$ is drawn from the conditional distribution $P_{X|z}$. We assume the joint distribution $P^{X,Z}$ to also be a member of the family $(P_\theta)_{\theta \in \Theta}$, such that the density of the conditional distribution of $(X|z)$ for a realisation z of Z, given by the expression

$$p_{X|z}(x) := \frac{p_{X,Z}(x, z)}{p_Z(z)},$$

can also be parameterised by Θ. We assume that the parameters θ^Z and $\theta^{(X,Z)}$ characterising the true distribution of Z and the true joint distribution of X and Z are unknown.

Intuitively, this scenario implies that the variation in our data is based on a few unobserved factors of variation corresponding to the dimensions of the latent random variable Z that we would like to recover along with the details of how the data are generated from the low-dimensional representation Z. To achieve that, since only X but not Z is observed, we have to reverse the process and use the observations of X to enrich our knowledge of Z, i.e., we condition the distribution of Z on the observations of X and compute the density of the **posterior distribution** of Z in Bayesian terminology (see [**18**, pp. 57-63], [**14**, p. 70] for more on Bayesian

inference). According to Bayes' rule, we obtain

$$p_{Z|x}(z) = \frac{p_{X,Z}(x,z)}{p_X(x)} = \frac{p_{X,Z}(x,z)}{\int p_{X,Z}(x,z)dz}. \tag{2.6}$$

In the last expression of (2.6), the marginal data likelihood is rewritten as an integral of the joint density of X and Z over all values of Z. A model that parameterises such a family of potential joint distributions of the observed and latent variables with FNNs is called a **deep latent variable model (DLVM)** as in [**14**, pp. 12f.]. Training a DLVM that parameterises $p_{X,Z}$ as a function of θ means to find the parameters $\theta_{X,Z}$ such that $p_{X,Z}(\cdot, \theta_{X,Z})$ equals the true density of $P^{X,Z}$. In such a DLVM, for a fixed θ the marginal likelihood of a sample x of X is given by

$$p_X(x,\theta) = \int_{-\infty}^{\infty} p_{X,Z}(x,z,\theta)dz, \tag{2.7}$$

which shows that an advantage of these models is their expressivity while maintaining a simple structure: even for relatively simple factors $p_{X,Z}(x,z,\theta)$, the marginal density p_X can be very complex and DLVMs can thus effectively model complicated underlying distributions P^X([**14**, pp. 12f.]).

To obtain a maximum likelihood estimate for θ in DLVMs modelling joint distributions of input data and latent variables, the data likelihood $p_X(x,\theta)$ has to be maximised with respect to θ. From (2.7), we see that calculating this marginal likelihood from the joint distribution involves integrating over all possible configurations (x,z). Even for moderately complex models, such as DLVMs with a non-linear hidden layer, this is computationally intractable, implying the intractability of the posterior ([**14**, p. 13]).

If, however, the density of the conditional distribution $p_{Z|x}(\cdot,\theta)$ is tractable to compute, the **expectation maximisation (EM)** algorithm ([**19**]) can be used to approximate the maximum likelihood estimate of the marginal likelihood, which consists of iteratively applying the following two steps until convergence:

1. **Expectation step:** Define $q(\theta \mid \theta^{(k)})$ as the expected value of the log likelihood function of the joint distribution of X and Z with respect to the conditional

distribution of Z given a sample x of X under the current estimate of $\theta^{(k)}$:

$$q(\theta \mid \theta^{(k)}) := \mathbb{E}_{p_{Z|x}(\cdot,\theta^{(k)})} \left[\log \left(p_{X,Z}(x, \cdot, \theta) \right) \right].$$

2. **Maximisation step:** Find the parameter $\theta^{(k+1)}$ that maximises this quantity:

$$\theta^{(k+1)} := \arg\max_{\theta} q\left(\theta \mid \theta^{(k)}\right).$$

2.3. Inference as an optimisation problem. In DLVMs, the marginal likelihood p_X of the data-generating random variable X is typically intractable. But since they are defined as a model parameterising the joint density $p_{X,Z}(x, z, \theta)$, this joint distribution is tractable to compute in DLVMS. Hence, we can see from (2.6) that the posterior $p_{Z|x}(z, \theta)$ is tractable if and only if the marginal likelihood is tractable ([**14**, p. 14]).

More broadly, this problem applies to any sufficiently complex Bayesian model including latent variables, since inference in a Bayesian model always amounts to conditioning on data and computing the posterior. If this is not computationally tractable, approximative inference is required ([**20**, p. 2]). A classical approach for that is Markov chain Monte Carlo (MCMC) sampling, which amounts to constructing a Markov chain on Z such that its stationary distribution is that of the posterior $P_\theta^{Z|x}$, sampling from the chain and approximating the posterior with an empirical estimate constructed from the samples ([**20**, p. 2]). MCMC sampling is a common and widely used tool to perform approximate inference in Bayesian statistics that has been extensively studied (for an overview see, e.g., [**21**]).

Variational inference ([**20**]) represents an alternative strategy to approximate intractable posterior distributions. The key idea is to reframe the problem of performing approximate inference as an optimisation problem rather than using sampling. To this end, we define a family $\mathcal{Q}$ of approximate densities over the latent variables Z and aim to find the member of the family that minimises a measure of distance $d(\cdot, \cdot)$ to the exact posterior density:

$$q^*_{Z|x} = \arg\min_{q_{Z|x} \in \mathcal{Q}} d(q_{Z|x}, p_{Z|x}). \tag{2.8}$$

Variational inference in general has advantages over MCMC in settings where a faster approximation of a posterior is desired than can be achieved with MCMC algorithms, such as for large datasets or very complex models. For a more detailed discussion of the strength and weaknesses of the two approaches and typical scenarios in which to use either, see [**20**, p. 3] and the references mentioned there.

2.4. The Kullback-Leibler divergence. As a measure of distance between probability distributions, the **Kullback-Leibler (KL) divergence** ([**9**, p. 72]) is used:

DEFINITION 7. Let $(\Omega, \mathcal{F})$ be a measurable space and let P, Q be probability measures on $(\Omega, \mathcal{F})$ such that P is absolutely continuous with respect to Q, i.e., admits a density with respect to Q denoted by $\frac{dP}{dQ}$. The **Kullback-Leibler (KL) divergence** from Q to P is defined as

$$D_{\mathrm{KL}}(P\|Q) = \int_{\Omega} \log\left(\frac{dP}{dQ}\right) dP.$$

If P and Q are absolutely continuous with respect to the Lebesgue measure λ, denoting the densities $p = \frac{dP}{d\lambda}$ and $q = \frac{dQ}{d\lambda}$, the Kullback-Leibler divergence can be rewritten as an expectation with respect to P:

$$D_{\mathrm{KL}}(P\|Q) = \mathbb{E}\left[\log\left(\frac{dP}{dQ}\right)\right] = \int_{\Omega} \log\left(\frac{dP}{dQ}\right) dP = \int_{-\infty}^{\infty} \log\left(\frac{p}{q}\right) p \, d\lambda = \mathbb{E}_p\left[\log\left(\frac{p}{q}\right)\right].$$

In this case, I equivalently write $D_{\mathrm{KL}}(p\|q)$ for $D_{\mathrm{KL}}(P\|Q)$.

REMARK. It follows from Jensen's inequality that the Kullback-Leibler divergence is always non-negative:

$$\begin{aligned} D_{\mathrm{KL}}(P\|Q) &= \int_{\Omega} \log\left(\frac{dP}{dQ}\right) dP = \int_{\Omega} -\log\left(\frac{dQ}{dP}\right) dP \\ &\geq -\log\left(\int_{\Omega} \frac{dQ}{dP} dP\right) = -\log\left(\int_{\Omega} dQ\right) = -\log(1) = 0, \end{aligned}$$

where the $\geq$ results from applying Jensen's inequality.

The KL divergence is equal to 0 if and only if the two probability measures P and Q are equal. However, the KL-divergence is not symmetric and hence is not a

metric. Nonetheless, it is widely used as a notion of distance between two probability distributions in machine learning and statistics.

Quantifying the difference between members of the variational family $\mathcal{Q}$ and the exact posterior with the KL-divergence, our optimisation problem (2.8) becomes

$$q^*_{Z|x} = \arg\min_{q_{Z|x}\in\mathcal{Q}} D_{\mathrm{KL}}(q_{Z|x}\|p_{Z|x}). \tag{2.9}$$

The complexity of this optimisation depends on the complexity of the family $\mathcal{Q}$ of candidate approximate densities. While a simpler structure of $\mathcal{Q}$ allows for a simpler and more efficient optimisation, a larger, more flexible family $\mathcal{Q}$ can potentially capture a density closer to the exact posterior.

2.5. Calculus of variation. The term "variational inference" refers to the theory of calculus of variation, a branch of functional analysis, which centers on the fundamental idea of minimising a functional on some (typically infinite-dimensional) function space. A classical example is given by the Dirichlet energy functional

$$E : \mathcal{H}^1(U) \to \mathbb{R}, \quad u \mapsto \frac{1}{2}\int_U \|\nabla u(x)\|^2 dx,$$

where $U \subset \mathbb{R}^n$ is an open subset and $\mathcal{H}^1(U)$ denotes the Sobolev space on U. The functions that minimise the Dirichlet energy and satisfy certain boundary conditions are exactly the solutions of the partial differential equation given by Laplace's equation $-\Delta u(x) = 0$ for all $x \in U$ subject to appropriate boundary conditions. More generally, many problems in the scope of calculus of variation have applications in the field of partial differential equations.

In variational inference, the functional to be minimised is given by the distance measure between the probability distributions, often the KL-divergence, while the space of functions over which it is minimised is given by the variational family $\mathcal{Q}$. In my setting, I will exclusively use a parametric family $\mathcal{Q} = (q_Z(\cdot,\phi))_{\phi\in\Phi}$ with $\Phi \subset \mathbb{R}^e$ for some $e \in \mathbb{N}$, so that the space of approximate posterior distributions is completely determined by the parameter space Φ and hence finite-dimensional.

Therefore, here the calculus of variation does not have to be employed to solve the approximation problem.

2.6. The evidence lower bound. The following derivation of the evidence lower bound is based on [**20**, pp. 6f.] and [**14**, pp. 16-18].

Returning to our optimisation problem of finding $q^*_{Z|x}$ as in (2.9), rearranging the terms of the KL-divergence yields

$$\begin{aligned} D_{\mathrm{KL}}(q_{Z|x}\|p_{Z|x}) &= \mathbb{E}_{q_{Z|x}}\left[\log\left(\frac{q_{Z|x}}{p_{Z|x}}\right)\right] \\ &= \mathbb{E}_{q_{Z|x}}[\log(q_{Z|x})] - \mathbb{E}_{q_{Z|x}}[\log(p_{Z|x})] \\ &= \mathbb{E}_{q_{Z|x}}[\log(q_{Z|x})] - \mathbb{E}_{q_{Z|x}}\left[\log\left(\frac{p_{X,Z}(x,\cdot)}{p_X(x)}\right)\right] \\ &= \mathbb{E}_{q_{Z|x}}[\log(q_{Z|x})] - \mathbb{E}_{q_{Z|x}}[\log(p_{X,Z}(x,\cdot))] + \log(p_X(x)), \end{aligned} \tag{2.10}$$

where we used the linearity of the expectation and the fact that $\log(p_X(x))$ is a constant with respect to $q_{Z|x}$. The last line shows that in fact the KL-divergence we aim to minimise depends on $\log(p_X(x))$ which we assumed to be intractable. That intractability implies that of the posterior $p_{Z|x}$ (as described in Section 2.2) and was our motivation to approximate the posterior with variational inference in the first place. However, the observation that $\log(p_X(x))$ is a constant with respect to $q_{Z|x}$ also implies that minimising the expression $\mathbb{E}_{q_{Z|x}}[\log(q_{Z|x})] - \mathbb{E}_{q_{Z|x}}[\log(p_{X,Z}(x,\cdot))]$ is equivalent to minimising $D_{\mathrm{KL}}(q_{Z|x}\|p_{Z|x})$ with respect to $q_{Z|x}$, i.e.,

$$q^*_{Z|x} = \arg\min_{q_{Z|x}\in\mathcal{Q}} D_{\mathrm{KL}}(q_{Z|x}\|p_{Z|x}) = \arg\min_{q_{Z|x}\in\mathcal{Q}} \mathbb{E}_{q_{Z|x}}[\log(q_{Z|x})] - \mathbb{E}_{q_{Z|x}}[\log(p_{X,Z}(x,\cdot))].$$

Re-organising the terms in the last line of (2.10) and using the non-negativity of the KL-divergence from Remark 2.4, we obtain

$$\begin{aligned} \log(p_X(x)) &= \mathbb{E}_{q_{Z|x}}[\log(p_{X,Z}(x,\cdot))] - \mathbb{E}_{q_{Z|x}}[\log(q_{Z|x})] + D_{\mathrm{KL}}(q_{Z|x}\|p_{Z|x}) \\ &\geq \mathbb{E}_{q_{Z|x}}[\log(p_{X,Z}(x,\cdot))] - \mathbb{E}_{q_{Z|x}}[\log(q_{Z|x})]. \end{aligned} \tag{2.11}$$

This shows that $\mathbb{E}_{q_{Z|x}}[\log(p_{X,Z}(x,\cdot))] - \mathbb{E}_{q_{Z|x}}[\log(q_{Z|x})]$, the negative of the equivalent objective $\mathbb{E}_{q_{Z|x}}[\log(q_{Z|x})] - \mathbb{E}_{q_{Z|x}}[\log(p_{X,Z}(x,\cdot))]$ defines a lower bound on the marginal data likelihood $\log(p_X(x))$. Because the marginal data likelihood is also called the

evidence in Bayesian terminology, this expression is called the **evidence lower bound (ELBO)**

$$\mathrm{ELBO}(x, q_{Z|x}) = \mathbb{E}_{q_{Z|x}}[\log(p_{X,Z}(x,\cdot))] - \mathbb{E}_{q_{Z|x}}[\log(q_{Z|x})].$$

Since minimising the KL divergence with respect to q is equivalent to minimising $\mathbb{E}_{q_{Z|x}}[\log(q_{Z|x})] - \mathbb{E}_{q_{Z|x}}[\log(p_{X,Z}(x,\cdot))]$, which is equivalent to maximising $\mathbb{E}_{q_{Z|x}}[\log(p_{X,Z}(x,\cdot))] - \mathbb{E}_{q_{Z|x}}[\log(q_{Z|x})]$, it follows that minimising the KL-divergence $D_{\mathrm{KL}}(q_{Z|x}\|p_{Z|x})$ is equivalent to maximising the ELBO and

$$q^*_{Z|x} = \arg\min_{q_{Z|x}\in\mathcal{Q}} D_{\mathrm{KL}}(q_{Z|x}\|p_{Z|x}) = \arg\max_{q_{Z|x}\in\mathcal{Q}} \mathrm{ELBO}(x, q_{Z|x}).$$

From (2.11) we can see that the KL-divergence $D_{\mathrm{KL}}(q_{Z|x}\|p_{Z|x})$ determines the gap between the ELBO and the marginal data likelihood $\log(p_X(x))$ and thus the tightness of the bound. Rewriting the ELBO can give us an intuition about the properties of the optimal variational density ([**20**, pp. 6f.]):

$$\begin{aligned}
\mathrm{ELBO}(x, q_{Z|x}) &= \mathbb{E}_{q_{Z|x}}[\log(p_{X,Z}(x,\cdot))] - \mathbb{E}_{q_{Z|x}}[\log(q_{Z|x})] \\
&= \mathbb{E}_{q_{Z|x}}[\log(p_{X|Z}(x)p_Z)] - \mathbb{E}_{q_{Z|x}}[\log(q_{Z|x})] \\
&= \mathbb{E}_{q_{Z|x}}[\log(p_{X|Z}(x))] + \mathbb{E}_{q_{Z|x}}[\log(p_Z)] - \mathbb{E}_{q_{Z|x}}[\log(q_{Z|x})] \\
&= \mathbb{E}_{q_{Z|x}}[\log(p_{X|Z}(x))] + \mathbb{E}_{q_{Z|x}}\left[\log\left(\frac{p_Z}{q_{Z|x}}\right)\right] \\
&= \mathbb{E}_{q_{Z|x}}[\log(p_{X|Z}(x))] - D_{\mathrm{KL}}(q_{Z|x}\|p_Z).
\end{aligned} \tag{2.12}$$

The first term in the last line shows that maximising the ELBO encourages densities placing mass on configurations of latent variables that explain the observed data, while the second term encourages densities that are close to the prior.

Parameterising the potential candidate prior and posterior densities of Z and $Z|x$ for a sample $x \in \mathcal{D}$ with θ as in Section 2.2 and (e.g.) Equation (2.6), we can define the ELBO as a function of not only the approximate posterior $q_{Z|x}$ and the data $x \in \mathcal{D}$, but also of the parameter θ defining the densities of Z and $Z|x$ that is called the **model parameter**. Additionally, we assume a parametric variational family $\mathcal{Q} = (q_{Z|x}(\cdot,\phi))_{\phi\in\Phi}$ with $\Phi \subset \mathbb{R}^e$ for an $e \in \mathbb{N}$, so that the finite-dimensional

space Φ completely characterises the family of approximate posteriors, and call the parameter $\phi \in \Phi$ the **variational parameter**. As a result, we can view the ELBO as a function of the data $x \in \mathcal{D}$, the model parameter θ and the variational parameter ϕ

$$\begin{aligned} \mathrm{ELBO}(x, \theta, \phi) &= \mathbb{E}_{q_{Z|x}(\cdot,\phi)}[\log(p_{X,Z}(x, \cdot, \theta))] - \mathbb{E}_{q_{Z|x}(\cdot,\phi)}\left[\log\left(q_{Z|x}(\cdot, \phi)\right)\right] \\ &= \mathbb{E}_{q_{Z|x}(\cdot,\phi)}[\log(p_{X|Z}(x, \theta))] - D_{\mathrm{KL}}(q_{Z|x}(\cdot, \phi) \| p_Z(\cdot, \theta)). \end{aligned} \tag{2.13}$$

Note that since the ELBO is a lower bound on $\log(p_X(x, \theta))$, maximising the ELBO with respect to θ yields an approximation to the maximum likelihood estimate for θ. That approximation is better the smaller the KL-divergence

$$D_{\mathrm{KL}}(q_{Z|x}(\cdot, \phi) \| p_{Z|x}(\cdot, \theta)),$$

that determines the tightness of the bound. We can thus obtain both approximate maximum likelihood estimates for θ and an optimal variational density q if we maximise the ELBO both with respect to θ and ϕ. The search of finding optimal parameters θ is referred to as **learning** and the process of finding parameters ϕ of an optimal approximate posterior q as **inference** ([**14**, p. 15]).

2.7. Variational expectation maximisation. Maximising the ELBO with respect to both θ and ϕ can be formulated as a variation of the EM-algorithm from Section 2.2 called the **variational expectation maximisation** algorithm and can also be viewed as coordinate ascent on the ELBO ([**14**, p. 71], [**22**]). It assumes a parametric variational familiy $\mathcal{Q} = (q_\phi(z))_{\phi \in \Phi}$ with $\Phi \subset \mathbb{R}^e$ for some $e \in \mathbb{N}$ and estimates **local** variational parameters ϕ_i for each individual sample x_i.

As for classical EM, the algorithm starts with (random) initial values of $\theta^{(0)}$ and $\phi^{(0)}_{i=1,\dots,n}$ and iteratively applies the following steps:

1. **Expectation step:** For all $i = 1, \dots, n$, maximise the ELBO with respect to ϕ_i and define

$$\phi_i^{(k+1)} := \arg\max_{\phi} \mathrm{ELBO}(x_i, \theta^{(k)}, \phi_i). \tag{2.14}$$

2. **Maximisation step:** Maximise the ELBO with respect to θ and define

$$\theta^{(k+1)} := \arg\max_{\theta} \sum_{i=1}^{n} \mathrm{ELBO}(x_i, \theta, \phi_i^{(k+1)}). \tag{2.15}$$

Note that, as shown before, the objective in (2.14) is equivalent to finding $\arg\min_{\phi} D_{\mathrm{KL}}(q_{Z|x}(\cdot, \phi) \| p_{Z|x}(\cdot, \theta^{(k)}))$.

3. Variational autoencoder

The variational autoencoder, briefly described in Section 1.4, is a framework to efficiently perform approximate inference and learning. In short, it is a generative deep learning model consisting of two coupled, jointly optimised neural networks that parameterise an approximate posterior of a latent random variable given observations and the conditional distribution of the data given a sample from the posterior over the latent variables. The training objective is given by the ELBO (2.13) that is optimised with respect to both the model parameters and the variational parameters using SGD. The following presentation is based on the paper first proposing VAEs [**17**] and [**14**, pp. 15-27].

3.1. Model overview. The VAE proposes a solution to the three related problems in the setting of intractabilities of the marginal likelihood

$$p_X(\cdot, \theta) = \int p_{X,Z}(\cdot, z, \theta) dz$$

and/or the posterior density $p_{Z|x}(\cdot, \theta)$ for a potentially large dataset where sampling-based MCMC methods would be too slow, namely efficient approximation of maximum likelihood estimates for the parameters $\theta^Z, \theta^{(X,Z)}$, efficient approximate posterior inference of $Z|X$ and efficient approximate marginal inference of the underlying variable X ([**17**, pp. 2f.]). Since the latent variable Z encodes a low-dimensional representation of data, the VAE can be seen as a method for inferring these ideally semantically meaningful, statistically independent factors of variation in data. This process is generally known as unsupervised representation learning, and VAEs have been extensively employed for that purpose ([**14**, p. 4]).

To handle the intractabilities of the marginal likelihood and posterior, a parametric family of distributions $\mathcal{Q} = (q_{Z|x}(\cdot, \phi))_{\phi \in \Phi}$ with $\Phi \subset \mathbb{R}^e$ for some $e \in \mathbb{N}$ is introduced. We then aim to optimise the variational parameters ϕ to find a family member $q_{Z|x}(\cdot, \phi)$ that approximates the true posterior as in Section 2, such that $q_{Z|x}(z, \phi) \approx p_{Z|x}(z, \theta)$ for all $z \in \mathbb{R}$, $x \in \mathcal{D}$.

The model $q_{Z|x}(z, \phi)$ as a function of the variational parameters ϕ and realisations x and z of the respective random variables is called the **inference model** or the **encoder**, since it encodes a sample $x \in \mathcal{D}$ into a lower-dimensional representation in the space of samples of Z. The density $q_{Z|x}(\cdot, \phi)$ is parameterised with a deep neural network, i.e., the parameters are given as output of a FNN.

A second neural network is employed to define the density $p_{X|z}(\cdot, \theta)$ called the **generator** or **decoder** part of the model. Given an input observation x, we obtain a sample z from the distribution given by the density $q_{Z|x}(\cdot, \phi)$ that we can use to evaluate the likelihood of the input observation x under the density $p_{X|z}(\cdot, \theta)$. Both parts together form the **variational autoencoder** and can be jointly trained by maximising the ELBO with respect to both the variational parameters ϕ and the parameters θ of our assumed statistical model. Note that in contrast to the Variational EM algorithm in Section 2.7, a single encoder neural network is used that is shared by all observations. Thus, a shared set of variational parameters is estimated, rather than local parameters determined for each individual. This strategy of sharing varational parameters across data points is also called **amortised variational inference** ([**14**, p. 16]).

The cost function defining the training objective for the model is given by the negative ELBO (2.13): Since it is a function of the model parameters θ and the variational parameters ϕ and therefore of the output of two FNNs, maximising it with respect to both θ and ϕ is equivalent to jointly training the encoder and decoder network with backpropagation as described in Section 1.3. As noted before, this is equivalent to jointly performing both variational inference by finding optimal parameters of the approximate variational posterior and performing approximate

maximum likelihood estimation on the marginal data likelihood by finding optimal model parameters. Note that by applying variational inference to approximate the posterior, the distribution of the latent variable does not have to be specified explicitly beforehand and no specific structure other than dimensionality of the latent representation has to be assumed a priori. Instead, one can simply define a variational family that determines the complexity of the optimisation problem and let the model freely infer a representation that fits the data.

Having trained the model, we can sample z from the prior and generate an artificial observation $\hat{x}$ by sampling from the distribution defined by $p_{X|z}(\cdot, \theta)$.

3.2. Stochastic gradient-based optimisation of the ELBO. Training the VAE model means optimising the ELBO with respect to both the model parameters θ and the variational parameters ϕ. To perform stochastic gradient descent, the ELBO has to be differentiated with respect to θ and ϕ. In the following, we derive unbiased estimators of the individual-datapoint ELBO and its gradients (that are generally intractable) based on [**14**, pp. 19-23] and [**17**, pp. 3-5].

Obtaining an unbiased estimator of the gradient with respect to the generative model parameters θ is straightforward. For any $x \in \mathcal{D}$ and a sample z from the posterior $q_{Z|x}(\cdot, \phi)$ it holds that

$$\begin{aligned} \nabla_\theta \text{ELBO}(x, \theta, \phi) &= \nabla_\theta \mathbb{E}_{q_{Z|x}(\cdot,\phi)}[\log(p_{X,Z}(x, \cdot, \theta)) - \log(q_{Z|x}(\cdot, \phi))] \\ &= \mathbb{E}_{q_{Z|x}(\cdot,\phi)}[\nabla_\theta(\log(p_{X,Z}(x, \cdot, \theta)) - \log(q_{Z|x}(\cdot, \phi)))], \end{aligned} \tag{2.16}$$

and (2.16) can be estimated with the Monte Carlo estimator

$$\nabla_\theta(\log(p_{X,Z}(x, z, \theta)) - \log(q_{Z|x}(z, \phi))) = \nabla_\theta(\log(p_{X,Z}(x, z, \theta))), \quad z \sim q_{Z|x}(\cdot, \phi).$$

Obtaining an unbiased estimator of the gradient with respect to the variational parameters ϕ is more challenging, since we take the expectation with respect to $q_{Z|x}(\cdot, \phi)$ which depends on ϕ, and in general

$$\begin{aligned}\nabla_\phi \text{ELBO}(x,\theta,\phi) &= \nabla_\phi \mathbb{E}_{q_{Z|x}(\cdot,\phi)}[\log(p_{X,Z}(x,\cdot,\theta)) - \log(q_{Z|x}(\cdot,\phi))] \\ &\neq \mathbb{E}_{q_{Z|x}(\cdot,\phi)}[\nabla_\phi(\log(p_{X,Z}(x,\cdot,\theta)) - \log(q_{Z|x}(\cdot,\phi)))].\end{aligned}$$

To derive an unbiased estimate of $\nabla_\phi \text{ELBO}(x,\theta,\phi)$, a change of variables called the **reparameterisation trick** ([**14**, pp. 20f.], [**17**, pp. 4f.]) can be employed. For that, we define a real-valued random variable $\mathcal{E}$ on $(\Omega, \mathcal{F}, P)$ that is independent of X and $(Z|x)$ and a differentiable, invertable transformation g such that for a given parameter value ϕ and a sample $x \in \mathcal{D}$ of X, we can express

$$(Z|x) = g(\mathcal{E}, \phi, x).$$

With that change of variables from $(Z|x)$ to $\mathcal{E}$, the expectation with respect to $q_{Z|x}(\cdot,\phi)$ can be rewritten in terms of an expectation with respect to $p_\mathcal{E}$, the density of $P^\mathcal{E}$. For $x \in \mathcal{D}$ as before, we thus obtain

$$\begin{aligned}\nabla_\phi \text{ELBO}(x,\theta,\phi) &= \nabla_\phi \mathbb{E}_{q_{Z|x}(\cdot,\phi)}[\log(p_{X,Z}(x,\cdot,\theta)) - \log(q_{Z|x}(\cdot,\phi))] \\ &= \nabla_\phi \mathbb{E}_{p_\mathcal{E}}[\log(p_{X,Z}(x, g(\cdot,\phi,x),\theta)) - \log(q_{Z|x}(\cdot,\phi))] \qquad (2.17) \\ &= \mathbb{E}_{p_\mathcal{E}}[\nabla_\phi(\log(p_{X,Z}(x, g(\cdot,\phi,x),\theta)) - \log(q_{Z|x}(\cdot,\phi)))].\end{aligned}$$

Now, (2.17) can be estimated as before with the Monte Carlo estimator

$$\nabla_\phi(\log(p_{X,Z}(x,z,\theta)) - \log(q_{Z|x}(z,\phi))), \quad z = g(\varepsilon,\phi,x), \quad \varepsilon \sim p_\mathcal{E}.$$

Furthermore, we can see that this estimator is unbiased:

$$\begin{aligned}&\mathbb{E}_{p_\mathcal{E}}[\nabla_\phi(\log(p_{X,Z}(x, g(\cdot,\phi,x),\theta)) - \log(q_{Z|x}(g(\cdot,\phi,x),\phi)))] \\ &\quad= \nabla_\phi(\mathbb{E}_{p_\mathcal{E}}[\log(p_{X,Z}(x,\cdot,\theta)) - \log(q_{Z|x}(g(\cdot,\phi,x),\phi))]) \\ &\quad= \nabla_\phi(\mathbb{E}_{q_{Z|x}(\cdot,\phi)}[\log(p_{X,Z}(x,\cdot,\theta)) - \log(q_{Z|x}(\cdot,\phi))]) \\ &\quad= \nabla_\phi(\text{ELBO}(x,\theta,\phi)).\end{aligned}$$

As a result, we obtain unbiased estimates of the ELBO with respect to both θ and ϕ and can optimise is using SGD (see Section 1.3). Figure 4 illustrates the reparameterisation trick. Note that in general, the ELBO is a non-convex function and thus potentially has several local optima.

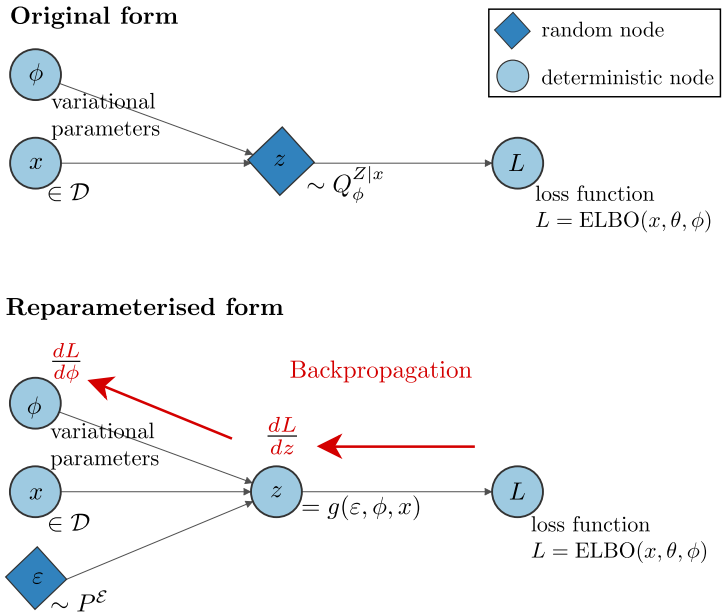


FIGURE 4. The reparameterisation trick. (adapted from [**14**, p. 22])

The complete training algorithm of the VAE, also termed **auto-encoding variational Bayes** algorithm by the original authors ([**14**, p. 21], [**17**, p. 4]), is summarised in Algorithm 1. The training process is visualised in Figure 5.

3.3. Example: VAE with factorised Gaussian posterior. In this section, I present a typical and commonly used example of a variational autoencoder with the standard choices for the distributions which enable an efficient computation of the ELBO and thus allow for efficient optimisation ([**14**, pp. 24f.], [**17**, p. 5]).

Let the prior over the latent variable Z be given by a standard normal distribution, i.e., $Z \sim P^Z = \mathcal{N}(0, I)$. In this case, the prior has no parameters and is thus not subject to any optimisation or change during training. Let $p_{X|z}(\cdot, \theta)$ be the density of a multivariate Gaussian (for real-valued data) or a Bernoulli distribution (for binary data). The distribution parameters θ are computed from realisations of the latent variable Z with a neural network with a single hidden layer and non-linear activation function. In this case, the true posterior $p_{Z|x}(\cdot, \theta)$ is intractable. To approximate it, we choose a multivariate Gaussian distribution with mean $\mu \in \mathbb{R}^K$ and diagonal covariance matrix $\mathrm{diag}(\sigma^2) \in \mathbb{R}^{K \times K}$, where the variational parameters

Algorithm 1: Auto-encoding variational Bayes algorithm: VAE training by stochastic optimisation of the ELBO

Data:

Input: dataset $\mathcal{D} = \{x_1, \ldots, x_n\}$,

Inference model: family of variational posterior distributions $(Q_\phi^{Z|x})_{\phi \in \Phi}$,

Generative model: family of joint distributions $(P_\theta^{X,Z})_{\theta \in \Theta}$

Result:

Learned parameters (θ, ϕ) of the generative and inference model

$(\theta, \phi) \leftarrow$ Random initialization;

while *SGD not converged* **do**

  Sample random minibatch of data $\mathcal{M} \sim \mathcal{D}$;

  for *each datapoint* $x \in \mathcal{M}$ **do**

    Sample random noise $\varepsilon \sim P^{\mathcal{E}}$;

    Calculate $z = g(\varepsilon, \phi, x)$;

  end

  Compute (estimators of) ELBO$(\mathcal{M}, \theta, \phi)$ and gradients $\nabla_{\theta,\phi}(\text{ELBO}(\mathcal{M}, \theta, \phi))$;

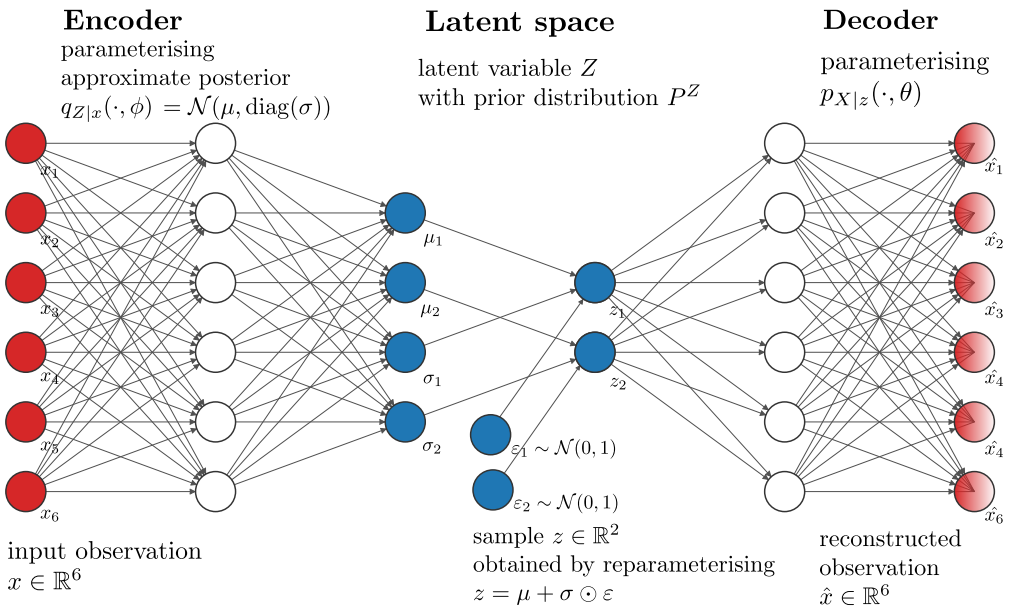
  Update parameters θ and ϕ using SGD optimiser;

end

(μ, σ) are given by a neural network:

$$\begin{aligned} (\mu, \log(\sigma)) &= \text{EncoderNeuralNet}(x) \\ q_{Z|x}(z, \phi) &= \prod_{k=1}^{K} q_{Z|x}(z_k, \phi) = \prod_{k=1}^{K} f_{\mathcal{N}(\mu, \text{diag}(\sigma^2))}(z_k). \end{aligned} \tag{2.18}$$

This factorised Gaussian encoder corresponds to the mean-field approach from variational inference, where we also assume the members of the variational family to be mutually independent, such that the distribution factorises ([**20**, pp. 7-10]).



Training objective: Maximise

$\mathrm{ELBO}(x,\theta,\phi) = \mathbb{E}_{q_{Z|x}(\cdot,\phi)}\left[\log(p_{X|z}(x,\theta))\right] - D_{\mathrm{KL}}\left(q_{Z|x}(\cdot,\phi)\|p_Z(\cdot,\theta)\right)$ w.r.t both θ and ϕ

FIGURE 5. VAE training process with reparameterisation trick.

Choosing $P^{\mathcal{E}} = \mathcal{N}(0, I)$, we can, for a sample ε from $P^{\mathcal{E}}$ and an observation x, obtain a sample z from the posterior $Q_\phi^{Z|x}$ via the reparameterisation

$$z = \mu + \sigma \odot \varepsilon,$$

where $\odot$ denotes the element-wise product. Analogously to the Gaussian example, for any family of distribution defined by a "location" and a "scale", we can choose the standard distribution with location 0 and scale 1 for the auxiliary random variable $\mathcal{E}$ and set $z = \text{location} + \text{scale} \odot \varepsilon$. Additionally, also for families of posterior distributions $\left(q_{Z|x}(\cdot,\phi)\right)_{\phi\in\Phi}$ with a tractable inverse cumulative distribution function, a differentiable transformation g and an auxiliary variable $\mathcal{E}$ can be chosen such that we can perform the reparameterisation trick (see for details [**17**, p. 5]).

We have seen in (2.12) that for any $x \in \mathcal{D}$

$$\mathrm{ELBO}(x,\theta,\phi) = \mathbb{E}_{q_{Z|x}(\cdot,\phi)}[\log(p_{X|z}(x,\theta))] - D_{\mathrm{KL}}(q_{Z|x}(\cdot,\phi)\|p_Z(\cdot,\theta)).$$

The KL-divergence between two multivariate Gaussian distributions with $\mu_1, \mu_2 \in \mathbb{R}^K, \Sigma_1, \Sigma_2 \in \mathbb{R}^{K\times K}$ can be computed in closed form according to the formula

$$\begin{aligned} &D_{\mathrm{KL}}(\mathcal{N}(\mu_1, \Sigma_1)\|\mathcal{N}(\mu_2, \Sigma_2)) \\ &\quad = \frac{1}{2}\left(\mathrm{tr}(\Sigma_2^{-1}\Sigma_1) + (\mu_2 - \mu_1)^\top \Sigma_2^{-1}(\mu_2 - \mu_1) - K + \log\left(\frac{\det(\Sigma_2)}{\det(\Sigma_1)}\right)\right). \end{aligned} \tag{2.19}$$

We thus obtain for the given choices of the prior and posterior the explicit expression

$$\begin{aligned} D_{\mathrm{KL}}(q_{Z|x}(\cdot, \phi)\|p_Z) &= D_{\mathrm{KL}}(\mathcal{N}(\mu, \mathrm{diag}(\sigma^2))\|\mathcal{N}(0, I)) \\ &= \frac{1}{2}\left(\sum_{k=1}^{K} \sigma_k^2 + \mu_k^2 - 1 - \log\left(\sigma_k^2\right)\right). \end{aligned} \tag{2.20}$$

Using L samples $z^{(1)}, \ldots, z^{(L)}$ from the posterior by sampling $\varepsilon^{(1)}, \ldots, \varepsilon^{(L)}$ from $P^{\mathcal{E}} = \mathcal{N}(0, I)$ and calculating $z^{(l)} = \mu + \sigma \odot \varepsilon^{(l)}$, we thus obtain the following estimator for the ELBO at datapoint $x \in \mathcal{D}$:

$$\mathrm{ELBO}(x, \theta, \phi) = \frac{1}{2}\sum_{k=1}^{K}(1 + \log(\sigma_k^2) - \mu_k^2 - \sigma_k^2) + \frac{1}{L}\sum_{l=1}^{L}\log(p_\theta(x \mid z^{(l)})). \tag{2.21}$$

Here, the decoding term $\log(p_{X|z^{(l)}}(x, \theta))$ can be seen as a negative reconstruction error. A theoretical justification of the particular choice of q is the fact that the Gaussian distribution maximises the entropy, i.e., is the potentially most informative distribution to use if we have no prior knowledge on the structure desired ([**9**, pp. 644f.]).

4. Ordinary differential equations

Differential equations relate functions and one or more of their derivatives and thus form the paradigmatic concept to describe quantities changing over time. In the following sections, I formally define differential equations and give basic central results about the existence and uniqueness of solutions.

4.1. Definitions.

Definition 8 ([**23**, p. 27, Definition 1]). Let $I \subset \mathbb{R}$ be an interval, $U \subset I \times \mathbb{R}^m$ open and $f : U \to \mathbb{R}^n$ a continuous function. Then, for an interval $J \subset I$, a

function $x : J \to \mathbb{R}^n$ is called a **solution of the (ordinary) differential equation** $x' = f(\cdot, x)$ **on the interval** J, if x is differentiable and $\frac{d}{dt}x(t) = f(t, x(t))$ for all $t \in J$.

Additionally, given a $t_0 \in J$ and $(t_0, x_0) \in U$, x is called a **solution of the initial value problem** defined by f and (t_0, x_0) in J, if x is a solution of the differential equation and $x(t_0) = x_0$, i.e., if

$$\frac{d}{dt}x(t) = f(t, x),$$
$$x(t_0) = x_0 \text{ for all } t \in J.$$

REMARK. Definition 8 can be generalised to differential equations of higher orders: For an $n \in \mathbb{N}$, an open subset $U \subset I \times \mathbb{R}^n$ and a continuous function f on U, a function $x \in \mathcal{C}^n(J)$ for $J \subset I$ is a solution of the differential equation $x^{(n)}(t) = f(t, x, x', x'', \ldots, x^{(n-1)})$ in J, if

$$\tfrac{d^{(n)}}{dt}x(t) = f(t, x(t), \tfrac{d}{dt}x(t), \ldots, \tfrac{d^{(n-1)}}{dt}x(t)) \text{ for all } t \in J.$$

Such a differential equation of order n can always be reduced to a first-order differential equation as in Definition 8 by changing to the new set of variables $y = (x, x', x'', \ldots, x^{(n-1)})$, which yields the first-order system

$$y_1' = y_2,$$
$$\vdots$$
$$y_{n-1}' = y_n$$
$$y_n' = f(t, y).$$

Then, any function $x \in \mathcal{C}^n(J)$ is a solution of $x^{(n)}(t) = f(t, x, x', x'', \ldots, x^{(n-1)})$ in J if and only if the function $t \mapsto y(t)$ is a solution of the first-order differential equation $y' = f(\cdot, y)$ in J ([**24**, p. 7]).

For the corresponding initial value problem, let $(t_0, x_0, \ldots, x_{n-1}) \in U$ for $t_0 \in J$. Then, $x \in \mathcal{C}^n(J)$ is a solution of the initial value problem defined by f and $(t_0, x_0, \ldots, x_{n-1})$ in J, if x is a solution of the differential equation and additionally $x(t_0) = x_0, \frac{d}{dt}x(t_0) = x_1, \ldots, \frac{d^{(n-1)}}{dt}x(t) = x_{n-1}$. We thus see that for an initial value

problem of order n, in addition to the initial value in t_0, the values of the first $n-1$ derivatives in t_0 have to be specified.

REMARK. Differential equations according to Definition 8 are called **ordinary differential equations (ODEs)**, because the solution function depends on only one variable that is denoted by t in the definition. In contrast to that, equations specifying a relation between partial derivatives of a function of two or more variables or of a vector with two or more dimensions and the function itself are called **partial differential equations (PDEs)**. An example is the Laplace equation briefly mentioned in Section 2.5 given by

$$\Delta u(x_1, \ldots, x_n) = \sum_{i=1}^{n} \frac{\partial^2}{\partial x_i^2} u(x_1, \ldots, x_n) = 0.$$

4.2. Existence and uniqueness of solutions. The main result on the conditions under which initial value problems have solutions is given by the Picard-Lindelöf theorem, which proves local existence and uniqueness for a locally Lipschitz-continuous function f defining the differential equation:

THEOREM 9 (Picard-Lindelöf, [**24**, p. 38, Theorem 2.2]). *Let $U \subset \mathbb{R} \times \mathbb{R}^m$ be an open subset, let $(t_0, x_0) \in U$ and $f : U \to \mathbb{R}^n$ be a continuous function. If f is locally Lipschitz continuous in the second argument, uniformly with respect to the first argument, then there exists a unique local solution of the initial value problem defined by f and (t_0, x_0). More specifically, if $V = [t_0, t_0 + T] \times \overline{B_\delta(x_0)} \subset U$ and $M = \max_{(t,x)\in V} |f(t, x)\|$, the solution exists at least for $t \in [t_0, t_0 + T_0]$ and remains in $\overline{B_\delta(x_0)}$, where $T_0 = \min\{T, \frac{\delta}{M}\}$. The analoguous result holds for $[t_0 - T, t_0]$.*

PROOF. I sketch the main steps of the proof and refer to [**24**, pp. 36-38] for details. The central idea is to reframe the statement of the theorem as a fixed-point equation of a functional on a complete metric space. Then, the claim follows from Banach's fixed-point theorem.

First, we observe that a function $x \in \mathcal{C}^1(J, \mathbb{R}^n)$ is a solution of the initial value problem defined by f and (t_0, x_0) in J if and only if the function $x \in \mathcal{C}(J, \mathbb{R}^n)$ solves

the integral equation ([**24**, p. 36])

$$x(t) = x_0 + \int_{t_0}^{t} f(s, x(s))ds.$$

Note that in order to satisfy the Lipschitz conditions, if suffices that $f \in \mathcal{C}^1(U)$: It then follows from the fact that any continuous function on a compact subset of a finite-dimensional real vector space attains its maximum in that subset (see [**23**, p. 71f.] for details) that for every compact set $V_0 \subset U$ the number

$$L = \sup_{(t,x)\neq(t,y)\in V_0} \frac{|f(t,x) - f(t,y)|}{|x - y|}$$

is finite ([**24**, p. 37]).

Next, we define the function space

$$\mathcal{X}_I := \{x \in \mathcal{C}(I, \mathbb{R}^n) : \sup_{t\in I} |x(t) - x_0| < \delta\}$$

for some $\delta > 0$ and an interval $I \subset \mathbb{R}$, and a functional K_I on $\mathcal{X}_I$

$$K_I : \mathcal{X}_I \to \mathcal{C}(I, \mathbb{R}^n),$$
$$(t \mapsto x(t)) \mapsto \left(t \mapsto T_I(x)(t) := x_0 + \int_{t_0}^{t} f(s, x(s))ds\right).$$

Then, according to our observation from the beginning of the proof, any function $x \in \mathcal{C}(I, \mathbb{R}^n)$ is a solution of the initial value problem defined by f and (t_0, x_0) on I if and only if $x(t) = K_I(x)(t)$ for all $t \in I$, in other words, if and only if the function x is a fixed point of K_I.

Defining $V := [t_0, T] \times \overline{B_\delta(x_0)} \subset U$ and denoting

$$M = \max_{(t,x)\in V} |f(t, x)|$$

(the maximum exists by continuity of f and compactness of V), we obtain

$$|K_I(x)(t) - x_0| \leq \int_{t_0}^{|} f(s, x(s))|ds \leq tM$$

for all $(t, x) \in V$. Hence, if we define

$$T_0 = \min\{T, \frac{\delta}{M}\},$$

it follows that $T_0 M \leq \delta$, and for all $t \leq T_0$,

$$|K_I(x)(t) - x_0| \leq T_0 M \leq \delta.$$

Defining $I := [t_0, T_0]$, this implies that $K_I(\mathcal{X}_I) \subset K_I(\mathcal{X}_I)$.

We can now use the fact that f is locally Lipschitz-continuous with respect to x to show that for $I := [t_0, T_0]$, K_I is a contraction on $\mathcal{X}_I$, i.e., $\sup_{t\in I} |(K_I(x))(t) - (K_I(y)(t))| \leq LT_0 \sup_{t\in I} |x(t) - y(t)|$ (see for details [**24**, pp. 40f.]). Since X_I is complete with respect to the supremum norm, the existence of a unique fixed point and thus also the existence of a unique solution of the initial value problem now follows from Banach's fixed-point theorem (see, e.g., [**24**, Theorem 2.1, p. 35]). □

REMARK. With the help of Gronwall's inequality ([**24**, p. 42f.]), it can be shown that the solution of an initial value problem depends continuously on the initial condition and the parameters of the system (see [**23**, Section 2.3, pp. 79-84] and [**24**, Section 2.4, pp. 41-48]). The existence of a solution can also be shown with Peano's existence theorem ([**24**, Theorem 2.19, pp. 56f.]) under weaker conditions on the function f. Here, the proof is based on applying a fixed-point theorem by Schauder and the Arzela-Ascoli theorem.

REMARK. If the coefficients of the differential equation grow at most linearly with respect to x, we can prove global existence of a solution: If in the setting of Theorem 9 for every $T > 0$ there are constants $M(T), L(T)$ such that

$$|f(t, x)| \leq M(T) + L(T)|x|,$$

the initial value problem defined by f and (t_0, x_0) has a unique solution defined for all $t \in \mathbb{R}$ (see [**24**, Theorem 2.17, p. 53]).

4.3. Linear ODEs.

DEFINITION 10. Let $I \subset \mathbb{R}$ be an interval. An ODE of the form

$$\frac{d}{dt}x(t) = A(t)x(t) + b(t)$$

for functions $A : I \to \mathbb{R}^{n\times n}, b : I \to \mathbb{R}^n$ is called **linear**. If $b \equiv 0$, the equation is called **homogeneuous**.

For homogeneuous linear ODEs with constant coefficients, i.e. with $A(t) \equiv A \in \mathbb{R}^{n\times n}$ for all $t \in I$, we can derive an explicit solution with the help of the following definition:

DEFINITION 11 ([**24**, p. 60]). For a matrix $B \in \mathbb{R}^{n\times n}$, we define the **matrix exponential**

$$\exp(B) := \sum_{k=0}^{\infty} \frac{B^k}{k!}.$$

THEOREM 12. *Let $B \in \mathbb{R}^{n\times n}$ be a matrix, let $J \in \mathbb{R}^{n\times n}$ be the Jordan normal form of B, such that $J = S^{-1}BS$ for a matrix $S \in \mathbb{R}^{n\times n}$. Then, for $f(t, x) = Bx$ and $(t_0, x_0) \in \mathbb{R} \times \mathbb{R}^n$, the solution of the initial value problem defined by f and (t_0, x_0) on $\mathbb{R}$ is given by the function*

$$x(t) = \exp(B(t - t_0))x_0 = S \exp(J(t - t_0))S^{-1}x_0.$$

PROOF. We can explicitly calculate that $x(t) = \exp(B(t - t_0))x_0$ solves the initial value problem. The second equality follows from the properties of the matrix exponential (for details see [**24**, Section 3.1, pp. 59-64]). □

REMARK. For a general linear homogeneous first-order systems, i.e., a system where the coefficient matrix A can depend on t, if $A \in \mathcal{C}(I, \mathbb{R}^{n\times n})$, existence and uniqueness of a solution on the entire interval I follow from the result mentioned in Remark 4.2 by choosing $L(T) = \max_{[0,T]} \|A(t)\|$ for every $T \in I$. To give an explicit form of the solution, it is neccessary to define the right generalisation of the expression $\exp((t - t_0)A)$. This is developed, e.g., in [**24**, Section 3.4, pp. 80-86].

CHAPTER 3

Methods

1. Modelling smooth latent dynamics within a VAE framework

We have seen that VAEs can infer lower-dimensional representations of the central factors of variation in a dataset. In the simulated data setting outlined in Chapter 1, we have such an underlying low-dimensional dynamical system from which the observations are generated. In an epidemiological cohort study such as the NAKO, we are typically interested in the underlying concept of the study participants' general condition and their mental and physical health status and try to access that health status, being a complex phenotype, by taking a multitude of individual measurements. It is thus plausible to assume the presence of groups of variables that jointly describe a more general underlying concept, e.g., we can imagine to collect data on individual's BMI, frequency of physical acticity, resting pulse rate and dietary habits all contributing to a latent representation of the individual's general physical fitness. Since we do not know specifically in what way single variables contribute to which part of a latent structure, the VAE provides a viable tool to infer such a latent representation directly from data in an unsupervised way, allowing for potentially complex, non-linear interactions. My model to infer the underlying lower-dimensional development patterns is therefore based on a VAE architecture.

The data are viewed as snapshots from a process evolving continuously over time that is observed with measurement noise. Consequently, also the underlying latent structure can be assumed to consist of dynamics changing smoothly over time and the latent space-representation has to be constrained to describe such smooth temporal development patterns. In my model, I formulate this constraint as the assumption that the latent space can be described by a system of differential equations

that capture the basic dynamics underlying the observations. This implies that the latent variable Z should represent both a random variable and a function changing over time, i.e., a stochastic process. Regarding the elements of stochasticity in the observed process, I assume a "true", inherently deterministic underlying dynamic process and a random force not acting on the process itself, but on a noisy observation of it. That observation is governed by some level of uncertainty, which I assume to be independent of the observation time. As a result, the latent space of the VAE model should encode a low-dimensional representation of the data that matches a smooth trajectory, with a stochastic element accounting for the uncertainty in the generating process of observations from an inherently deterministic underlying trajectory. This implies to model the stochasticity of the process as the random measurement error in the observations that can be assumed to be distributed according to $\mathcal{N}(0, \sigma^2)$. Consequently, I employ a Gaussian prior and posterior for the latent random variable Z. Here, the posterior mean represents its value according to the deterministic trajectory described by the ODE system, while the variance accounts for the uncertainty in the observation of that value.

Summing up, I describe the true deterministic process from which data are generated as an ODE system imposed on the latent space formulated in terms of the posterior mean μ_{post} of $Q^{Z|x} = \mathcal{N}(\mu_{\text{post}}, \sigma^2_{\text{post}})$ to account for the assumption of modelling a deterministic process with a random element in its observation, but not the temporal development itself. I therefore constrain the latent space to describe smooth dynamics by solving an ODE system for the posterior mean. To model the stochasticity of the measurement process, I subsequently sample z from the posterior distribution according to smooth dynamics after solving the ODE and decode it to the data space to obtain a reconstructed observation.

2. Fitting individual ODE parameters with baseline variables

As a further important feature, the model should be capable of extracting individual-specific development patterns, i.e., to fit individual ODE systems to the posterior mean of the latent representation. Since this representation should recover the

main factors of variation that govern the observed developments, I define a low-dimensional system of ODEs with one equation for every dimension of the latent representation that can be thought of as an underlying common trend shared by several variables in the original dataset.

The general structure of the ODE is explicitly specified as a linear system in Chapter 4, Sections 3 and 5 or as a non-linear system in Chapter 4, Section 4, but the parameters of the pre-defined system are determined individually for each input observation to account for different development patterns in the same set of measured variables. To infer such an individual-specific set of ODE parameters, I employ the additional variables measured only at the baseline timepoint, assuming that this more extensive characterisation carries information about each individual's development. Since I assume it to be unknown how they specifically relate to the individual ODE parameters, I employ an additional neural network to map the observations of the baseline variable to a set of ODE parameters for each individual. I train this network jointly with the VAE model, since the model should find a latent representation matching the assumption that the latent space can be described by an ODE system. By training jointly, the ODE structure imposed on the latent space is allowed to influence the VAE training and guide the model to find an appropriate latent representation.

As an example to motivate this method, we can imagine to measure several time-dependent variables describing lung function and at baseline also collect information about chronic lung diseases, age, physical activity and smoking habits. Then, these additional baseline measurements can potentially be informative about the development of each individual's lung function. More generally, if there are groups of individuals sharing common development trends, e.g., chain smokers, the baseline variables could contain information from which an individual's group membership can be deduced.

3. Derivation of the ODE-VAE training objective

I now formalise my method by describing the training process of the model and deriving an adapted version of the ELBO to define the loss function.

The dataset $\mathcal{D} = \{x_1, \ldots, x_n\}$ includes n observations that each consist of two measurements of p variables, i.e. $x_i = (x_i^{t_0} \quad x_i^{t_1}) \in \mathbb{R}^{p\times 2}$ with $x_i^{t_j} = (x_{i,1}^{t_j}, \ldots, x_{i,p}^{t_j})^\top \in \mathbb{R}^p$ for $j = 0, 1$. The encoder part of the VAE maps an input observation x_i column-wise to the mean $\mu_i = (\mu_i^{t_0} \quad \mu_i^{t_1}) \in \mathbb{R}^{k\times 2}$ and standard deviation $\sigma_i = (\sigma_i^{t_0} \quad \sigma_i^{t_1}) \in \mathbb{R}^{k\times 2}$ of the posterior distribution $\mathbb{Q}^{Z|x_i} = \mathcal{N}(\mu_i, \sigma_i^2)$, where k is the dimension of the latent space.

For each individual i, I denote with $y_{i,1}, \ldots, y_{i,q}$ the measurements of the additional baseline variables used to infer the parameters of the ODE system in latent space. A neural network that I refer to as the **ODE-net** maps the corresponding observation $y_i = (y_{i,1}, \ldots, y_{i,q})^\top \in \mathbb{R}^q$ to a set of individual ODE parameters $\eta_i \in \mathbb{R}^l$, with l being the number of parameters of the ODE system.

The latent space dynamics are given by a pre-specified function $f(\mu_i(t), t, \eta_i)$ that describes either a homogeneous two-dimensional linear ODE system or a non-linear Lotka-Volterra ODE system. The parameters of this ODE system are inferred from the individual-specific baseline measurements with the ODE-net. Then, we can solve the initial value problem

$$\begin{aligned} \frac{d}{dt}\mu_i(t) &= f(\mu_i(t), t, \eta_i) \\ \mu_i(t_0) &= \mu_i^{t_0} \end{aligned}$$

at t_i^1, the time point of the individual's second measurement, and obtain a vector $\widetilde{\mu}_i^{t_1} = \text{ODESolve}(f(\mu_i(t), t, \eta_i), \mu_i^{t_0}, t_i^1) \in \mathbb{R}^k$. We now define $\widetilde{\mu}_i := (\mu_i^{t_0} \quad \widetilde{\mu}_i^{t_1})$ as the mean of the posterior distribution constrained to a smooth dynamic and draw a sample $z_i \sim \widetilde{Q}_{\eta_i,\phi}^{Z|x_i} = \mathcal{N}(\widetilde{\mu}_i, \sigma_i^2)$ from the approximate posterior. Subsequently, z_i is passed to the decoder of the VAE that parameterises $P_\theta^{X|z_i}$, the conditional distribution of the data given a sample of the posterior, to obtain a reconstructed observation $\widehat{x}_i$ as a sample from $P_\theta^{X|z_i}$.

To define a training objective for jointly optimising the VAE model constrained to smooth latent space dynamics and the ODE-net, I adapt the ELBO of (2.13): To learn the latent space dynamics according to the ODE system jointly with the network supplying the parameters of this system, the posterior mean as obtained from solving the ODE is used. Because this mean and therefore also the posterior distribution depends on the parameters η_i of the individual's ODE system, we thereby introduce dependence of the ELBO on the ODE parameters and hence provide feedback for learning the weights and biases of the ODE-net:

$$\begin{aligned}\mathrm{ELBO}_{\mathrm{smooth}}(x_i^{t_0}, x_i^{t_1}, \eta_i, \theta, \phi) = &-D_{\mathrm{KL}}(\widetilde{q}_{Z|x_i^{t_0},x_i^{t_1}}(\cdot, \eta_i, \phi) \| p_Z(\cdot, \theta)) \\ &+ \mathbb{E}_{\widetilde{q}}[\log(p_{X|z_i^{t_0},z_i^{t_1}}(x_i^{t_0}, x_i^{t_1}, \theta))],\end{aligned} \tag{3.1}$$

where $\widetilde{q}$ is used as an abbreviation for $\widetilde{q}_{Z|x_i^{t_0},x_i^{t_1}}(\cdot, \eta_i, \phi)$ in the expectation. Since the solution of an ODE is uniquely defined (if it exists) by the function specifying the derivative and the initial value according to Theorem 9 (Picard-Lindelöf), the ODE solution $\widetilde{\mu}_i^{t_1}$ depends exclusively on the initial value $\mu_i^{t_0}$ and the ODE parameters η_i, but not on the observations at the second time point $x_i^{t_1}$. It follows that the posterior distribution $\widetilde{Q}_{\eta_i,\phi}^{Z|x_i}$ is independent of $x_i^{t_1}$, hence $\widetilde{q}_{Z|x_i^{t_0},x_i^{t_1}}(\cdot, \eta_i, \phi) = \widetilde{q}_{Z|x_i^{t_0}}(\cdot, \eta_i, \phi)$. To improve the capability of the model to still reconstruct $x_i^{t_1}$ along with $x_i^{t_0}$ and to provide feedback for the encoder weights and biases with respect to $x_i^{t_1}$, the loss is augmented with the squared euclidean distance between $\mu_i^{t_1}$ as obtained from directly passing $x_i^{t_1}$ through the encoder and $\widetilde{\mu}_i^{t_1}$ as obtained from solving the ODE. By minimising that distance, the posterior distribution from the encoder $q_{Z|x_i^{t_0},x_i^{t_1}}(\cdot, \phi)$ is brought to match the posterior distribution $\widetilde{q}_{Z|x_i^{t_0}}(\cdot, \eta_i, \phi)$ constrained to smooth latent space dynamics. In that way, the model is encouraged to encode the data from both measurement time points into a lower-dimensional representation that matches the smooth structure reinforced by the ODE system even before applying the ODE solving step. Since the encoder posterior $q_{Z|x_i^{t_0},x_i^{t_1}}(\cdot, \phi)$ directly depends on $x_i^{t_1}$, bringing it close to $\widetilde{q}_{Z|x_i^{t_0}}(\cdot, \eta_i, \phi)$ implicitly provides feedback from $x_i^{t_1}$ on $\widetilde{\mu}_i^{t_1}$ and thus achieves an implicit conditioning of $\widetilde{q}_{Z|x_i^{t_0}}(\cdot, \eta_i, \phi)$ on $x_i^{t_1}$. Adding the term

with a weighting factor of $\alpha \in [0, 1]$ results in the final loss

$$\begin{aligned}\mathcal{L}(x_i^{t_0}, x_i^{t_1}, \eta_i, \theta, \phi) &= -\text{ELBO}_{\text{smooth}}(x_i^{t_0}, x_i^{t_1}, \eta_i, \theta, \phi) + \alpha \|\mu_i^{t_1} - \widetilde{\mu}_i^{t_1}\|_2^2 \\ &= D_{\text{KL}}(\widetilde{q}_{Z|x_i^{t_0}, x_i^{t_1}}(\cdot, \eta_i, \phi) \| p_Z(\cdot, \theta)) \\ &\quad - \mathbb{E}_{\widetilde{q}}[\log(p_{X|z_i^{t_0}, z_i^{t_1}}(x_i^{t_0}, x_i^{t_1}, \theta))] \\ &\quad + \alpha \|\mu_i^{t_1} - \widetilde{\mu}_i^{t_1})\|_2^2.\end{aligned} \tag{3.2}$$

Note that by minimising $\mathcal{L}$, we still maximise a lower bound on the data likelihood, since the negative loss equals the ELBO with a positive term subtracted. Heuristically, by jointly maximising the ELBO and minimising the distance of $\mu_i^{t_1}$ and $\widetilde{\mu}_i^{t_1}$ via minimising $\mathcal{L}$, we can even obtain a lower bound closer to the true data likelihood due to the implicit conditioning of the smooth posterior $\widetilde{Q}_{\eta_i,\phi}$ on $x_i^{t_1}$, which potentially leads to a better reconstruction error while still maintaining a smooth dynamic structure of the latent state. Figure 6 provides a graphical overview of my model and its training process.

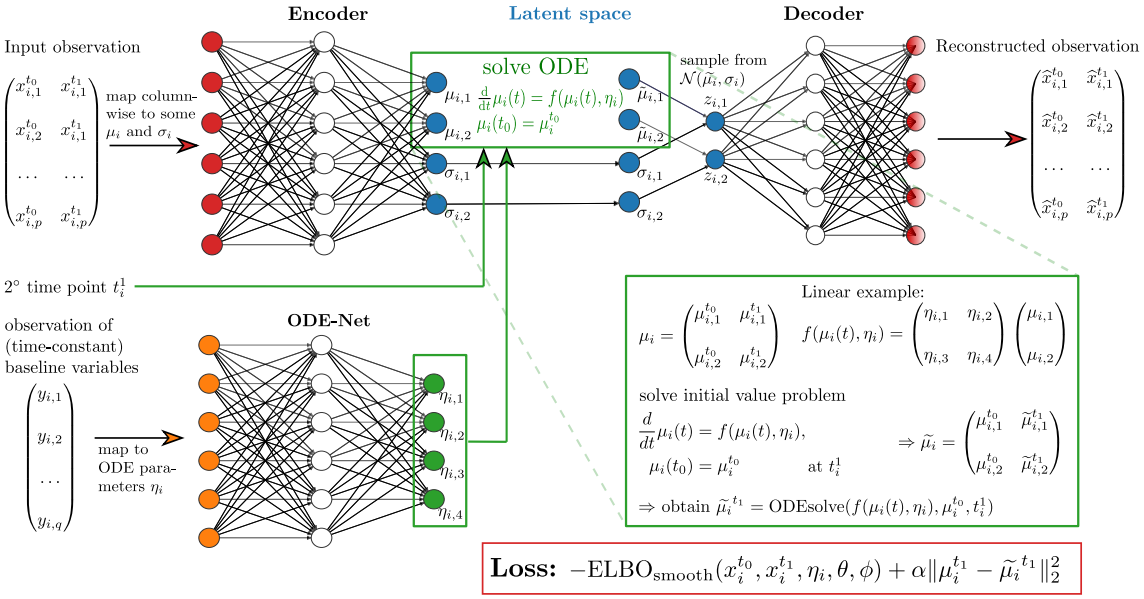


FIGURE 6. Overview of model architecture and training.

4. Model extension: training on batches of similar individuals

4.1. Time sparsity and notion of similarity. A major challenge for my approach of modelling dynamic processes with individual-specific ODE systems in latent space is the strong sparsity and irregularity of the time grid with only two observations for every individual, which implies constraints on the complexity of ODE systems that can be fitted with these data. While it is natural to assume that two time points allow to estimate two unknown ODE parameters for each individual, the model should optimally also capture more complex development patterns.

This is achieved with an extension of my method: To every individual, a batch of individuals with similar underlying development patterns is assigned. Then, the combination of all their second time point measurements can be used as proxy information on the common dynamics at multiple time points. Thus, each observation is enriched with a group of similar ones observed at different time points and the irregularity of the time grid is exploited to address the problem of strong time sparsity, which ultimately allows to model more complex development patterns.

More specifically, first a measure of similarity has to be defined to identify individuals with similar development patterns. Since each individual is observed at a different second time point, their measurements cannot be directly compared in data space. Additionally, if the random measurement noise in the observations superimposes the true underlying developments, it can be impossible to identify similar trajectories from the data alone.

Figure 7 provides an illustration of the batch assignment: For each individual, the observations are generated from one of two different underlying development patterns with a moderate level of noise. The original underlying pattern is indicated by the frame colour of each panel (violet/green). The development of the individual at the top is different from the left one in the second row, but similar to the right one in the third row. However, it is challenging to judge from these observations alone whether the right individual in the second row and the left in the third row are based on different underlying development patterns. This becomes clear only after passing

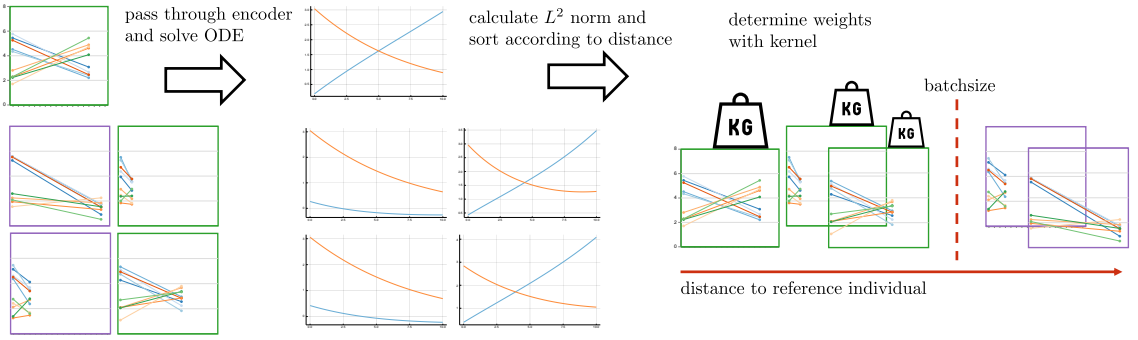


FIGURE 7. Assigning to an individual a batch of similar ones.

each observation through the encoder and comparing the latent representation of the trajectories: In latent space, depicted in the middle part of the figure, we can clearly distinguish the two general development patterns.

Consequently, I define the similarity of individuals based on their latent trajectories represented by the solution of each individual's ODE system in latent space. To infer for a given individual a batch of similar ones, I therefore first pass all observations through the VAE encoder. In the latent space of ODE solutions, I calculate the L^2-norm of distances of the ODE solutions and sort individuals according to distance to the reference individual. Additionally, I use a kernel to weigh the individuals with respect to their similarity to the reference individual. The complete method is formalised in the following sections.

4.2. Defining an iterative optimisation framework. The proposed notion of similarity between individuals refers to the similarity not of their observed measurements, but of their unobserved latent trajectories, i.e., their ODE solutions in the latent space of the ODE-VAE model, that should ideally reflect the true underlying development patterns. However, inferring such a latent representation that recovers the true ODE system underlying every individual's observed measurements is the objective of the training process that we aim to improved by using the batches of similar individuals in the first place. In other words, a notion of similarity based on the underlying trajectories is needed to group individuals to train the model, i.e., to find a latent representation recovering the true dynamics, but at the same

time such a meaningful latent representation is already needed to accurately infer individuals' similarity.

To overcome this problem, I introduce an iterative optimisation framework that alternates between the assignment of batches of similar individuals and the joint training of the ODE-VAE model. In this way, the latent representation of the trajectories are improved iteratively, so that they more closely reflect the true developments and individuals can be grouped more accurately based on these trajectories. Training on these groups – that more accurately capture the similarity of the underlying patterns – in turn improves individuals' latent representations due to better proxy information at other time points from the batch. This results in an overall training method that resembles an expectation-maximisation framework (Chapter 2, Section 2.2).

4.2.1. *Assigning a batch to each individual.* To group individuals in batches based on the similarity of their ODE solutions in latent space, every individual observation $x_i \in \mathcal{D}, i \in \{1, \ldots, n\}$ is first passed through the VAE encoder to obtain an approximate posterior mean $\mu_i = (\mu_i^{t_0} \quad \mu_i^{t_1}) \in \mathbb{R}^{k\times 2}$ and variance $\sigma_i = (\sigma_i^{t_0} \quad \sigma_i^{t_1}) \in \mathbb{R}^{k\times 2}$ as in Section 3. Also, the respective baseline measurements y_i are passed through the ODE-net to obtain a set of individual ODE parameters η_i. Next, each individual's ODE system as defined by the function f denoting the common ODE structure and the individual parameters η_i is solved to obtain for each individual the posterior mean according to smooth dynamics as a function of t:

$$\mu_i(t) = \text{ODEsolve}(f(\mu_i(t), t, \eta_i)).$$

For a $T \in \mathbb{R}$, it holds that for all i, $\mu_i \in L^2([t_0, T])$. Now, we calculate a symmetric distance matrix

$$D = (d_{i,j})_{i,j=1,\ldots,n} = \left(\|\mu_i - \mu_j\|_{L^2([t_0,T])}\right)_{i,j=1,\ldots,n}$$

where the i, j-th entry denotes the distance in L^2-norm between the ODE solutions $\mu_i(t)$ and $\mu_j(t)$ of individuals i and j. In my implementation, the ODE solutions

are evaluated at a fixed grid of $m+1$ time points $t_0, \ldots, t_m$ to approximate the L^2-integral:

$$d_{i,j} = \left(\frac{1}{m} \sum_{t=t_0}^{t_m} \|\mu_i(t) - \mu_j(t)\|_2^2 \right)^{\frac{1}{2}}. \tag{3.3}$$

For a pre-defined batch size of b, to each individual the $(b-1)$ individuals closest to that reference individual are assigned: Let $i \in \{1, \ldots, n\}$ be the index of the reference individual. Of course, $d_{i,i} = 0$. The remaining entries in row i are sorted such that $d_{i,j_1} \leq d_{i,j_2} \leq \cdots \leq d_{i,j_{n-1}}$ where $j_k \in \{1, \ldots, i-1, i+1, \ldots, n\}$ for $k = 1, \ldots, n-1$. For any $a < n$, the indices $j_1, \ldots, j_a$ now denote the indices of the a closest individuals to the reference individual. Thus, a batch B_i of the $b-1$ individuals closest to x_i can be defined by setting

$$B_i = \{x_i, x_{j_1}, \ldots, x_{j_{b-1}}\}.$$

The closer an individual is to the reference individual, the more similar is its underlying development pattern. This implies that also the substistute information provided by that individual is potentially more accurate and should thus be regarded as more important for the training of the specific batch. To account for that, each individual in the batch is weighted according to its calculated distance to the reference individual. Specifically, weights $w_i, w_{j_1}, \ldots, w_{j_{b-1}}$ are determined for each individual in the batch B_i with a smoothing kernel K_h, such that

$$w_k = \frac{K_h(d_{i,k})}{\sum_{k=i,j_1,\ldots,j_{b-1}} K_h(d_{i,k})} \quad \text{for all } k = i, j_1, \ldots, j_{b-1}. \tag{3.4}$$

The kernel is defined as the tricube function given by

$$K(x) = \begin{cases} (1- \mid x \mid^3)^3 & \mid x \mid \leq 1, \\ 0 & \text{else} \end{cases}$$

and can be scaled with the bandwidth parameter h by defining $K_h(x) = \frac{1}{h} K(\frac{x}{h})$.

4.2.2. *Adapting the ODE-VAE loss to training on batches.* Having determined for each individual a batch of individuals exhibiting similar temporal development patterns, the ODE-VAE model is trained on the resulting batches in order to improve the estimates of the variational parameters of the approximate posterior ϕ, the model parameters θ and the individual ODE parameters η_i with respect to the training objective $\mathcal{L}(x_i^{t_0}, x_i^{t_1}, \eta_i, \theta, \phi)$ of (3.2):

For each individual in the batch, the respective loss value is calculated as described in Section 3. Since for each individual the solution at the individual-specific second measurement time point is used to calculate its loss, those additional values now serve as proxy information for measurements at more time points (one from each individual in the batch) of the reference individual. To use them to enrich this reference individual's information and stabilise the gradient of its loss function, we derive a common loss value for the entire batch as the weighted average of individual loss values, using the weights derived from the distance matrix of the first step. The final training objective is thus given by the loss of each batch B_i, $i = 1, \ldots, n$:

$$\mathcal{L}_{\text{batch}}(B_i, (\eta_k)_{k=i,j_1,\ldots,j_{b-1}}, \theta, \phi) = \sum_{k=i,j_1,\ldots,j_{b-1}} w_k \cdot \mathcal{L}(x_k^{t_0}, x_k^{t_1}, \eta_k, \theta, \phi). \tag{3.5}$$

The weighting approach ensures that the individuals contributing most strongly to the common loss are the ones which best match the trajectory of the reference individual and therefore provide a good proxy for the observations of the reference individual at additional time points.

The overall method to train the ODE-VAE model on batches now consists of iteratively applying the two steps and is summarised in Algorithm 2.

Algorithm 2: Training procedure of the ODE-VAE on batches

Data:

Dataset $\mathcal{D} = \{x_1, \ldots, x_n\}$ of time-dependent variables,

additional baseline variables $y_1, \ldots, y_n$

Inference model: family of approximate variational posterior distributions $(Q_\phi^{Z|x})_{\phi\in\Phi} = (\mathcal{N}(\mu, \sigma^2))_{(\mu,\sigma^2)\in\mathbb{R}^{k\times 2}\times\mathbb{R}^{k\times 2}}$,

Generative model: family of joint distributions $(P_\theta^{X,Z})_{\theta\in\Theta}$,

Function $f : \mathbb{R}^k \times \mathbb{R} \times \mathbb{R}^l \to \mathbb{R}^k$ defining general structure of latent ODE system, parameterised by $\eta \in \mathbb{R}^l$

Result:

Learned parameters (θ, ϕ) of the generative and inference model,

individual-specific ODE parameters η_i

$(\theta, \phi, \text{ODEnet}) \leftarrow$ Random initialization;

while *SGD not converged* **do**

  for *each datapoint* $x_i \in \mathcal{D}$ **do**

    Compute $\eta_i = \text{ODEnet}(y_i)$;

    Compute $\mu_i(t) = \text{ODEsolve}(f(\mu(t), t, \eta_i), \mu_i^{t_0})$;

    Compute distance matrix $D = (\|\mu_i(t) - \mu_j(t)\|_{L^2})_{i,j=1,\ldots,n}$;

    Sort $d_{i,j_1} \le d_{i,j_2} \le \cdots \le d_{i,j_{n-1}}$, $j_k \in \{1, \ldots, i-1, i+1, \ldots, n\}$;

    Define $B_i = \{x_i, x_{j_1}, \ldots, x_{j_{b-1}}\}$;

    Calculate $w_k = \frac{K_h(d_{i,k})}{\sum_{k=i,j_1,\ldots,j_{b-1}} K_h(d_{i,k})}$;

  end

  for *each batch* $B_i = \{x_i, x_{j_1}, \ldots, x_{j_{b-1}}\}$ **do**

    for *each* $x \in B_i$ **do**

      Sample random noise $\varepsilon \sim P^{\mathcal{E}}$ and compute $z = g(\varepsilon, \phi, x)$;

      Compute $\mathcal{L}(x_k^{t_0}, x_k^{t_1}, \eta_k, \theta, \phi)$;

    end

    Compute

    $\mathcal{L}_{\text{batch}}(B_i, (\eta_k)_{k=i,j_1,\ldots,j_{b-1}}, \theta, \phi) = \sum_{k=i,j_1,\ldots,j_{b-1}} w_k \cdot \mathcal{L}(x_k^{t_0}, x_k^{t_1}, \eta_k, \theta, \phi)$;

  end

  Update parameters θ, ϕ and ODE-net using SGD optimiser;

end

CHAPTER 4

Simulation study

In this chapter, I apply the ODE-VAE model developed in Chapter 3 to different simulated data settings inspired by the scenario of the NAKO sub study outlined in Chaper 1. First, I explain the simulation design and give details on the model implementation. Next, I present results from training the model on data generated from linear and non-linear two-dimensional ODE systems with two unknown parameters and on a linear system with four unknown parameters, where I employ the approach of Chapter 3, Section 4 to train on batches of similar individuals. I compare scenarios with different variants of simulated baseline information, with different amounts of noise present in the baseline variables, and with different noise levels in the simulated measurements of the true underlying trajectory.

1. Simulation design

In all simulated data settings, two distinct underlying development patterns are modelled, each given as the solution of a two-dimensional ODE system. Thus, all observations can be categorised into two groups with distinct development patterns and there are two central components characterising each development pattern. Since our aim is to recover those central factors of variation as time-continuous functions in the latent representation, the VAE latent space is set to be two-dimensional.

After specifying the general structure of the two-dimensional ODE, two sets of parameters are defined for the common underlying system structure, such that the two resulting ODE systems have qualitatively different – and thus easily distinguishable – solutions. For example, in a linear case, I define one system where both components display a downward trend, and the other one with an upward trend in one component and a downward trend in the other component.

Both the linear and the non-linear ODE systems are governed by four parameters. In the applications of Sections 3 and 4, two of these parameters are assumed as known and explicitly defined beforehand together with the general linear or Lotka-Volterra-like structure of the ODE system. During training, the model then only learns estimates for the two remaining parameters as outputs of the ODE-net. In the application of Section 5, all four parameters are assumed to be unknown and estimated with the ODE-net.

Generally, I simulate observations of n individuals by drawing from the true underlying trajectories given as solutions of the pre-defined ODE systems with random noise to account for the measurement error. Individuals are randomly divided into two groups of equal size defined by the two distinctly parameterised underlying ODE systems. Algorithm 3 summarises the simulation procedure. For each individual, two observations of $p = 10$ variables are generated, one observation at a common baseline timepoint $t_0 = 0$ and one at an individual-specific second time point t_1^i sampled uniformly from the interval $[1.5, 10]$. I chose 1.5 as earliest second time point to account for the assumption that no individual was measured again directly after the basline time point. The simulation should reflect the assumption that the underlying development patterns form a compressed representation of the measurements and that there are groups of observed variables sharing a common trend. Thus, for each individual, measurements of the first five variables are simulated by sampling from the first component of the individual's true trajectory both at the common baseline time point t_0 and at the individual-specific second time point t_1^i. For variables 6 to 10, I sample in the same fashion from the second component of the individual's true trajectory at both time points. Specifically, for each time point, I draw a variable-specific measurement error $\delta_j^{t_0}, \delta_j^{t_1} \sim \mathcal{N}(0, \sigma_{\text{var}}^2)$ for $j = 1, \ldots, p = 10$ and an individual-specific measurement error $\varepsilon_{i,j}^{t_0}, \varepsilon_{i,j}^{t_1} \sim \mathcal{N}(0, \sigma_{\text{ind}}^2)$ for $i = 1, \ldots, n$, $j = 1, \ldots, p$ and add them to the true value of the corresponding component of the ODE solution at t_0 and t_1^i, respectively. The standard deviations σ_{var} and σ_{ind} of the measurement error distributions determine the level of noise in the data.

Algorithm 3: Simulation of time-dependent variables

Data: $n \in 2 \cdot \mathbb{N}, \quad p \in \mathbb{N}, \quad \sigma_{\text{var}}, \sigma_{\text{ind}} \in \mathbb{R}^+$

Result: $\mathcal{D} = \{x_1, \ldots, x_n\}$ with $x_i = (x_i^{t_0}, x_i^{t_1}), i = 1, \ldots, n$

Initialise: sample random permutation $\pi(1, \ldots, n)$ and set

$\text{group}_1 = \{\pi(1, \ldots, n)_k \mid k = 1, \ldots, \frac{n}{2}\}$,

$\text{group}_2 = \{\pi(1, \ldots, n)_k \mid k = (\frac{n}{2} + 1), \ldots, n\}$;

for $j = 1, \ldots, p = 10$ **do**

  sample $\delta_j^{t_0}, \ \delta_j^{t_1} \sim \mathcal{N}(0, \sigma_{\text{var}}^2)$;

end

for $i = 1, \ldots, n$ **do**

  sample $t_i^1 \sim \text{UC}_{[1.5,10]}$;

  solve $v_1^{t_i^1} = \text{ODEsolve}(f_{\text{group}_i}(u_1, t), u_1(t_0), t_i^1)$,

   $v_2^{t_i^1} = \text{ODEsolve}(f_{\text{group}_i}(u_2, t), u_2(t_0), t_i^1)$;

  for $j = 1, \ldots, \frac{p}{2}$ **do**

    sample $\varepsilon_{i,j}^{t_0}, \ \varepsilon_{i,j}^{t_1} \sim \mathcal{N}(0, \sigma_{\text{ind}}^2)$;

    compute $x_{i,j}^{t_0} = u_1(t_0) + \delta_j^{t_0} + \varepsilon_{i,j}^{t_0}$, $x_{i,j}^{t_1} = v_1^{t_i^1} + \delta_j^{t_1} + \varepsilon_{i,j}^{t_1}$;

  end

  for $j = (\frac{p}{2} + 1), \ldots, p$ **do**

    sample $\varepsilon_{i,j}^{t_0}, \ \varepsilon_{i,j}^{t_1} \sim \mathcal{N}(0, \sigma_{\text{ind}}^2)$;

    compute $x_{i,j}^{t_0} = u_2(t_0) + \delta_j^{t_0} + \varepsilon_{i,j}^{t_0}$, $x_{i,j}^{t_1} = v_2^{t_i^1} + \delta_j^{t_1} + \varepsilon_{i,j}^{t_1}$;

  end

end

In addition to the time-dependent variables, I simulate measurements of $q = 50$ additional baseline variables representing the individuals' more extensive characterisation that are assumed to be potentially informative about the individual development pattern in the time-dependent variables (see Chapter 3, Section 2).

Here, two different approaches are applied as summarised in Algorithms 4 and 5. In the first version, the values of the baseline variables are drawn from the

individual's true ODE parameters with random noise. For each unknown ODE parameter, I simulate a number of variables containing noisy values of this one ODE parameter and add noise variables not containing any information, such that I end up with a total of 50 baseline variables. In Section 4.3, I discuss the effect of changing the number of noise variables and present results for different settings.

The second version models a scenario where only some knowledge about the group membership of each individual is assumed. We can imagine to know that the observed individuals can be classified into two groups that are expected to exhibit different development patterns. As an example, if the time-dependent variables measure lung functions and there are smoking and non-smoking individuals in the dataset, it can be assumed that the general development of lung functionality for smoking people will differ from that of non-smokers and we can use that information to estimate individual-specific ODE parameter sets. Again, I simulate a number of variables with values sampled from the individual's group membership (encoded as $+1$ or -1) with random noise and add noise variables as in the first version of simulated baseline variables with the true ODE parameters. The number of baseline variables containing information either about the true ODE parameters or the group membership is denoted with q_{info}.

2. Implementation details

The following section provides details on the implementation of the proposed method. The code to run all simulations and produce all figures of this chapter is written in the Julia programming language of version 1.1.1 with the additional packages DataFrames.jl (v0.20.0), DiffEqFlux.jl (v0.10.0), DifferentialEquations.jl (v6.9.0), Distributions.jl (v0.21.9), Flux.jl (v0.9.0), LaTeXStrings.jl (v1.0.3), Plots.jl (v0.28.4), StatsBase.jl (v0.32.0) and VegaLite.jl (v1.0.0).

For the VAE model, I aimed at keeping the architecture simple and close to established models from literature. Specifically, I used one hidden layer with the number of hidden units equal to the number of input dimensions (i.e., 10 in all applications) and a tanh-activation function shifted by $+1$. The latent space is set to be

Algorithm 4: Simulation of baseline variables with true ODE parameters

Data: $n \in 2\mathbb{N}, \quad q, q_{\text{info}} \in \mathbb{N}, \quad \sigma_{\text{info}}, \sigma_{\text{noise}} \in \mathbb{R}^+$

Result: $\{y_1, \ldots, y_n\}$ with $y_i = (y_{i,1}, \ldots, y_{i,q})^\top, i = 1, \ldots, n$

for $j = 1, \ldots, q_{\text{info}}$ **do**

  for $i = 1, \ldots, n$ **do**

    sample $y_{i,j} \sim \mathcal{N}(\text{ODEparameters}_{\text{group}_\text{i}}, \sigma^2_{\text{info}})$;

  end

end

for $j = (q_{\text{info}} + 1), \ldots, q$ **do**

  for $i = 1, \ldots, n$ **do**

    sample $y_{i,j} \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$;

  end

end

Algorithm 5: Simulation of baseline variables with group membership

Data: $n \in 2\mathbb{N}, \quad q, q_{\text{info}} \in \mathbb{N}, \quad \sigma_{\text{info}}, \sigma_{\text{noise}} \in \mathbb{R}^+$

Result: $\{y_1, \ldots, y_n\}$ with $y_i = (y_{i,1}, \ldots, y_{i,q})^\top, i = 1, \ldots, n$

for $j = 1, \ldots, q_{\text{info}}$ **do**

  for $i = 1, \ldots, n$ **do**

    sample $y_{i,j} \sim \mathcal{N}(\pm 1_{\text{group}_i}, \sigma^2_{\text{info}})$;

  end

end

for $j = (q_{\text{info}} + 1), \ldots, q$ **do**

  for $i = 1, \ldots, n$ **do**

    sample $y_{i,j} \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$;

  end

end

two-dimensional and the latent space mean and variance are obtained as affine linear transformations of the hidden layer values without a non-linear activation. The decoder includes one hidden layer with 10 hidden units and a tanh-activation function and outputs mean and variance of a Gaussian distribution that are calculated from the hidden layer by using an affine linear transformation without activation function. In the loss function, the KL-divergence between the prior and posterior is downweighed with a factor of 0.5 to reduce the regularising effect. The sum of squared values of all decoder parameters with a weighting factor of 0.01 is added as commonly used penalty term to prevent exploding model parameters.

The ODE-net consists of three layers in addition to the input layer, two hidden layers and one output layer. In the first hidden layer, the number of units equals the number of input dimensions (i.e., 50 in all the applications) and a tanh-activation function is used. In the second layer, the number of hidden units equals the number of parameters to be estimated and the activation function is a shifted sigmoid function. The shifting of the activation function serves as a prior to guide the model in identifying the range in which to estimate the ODE parameters: For the applications with a linear underlying ODE systems and true parameters of around 0.2 or -0.2 (Sections 3 and 5), the sigmoid function is shifted by -0.5 such that it outputs values in the interval $[-0.5, 0.5]$. For the application on a non-linear systems with true parameters between 0.5 and 2, it is shifted by $+1$, such that it outputs values in $[0, 2]$. However, in order not to restrict the model to those ranges but let it freely shift and scale the sigmoid-transformed values, an affine linear transformation with a diagonal matrix is added as final output layer.

All reported numbers of epochs and learning rates are chosen based on monitoring convergence of the loss function and visualising training results. Neither a systematic grid search nor excessive tuning of these hyperparameters was performed.

For the applications with the non-linear ODE system from Section 4, I frequently observed training runs where the random weight initialisation of the network produced instabilities in the numerical solving of the ODE with the Tsit5-ODE solver

implementing a Tsitouras 5/4 Runge-Kutta method. To reduce the number of resulting training break-offs, I changed the default initialisation method for the weights to sampling from smaller values. While this did result in less solver instabilities, it did not prevent them completely.

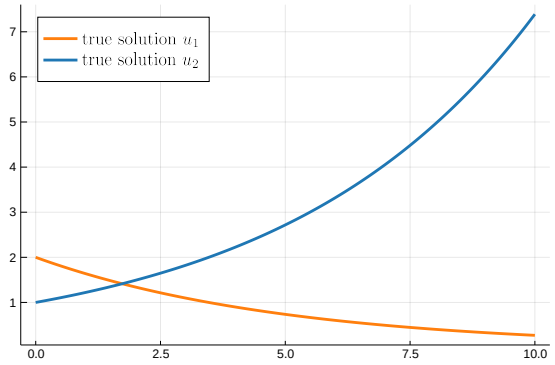
Generally, backpropagating through the ODE solving step within the neural network framework is realised by the DiffEqFlux.jl-package (for details, see [**25**]).

In the scenario with four unknown ODE parameters, I employed a discretisation to efficiently approximate the L^2-norm to determine the distance matrix (see Equation 3.3 in Chapter 3, Section 4.2) with $m + 1 = 11$ equidistant values. Here, as an additional centralising step, from each value the mean of all $m + 1$ values from the discretisation is subtracted to ensure the distance is based on the actual development over the time interval, not on random similarity of absolute values due to the neccessary discretisation.

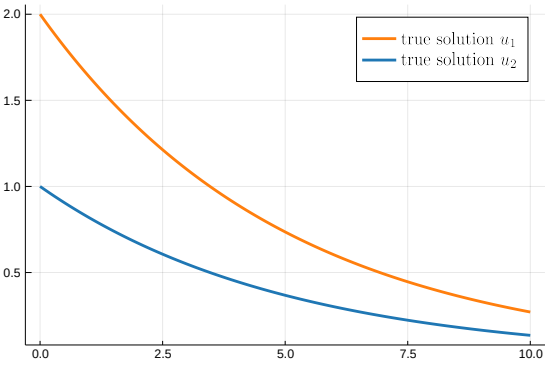
3. Linear ODE system with two unknown parameters

3.1. Structure of input data. First, two distinct underlying development patterns are defined as solutions of the two ODE systems given in Equations (4.1) and (4.2) and visualised in Figure 8. The simulated individuals fall into two groups, one with an upward trend in one component and a downward trend in the other component, and the other group with a downward trend in both components.

The data is simulated by drawing measurement points from these true trajectories with random noise as described in Algorithm 3. More precisely, I generate observations of 100 individuals, 50 from each of the two groups, and generate observations of 10 variables by adding variable-specific and the individual-specific measurement errors to 5 values from each of the two ODE solution components both at the initial time point $t_0 = 0$ and an individual-specific second time point t_1^i. In this first setting, I simulate a low level of noise and set $\sigma_{\text{var}} = \sigma_{\text{ind}} = 0.1$. Figure 9 visualises all simulated observations together with the true ODE solutions. To illustrate the structure of each simulated individual's observations, Figure 10 exemplarily shows the observations of 20 simulated study participants on an individual level. Each

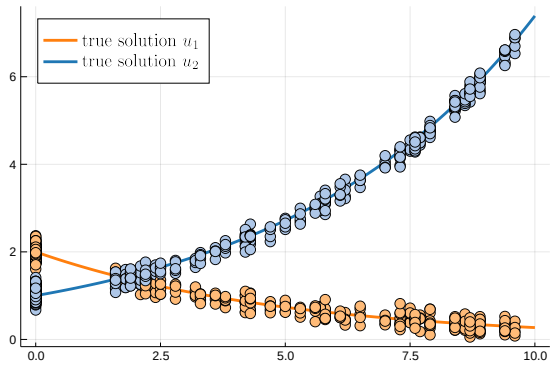


$$\frac{d}{dt}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}(t) = \begin{pmatrix} -0.2 & 0 \\ 0 & 0.2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}(t); \quad \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}(0) = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \tag{4.1}$$



$$\frac{d}{dt}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}(t) = \begin{pmatrix} -0.2 & 0 \\ 0 & -0.2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}(t); \quad \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}(0) = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \tag{4.2}$$

FIGURE 8. True underlying development patterns characterised as solutions of two-dimensional linear ODE systems.



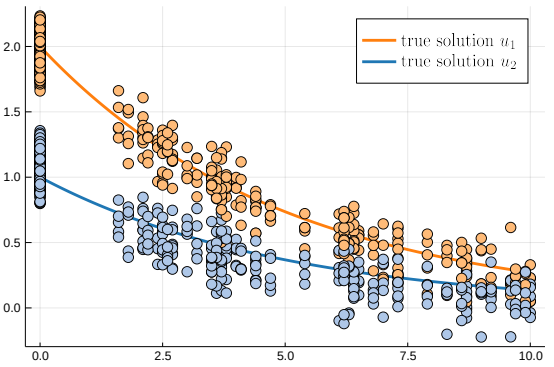


FIGURE 9. Simulated observations of 100 individuals (50 in each group) at t_0 and an individual-specific time point t_1^i.

panel represents one individual, dots represent individual measurements and colors indicate different variables similar to Figure 1 in the introduction.

3.2. Comparison of different simulation scenarios of baseline variables. On the simulated data described in the previous section, the ODE-VAE model is trained as described in Chapter 3, Section 3. Specifically, the ODE-net estimates the

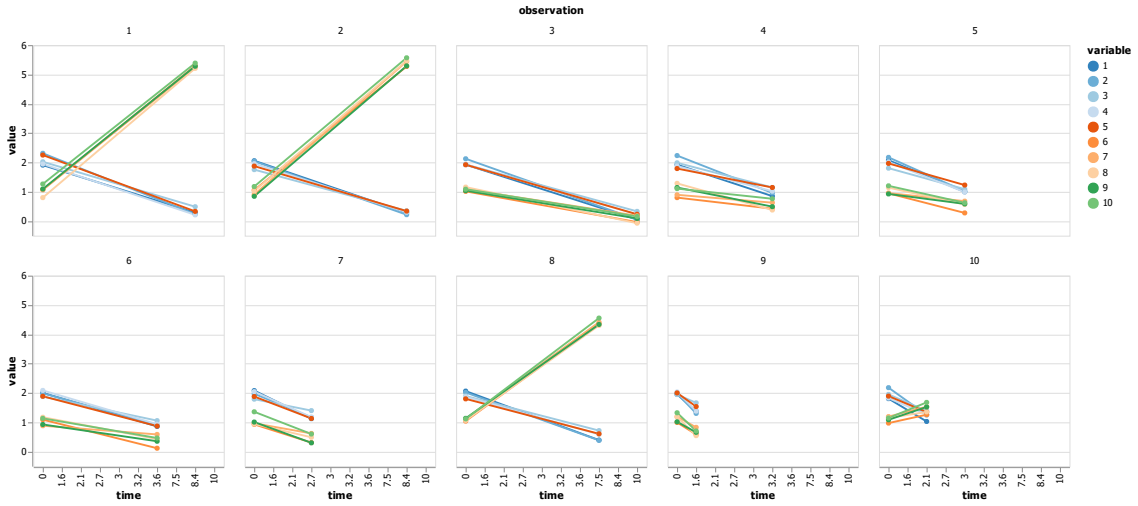


FIGURE 10. Individual-level simulated data structure of 20 individuals.

two non-zero parameters of each of the two ODE systems while the two off-diagonal entries are set to zero, i.e., no interaction is allowed between the components of the ODE solution. Thus, for each individual i the initial value problem

$$\frac{d}{dt}\mu_i(t) = \begin{pmatrix} \eta_{i,1} & 0 \\ 0 & \eta_{i,2} \end{pmatrix} \begin{pmatrix} \mu_{i,1} \\ \mu_{i,2} \end{pmatrix}(t);$$
$$\mu_i(t_0) = \mu_i^{t_1}$$

is solved in latent space at the individual's second measurement time point t_i^1 with ODE-net outputs $\eta_{i,1}, \eta_{i,2}$.

As a first application, I compare the different simulations scenarios of baseline variables described in Algorithms 4 and 5. For the first scenario with the true ODE parameters, I set $\sigma_{\text{info}} = \sigma_{\text{noise}} = 0.1$ and for the second scenario with the group membership, I set $\sigma_{\text{info}} = \sigma_{\text{noise}} = 0.5$, such that for both scenarios, the distributions of noise and the information overlap beyond one standard deviation. In each scenario, I simulate 10 variables containing the respective baseline information, add 40 noise variables and train the model for 35 epochs with the ADAM SGD-optimiser ([**26**]) and a learning rate of 0.0005.

Figure 11 depicts the fitted ODE solutions of two individuals both for a model trained with the true ODE parameters and one trained with only the group membership as baseline information. In both scenarios, the model recovers the general structure of the underlying patterns, recognising the distinct upward/downward trends of variables for the two individuals. It tends to overestimate the initial value of the first component of the solution and to underestimate it in the second component. For the training with the true ODE parameters, the overestimation of the first component initial value is more pronounced, whereas the underestimation in the second component is smaller than for training with only the group information. In both scenarios, the first dimension of the mean obtained directly from the VAE encoder before solving the ODE (blue dots) closely matches the first dimension of the smooth mean obtained as ODE solution (solid blue lines), whereas for the second component, in the second group, the means from the encoder for the second time point (orange dots) match the true ODE solution (dotted orange lines) more closely than the fitted ODE solutions. Thus, in general the training strategy of matching the means before and after solving the ODE – to both find a latent representation modelling a smooth trajectory even before solving the ODE and to implicitly condition the ODE solution on the data from the second time point – works, while there is some variation in whether the focus on matching the ODE solution or matching the data dominates.

To get a broader overview, the distributions of ODE solutions are compared across the whole dataset. In Figure 12, the fitted ODE solutions and encoder means from all individuals of one group are overlayed, and again both groups of underlying development patterns and both scenarios of baseline variables are compared.

As expected, the results from training with only the group membership as baseline information, representing the more difficult scenario, display a higher variability between individual ODE solutions in both groups. In this setting, the model also has greater difficulty matching the upward development in the first group of individuals with the posterior mean before solving the ODE (blue dots). While this trend is

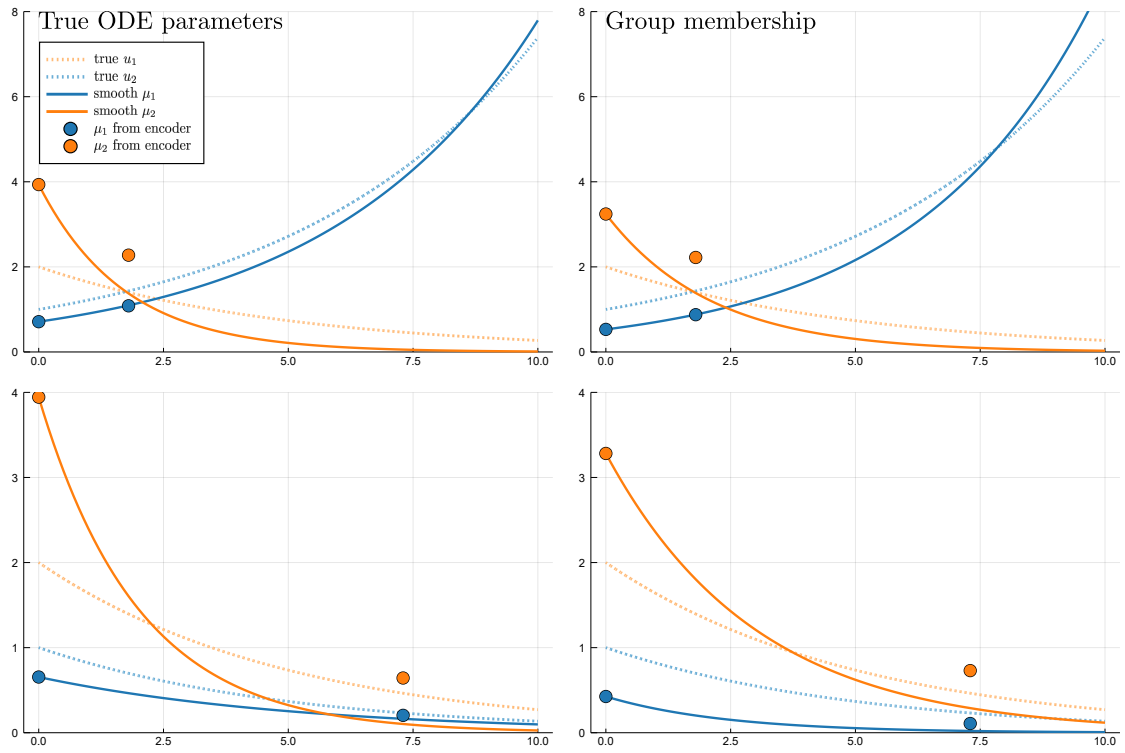


FIGURE 11. Comparison of fitted ODE solutions of one individual from each group for both scenarios of simulated baseline variables.

underestimated especially for later timepoints ($t > 6$) also with the true ODE parameters as baseline information, this becomes more pronounced for training with only the group membership. For both groups and both training scenarios, it seems to be difficult to capture the exponential growth or decay in the latent representation mean before solving the ODE, as can be seen from the blue dots in the panels in the first row and the orange dots in the second row forming a linear rather than an exponential trend. Only the exponential decay of the first ODE component for the first group of individuals is captured in the encoder means (orange dots in the first row).

3.3. Comparison of different noise levels in the simulated data. To investigate how different levels of noise in the time-dependent variables affect the training of the model, I compare the scenario described in Section 3.1 to a scenario

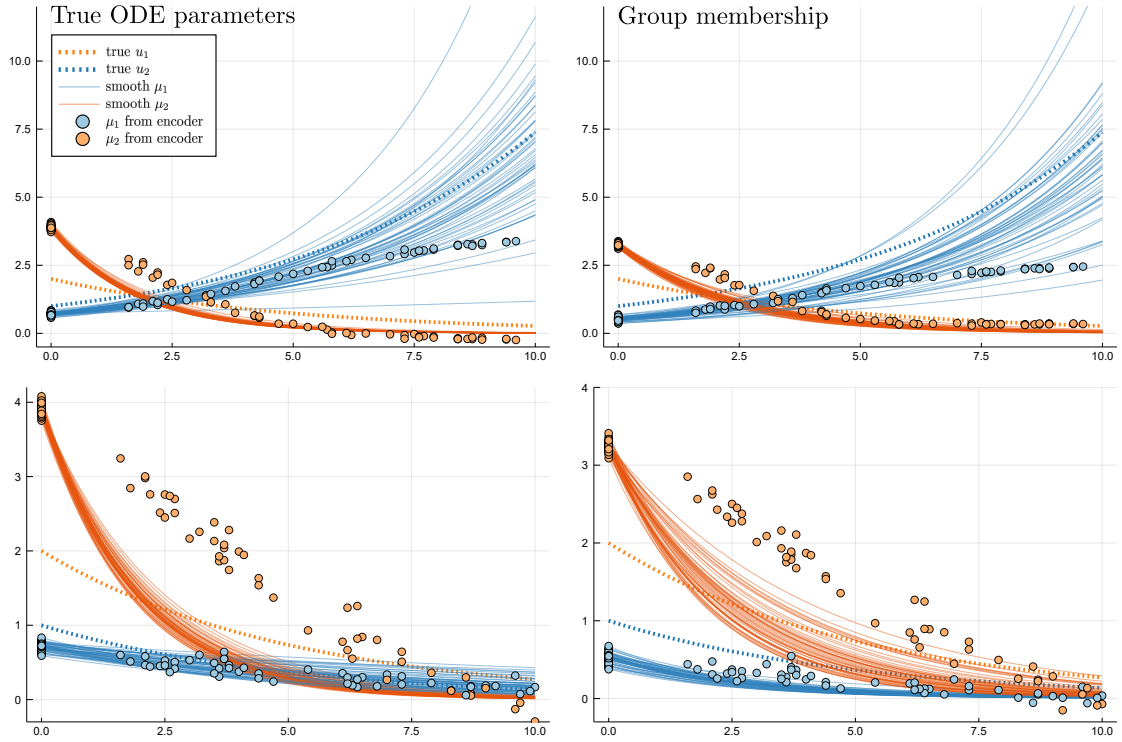


FIGURE 12. Comparison of fitted ODE solutions of all individual from each group for both scenarios of simulated baseline variables.

with the same number of observations from the same underlying ODE solutions, but a higher level of random noise. Specifically, I set $\sigma_{\text{var}} = 0.5$ and $\sigma_{\text{ind}} = 1$. As in Figures 9 and 10 for the setting with a low level of noise, Figure 13 visualises all simulated measurements together with the true ODE solutions while the individual-level structure of 20 simulated individuals is exemplarily illustrated in Figure 14.

For this setting of noisier observations of the true trajectories, I simulate the two versions of baseline variables exactly as in the previous Section 3.2 and train the model for 35 epochs with the ADAM-optimiser and a learning rate of 0.001. Figure 15 shows a comparison of the results with those from the less noisy setting.

First, we can observe that the noisier data setting results in higher variability between fitted ODE solutions (solid lines) as well as the means obtained directly from the encoder (dots) of different individuals. Especially for the second group of individuals (last row), the two components of the mean both from the encoder and

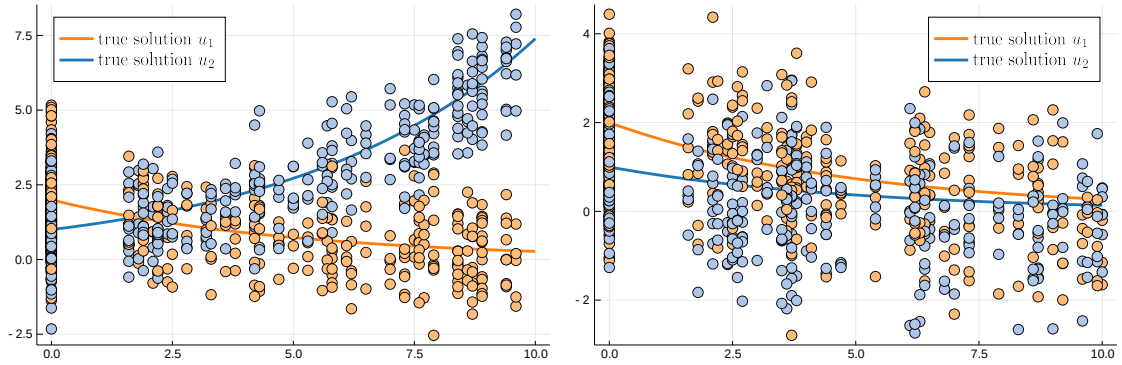


FIGURE 13. Simulated observations of 100 individuals (50 in each group) at t_0 and an individual-specific time point t_1^i.

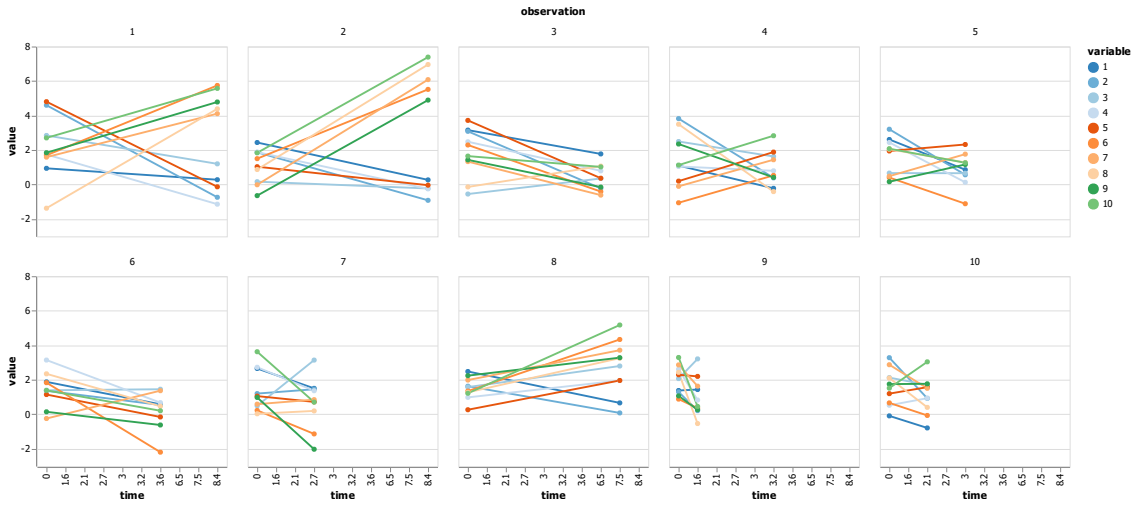


FIGURE 14. Individual-level simulated data structure of 20 individuals.

as ODE solution are not as distinctly separated. This reflects the setting in the input data: Here, also, the higher noise level results in a substantial overlay of the simulated measurements of both components (see Figure 13 and the small figure in the last row of Figure 15). When trained with only the group membership as baseline information, the variability further increases as before, as does the underestimation of the upward trend in the first group of individuals. In the noisier setting, also the downward trend of the first ODE solution component (dotted orange lines) is underestimated in both groups of individuals. This is potentially due to the

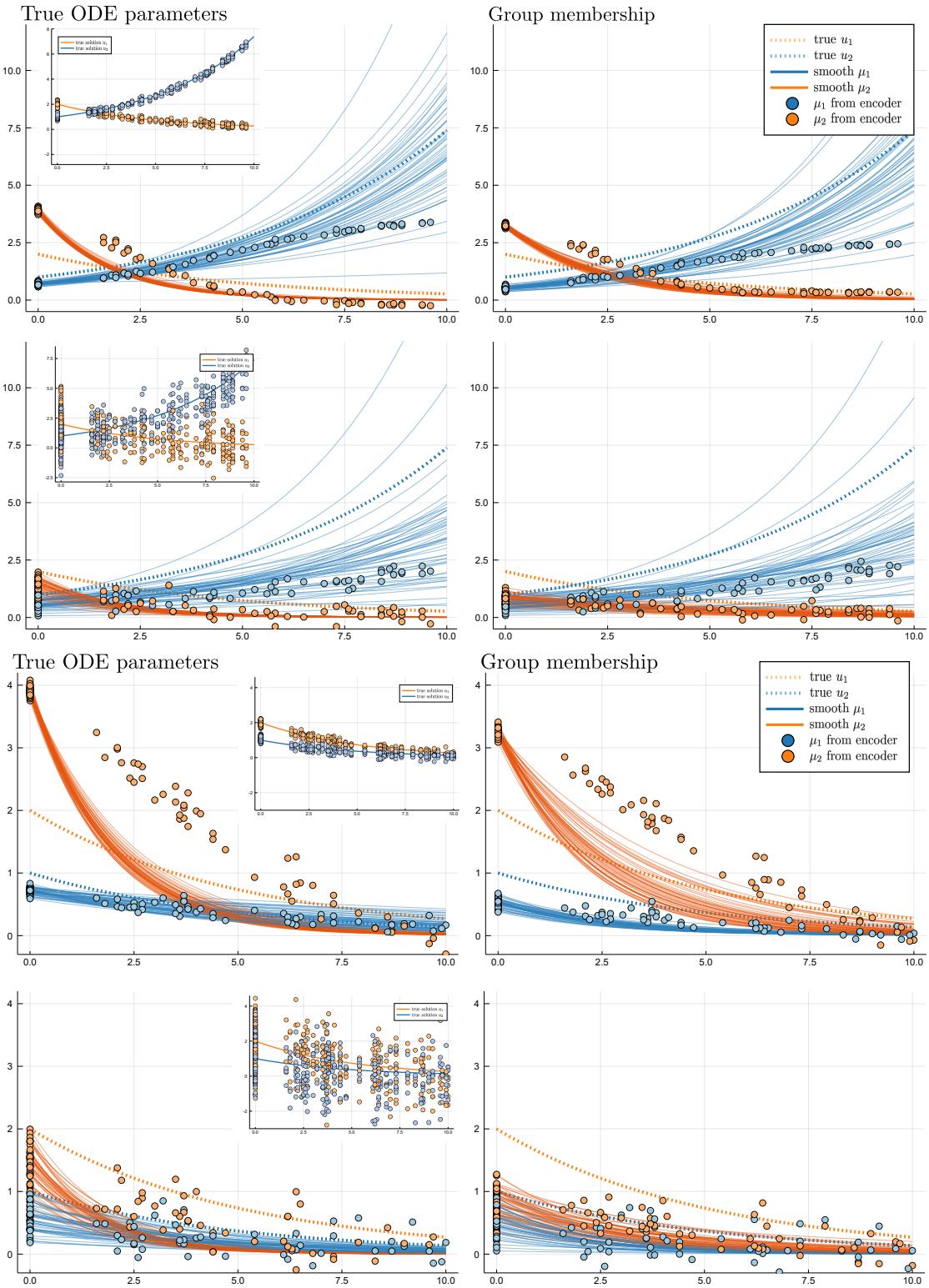


FIGURE 15. Comparison of fitted ODE solutions of all individual from both groups for both scenarios of simulated baseline variables and two different noise levels.

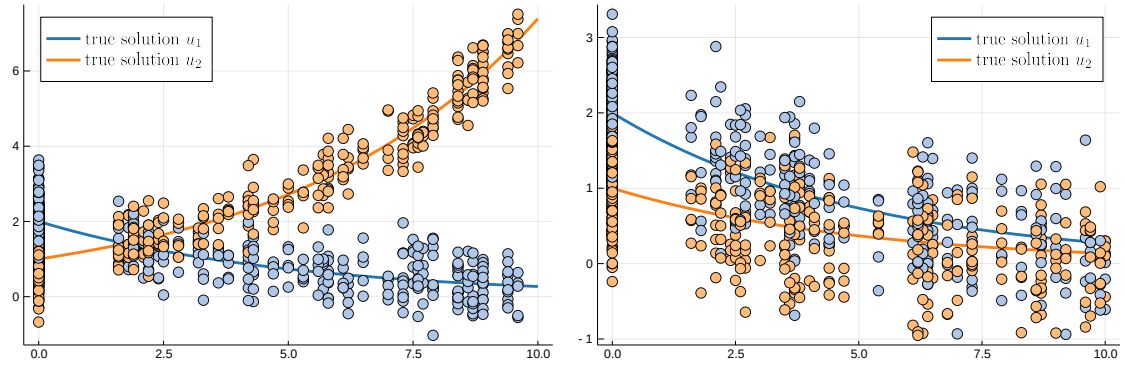


FIGURE 16. Simulated observations of 100 individuals (50 in each group) at t_0 and an individual-specific time point t_1^i.

variational posterior being more heavily influenced by the regularising $\mathcal{N}(0, I)$-prior when it gets a weaker input from the data.

Generally, as the data becomes noisier and the baseline variables become less informative, the latent representations get blurrier, the variability between individuals increases and the two components become less discintly separated, as becomes particularly apparent at the initial value. Nonetheless, despite the substantial level of noise in the data, the model still recovers the general trends and the individual-specific structure of the development patterns.

3.4. Training variability. As the training objective (3.2) of the ODE-VAE model represents a non-convex function and hence does not have a unique minimum, the SGD iterative optimisation procedure only approximates a local optimum that depends on the starting point of the algorithm, i.e., the random initialisation of the model weights and biases. To assess the stability with respect to the initialisation, I compare the results across several training runs.

For this application, I employ a moderate level of noise in the time-dependent variables in between the more extreme scenarios compared in the previous Section 3.3 and set $\sigma_{\text{var}} = 0.1$ and $\sigma_{\text{ind}} = 0.5$. The resulting simulated measurements of the true trajectories are depicted in Figure 16.

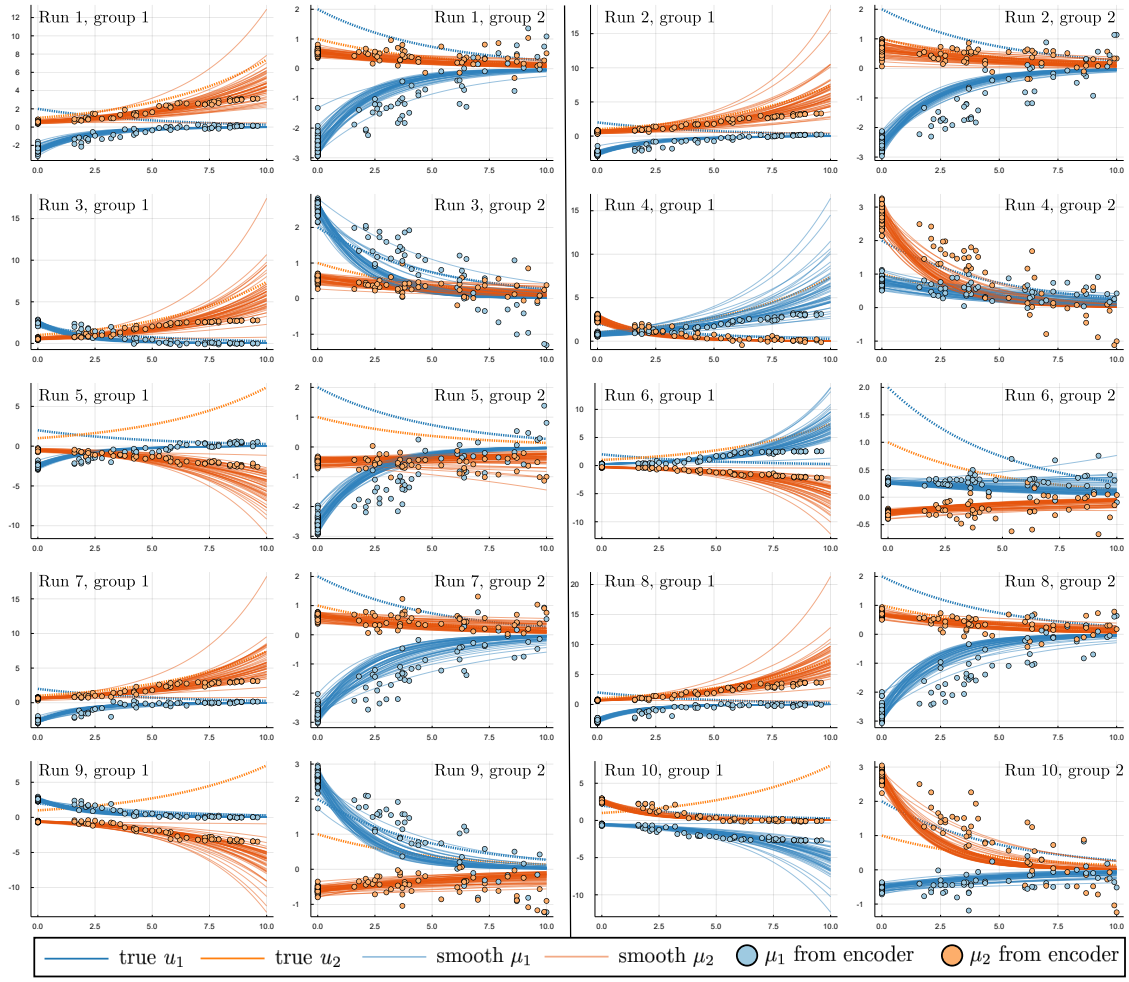


FIGURE 17. Results of 10 training runs for model trained with true ODE parameters as baseline information. Each row depicts two training runs and for each run, the two groups of individuals are shown in separate panels.

I employ the same two scenarios of simulated baseline variables as before and for each scenario conduct 10 training runs of the model with different random weight initialisations. In each run, I train the model for 35 epochs with the ADAM-optimiser and a learning rate of 0.001. The results are shown in Figure 17 (true ODE parameters as baseline information) and Figure 18 (group membership).

Note that the model quite frequently flips one or both components of the initial condition along the x-axis. However, the model then also reverses the trend shown

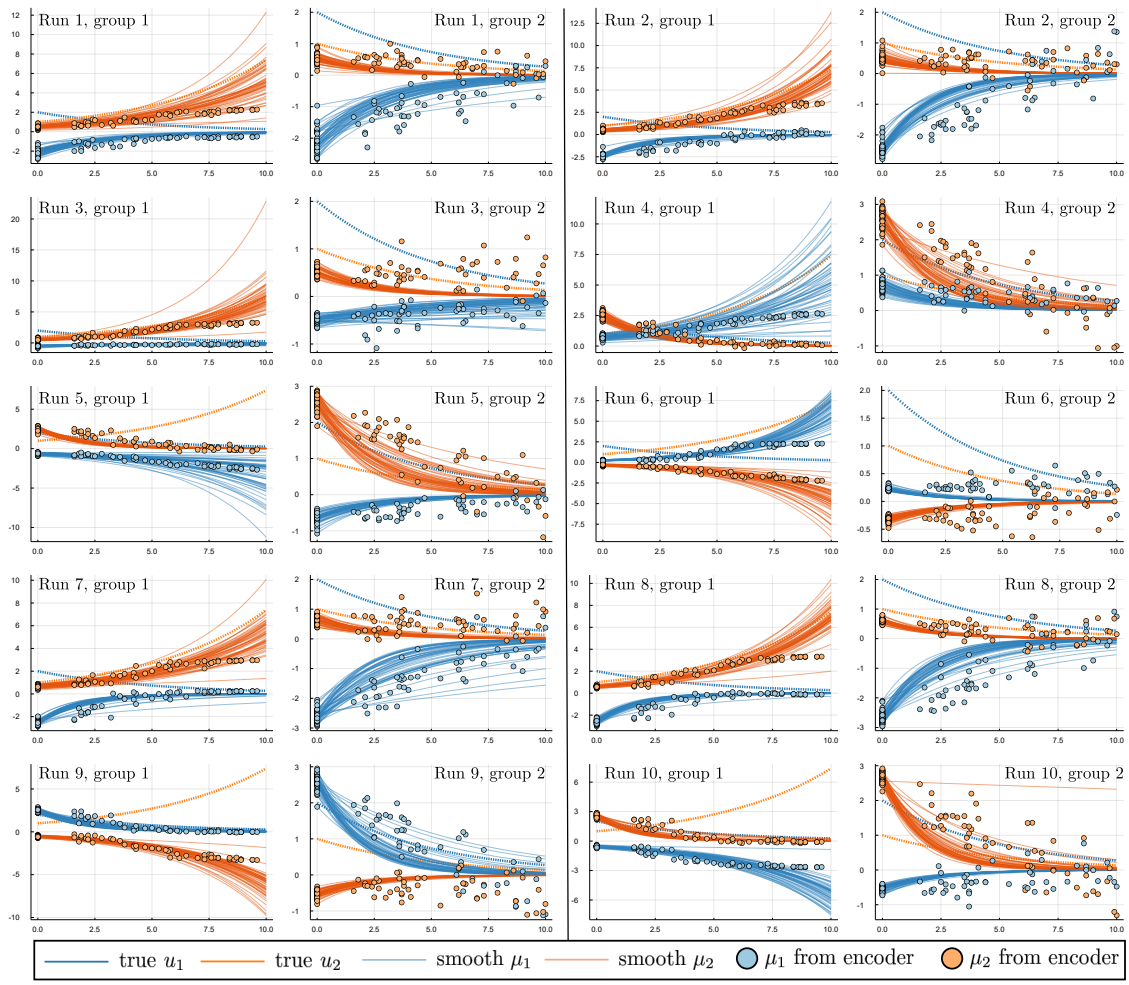


FIGURE 18. Results of 10 training runs for model trained with group membership as baseline information.

by the true trajectories when it flips them to negative values, such that the sign of the corresponding ODE parameter remains the same (a positive parameter in the system defines an upward trend for a positive value of the initial condition and a downward trend for a negative value). The downward trends of the true solution turn into upward trends of the fitted solution when the model flips the initial condition, and vice versa. The model thus still infers the distinct group-specific development patterns and assigns to them the correct parameters of the ODE system, it merely embeds the data in a latent representation that flips the signs in one or both dimensions. The neural network architecture defining the model

encoder and decoder grants the model freedom to structure its latent representation so that it can potentially flip the signs of the data values in the process of embedding them in latent space and flip them back again in the decoder, and does not enforce or even encourage a latent representation that keeps the signs of the original ODE solution. Additionally, the model freedom in structuring the latent space involves the possibility to rescale the input values, such that the entire system is captured in, e.g., a downscaled version. Potentially, this phenomenon may also account for the downscaled representation compared to the original true ODE solutions that we observed in some of the previous applications.

Regularly, the model also swaps the dimensions, as can be seen from the training runs that appear to have an inverted colour coding: E.g., in the training run depicted in the two right panels in the second row, the orange fitted ODE solutions belonging to the first latent mean component μ_1 match the second component of the ground truth-ODE solution u_2 shown in blue and vice versa. This phenomenon can be explained by the fact that the underlying system is symmetric in the two variables and again, the model structure allows it to fit the first five variables of each observation drawn from the first component u_1 of the true solution with the second component μ_2 of its latent representation and vice versa.

In all but one run (the two right panels in the third row in Figures 17 and 18), the model correctly identifies the distinct underlying development patterns in the two groups. This run remained the only one where the general structure was not captured by the model, also when I ran 10 more runs with different initialisations (results not shown). The variability across the dataset differs and is generally higher in the means obtained directly from the encoder. Often, the variability in the second group of individuals appears higher, but this is mostly related to the different scale of the y-axis in the two groups. Overall, the observations made in Section 3.2 apply also to the majority of training runs depicted in Figures 17 and 18, suggesting that while the ODE-VAE model sometimes is subject to substantial variability in the fitted

ODE solutions, it is generally capable of stably inferring distinctly parameterised underlying ODE systems.

4. Non-linear ODE system with two unknown parameters

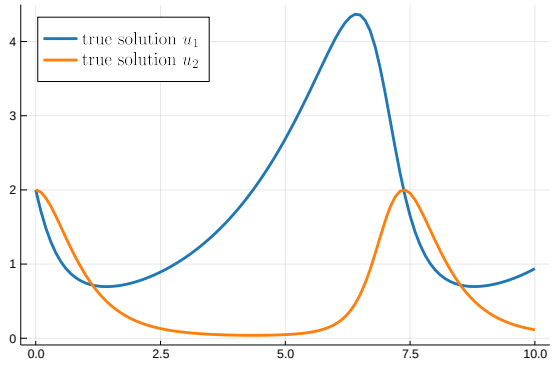
4.1. Structure of input data. In the following section, the model is applied on data from a non-linear ODE system. Specifically, I define two distinctly parameterised Lotka-Volterra systems. This two-dimensional non-linear system is used in ecology to describe the dynamics of the interaction of two populations generally denoted the predator and the prey species ([**24**, p. 209]). The pair of equations is governed by four parameters and has the following form:

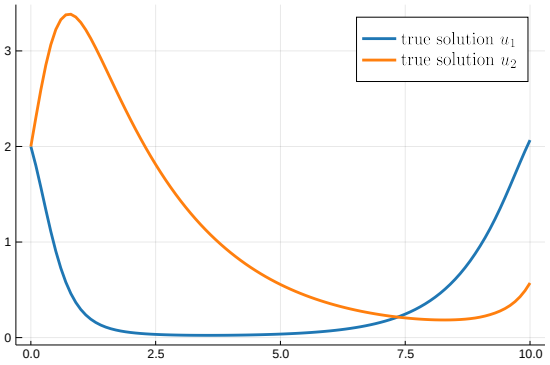
$$\begin{aligned} \frac{d}{dt}u_1(t) &= \alpha u_1(t) - \beta u_1(t)u_2(t), \\ \frac{d}{dt}u_2(t) &= \delta u_1(t)u_2(t) - \gamma u_2(t) \end{aligned} \tag{4.3}$$

In the predator-prey model, the first component u_1 describes the dynamics of the prey species and the second component u_2 describes that of the predator species. The model then assumes that the growth rate of the prey is α, if there are no predators present, and that in the presence of predators, the growth rate is reduced proportional to the number of predators, i.e., β can be understood as predation rate or meeting rate of prey and predator. Similarly, without prey, i.e., without food supply, the model assumes the number of predators to decay at rate $-\gamma$, and to increase, if there is prey, proportional to the amount of prey, such that δ describes the growth rate of the predator in the presence of prey.

Unlike the linear system employed in the previous Section 3, the periodic solutions of the Lotka-Volterra system cannot be calculated analytically in closed form, which means that the solution has to be approximated numerically. For more details on the properties of Lotka-Volterra systems, see [**24**, pp. 209-215].

As for the linear system in Section 3, I simulate data from two different underlying development patterns given as solutions of the two distinctly parameterised Lotka-Volterra ODE systems defined in Equations (4.4) and (4.5) and visualised in Figure 19.





$$\begin{aligned} \frac{d}{dt}u_1(t) &= 0.5u_1(t) - u_1(t)u_2(t), \\ \frac{d}{dt}u_2(t) &= u_1(t)u_2(t) - 2u_1(t); \\ u_1(0) &= 2, \\ u_2(0) &= 2 \end{aligned} \tag{4.4}$$

$$\begin{aligned} \frac{d}{dt}u_1(t) &= u_1(t) - u_1(t)u_2(t), \\ \frac{d}{dt}u_2(t) &= u_1(t)u_2(t) - 0.5u_1(t); \\ u_1(0) &= 2, \\ u_2(0) &= 2 \end{aligned} \tag{4.5}$$

FIGURE 19. True underlying development patterns characterised as solutions of two-dimensional Lotka-Volterra ODE systems.

As described in Algorithm 3, observations are simulated by adding random noise to values from these true trajectories. To account for the higher complexity of the true system, observations of 200 individuals are generated in this application, such that the randomly drawn second measurement time points more densely cover the entire trajectory. Thus, the model gets observations from all different sections of the true underlying patterns (e.g., observations from the peak at the beginning of the time interval in the second ODE component of the second system, and the upward trend at the end of the time interval).

Consequently, I generate observations of 100 individuals in each of the two groups, and again generate observations of 10 variables by adding variable-specific and the individual-specific measurement errors to 5 values from each of the two ODE solution components both at the initial time point $t_0 = 0$ and an individual-specific second time point t_1^i. To optimally observe all parts of the trajectory and cover the entire time interval, I sample the second measurement time points uniformly from

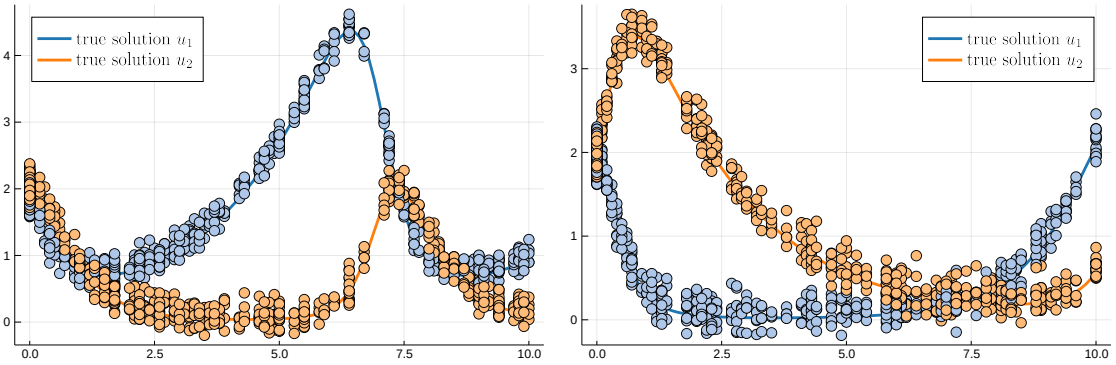


FIGURE 20. Simulated observations of 200 individuals (100 in each group) at t_0 and an individual-specific time point t_1^i.

the interval $[0, 10]$. In all applications of this section, I set $\sigma_{\text{var}} = \sigma_{\text{ind}} = 0.1$. Figure 20 visualises all simulated observations together with the true ODE solutions.

4.2. Individual fitted ODE solutions. On these simulated data, the ODE-VAE model is trained to estimate the two parameters that differ between the two ODE systems, i.e., the parameters β and δ from (4.3) describing the interaction of the two components are set to the true value of 1 and the model learns the two other parameters α and γ. Thus, for each individual i, the initial value problem

$$\begin{aligned}
\frac{d}{dt}\mu_1(t) &= \eta_{i,1}\mu_1(t) - \mu_1(t)\mu_2(t), \\
\frac{d}{dt}\mu_2(t) &= \mu_1(t)\mu_2(t) - \eta_{i,2}\mu_1(t); \\
\mu_i(t_0) &= \mu_i^{t_0}
\end{aligned}$$

is solved in latent space at the individual's second measurement time point t_i^1 with the ODE-net outputs $\eta_{i,1}, \eta_{i,2}$.

In this setting, we first look at fitted ODE solutions for selected individuals and compare the two groups of underlying development patterns. For this application, I simulate baseline variables based on the true ODE parameters as in Algorithm 4 and set $\sigma_{\text{info}} = \sigma_{\text{noise}} = 0.5$ and $q_{\text{info}} = 30$, i.e., simulate 30 variables containing the

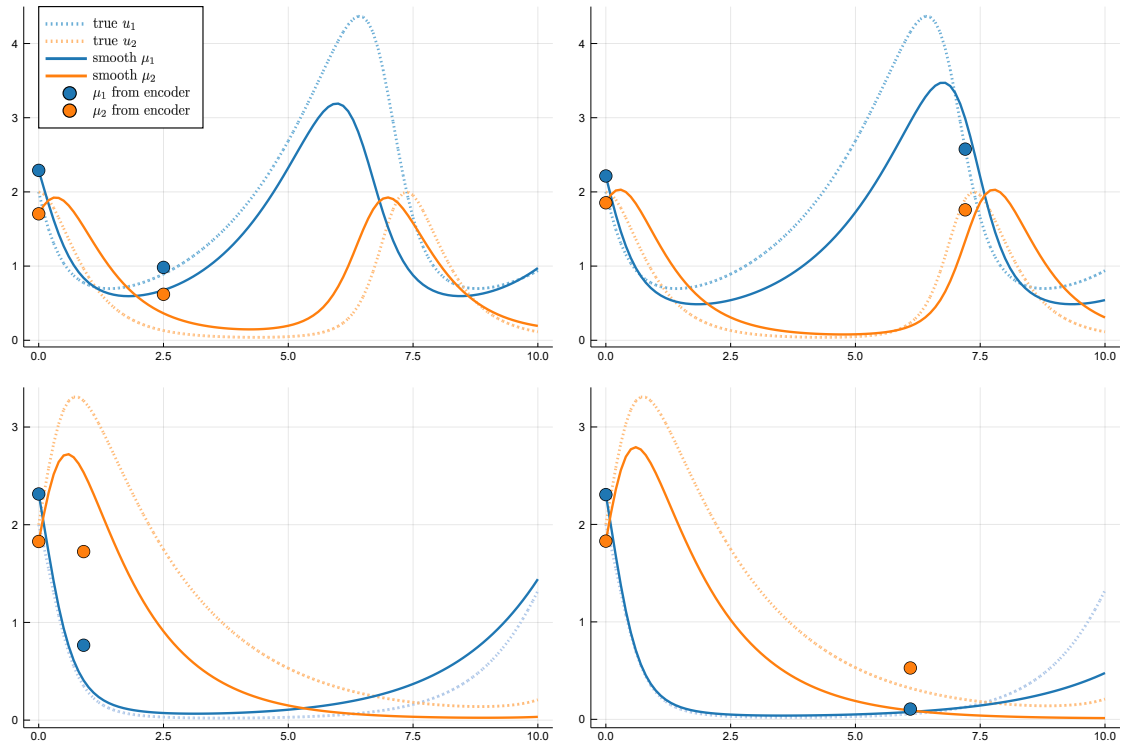


FIGURE 21. Fitted ODE solutions of four individuals with true ODE parameters as baseline information: Two individuals from each group are shown, one with a second measurement time point rather at the beginning of the time interval (left column) and one with a rather late simulated second measurement (right column).

respective baseline information and add 20 noise variables. Then, I train the model for 40 epochs with the ADAM SGD-optimiser and a learning rate of 0.0001.

Figure 21 shows the fitted ODE solutions of four individuals. Generally, the model learns the structure of the underlying patterns in both groups. It slightly overestimates the initial value of the first component of the solution and slightly underestimates it in the second component. Comparable to the linear setting where the model tends to underestimate the upward trend, in this scenario the model also underestimates the high peaks of the first component u_1 for the first underlying pattern and of the second component u_2 in the second underlying pattern. At the peak of the second component in the left panel of the second row, the model has

difficulty matching the mean obtained directly from the encoder (orange dot) to the trajectory of the mean obtained from solving the ODE (solid orange line). As in the linear case, this could be explained by the regularising effect of the $\mathcal{N}(0, I)$-prior on the fitted model posterior and the fact that the true trajectory is rather steep around the peak, such that the data from this region is not dense enough to sufficiently encourage the model to move the posterior to more extreme values away from the prior. As the encoder part of the VAE outputting the posterior means before solving the ODE has no access to the baseline variables containing information about the true parameters governing the ODE system, this phenomenon is less pronounced for the mean obtained as ODE solution which can be influenced by the baseline information.

For the individual from the second group with a late second time point (right panel in second row), the mean from the VAE encoder before solving the ODE in the second component (orange dot) more closely matches the true trajectory (dotted orange line) than the fitted ODE solution (solid orange line). This tendency, albeit less evident, can also be observed for the first component of the mean from the VAE encoder in the both individuals from the first group (blue dots in the panels in the first row) and for the individual with a late second time point also in the second component (orange dot in the right panel of the first row). Probably related to the rather low level of noise in the simulated data, the model thus tends to fit a more data-oriented representation of the mean before solving the ODE. This gets more difficult if there is no sufficient data, in which case the regularising $\mathcal{N}(0, I)$-prior gains influence, as for the second mean component in the left panel in the second row. Generally, also in the non-linear case the basic distinct trends in the developments are captured and reflected by both the ODE solutions and the means from the VAE encoder.

4.3. Training with different numbers of informative baseline variables. The aim of this section is to investigate the effect of the numbers of infomative and noise variables at t_0. To visualise the variability of the results across the entire

dataset and to be able to evaluate the effect of more or less informative baseline variables on the overall model performance, I overlay the fitted ODE solutions of all individuals from each of the two groups and their posterior means from the VAE encoder as in Figures 12 or 15. I conduct four training runs with a different number of informative baseline variables with the time-dependent data described in Section 4.1 (Figure 20). Specifically, I set $q_{\text{info}} = 10, 20, 30, 40$ and for each setting, simulate the baseline information based on the true ODE parameters and the noise variables as described in Algorithm 4 with $\sigma_{\text{info}} = \sigma_{\text{noise}} = 0.5$. I keep all hyperparameters fixed across the four training runs and always start with the same initialisation of the model weights and biases. As in the previous Section 4.2, in each setting, the model is trained for 40 epochs with the ADAM-optimiser and a learning rate of 0.0001 without individual tuning of hyperparameters. Figure 22 shows the resulting fitted ODE solutions.

In the first two rows corresponding to the training results with 10 and 20 baseline variables carrying information about the true ODE parameters, the model has difficulty inferring the distinct group-specific ODE parameters, as the fitted ODE solutions of both groups are more similar than the actual ground-truth trajectories. While the first underlying pattern is essentially recognised by the model, although with substantial variability, the second underlying ground-truth pattern (right column) is not captured well: The peaks that the model generates in the solutions will in fact also appear in the ground truth solution (due to its periodicity), but at a later time point outside the depicted time interval, indicating that the model underestimates the length of the period of the solution and hence does not infer the correct parameters of the system.

Additionally, in the first two rows, the model again underestimates the peaks both of the first solution component for the first group of individuals (left column) and of the second solution component in the second group of individuals (right column). With 20 baseline variables, the model matches the peaks a bit better,

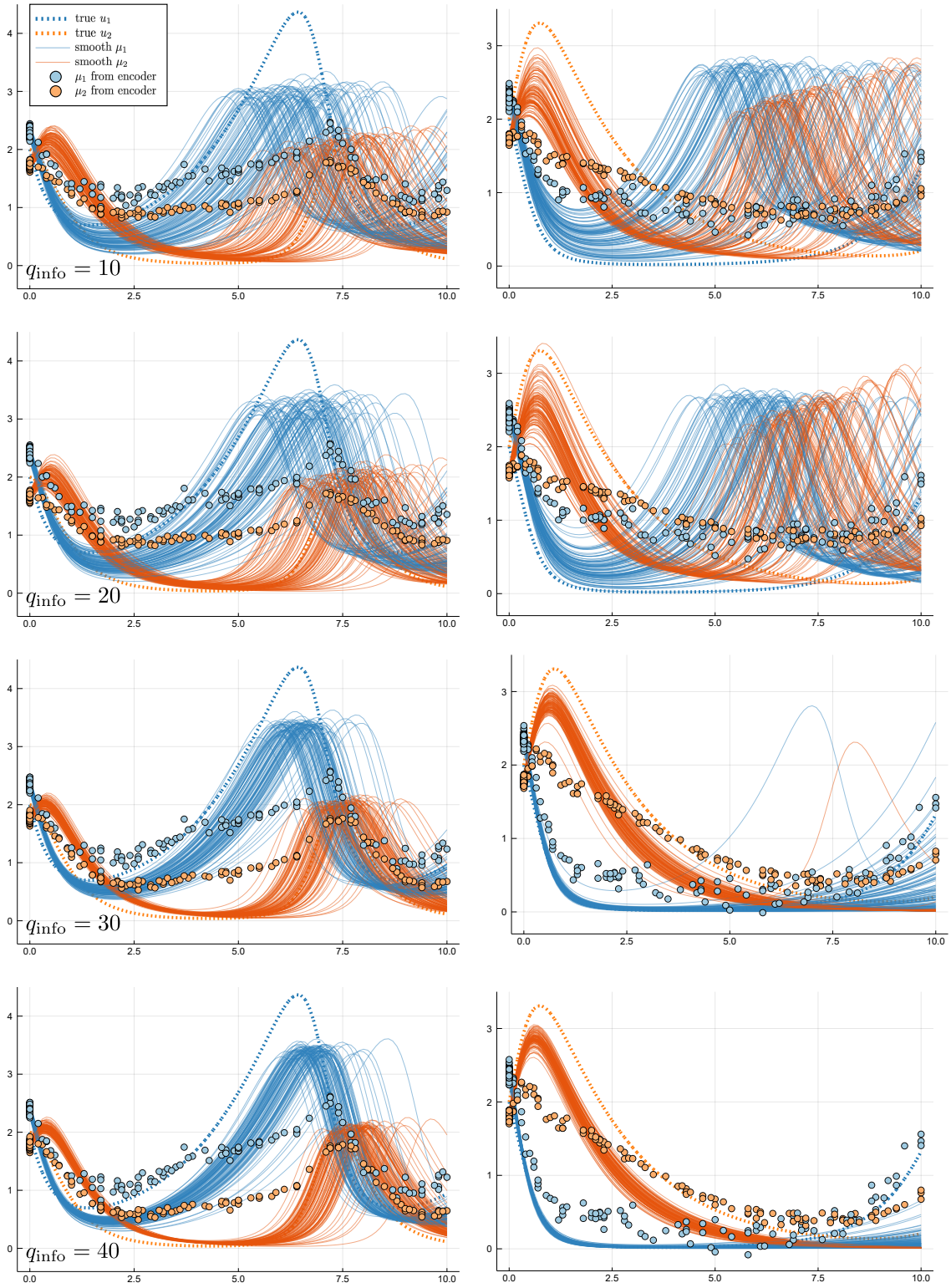


FIGURE 22. Comparison of all individuals' fitted ODE solutions for different baseline variables based on the true ODE parameters. One panel shows the solutions of all individuals from one group of underlying development patterns. One row of panels belongs to one training run with the indicated number of informative baseline variables.

reinforcing our previous observation that a lower level of noise helps the model to fit a posterior closer to the data and further away from the $\mathcal{N}(0, I)$-prior.

Interestingly, in these first two scenarios, the means from the VAE encoder before solving the ODE actually describe the ground truth trajectories better than the fitted ODE solutions and, unlike them, do reflect the different underlying development patterns in the data. With 30 out of 50 baseline variables carrying information, the model also recovers the second underlying pattern in the fitted ODE solutions. The variability between individuals in both groups is strongly reduced in comparison to the setting with only 10 informative baseline variables. There is also less variability in the means from the encoder and they better match the ODE solution, although the improvement in performance with respect to the means from the encoder is not as pronounced as for the fitted ODE solutions. This is hardly surprising as the VAE encoder is not directly influenced by the baseline information and hence not as sensitive to a higher level of noise in them as the fitted ODE solutions that depend directly on the ODE parameters inferred from the baseline variables. Rather, the fact that they do improve at all confirms again that our training objective of bringing these posterior means from before and after solving the ODE together can successfully guide the model towards finding a meaningful, consistent latent representation and ODE parameters that fit the data.

This becomes evident particularly in the second group of individuals (right column): In the first two rows, where the encoder means describe the ground truth trajectories better than the fitted ODE solutions, there is not much change between 10 and 20 informative baseline variables. However, for 30 baseline variables carrying information, the fitted ODE solutions reflect the underlying true pattern, and also the improvement in the fit of the encoder means is larger than before. Both mean representations are subject to a lower level of variability between individuals and in both components, the means from the encoder match both the ground truth and the fitted ODE solutions much better. This implies that more baseline information

helps the VAE to find a better fitting ODE system, and a better fitting ODE system in latent space in turn helps the VAE to find a good latent representation that matches such a smooth development according to the ODE system already as direct output of the encoder, even before solving the ODE. In the last row corresponding to the results with 40 informative baseline variables, compared to the results for 30 informative baseline variables, mainly the variability across the dataset is further reduced. Even in this last setting, however, the model does not fit the peak of the first solution component in the first group of individual (solid blue lines in left panel) perfectly, and the means from the encoder (blue dots) underestimate it more significantly. In the second component, the means from the encoder do not match the ground-truth trajectory exactly, but still reflect the general non-linear structure.

When performing the same experiments in the second simulation scenario of baseline variables, i.e., with only the information about the group membership (see Algorithm 5), I obtained much better fitting ODE solutions for low numbers of informative baseline variables (see Figure 23). Even with only 10 informative baseline variables and despite the generally more difficult scenario of simulated baseline information, the model still clearly distinguishes the two development patterns and fits them with as little variability as in the setting with the true ODE parameters as baseline information and 30 informative baseline variables. A possible explanation for this phenomenon might be that it can be easier for the model to distinguish between two groups if only the group membership is encoded in the baseline information: With the true ODE parameters as information, it has to take an additional step, i.e., it has to both infer the true parameters from the noisy information and then infer from that there are two groups, while with only the group membership, it is easier for the model to infer the general presence of two distinct groups from the data. As in the setting shown in Figure 23, the variability between individuals further decreases for an increased number of informative variables and the model generally has difficulty matching the means from the encoder and the means after solving the ODE in the regions of high peaks in the data.

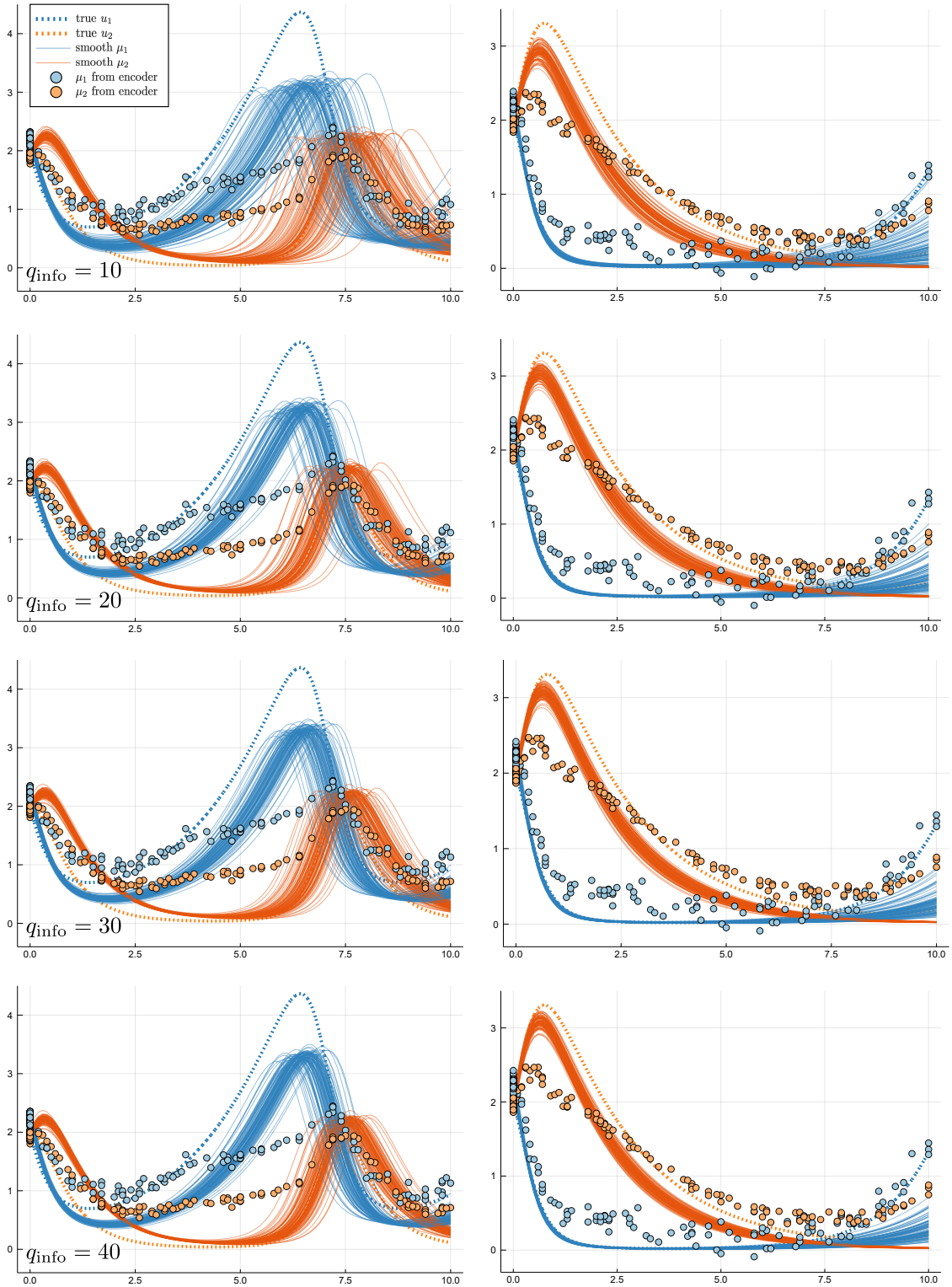
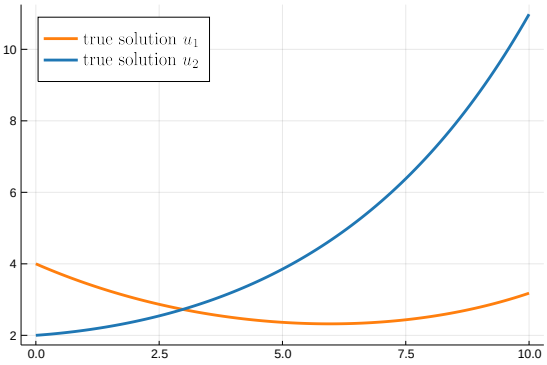
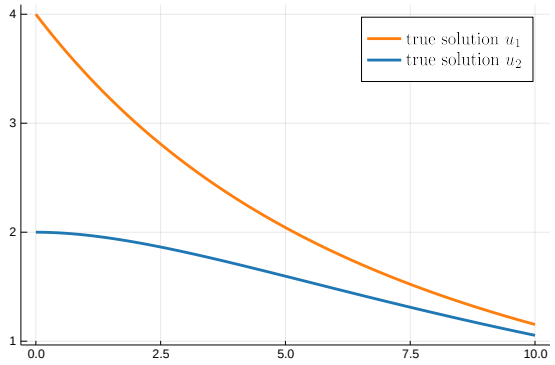


FIGURE 23. Comparison of all individuals' fitted ODE solutions for different baseline variables based on the group membership. One panel shows the solutions of all individuals from one group of underlying development patterns. One row of panels belongs to one training run with the indicated number of informative baseline variables.





$$\frac{d}{dt}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}(t) = \begin{pmatrix} -0.2 & 0.1 \\ -0.1 & 0.25 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}(t); \quad \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}(0) = \begin{pmatrix} 4 \\ 2 \end{pmatrix} \tag{4.6}$$

$$\frac{d}{dt}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}(t) = \begin{pmatrix} -0.2 & 0.1 \\ 0.1 & -0.2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}(t); \quad \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}(0) = \begin{pmatrix} 4 \\ 2 \end{pmatrix} \tag{4.7}$$

FIGURE 24. True underlying development patterns characterised as solutions of two-dimensional linear ODE systems.

5. Linear ODE system with four unknown parameters

While the applications presented so far have covered settings where the number of parameters was equal to the number of simulated measurement time points, I now investigate the capability of the ODE-VAE model to fit more parameters in order to model more complex dynamical systems. In the following section, I therefore apply the extension of our training procedure described in Section 4 and infer batches of individuals with similar trajectories to enrich individual's observations with proxy information at more time points from other individuals in the batch.

5.1. Structure of input data. The model is trained on data from a linear ODE system as in Section 3 and all four ODE parameters are estimated as outputs of the ODE-net. Again, two distinct underlying development patterns are defined as solutions of the ODE systems in Equations (4.6) and (4.7) and visualised in Figure 24, similar to those in Section 3.1.

As in Section 3.1, I generate observations of 10 variables for 100 individuals, 50 from each of the two groups, by adding variable-specific and the individual-specific

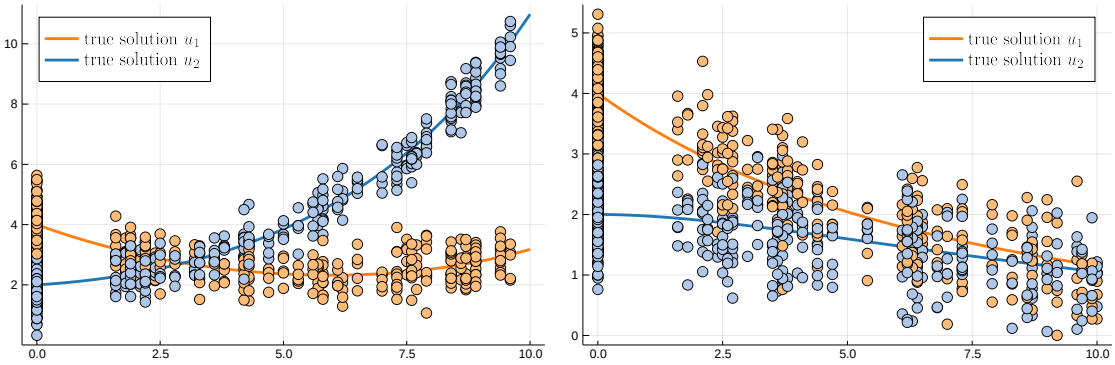


FIGURE 25. Simulated observations of 100 individuals (50 in each group) at t_0 and an individual-specific time point t_1^i.

measurement errors to 5 values from each of the two ODE solution components both at the initial time point $t_0 = 0$ and the second time point t_1^i drawn uniformly from the interval $[1.5, 10]$ as in Algorithm 3. For this setting with four unknown parameters, I set $\sigma_{\text{var}} = 0.1$ and $\sigma_{\text{ind}} = 0.5$. Figure 25 visualises all simulated observations together with the true ODE solutions.

5.2. Comparison of different simulation scenarios of baseline variables. On these data, the ODE-VAE model is trained as described in Chapter 3, Section 4 and Algorithm 2. Specifically, all four parameters of each of the two ODE systems are estimated and in latent space, for each individual i the initial value problem

$$\frac{d}{dt}\mu_i(t) = \begin{pmatrix} \eta_{i,1} & \eta_{i,2} \\ \eta_{i,3} & \eta_{i,4} \end{pmatrix} \begin{pmatrix} \mu_{i,1} \\ \mu_{i,2} \end{pmatrix}(t);$$
$$\mu_i(t_0) = \mu_i^{t_1}$$

is solved at the individual's second time point t_i^1 with the ODE-net outputs $\eta_{i,1}, \ldots, \eta_{i,4}$.

As in Section 3.2, the different simulations scenarios of baseline variables described in Algorithms 4 and 5 are compared. For the first scenario with the true ODE parameters, I set $\sigma_{\text{info}} = \sigma_{\text{noise}} = 0.1$ and for the second scenario with the group

membership, I set $\sigma_{\text{info}} = \sigma_{\text{noise}} = 0.5$. In each scenario, I simulate 20 variables containing the respective baseline information, add 30 noise variables and train the model for 15 epochs with the ADAM SGD-optimiser and a learning rate of 0.001.

Empirically, I found that for my setting with two distinct underlying patterns, a small batchsize ($b \leq 5$) is insufficient to capture the complete dynamics because it provides too little time points and proxy observations. On the other hand, a large batchsize with respect to the dataset size ($b \gg \frac{n}{10}$) introduces training instability due to too many individuals with different development pattern being grouped together. Between those extremes, I found the results to be robust against different choices of batchsizes and present in the following only results for a batchsize of $b = 10$.

I also found that the results are stable with respect to different choices of kernel functions, including using equal weights for all individuals in the batch. In the more difficult scenario of using only the group membership as baseline information, however, the weighting approach does improve the model performance. Thus, I chose to use the kernel approach for all applications with the tricube function as kernel and a bandwidth of 1 without tuning this hyperparameter.

Figure 26 compares the ODE solutions of individual minibatches for the two simulation scenarios of baseline variables.

Generally, we can see that the second measurement time points of all individuals in the batch do cover the entire time interval, and that all individuals in one batch share a common development trend in both components of the ODE solution. For training with the group membership as baseline information, we observe the phenomenon mentioned in Section 3.4 of the inverted signs of the initial condition and the model flipping the first component of the fitted solution at the x-axis. In both compared settings, the model again inverts the dimensions, i.e., fits the first component of the true ODE solution with the second component of its latent representation mean and vice versa.

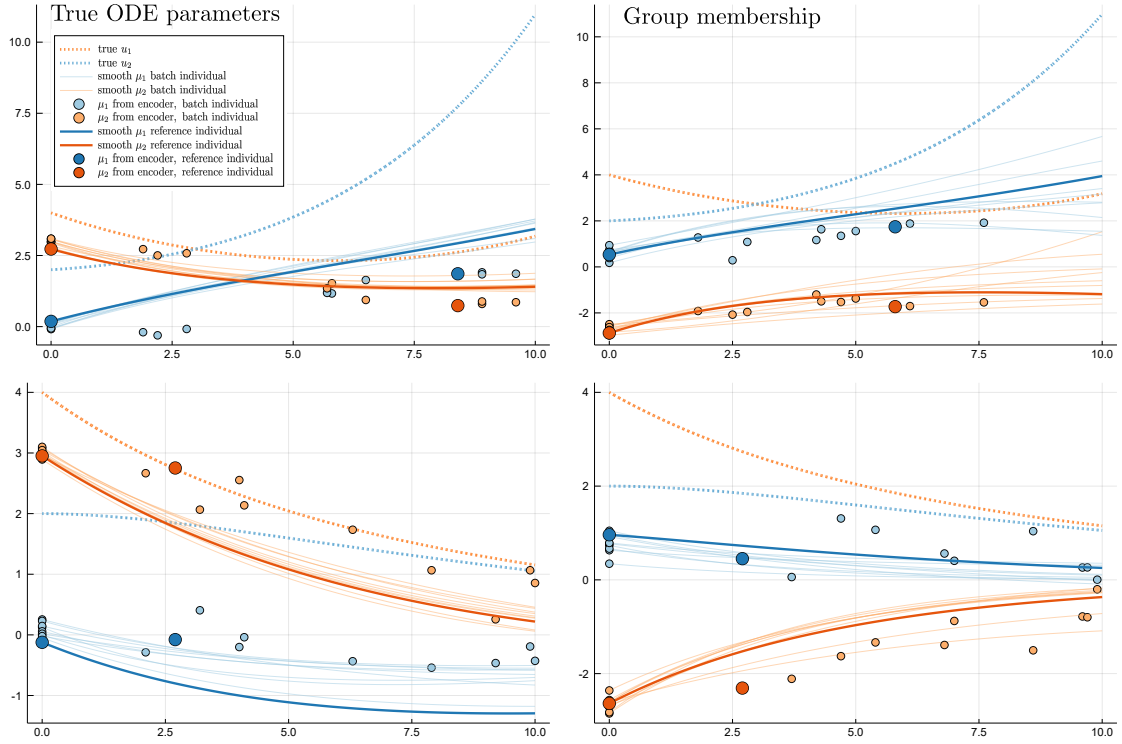


FIGURE 26. Comparison of fitted ODE solutions of selected mini-batches for both scenarios of simulated baseline variables. One panel depicts the fitted ODE solutions of all individuals from one batch. Thick, dark lines and dots represent the latent representation means of the reference individual around which the batch was grouped, while lighter, thin lines and dots belong to the other individuals in the batch.

In the scenario with the true ODE parameters, the fitted ODE solutions show the correct trends of the ground-truth solutions, but, especially in the first component, underestimate the true solution. This is even more pronounced in the first component of the means obtained from the VAE encoder before solving the ODE. However, these means also reflect the true developments with some variability (the apparently higher variability in the second row is due to the different scaling of the axis). When trained with only the group membership as baseline information, the

fitted ODE solutions vary more strongly between individuals from the batch, especially in the upper panel. Again, in particular the first component of the fitted ODE solution underestimates the true trend.

Figure 27 compares the results of all individuals from each of the two groups in the two simulation scenarios of baseline variables by overlaying all individual ODE solutions and encoder means from one group of underlying development patterns in one panel. In general, the variability between individual's solutions from the same group is higher with only the group membership as baseline information. In both scenarios, particularly the upward trend of the second ground-truth ODE solution component is underestimated, while the fitted ODE solutions show that the integrated ODE solving step can partially improve this. The variability of the means from the encoder tends to be higher later in the time interval, reflecting the greater uncertainty in the fitted ODE solutions at those later time points. While for the first scenario with the true ODE parameters, the general trends of the trajectories are still distinguishable, they become more and more masked by individual variability in the second scenario. Our observations are thus generally similar to those for the linear system with only two unknown parameters, although there is more variability and in general a weaker signal present in the application with four unknown parameters. On the other hand, the underlying problem of inferring more parameters than observed time points is much more difficult.

Additionally, I investigated the use of randomly assigned minibatches without inferring individuals' similarity and training without the batches as in the previous sections. Here, I found that without any batches, the model is not able to identify two distinct development patterns at all, and has significantly more difficulty inferring the correct trends of the trajectories with random minibatches.

Overall, although the results are subject to substantial variation for different initialisations of model weights and biases, the proposed method of training on batches can still enable the achievement of similar results to the two-parameter case in this substantially more difficult setting.

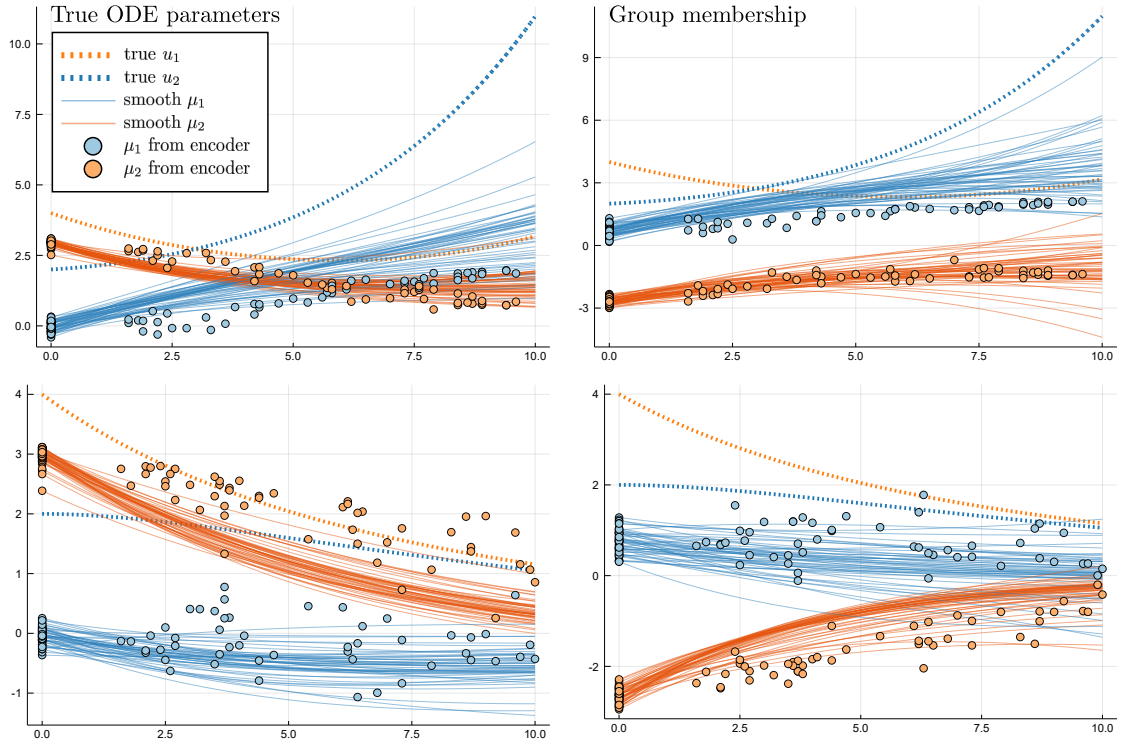


FIGURE 27. Comparison of fitted ODE solutions of all individuals from both groups for both scenarios of simulated baseline variables.

CHAPTER 5

Discussion

In this thesis, I have adapted the architecture of the variational autoencoder, a generative deep learning framework, to develop a model that can identify the central factors of variation in temporal development patterns and recover them in a non-linear and unsupervised fashion. I have constrained the latent space to model smooth trajectories by integrating an ODE system and augmented the model with another neural network used to infer individual-specific ODE parameters from additional baseline variables. Furthermore, I have extended the model to train on batches of similar individuals, using combined measurements of other individuals from the batch as proxy information on the common trajectory at multiple time points.

Inspired by a data scenario from a NAKO sub study, I have devised a simulation design where data are generated from distinct underlying temporal developments with random measurement noise at a common baseline time point and an irregularly sampled individual second time point. Then, the model has been applied and evaluated on these simulated data from both a linear and a non-linear ODE system with two unknown parameters and trained on batches of similar individuals on data from a linear system with four unknown parameters. I have compared different simulation scenarios of baseline information and different numbers of informative baseline variables and investigated the model performance for different levels of simulated measurement noise and across several training runs.

Overall, it could be shown that the proposed ODE-VAE model can recover the distinct underlying development patterns and infer individual-specific ODE parameters in various simulated data settings and accurately identify groups of individuals with similar trajectories. In conclusion, it thus provides an individual-level understanding of the temporal dynamics underlying individuals' developments and, rather

than estimating average effects, has the potential to plan interventions based on the knowledge of full individual-specific dynamical systems.

With respect to related research, the main novelties of my method are the employment of batches of similar individuals to get proxy information on additional measurement time points and training the model in an iterative expectation-maximisation-like procedure to be able to afford more ODE parameters and thus model more complex dynamics. Also, the idea to infer individual-specific ODE parameter sets from additional baseline information has been introduced. What further differentiates my approach from, e.g., the latent time-series models in [**3**] and [**4**] is solving the ODEs at the level of the latent representation and implicitly conditioning the smooth mean obtained from solving the ODE on the data from the second measurement time point by adapting the training objective to explicitly encourage consistency of the latent representation means before and after solving the ODE.

The main limitations of the method are that some knowledge on the underlying dynamics is required to define the general structure and dimensionality of the ODE system and the general sensitivity of the model to noisy information in the time-dependent and baseline variables as well as to how the data is scaled with respect to the $\mathcal{N}(0, 1)$-prior. Additionally, even for only moderately complex scenarios, the model performance is subject to substantial variability for different random initialisations of the model parameters. In its current form, the method is restricted to a rather specific setting.

For future research, it would thus be promising to leverage the method to model more complex or higher-dimensional dynamical systems, possibly also with more measurement time points, and to more thoroughly investigate its performance by defining rigorous evaluation criteria and averaging over many training runs. Finally, it would be crucial to test and evaluate the model performance on real-world datasets such as the NAKO study, which could then guide the modelling of more complex scenarios based on results from real data once they become available.

References

[1] Leo Breiman. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199–231, 08 2001.

[2] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, and Ali Ramadhan. Universal differential equations for scientific machine learning, 2020. arXiv:2001.04385.

[3] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, 2018.

[4] Yulia Rubanova, Tian Qi Chen, and David Duvenaud. Latent ODEs for irregularly-sampled time series, 2019. arXiv:1907.03907.

[5] Çağatay Yıldız, Markus Heinonen, and Harri Lähdesmäki. ODE^{2}VAE: Deep generative second order odes with bayesian neural networks, 2019. arXiv:1905.10994.

[6] Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series, 2019. arXiv:1905.12374.

[7] Vincent Fortuin, Dmitry Baranchuk, Gunnar Rätsch, and Stephan Mandt. GP-VAE: Deep probabilistic time series imputation, 2019. arXiv:1907.04155.

[8] Karol Gregor, George Papamakarios, Frederic Besse, Lars Buesing, and Theophane Weber. Temporal difference variational auto-encoder, 2018. arXiv:1806.03107.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[10] George Cybenko. Approximations by superpositions of sigmoidal functions. *Math. Control Signals Systems*, 2(4):303–314, 1989.

[11] Moshe Leshno, Vladimir Y. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993.

[12] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*. 2017.

[13] David Rumelhart, Geoffrey Hinton, and Ronald Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[14] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[15] Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, 2009.

[16] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. arXiv:1406.2661.

[17] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations (ICLR), Conference Track Proceedings*, 2014.

[18] Claudia Czado and Thorsten Schmidt. *Mathematische Statistik*. Springer, Berlin / Heidelberg, 2011.

[19] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[20] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[21] Christian Robert and George Casella. *Monte Carlo Statistical Methods*. Springer, New York, 2005.

[22] Radford Neal and Geoffrey Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In Jordan M.I., editor, *Learning in Graphical Models. NATO ASI Series*. 1998.

[23] Lawrence Perko. *Differential Equations and Dynamical Systems*. Springer, New York, 2001.

[24] Gerald Teschl. *Ordinary Differential Equations and Dynamical Systems*. American Mathematical Society, Providence, Rhode Island, 2012.

[25] Chris Rackauckas, Mike Innes, Yingbo Ma, Jesse Bettencourt, Lyndon White, and Vaibhav Dixit. Diffeqflux.jl - A julia library for neural differential equations, 2019. arXiv:1902.02376.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations (ICLR), Conference Track Proceedings*, 2015.

Zusammenfassung in deutscher Sprache

Bei der Betrachtung biomedizinischer Zeitreihendaten ist das Verständnis der zugrunde liegenden dynamischen Systeme auf individueller Ebene der erste und wesentliche Schritt für eine personalisierte Behandlung. Solche Daten weisen jedoch oftmals ein spärliches und unregelmäßiges Zeitraster von Messungen sowie individuell verschiedene Entwicklungen auf, was die Modellierung erschwert.

In dieser Arbeit wird basierend auf einem generativen Deep Learning-Verfahren ein Modell entwickelt, das aus solchen spärlichen und unregelmäßig beobachteten Zeitreihendaten einen niedrigdimensionalen latenten Raum lernt, in dem die individuellen Entwicklungsmuster als Lösungen gewöhnlicher Differentialgleichungen repräsentiert werden. Meine Methodik ist dabei von aktuellen Forschungsarbeiten zur Kombination von Deep Learning-Verfahren mit einer expliziten mechanistischen Modellierung durch Differentialgleichungen inspiriert und durch ein Szenario aus der Nationalen Kohorte (NAKO), einer groß angelegten epidemiologischen Kohortenstudie, motiviert.

Basierend auf einer NAKO-Substudie simuliere ich Daten, die sich durch eine umfangreiche Charakterisierung jedes Individuums mit Messungen vieler Variablen zu einem Baseline-Zeitpunkt auszeichnen, wobei eine kleinere Teilmenge dieser Variablen zu einem zweiten, für jedes Individuum unterschiedlichen, Zeitpunkt erneut gemessen wird. Insgesamt liegen somit individuelle dynamische Prozesse vor, die nur selten (je zwei Zeitpunkte pro Individuum) und unregelmäßig mit einer gewissen Messunsicherheit beobachtet werden. Ziel meiner Arbeit ist es, in einem solchen Szenario ein Modell zu entwickeln, das trotz der Messfehler sowie des groben und unregelmäßigen Zeitrasters die zugrunde liegenden individuellen Entwicklungsmuster extrahieren kann.

Hierbei verwende ich einen Variational Autoencoder (VAE), ein generatives Deep-Learning-Modell, um anhand der beobachteten Entwicklungsmuster in einem nicht-linearen, unüberwachten Lernverfahren einen niedrigdimensionalen latenten Raum abzuleiten, der die zentrale, den Daten zugrunde liegende Dynamik repräsentiert. Um glatte Trajektorien zu modellieren, wird der latente Raum auf einen Raum differenzierbarer Funktionen eingeschränkt, die als Lösungen eines vorab definierten ODE-Systems vorliegen. Basierend auf der Annahme, dass individuelle Unterschiede in den Entwicklungsmustern auf Unterschiede in den nur bei zum Baseline-Zeitpunkt gemessenen Variablen zurückgeführt werden können, verwende ich diese Baseline-Variablen, um mithilfe eines weiteren neuronalen Netzes individuelle ODE-Parameter zu bestimmen.

Um komplexere zugrunde liegende dynamische Systeme modellieren zu können, erweitere ich das Modell und bestimme für jedes Individuum eine Gruppe von Individuen mit ähnlichen Entwicklungsmustern. Die Kombination aller zweiten Messungen in der Gruppe dient dann als Proxy-Information für das betrachtete Individuum über die gemeinsame dynamische Entwicklung zu mehreren Zeitpunkten. Durch das Trainieren des Modells auf diesen Gruppen wird so die Unregelmäßigkeit der zweiten Messzeitpunkte ausgenutzt und die Information jedes einzelnen Individuums wird um zusätzliche, stellvertretende Informationen von anderen Individuen erweitert.

In den Anwendungen auf die oben beschriebenen simulierte Daten konnte ich zeigen, dass das entwickelte Modell in der Lage ist, individuelle Entwicklungsmuster basierend auf linearen und nicht-linearen zweidimensionalen ODE-Systemen mit zwei oder vier unbekannten Parametern zu rekonstruieren und für ein gegebenes Individuum eine Gruppe von Individuen mit ähnlichen Entwicklungen zu ermitteln.

Zusammenfassend bietet die vorgestellte Methode somit auf individueller Ebene Einsicht in die dynamischen Systeme, die den Entwicklungen verschiedener Individuen im Zeitverlauf zugrunde liegen, und kann die Planung personalisierter Interventionen anhand der Kenntnis solcher vollständigen individualspezifischen dynamischen Systeme ermöglichen.