Heinonen, Markus; Yildiz, Cagatay; Mannerström, Henrik; Intosalmi, Jukka; Lähdesmäki, Harri

**Learning unknown ODE models with Gaussian processes**

*Published in:*
Proceedings of the 35th International Conference on Machine Learning, ICML 2018

*Published:* 01/01/2018

*Document Version*
Publisher's PDF, also known as Version of record

*Please cite the original version:*
Learning unknown ODE models with Gaussian processes

Markus Heinonen 1 2 Çagatay Yıldız 1 Henrik Mannerström 1 Jukka Intosalmi 1 Harri Lähdesmäki 1

Abstract

In conventional ODE modelling coefficients of an equation driving the system state forward in time are estimated. However, for many complex systems it is practically impossible to determine the equations or interactions governing the underlying dynamics. In these settings, parametric ODE model cannot be formulated. Here, we overcome this issue by introducing a novel paradigm of non-parametric ODE modelling that can learn the underlying dynamics of arbitrary continuous-time systems without prior knowledge. We propose to learn non-linear, unknown differential functions from state observations using Gaussian process vector fields within the exact ODE formalism. We demonstrate the model’s capabilities to infer dynamics from sparse data and to simulate the system forward into future.

1. Introduction

Dynamical systems modelling is a cornerstone of experimental sciences. In biology, as well as in physics and chemistry, modelers attempt to capture the dynamical behavior of a given system or a phenomenon in order to improve its understanding and make predictions about its future state. Systems of coupled ordinary differential equations (ODEs) are undoubtedly the most widely used models in science. Even simple ODE functions can describe complex dynamical behaviours (Hirsch et al., 2004). Typically, the dynamics are firmly grounded in physics with only a few parameters to be estimated from data. However, equally ubiquitous are the cases where the governing dynamics are partially or completely unknown.

We consider the dynamics of a system governed by multivariate ordinary differential functions:

\[ \dot{x}(t) = \frac{dx(t)}{dt} = f(x(t)) \]  

where \( x(t) \in \mathcal{X} = \mathbb{R}^D \) is the state vector of a \( D \)-dimensional dynamical system at time \( t \), and the \( \dot{x}(t) \in \mathcal{X} = \mathbb{R}^D \) is the first order time derivative of \( x(t) \) that drives the state \( x(t) \) forward, and where \( f : \mathbb{R}^D \rightarrow \mathbb{R}^D \) is the vector-valued derivative function. The ODE solution is determined by

\[ x(t) = x_0 + \int_0^t f(x(\tau))d\tau, \]

where we integrate the system state from an initial state \( x(0) = x_0 \) for time \( t \) forward. We assume that \( f(\cdot) \) is completely unknown and we only observe one or several multivariate time series \( Y = (y_1, \ldots, y_N)^T \in \mathbb{R}^{N \times D} \) obtained from an additive noisy observation model at observation time points \( T = (t_1, \ldots, t_N) \in \mathbb{R}^N \),

\[ y(t) = x(t) + \epsilon_t, \]

where \( \epsilon_t \sim \mathcal{N}(0, \Omega) \) follows a stationary zero-mean multivariate Gaussian distribution with diagonal noise variances \( \Omega = \text{diag}(\omega_1^2, \ldots, \omega_D^2) \). The observation time points do not need to be equally spaced. Our task is to learn the differential function \( f(\cdot) \) given observations \( Y \), with no prior knowledge of the ODE system.

There is a vast literature on conventional ODEs (Butcher, 2016) where a parametric form for function \( f(x; \theta, t) \) is assumed to be known, and its parameters \( \theta \) are subsequently optimised with least squares or Bayesian approach, where the expensive forward solution \( x_\theta(t_i) = \int_0^{t_i} f(x(\tau); \theta, t)d\tau \) is required to evaluate the system responses \( x_\theta(t_i) \) from parameters \( \theta \) against observations \( y(t_i) \). To overcome the computationally intensive forward solution, a family of methods denoted as gradient matching (Varah, 1982; Ellner et al., 2002; Ramsay et al., 2007) have proposed to replace the forward solution by matching \( f(y_i) \approx \dot{y}_i \) to empirical gradients \( \dot{y}_i \) of the data instead, which do not require the costly integration step. Recently several authors have proposed embedding a parametric differential function within a Bayesian or Gaussian process (GP) framework (Graepel, 2003; Calderhead et al., 2008;
Dondelinger et al., 2013; Wang and Barber, 2014; Macdonald, 2017) (see Macdonald et al. (2015) for a review). GPs have been successfully applied to model linear differential equations as they are analytically tractable (Gao et al., 2008; Raissi et al., 2017).

However, conventional ODE modelling can only proceed if a parametric form of the driving function \( f(\cdot) \) is known. Recently, initial work to handle unknown or non-parametric ODE models have been proposed, although with various limiting approximations. Early works include spline-based smoothing and additive functions \( \sum_j f_j(x_j) \) to infer gene regulatory networks (De Hoon et al., 2002; Henderson and Michailidis, 2014). Äijö and Lähdesmäki (2009) proposed estimating the unknown nonlinear function with GPs using either finite time differences, or analytically solving the derivative function as a function of only time, \( \dot{x}(t) = f(t) \) (Äijö et al., 2013). In a seminal technical report of Heinonen and d’Alche Buc (2014) a full vector-valued kernel model \( f(x) \) was proposed, however using a gradient matching approximation. To our knowledge, there exists no model that can learn non-linear ODE functions \( \dot{x}(t) = f(x(t)) \) over the state \( x \) against the true forward solutions \( x(t_i) \).

In this work we propose npODE\(^1\): the first ODE model for learning arbitrary, and a \emph{priori} completely unknown non-parametric, non-linear differential functions \( f : \mathcal{X} \to \mathcal{X} \) from data in a Bayesian way. We do not use gradient matching or other approximative models, but instead propose to directly optimise the exact ODE system with the fully forward simulated responses against data. We parameterise our model as an augmented Gaussian process vector field with inducing points, while we propose sensitivity equations to efficiently compute the gradients of the system. Our model can forecast continuous-time systems arbitrary amounts to future, and we demonstrate the state-of-the-art performance in human motion datasets.

2. Nonparametric ODE Model

The differential function \( f(x) \) to be learned defines a vector field\(^2\) \( f \), that is, an assignment of a gradient vector \( f(x) \in \mathbb{R}^D \) to every state \( x \in \mathbb{R}^D \). We model the vector field as a vector-valued Gaussian process (Rasmussen and Williams, 2006)

\[
f(x) \sim GP(0, K(x, x')), \tag{4}
\]

which defines \emph{a priori} distribution over function values \( f(x) \) whose mean and covariances are

\[
\mathbb{E}[f(x)] = 0, \tag{5}
\]
\[
\text{cov}[f(x), f(x')] = K(x, x'), \tag{6}
\]

\(^1\)The implementation is publicly available in http://www.github.com/cagatayyildiz/npode
\(^2\)We use vector field and differential function interchangeably.

Figure 1. (a) Illustration of an ODE system vector field induced by the Gaussian process. The vector field \( f(x) \) (gray arrows) at arbitrary states \( x \) is interpolated from the inducing points \( u, z \) (black arrows), with the trajectory \( x(t) \) (red points) following the differential system \( f(x) \) exactly. (b) The trajectory \( x(t) \) plotted over time \( t \).

and where the kernel \( K(x, x') \in \mathbb{R}^{D \times D} \) is matrix-valued. A GP prior defines that for any collection of states \( X = (x_1, \ldots, x_N)^T \in \mathbb{R}^{N \times D} \), the function values \( F = (f(x_1), \ldots, f(x_N))^T \in \mathbb{R}^{N \times D} \) follow a matrix-valued normal distribution,

\[
p(F) = N(\text{vec}(F)|0, K(X, X)), \tag{7}
\]

where \( K(X, X) = (K(x_i, x_j))_{i,j=1}^N \in \mathbb{R}^{ND \times ND} \) is a block matrix of matrix-valued kernels \( K(x_i, x_j) \). The key property of Gaussian processes is that they encode functions where similar states \( x, x' \) induce similar differentials \( f(x), f(x') \), and where the state similarity is defined by the kernel \( K(x, x') \).

In standard GP regression we would obtain the posterior of the vector field by conditioning the GP prior with the data (Rasmussen and Williams, 2006). In ODE models the conditional \( f(x)|Y \) of a vector field is intractable due to the integral mapping (2) between observed states \( y(t_i) \) and differentials \( f(x) \). Instead, we resort to augmenting the Gaussian process with a set of \( M \) inducing points \( u \in \mathcal{X} \) and \( u \in \mathcal{X} \), such that \( f(z) = u \) (Quiñonero-Candela and
We note that more complex kernels can also be considered with differential variance \( \sigma_f \) (Wilson et al., 2013) or non-stationary (Remes et al., 2017). The vector-valued kernel function (8) uses this definition is then compatible with the deterministic vector fields that have zero divergence (Wahlström et al., 2016), while spectral kernels can represent stationary recurring patterns in the differential function. Non-stationary vector fields can be modeled with input-dependent lengthscales (Heinonen et al., 2017). Non-stationary vector fields can be modeled with input-dependent lengthscales (Heinonen et al., 2017). Non-stationary vector fields can be modeled with input-dependent lengthscales (Heinonen et al., 2017). Non-stationary vector fields can be modeled with input-dependent lengthscales (Heinonen et al., 2017). Non-stationary vector fields can be modeled with input-dependent lengthscales (Heinonen et al., 2017). Non-stationary vector fields can be modeled with input-dependent lengthscales (Heinonen et al., 2017). Non-stationary vector fields can be modeled with input-dependent lengthscales (Heinonen et al., 2017). Non-stationary vector fields can be modeled with input-dependent lengthscales (Heinonen et al., 2017).

2.1. Operator-valued Kernels

The vector-valued kernel function (8) uses operator-valued kernels, which result in matrix-valued kernels \( K_\theta(z, z') \in \mathbb{R}^{D \times D} \) for real valued states \( z, z' \), while the kernel matrix over data points becomes \( K_\theta = (K(z_i, z_j))_{i,j=1}^M \in \mathbb{R}^{M \times M} \) (See Alvarez et al. (2012) for a review). Most straightforward operator-valued kernel is the identity decomposable kernel \( K_{dec}(z, z') = k(z, z') \cdot I_D \), where the scalar Gaussian kernel

\[
K_\theta(z, z') = \sigma_f^2 \exp \left( -\frac{1}{2} \sum_{j=1}^D \frac{(z_j - z'_j)^2}{\ell_j^2} \right)
\]

with differential variance \( \sigma_f^2 \) and dimension-specific lengthscales \( \ell = (\ell_1, \ldots, \ell_D) \) are expanded into a diagonal matrix of size \( D \times D \). We collect the kernel parameters as \( \theta = (\sigma_f, \ell) \).

We note that more complex kernels can also be considered given prior information of the underlying system characteristics. The divergence-free matrix-valued kernel induces vector fields that have zero divergence (Wahlström et al., 2013; Solin et al., 2015). Intuitively, these vector fields do not have sinks or sources, and every state always finally returns to itself after sufficient amount of time. Similarly, curl-free kernels induce curl-free vector fields that can contain sources or sinks, that is, trajectories can accelerate or decelerate. For theoretical treatment of vector field kernels, see (Narcowich and Ward, 1994; Bhatia et al., 2013; Fusiello and Wright, 2017). Non-stationary vector fields can be modeled with input-dependent lengthscales (Heinonen et al., 2016), while spectral kernels can represent stationary (Wilson et al., 2013) or non-stationary (Remes et al., 2017) recurring patterns in the differential function.

2.2. Joint Model

We assume a Gaussian likelihood over the observations \( y_i \) and the corresponding simulated responses \( \tilde{x}(t_i) \) of Equation (2),

\[
p(Y|z_0, U, \omega) = \prod_{i=1}^N p(y_i|\tilde{x}(t_i), \Omega),
\]

where \( \tilde{x}(t_i) \) are forward simulated responses using the integral Equation (2) and differential Equation (8), and \( \Omega = \text{diag}(\omega_{t1}^2, \ldots, \omega_{tD}^2) \) collects the dimension-specific noise variances.

The inducing vectors have a Gaussian process prior

\[
p(U|z, \theta) = \mathcal{N}(\text{vec}(U)|0, K_\theta(z, z)).
\]

The model posterior is then

\[
p(U, x_0, \theta, \omega|Y) \propto p(Y|x_0, U, \omega)p(U|\theta) = \mathcal{L},
\]

where we have for brevity omitted the dependency on the locations of the inducing points \( Z \) and also the parameter hyperpriors \( p(\theta) \) and \( p(\omega) \) since we assume them to be uniform, unless there is specific domain knowledge of the priors.

The model parameters are the initial state \( x_0 \), the inducing vectors \( U \), the noise standard deviations \( \omega = (\omega_1, \ldots, \omega_D) \), and the kernel hyperparameters \( \theta = (\sigma_f, \ell_1, \ldots, \ell_D) \).

2.3. Noncentral Parameterisation

We apply a latent parameterisation using Cholesky decomposition \( L_\theta L_\theta^T = K_\theta(z, z) \), which maps the inducing vectors to whitened domain (Kuss and Rasmussen, 2005)

\[
U = L_\theta \tilde{U}, \quad \tilde{U} = L_\theta^{-1} U.
\]

The latent variables \( \tilde{U} \) are projected on the kernel manifold \( L_\theta \) to obtain the inducing vectors \( U \). This non-centered parameterisation (NCP) transforms the hierarchical posterior \( \mathcal{L} \) of Equation (12) into a reparameterised form

\[
p(\tilde{x}_0, \tilde{U}, \Theta, \omega|Y) \propto p(Y|\tilde{x}_0, \tilde{U}, \omega, \theta)p(\tilde{U}),
\]

where all variables to be optimised are decoupled, with the latent inducing vectors having a standard normal prior \( \tilde{U} \sim \mathcal{N}(0, I) \). Optimizing \( \tilde{U} \) and \( \theta \) is now more efficient since they have independent contributions to the vector field via \( U = L_\theta \tilde{U} \). The gradients of the whitened posterior can be retrieved analytically as (Heinonen et al., 2016)

\[
\nabla_{\tilde{U}} \log \mathcal{L} = L_\theta^T \nabla_{U} \log \mathcal{L}.
\]

In case of multiple time-series, we will use one initial state for each time-series.
Finally, we find a maximum a posteriori (MAP) estimate for the initial state $x_0$, latent vector field $\tilde{U}$, kernel parameters $\theta$ and noise variances $\omega$ by gradient ascent,

$$x_{0,\text{MAP}}, \tilde{U}_{\text{MAP}}, \theta_{\text{MAP}}, \omega_{\text{MAP}} = \arg\max_{x_0, \tilde{U}, \theta, \omega} \log p(Y), \quad (16)$$

while keeping the inducing locations $Z$ fixed on a sufficiently dense grid (See Figure 1). The partial derivatives of the posterior with respect to noise parameters $\omega$ can be found analytically, while the derivative with respect to $\sigma_f$ is approximated with finite differences. We select the optimal length scales $\ell$ by cross-validation.

### 3. Sensitivity Equations

The key term to carry out the MAP gradient ascent optimization is the likelihood

$$\log p(Y|x_0, \tilde{U}, \omega)$$

that requires forward integration and computing the partial derivatives with respect to the whitened inducing vectors $\tilde{U}$. Given Equation (15) we only need to compute the gradients with respect to the inducing vectors $u = vec(U) \in \mathbb{R}^{MD}$,

$$\frac{d \log p(Y|x_0, u, \omega)}{du} = \sum_{s=1}^{N} \frac{d \log N(y_s|x(t_s, u), \Omega)}{dx} \frac{dx(t_s, u)}{du}. \quad (17)$$

This requires computing the derivatives of the simulated system response $x(t, u)$ against the vector field parameters $u$,

$$\frac{dx(t, u)}{du} \equiv S(t) \in \mathbb{R}^{D \times MD}, \quad (18)$$

which we denote by $S_{ij}(t) = \frac{\partial x(t, u)}{\partial u_j}$, and expand the notation to make the dependency on $u$ explicit. Approximating these with finite differences is possible in principle, but is highly inefficient and has been reported to cause instability (Raue et al., 2013). We instead turn to sensitivity equations for $u$ and $x_0$ that provide computationally efficient, analytical gradients $S(t)$ (Kokotovic and Heller, 1967; Fröhlich et al., 2017).

The solution for $\frac{dx(t, u)}{du}$ can be derived by differentiating the full nonparametric ODE system with respect to $u$ by

$$\frac{d}{dt} \frac{dx(t, u)}{du} = \frac{d}{du} \mathbf{f}(x(t, u)). \quad (19)$$

The sensitivity equation for the given system can be obtained by changing the order of differentiation on the left hand side and carrying out the differentiation on the right hand side.

The resulting sensitivity equation can then be expressed in the form

$$\frac{d}{dt} \frac{dx(t, u)}{du} = \frac{\partial \mathbf{f}(x(t, u))}{\partial x} \frac{dx(t, u)}{du} + \frac{\partial \mathbf{f}(x(t, u))}{\partial u}, \quad (20)$$

where $J(t) \in \mathbb{R}^{D \times D}$, $R(t) \in \mathbb{R}^{D \times MD}$ (See Supplements for detailed specification). For our nonparametric ODE system the sensitivity equation is fully determined by

$$J(t) = \frac{\partial \mathbf{K}(x, Z)}{\partial x} \mathbf{K}(Z, Z)^{-1} \mathbf{u} \quad (21)$$

$$R(t) = \mathbf{K}(x, Z) \mathbf{K}(Z, Z)^{-1}. \quad (22)$$

The sensitivity equation provides us with an additional ODE system which describes the time evolution of the derivatives with respect to the inducing vectors $S(t)$. The sensitivities are coupled with the actual ODE system and, thus both systems $x(t)$ and $S(t)$ are concatenated as the new augmented state that is solved jointly by Equation (2) driven by the differentials $x(t)$ and $\dot{S}(t)$ (Leis and Kramer, 1988). The initial sensitivities are computed as $S(0) = \frac{dx(t, u)}{dt}$. In our implementation, we merge $x_0$ with $u$ for sensitivity analysis to obtain the partial derivatives with respect to the initial state which is estimated along with the other parameters. We use the CVODES solver from the SUNDIALS package (Hindmarsh et al., 2005) to solve the nonparametric ODE models and the corresponding gradients numerically. The sensitivity equation based approach is superior to the finite differences approximation because we have exact formulation for the gradients of state over inducing points, which can be solved up to the numerical accuracy of the ODE solver.

### 4. Simple Simulated Dynamics

As first illustration of the proposed nonparametric ODE method we consider three simulated differential systems: the Van der Pol (VDP), FitzHugh-Nagumo (FHN) and Lotka-Volterra (LV) oscillators of form

\begin{align*}
\text{VDP} : \quad \dot{x}_1 &= x_2 \\
\dot{x}_2 &= (1 - x_1^2)x_2 - x_1 \\
\text{FHN} : \quad \dot{x}_1 &= 3(x_1 - x_1^3 + x_2) \\
\dot{x}_2 &= 0.2 - 3x_1 - 0.2x_2 \\
\text{LV} : \quad \dot{x}_1 &= 1.5x_1 - x_1x_2 \\
\dot{x}_2 &= -3x_2 + x_1x_2.
\end{align*}

In the conventional ODE case the coefficients of these equations can be inferred using standard statistical techniques if sufficient amount of time series data is available (Girolami, 2008; Raue et al., 2013). Our main goal is to infer unknown dynamics, that is, when these equations are unavailable and we instead represent the dynamics with a nonparametric
vector field of Equation (8). We use these simulated models to only illustrate our model behavior against the true dynamics.

We employ 25 data points from one cycle of noisy observation data from VDP and FHN models, and 25 data points from 1.7 cycles from the LV model with a noise variance of $\sigma^2_n = 0.12$. We learn the npODE model with five training sequences using $M = 6^2$ inducing locations on a fixed grid, and forecast between 4 and 8 future cycles starting from true initial state $x_0$ at time 0. Training takes approximately 100 seconds per oscillator. Figure 2 (bottom) shows the training datasets (grey regions), initial states, true trajectories (black lines) and the forecasted trajectory likelihoods (colored regions). The model accurately learns the dynamics from less than two cycles of data and can reproduce them reliably into future.

Figure 2 (top) shows the corresponding true vector field (black arrows) and the estimated vector field (grey arrows). The vector field is a continuous function, which is plotted on a 8x8 grid for visualisation. In general the most difficult part of the system is learning the middle of the loop (as seen in the FHN model), and learning the most outermost regions (bottom left in the LV model). The model learns the underlying differential $f(x)$ accurately close to observed points, while making only few errors in the border regions with no data.

5. Unknown System Estimation

Next, we illustrate how the model estimates realistic, unknown dynamics from noisy observations $y(t_1), \ldots, y(t_N)$. As in Section 4, we make no assumptions on the structure or form of the underlying system, and capture the underlying dynamics with the nonparametric system alone. We employ no subjective priors, and assume no inputs, controls or other sources of information. The task is to infer the underlying dynamics $f(x)$, and interpolate or extrapolate the state trajectory outside the observed data.

We use a benchmark dataset of human motion capture data from the Carnegie Mellon University motion capture (CMU mocap) database. Our dataset contains 50-dimensional pose measurements $y(t_i)$ from humans walking, where each pose dimension records a measurement in different parts of the body during movement (Wang et al., 2008). We apply the preprocessing of Wang et al. (2008) by downsampling the datasets by a factor of four and centering the data. This resulted in a total of 4303 datapoints spread across 43 trajec-
We compare our method to the dynamical model GPDM. We perform model selection for lengthscales \( \ell \). We evaluate the method with two types of experiments: VGPLVM of Damianou et al. (2011), where we di-vide the trajectory, which are to be filled by the models. We perform model selection for lengthscales \( \ell \) with cross-validation split of 80/20. We record the root mean square error (RMSE) over test points in the original feature space within a low-dimensional space, and subsequently projected back into the original space via inverting the PCA or with GPLVM mean predictions. As all methods optimize their latent spaces separately, they are not directly comparable. Thus, the mean errors are computed in the original high-dimensional space. Note that the low-dimensional representation necessarily causes some reconstruction errors.

Table 1. Means and standard deviations of RMSEs of 43 datasets in forecasting and filling experiments.

<table>
<thead>
<tr>
<th>Model</th>
<th>Forecasting</th>
<th>Imputation</th>
</tr>
</thead>
<tbody>
<tr>
<td>npODE</td>
<td>4.52 ± 2.31</td>
<td>3.94 ± 3.50</td>
</tr>
<tr>
<td>GPDM</td>
<td>4.94 ± 3.3</td>
<td>5.31 ± 3.39</td>
</tr>
<tr>
<td>VGPLVM</td>
<td>8.74 ± 3.43</td>
<td>3.91 ± 1.80</td>
</tr>
</tbody>
</table>

Due to the current lack of ODE methods suitable for this non-parametric inference task, we instead compare our method to the state-of-the-art state-space models where such problems have been previously considered (Wang et al., 2008). In a state-space or dynamical model a transition function \( x(t_{k+1}) = g(x(t_k)) \) moves the system forward in discrete steps. With sufficiently high sampling rate, such models can estimate and forecast finite approximations of smooth dynamics. In Gaussian process dynamical model (Wang et al., 2006; Frigola et al., 2014; Svensson et al., 2016) a GP transition function is inferred in a latent space, which can be inferred with a standard GPLVM (Lawrence, 2004) or with a dependent GPLVM (Zhao and Sun, 2016). In dynamical systems the transition function is replaced by a GP interpolation (Damianou et al., 2011). The discrete time state-space models emphasize inference of a low-dimensional manifold as an explanation of the high-dimensional measurement trajectories.

We compare our method to the dynamical model GPDM of Wang et al. (2006) and to the dynamical system VGPLVM of Damianou et al. (2011), where we directly apply the implementations provided by the authors at inverseprobability.com/vargplvm and dgp.toronto.edu/~jmwang/gpdm. Both methods optimize their latent spaces separately, and they are thus not directly comparable.

5.1. Forecasting

In the forecasting task we train all models with the first half of the trajectory, while forecasting the second half starting from the first frame. The models are trained and forecasted within a low-dimensional space, and subsequently projected back into the original space via inverting the PCA or with GPLVM mean predictions. As all methods optimize their latent spaces separately, they are not directly comparable. Thus, the mean errors are computed in the original high-dimensional space. Note that the low-dimensional representation necessarily causes some reconstruction errors.

Figure 3 illustrates the models on one of the trajectories 35_12.amc. The top part (a) shows the training data in the PCA space for npODE, and optimized training data representation for GPDM and VGPLVM (black points). The colored lines (npODE) and points (GPDM, VGPLVM) indicate the future forecast. The bottom part (b) shows the first 9 reconstructed original pose dimensions reconstructed from the latent forecasted trajectories. The training data is shown in grey background, while test data is shown with circles.

The VGPLVM has most trouble forecasting future points, and reverts quickly after training data to a value close to zero, failing to predict future points. The GPDM model produces more realistic trajectories, but fails to predict any of the poses accurately. Finally, npODE can accurately predict five poses, and still retains adequate performance on remaining poses, except for pose 2.

Furthermore, Table 1 indicates that npODE is also best performing method on average over the whole dataset in the forecasting.

5.2. Imputation

In the imputation task we remove approximately 20% of the training data from the middle of the trajectory. The goals are to learn a model with the remaining data and to forecast the missing values. Figure 4 highlights the performance of the three models on the trajectory 07_07.amc. The top part (a) shows the training data (black points) in the PCA space (npODE) or optimized training locations in the latent space (GPDM, VGPLVM). The middle part imputation is shown with colored points or lines. Interestingly both npODE and GPDM operate on cyclic representations, while VGPLVM is not cyclic.

The bottom panel (b) shows the first 9 reconstructed pose
Learning unknown ODE models with GP

Figure 3. Forecasting 50 future frames after 49 frames of training data of human motion dataset \texttt{3512.amc}. (a) The estimated locations of the trajectory in a latent space (black points) and future forecast (colored lines). (b) The original features reconstructed from the latent predictions with grey region showing the training data.

dimensions from the three models. The missing values are shown in circles, while training points are shown with black dots. All models can accurately reproduce the overall trends, while npODE seems to fit slightly worse than the other methods. The PCA projection causes the seemingly perfect fit of the npODE prediction (at the top) to lead to slightly warped reconstructions (at the bottom). All methods mostly fit the missing parts as well. Table 1 shows that on average the npODE and VGPLVM have approximately equal top performance on the imputing missing values task.

6. Discussion

We proposed the framework of nonparametric ODE model that can accurately learn arbitrary, nonlinear continuous-time dynamics from purely observational data without making assumptions of the underlying system dynamics. We demonstrated that the model excels at learning dynamics that can be forecasted into the future. We consider this work as the first in a line of studies of nonparametric ODE systems, and foresee several aspects as future work. Currently we do not handle non-stationary vector fields, that is time-dependent differentials $f_t(x)$. Furthermore, an interesting future avenue is the study of various vector field kernels, such as divergence-free, curl-free or spectral kernels (Remes et al., 2017). Finally, including inputs or controls to the system would allow precise modelling in interactive settings, such as robotics.

The proposed nonparametric ODE model operates along a continuous-time trajectory, while dynamic models such as hidden Markov models or state-space models are restricted to discrete time steps. These models are unable to consider system state at arbitrary times, for instance, between two successive timepoints.

Conventional ODE models have also been considered from the stochastic perspective with stochastic differential equation (SDE) models that commonly model the deterministic...
Figure 4. Imputation of 17 missing frames in the middle of a 94-length trajectory of human motion dataset 07_07.amc (subsampled every fourth frame). (a) The estimated locations of the missing points in the latent space are colored. (b) The original features reconstructed from the latent trajectory.

system drift and diffusion processes separately leading to a distribution of trajectories \( p(x(t)) \) (Archambeau et al., 2007; García et al., 2017). As future work we will consider stochastic extensions of our nonparametric ODE model, as well as MCMC sampling of the inducing point posterior \( p(U|Y) \), leading to trajectory distribution as well.

Acknowledgements. The data used in this project was obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217. This work has been supported by the Academy of Finland Center of Excellence in Systems Immunology and Physiology, the Academy of Finland grants no. 284597, 311584, 313271, 299915.

References


Learning unknown ODE models with GP


