DEEP GAUSSIAN PROCESS STATE-SPACE MODEL FOR MOTION GENERATION VIA STOCHASTIC EXPECTATION PROPAGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Gaussian Processes (GPs) and related unsupervised learning techniques such as Gaussian process dynamical models (GP-DMs) have been very successful in the accurate modeling of high-dimensional data based on limited amounts of training data. Usually these techniques have the disadvantage of a high computational complexity. This makes it difficult to solve the associated learning problems for large data sets, since the related computations, as opposed to neural networks, are not node-local. Combining sparse approximation techniques for GPs and stochastic expectation propagation (SEP), we present a framework for the computationally efficient implementation of deep Gaussian process (state-space) models. We provide implementations of this approach on the GPU as well as on the CPU. We present the first implementation of such deep GP-SSMs and demonstrate the computational efficiency of our GPU implementation.

1 INTRODUCTION

Many applications, e.g., computer graphics and robotics, require real-time generative models for complex, high-dimensional and coordinated human motion. One possible approach for the solution to this problem is the use of neural networks Harvey & Pal (2019); Holden et al. (2017). However, the generalization properties of such models are difficult to control and often they require substantial amounts of training data to accomplish high accuracy and robustness of the generated motion Mourot et al. (2022). As an alternative, which allows a better control aof generalization, we propose probabilistic graphical models Bishop (2007). They provide an attractive theoretical framework for constructing modular and hierarchical models that enable inference for arbitrary variables within their latent space. In addition, they are suitable for the formulation of multi-layer architectures in the form of Deep Gaussian processes (DGP) Kaiser et al. (2018); van der Wilk et al. (2020). Such architectures are equivalent to deep neural networks with infinitely many hidden units per layer Damianou & Lawrence (2013); Neal (1994). Furthermore, Gaussian process state-space models (GP-SSM), which are non-parametric generalizations of finite state-space models, allow to implement inference about time series Deisenroth & Mohamed (2012). Consequently, these models provide a framework for the accurate approximation of high-dimensional human motion.

For these reasons, it is not surprising that motion synthesis and editing techniques in computer graphics Chai & Hodgins (2005); Brand & Hertzmann (2000); Ikemoto et al. (2009) and robotics Zhao et al. (2018); Schmerling et al. (2018) have extensively used Gaussian process models. Notably, kinematic modelling and motion interpolation Grochow et al. (2004), inverse kinematics Levine et al. (2012), and learning of low-dimensional dynamical models Wang et al. (2008); Ye & Liu (2010) have all successfully applied Gaussian process latent variable (GP-LVM) models.

However, these models also come with their drawbacks. Due to their high computational complexity, many of these techniques result in offline models unsuitable for embedding online control systems. Moreover, they tend to overfit, making them inappropriate for various applications, e.g., large datasets. Nevertheless, different methods have addressed these issues. By the inclusion of sparse approximation techniques, they become suitable for real-time applications Taubert et al. (2013). By an approximate inference exploiting sparse approximations of Gaussian processes (GP) within a variational free energy (VFE) framework, their overfitting is controllable Titsias (2009). Nevertheless, the VFE approach is not parallelizable. It produces unnecessarily large memory footprints during learning, making it unsuitable for GPU implementations and thus preventing these models from being used with large scale datasets.

Therefore, to address this issue, we propose here to use approximate inference exploiting expectation propagation (EP) Minka (2001). EP results in a more suitable algorithm for GPU implementations as it is parallelizable during learning. Moreover, it provides significantly more accurate approximations. The use of stochastic expectation propagation (SEP) Li et al. (2015) also reduces the memory overhead. In addition, the underlying learning scheme of Power EP Minka (2004) combines the advantages of VFE and EP within a single algorithm.

Combining all these elements, we propose a probabilistic graphical model in the form of deep Gaussian process state-space models (DGP-SSM) which leverage GPU computing by exploiting approximate inference using SEP. We demonstrate the suitability of this approach for learning online generative models for complex full-body movements of two interacting humans with 189 degrees of freedom.

The following sections first develop the underlying mathematical theory. We then discuss the implementation details briefly and present our results, comparing the GPU and CPU implementations, followed by a conclusion.

2 Preliminaries

2.1 GP Regression and Approximation

In the function space view GPs can be considered as nonlinear mapping, $f(\mathbf{x})$, from an input variable $\mathbf{x}$ to a one-dimensional real-valued output variable. The function value $\mathrm{f}_n := f(\mathbf{x}_n)$, at a particular input point, $\mathbf{x}_n$ of an input set $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^{\mathrm{T}} \in \mathbb{R}^{N \times Q}$, is a random variable, and a GP is an infinite collection of random variables, any finite number of which have a joint Gaussian distribution Rasmussen & Williams (2008); Bui (2017), where N being the sample size and Q the dimensionality of the input space.

A real Gaussian process $f(\mathbf{x})$ is characterized through its mean function $m(\mathbf{x})$ and kernel function $k(\mathbf{x}, \mathbf{x}')$,

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})],$$
$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))].$$

This can be interpreted as $f(\mathbf{x})$ being drawn from a Gaussian process prior with mean function $m(\mathbf{x})$ and kernel function $k(\mathbf{x}, \mathbf{x}')$,

$$f(\mathbf{x}) \sim \mathcal{GP}(f(\mathbf{x}); m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')).$$

In most cases we assume a zero mean function, $m(\mathbf{x}) = 0$, since for our application the prior knowledge about $f(\cdot)$ can be encoded by the kernel function and its hyperparameters. For our model we used in all layers an automatic relevance determination (ARD) kernel of the form,

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{1}{2}\sum_{q=1}^{Q} \frac{|x_q - x'_q|^2}{l_q^2}\right), \tag{1}$$

where σ is the variance and l_q the length-scale of the q-th input dimension of the kernel. For this particular kernel $\theta = (\sigma, \{l_q\}_{q=1}^{Q})$ are the hyperparameters.

Typically for regression models, a data set $\{y_n\}_{n=1}^{N}$ is modeled by an unknown function $f(\cdot) := f$, evaluated at input location $\mathbf{x}_n$ and corrupted by some independent noise ε_n,

$$y_n = f(\mathbf{x}_n) + \varepsilon_n.$$

With the prior $p(\varepsilon_n) = \mathcal{N}(\varepsilon_n; 0, \beta^{-1})$ the corresponding probabilistic model can be written as follows,

$$f|\theta \sim \mathcal{GP}(f; 0, k(\cdot, \cdot)), \tag{2}$$

$$p(\mathbf{y}|f, \beta) = \prod_{n=1}^{N} \mathcal{N}(y_n; f(\mathbf{x}_n), \beta^{-1}), \tag{3}$$

where $\mathbf{y}$ specifies the vector of all one dimensional data points. To find the noise free function values $\mathbf{f}$ and the right hyperparameter set $\{\theta, \beta\}$ by hand is quite difficult. The usual Bayesian approach would be to specify a prior over the hyperparameter set to compute the joint posterior $p(\mathbf{f}, \theta, \beta|\mathbf{y})$. For Gaussian processes this inference problem is typically not analytically solveable, due to the intrinsic non-linearity of the GP. A common practice to handle this problem is to optimize the hyperparameter set instead by minimizing the negative log-marginal likelihood,

$$\begin{aligned}\mathcal{L}(\theta, \beta) &= -\log p(\mathbf{y}|\theta, \beta), \\ &= -\log \int p(\mathbf{y}|f(\mathbf{x}_n), \beta) p(f|\theta)\, \mathrm{d}f, \\ &= -\log \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{K}_{\mathbf{ff}} + \beta^{-1}\mathbf{I}), \\ &= -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|\mathbf{K}_{\mathbf{ff}} + \beta^{-1}\mathbf{I}| \\ &\quad -\frac{1}{2}\mathbf{y}^{\mathrm{T}}(\mathbf{K}_{\mathbf{ff}} + \beta^{-1}\mathbf{I})^{-1}\mathbf{y},\end{aligned}$$

where the constructed matrix $\mathbf{K}_{\mathbf{ff}}$ results from the covariance function evaluations of the training inputs $\{\mathbf{x}_n\}_{n=1}^N$, i.e $[\mathbf{K}_{\mathbf{ff}}]_{n,n'} = k(\mathbf{x}_n, \mathbf{x}_{n'})$. To derive a conditional posterior $p(\mathbf{y}_*|\mathbf{y}, \theta, \beta)$, based on Gaussian identities Bishop (2007), from the joint probability of the prior function of the training set $p(\mathbf{y}|\theta, \beta)$ and the prior function of the test set $p(\mathbf{y}_*|\theta, \beta)$ for the prediction of new test outputs $\mathbf{y}_*$ with given test inputs $\mathbf{x}_*$ and a *fixed* hyperparameter set Lawrence (2005),

$$\begin{aligned}\tilde{m}(\mathbf{x}_*) &= \mathbf{k}_{*,\mathbf{f}}(\mathbf{K}_{\mathbf{ff}} + \beta^{-1}\mathbf{I})^{-1}\mathbf{y}, \\ \tilde{k}(\mathbf{x}_*, \mathbf{x}'_*) &= k(\mathbf{x}_*, \mathbf{x}'_*) - \mathbf{k}_{*,\mathbf{f}}(\mathbf{K}_{\mathbf{ff}} + \beta^{-1}\mathbf{I})^{-1}\mathbf{k}_{*,\mathbf{f}}^{\mathrm{T}},\end{aligned}$$

which is also a GP. Similar to $\mathbf{K}_{\mathbf{ff}}$, the constructed matrices result from the covariance function evaluations between the test inputs $\{[\mathbf{x}_*]_i\}_{i=1}^I$ and training input locations $\{\mathbf{x}_n\}_{n=1}^N$, $[\mathbf{k}_{*,\mathbf{f}}]_{i,n} = k([\mathbf{x}_*]_i, \mathbf{x}_n)$.

This minimization procedure has several disadvantages: First, it often gets stuck at local minima and has a computational cost, due to the inversion of $\mathbf{K}_{\mathbf{ff}} + \beta^{-1}\mathbf{I}$, see equation (4), of $O(N^3)$ at each iteration step and a $O(N^2)$ for prediction. Second, one has to store the full observation vector $\mathbf{y}$ for learning and prediction, see equations (4) and (4). These limitations prohibit larger scale learning using this approach, because of time and memory limitations. Especially the memory aspect is critical if optimization is implemented by GPU computing. Further, due to intractability of the posterior joint probability $p(\mathbf{f}, \theta, \beta|\mathbf{y})$, the construction of hierarchical (deep) models is not possible, because of the conditional dependency between the function value sets in the different layers.

In this paper, we show that approximate inference, exploiting Expectation Propagation (EP) framework in combination with sparse approximations of the Gaussian processes, offers an elegant solution for these problems.

2.2 GP Sparse Approximations

In order to implement the proposed GP framework for large data sets it is essential to reduce the computational complexity. This can be accomplished by a the GP sparse approximation approach Quiñonero Candela & Rasmussen (2005). For this purpose, the noise-free function value set $\mathbf{f}$, is approximated by selection of a small set of $M \ll N$ pseudo-point inputs $\{[\mathbf{x}_{\mathbf{u}}]_m\}_{m=1}^M$, mapped to the function values $\mathbf{u}$. It is assumed that training and test points of the Gaussian process are approximately conditionally independent, if conditioned on their pseudo-points. Further, the function values of $\mathbf{f}$ and $\mathbf{u}$ are disjoint so that $f = \{\mathbf{f}, \mathbf{u}, f_{\neq \mathbf{f},\mathbf{u}}\}$. Under this assumptions, the GP prior, see equation (2), can be approximated as follows:

$$q(f|\theta) = q(\mathbf{f}|\mathbf{u}, \theta) p(\mathbf{u}|\theta) p(f_{\neq \mathbf{f},\mathbf{u}}|\mathbf{f}, \mathbf{u}, \theta), \tag{4}$$

where $p(\mathbf{u}|\theta)$ is the GP prior over $\mathbf{u}$. The conditional relationship between $\mathbf{u}$ and $\mathbf{f}$ is fundamental for this equation. With the GP priors $p(\mathbf{f}|\theta)$ and $p(\mathbf{u}|\theta)$ it is possible to derive from their joint probability the conditional dependency $p(\mathbf{f}|\mathbf{u}, \theta) = \mathcal{N}(\mathbf{f}; \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \mathbf{D}_{\mathbf{ff}})$, where $\mathbf{D}_{\mathbf{ff}} = \mathbf{K}_{\mathbf{ff}} - \mathbf{Q}_{\mathbf{ff}}$ and $\mathbf{Q}_{\mathbf{ff}} = \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{uf}}$. The constructed matrices correspond to the covariance function evaluations at the pseudo-point input locations $\{[\mathbf{x_u}]_m\}_{m=1}^M$, i.e. $[\mathbf{K_{uu}}]_{m,m'} = k_\mathbf{f}([\mathbf{x_u}]_m, [\mathbf{x_u}]_{m'})$ and similarly covariance function evaluations between pseudo-point input and data locations $[\mathbf{K_{fu}}]_{n,m} = k_\mathbf{f}(\mathbf{x}_n, [\mathbf{x_u}]_m)$. The matrices of $p(\mathbf{f}|\mathbf{u}, \theta)$ can be approximated by simpler forms using several approaches, compactly summarized Bui (2017) by:

$$q(\mathbf{f}|\mathbf{u}, \theta) = \prod_{b=1}^{B} \mathcal{N}(\mathbf{f}_b; \mathbf{K}_{\mathbf{f}_b\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \alpha\mathbf{D}_{\mathbf{f}_b,\mathbf{f}_b}). \tag{5}$$

where b indexes B disjoint blocks of data-function values with $\mathbf{f}_b = [f_1, \ldots, f_B]^\mathrm{T}$. The Deterministic Training Conditional (DTC) approximation uses $\alpha \to 0$; the Fully Independent Training Conditional (FITC) approximation uses $\alpha = 1$ and $B = N$; the Partially Independent Training Conditional (PITC) approximation uses $\alpha = 1$ Quiñonero Candela & Rasmussen (2005); Schwaighofer & Tresp (2002).

This approximation assumes that $f(\mathbf{x})$ is fully determined by the pseudo-point inputs and reduces the computational cost to $O(M^2N)$ during learning and $O(MN)$ for prediction.

The new prior $q(f|\theta)$ with approximation can be combined with the data likelihood to obtain the modified generative model:

$$q(\mathbf{y}, f|\theta) = q(f|\theta) \prod_{n=1}^{N} p(y_n|f(\mathbf{x}_n), \theta). \tag{6}$$

This modified model is suitable for the construction of hierarchical models, keeping the memory inference steps *node-local* because each layer is associated with its own GP approximation. However, this formulation still leaves some problems unsolved. One is the intrinsic non-linearity of the data likelihood and its related analytic intractability. The second problem is that, due to GP approximation (especially for DTC), the model tends to overfit Titsias (2009). Approximate inference exploiting the expectation propagation framework can solve these remaining problems.

3 Methods

3.1 Stochastic Expectation Propagation

Expectation Propagation (EP) is a deterministic Bayesian inference algorithm Minka (2001) which allows to approximate intractable, but factorizable joint-distributions. EP returns a tractable form of the model joint-distribution, evaluated on the observed data. In the case of GP regression, the approximation takes the form of an unnormalized process $q^*(f|\theta) \approx p(\mathbf{y}, f|\theta)$ (the superscript $*$ defines an unnormalized process). The basic concept of distributional inference approximations, like Variational Free Enery (VFE) Titsias (2009), EP Minka (2001) and Power EP Minka (2004) is the decomposition of the joint-distribution into terms of interest, such that $p(f, \mathbf{y}|\theta) = p^*(\mathbf{y}|f, \theta) = p(\mathbf{y}|\theta)p(f|\mathbf{y}, \theta)$. They are fully reflected by tractable terms of the decomposed approximation $q^*(f|\theta) = Zq(f|\theta)$, where the normalization constant Z approximates the marginal likelihood $p(\mathbf{y}|\theta) \approx Z$ and the posterior is approximated by GP sparse approximation, equation (4), so that $p(f|\mathbf{y}, \theta) \approx q(f|\theta)$, i.e. the approximate inference schemes return simultaneously approximations of the posterior and marginal likelihood in form of an unnormalized Gaussian process $q^*(f|\theta)$.

For the implementation of EP we reformulate the unnormalized form $p^*(\mathbf{y}|f, \theta)$ in terms of a dense Gaussian process prior $p(f|\theta)$, see equation (2), and the independent likelihoods $\{p(y_n|f, \theta)\}_{n=1}^N$. EP constructs the approximate posterior as a product of *site functions* t_n Naish-guzman & Holden (2008) and employs an approximating family whose form mirrors that of the target Bui (2017),

$$\begin{aligned} p^*(f|\mathbf{y}, \theta) = p(f|\mathbf{y}, \theta)p(\mathbf{y}|\theta) = p(f|\theta) \prod_{n=1}^{N} p(y_n|f, \theta) \\ \approx p(f|\theta) \prod_{n=1}^{N} t_n(\mathbf{u}) = Z\, q(f|\theta) = q^*(f|\theta), \end{aligned} \tag{7}$$

where $t_n(\mathbf{u})$ is approximated by a simple Gaussian. The site functions were iterativly refined by minimizing an unnormalized Kullback-Leibler divergence, $\overline{\mathrm{KL}}$ Bui (2017), between the real posterior and each of the distributions formed by replacing one of the likelihoods by the corresponding approximating factor Li et al. (2015),

$$\underset{t_n(\mathbf{u})}{\arg\min}\,\overline{\mathrm{KL}}\left(p(f,\mathbf{y}|\theta)\;\middle\|\;\frac{p(f,\mathbf{y}|\theta)\,t_n(\mathbf{u})}{p(y_n|\mathrm{f}_n,\theta)}\right) = \underset{t_n(\mathbf{u})}{\arg\min}\,\overline{\mathrm{KL}}(p^*_{\setminus n}(f|\theta)p(y_n|\mathrm{f}_n,\theta)\;\|\;p^*_{\setminus n}(f|\theta)\,t_n(\mathbf{u})).$$

Unfortunately, such an update is still intractable as it involves the computation of the full posterior. Instead, EP replaces the leave-one-out posteriors $p^*_{\setminus n}(f|\theta) \propto p(f,\mathbf{y}|\theta)/p(y_n|\mathrm{f}_n,\theta)$ on both arguments of KL by approximate leave-one-out posterior $q^*_{\setminus n}(f|\theta) \propto q^*(f|\theta)/t_n(\mathbf{u})$, called the cavity, so that:

$$\overline{\mathrm{KL}}(q^*_{\setminus n}(f|\theta)p(y_n|\mathrm{f}_n,\theta)\;\|\;q^*_{\setminus n}(f|\theta)t_n(\mathbf{u})) = \overline{\mathrm{KL}}(q^*_{\setminus n}(f|\theta)p(y_n|\mathrm{f}_n,\theta)\;\|\;q^*(f|\theta)).$$

The update for the approximating factors are coupled and must be optimized by iterative updating. EP is doing this in four steps. We apply here a generalized version, called Power EP Minka (2004), where only a fraction α of the approximate or true likelihood is removed (or included). The steps are as follows:

1. Compute the cavity distribution by removing a fraction of one approximate factor, $q^*_{\setminus n}(f|\theta) \propto q^*(f|\theta)/t_n^\alpha(\mathbf{u})$.
2. Compute a hybrid or tilted distribution, $\tilde{p}(f|\theta) = q^*_{\setminus n}(f|\theta)p^\alpha(y_n|\mathrm{f}_n,\theta)$.
3. Project the hybrid distribution onto the approximate posterior by minimizing unnormalized KL divergence,
$$q^*(f|\theta) \leftarrow \underset{q^*(f|\theta)\in\mathcal{Q}}{\arg\min}\,\overline{\mathrm{KL}}(\tilde{p}(f|\theta)\;\|\;q^*(f|\theta)),$$
where $\mathcal{Q}$ is the approximation set defined in equation (7).
4. Update the approximate factor by multiplicative mixing of the new fraction of the approximate factor, $t_n(\mathbf{u}) = [t_n^{1-\alpha}(\mathbf{u})]_{\mathrm{old}}[t_n^\alpha(\mathbf{u})]_{\mathrm{new}}$, with $[t_n^\alpha(\mathbf{u})]_{\mathrm{new}} = q^*(f|\theta)/q^*_{\setminus n}(f|\theta)$.

The fractional updates are equivalent to running the original EP procedure, but replacing the KL minimisation with an α-divergence minimisation Zhu & Rohwer (1995); Minka (2004), where

$$\overline{\mathrm{D}}_\alpha(p^*(f|\theta)\;\|\;q^*(f|\theta)) = \frac{1}{\alpha(1-\alpha)}\int[\alpha\,p^*(f|\theta) + (1-\alpha)q^*(f|\theta) - p^*(f|\theta)^\alpha q^*(f|\theta)^{1-\alpha}]\,\mathrm{d}f. \quad (8)$$

The α-divergence becomes the inclusive KL divergence, $\overline{\mathrm{KL}}(p^*(f|\theta)\;\|\;q^*(f|\theta))$, for $\alpha = 1$. It becomes the exclusive KL divergence, $\overline{\mathrm{KL}}(q^*(f|\theta)\;\|\;p^*(f|\theta))$, when $\alpha \to 0$. Minimising a set of local exclusive KL divergences is equivalent to minimizing a single global exclusive KL divergence Minka (2004) and the Power EP solution is the minimum of a variational free energy (VFE) Bui (2017). Since the unnormalized approximation, $q^*(f|\theta)$, consists of the product of independent approximate factors $t_n(\mathbf{u})$, the data can be partitioned in B disjoint blocks $\mathbf{y}_b = \{y_n\}_{n\in\mathcal{B}_b}$. The choice of α and corresponding approximate factors $t_b(\mathbf{u})$ have a strong influence on the GP sparse approximation scheme, see equation (5) (and Bui (2017) for more details) and enables batch learning, which makes the GP regression task scaleable. However, local computation comes at the cost of memory overhead that grows with the number of datapoints, since local approximating factors need to be stored for every datapoint.

Stochastic Expectation Propagation (SEP) reduces the memory complexity of Power EP by a factor of N. This is accomplished by parameterizing a global factor, $t(\mathbf{u})$, that captures the average effect of a likelihood on the posterior $t(\mathbf{u})^N := \prod_{n=1}^N t_n(\mathbf{u})$ and combines the benefits of local

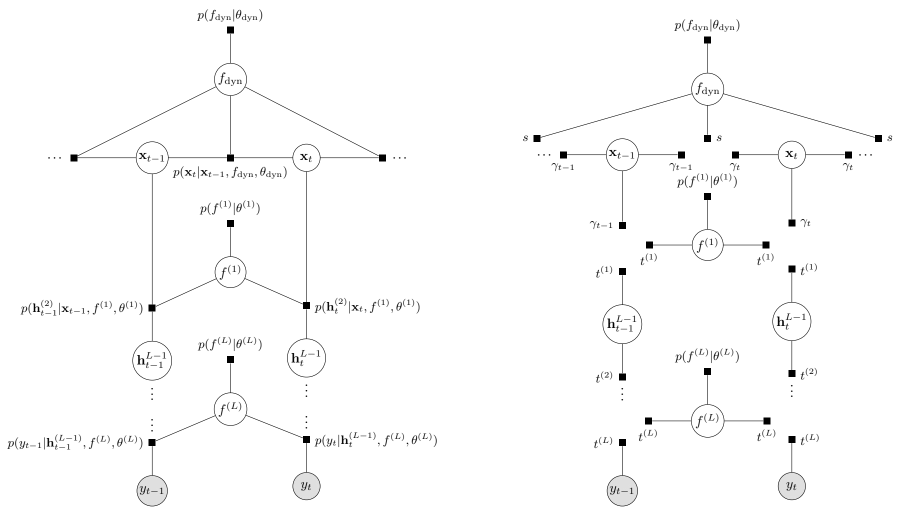

Figure 1: *Factor graphs of the DGP-SSM.* The model consists of a dynamic layer on top and multiple emission layers. Left: The model is fully factorized with tied factor constraints.

approximation (tractability of updates, distributability, and parallelisability) with global approximation (reduced memory demands) Li et al. (2015). The set of the approximate posterior in equation (7) can be rewritten as,

$$p^*(f|\mathbf{y},\theta) = p(f|\mathbf{y},\theta)p(\mathbf{y}|\theta) = p(f|\theta)\prod_{n=1}^{N} p(y_n|f,\theta)$$

$$\approx p(f|\theta)\prod_{n=1}^{N} t_n(\mathbf{u}) = p(f|\theta)t(\mathbf{u})^N = q^*_{\text{SEP}}(f|\theta). \quad (9)$$

One method to implement SEP would be to compute the approximate factorizations by the iterative procedure of Power EP and optimizing the hyperparameters in an outer loop with Power EP Li et al. (2015). But, it was shown in Hernandez-Lobato & Hernandez-Lobato (2016) that the factor tying approximation turns the optimization problem into a minimization problem, i.e. the approximate Power EP energy can be optimized with standard optimization algorithms (e.g. ADAM, L-BFGS-B) to find the approximate posterior and hyperparameters at the same time for each iteration step. This makes construction of hierarchical deep GP-LVMs feasible.

3.2 Deep Gaussian Process State-Space Model

GP-SSMs are a general form of discrete-time state-space models with nonlinear transitions between continuous latent variables. They are able to compress high dimensional data into low dimensional latent factors and to separate transition and measurment noise Bui (2017). Standard GP models with sparse approximation have strong limitations in terms of real world data sets with large numbers of training examples. This typically requires a substantial increase of the pseudo-inputs for a good approximation, resulting in a quadratic increase of computational complexity with the number of data points. This renders larger-scale learning not practicable. Constructing a multi-layer GP model reduces the computational cost to $O(NLM^2)$, where L is the number of layers. The general DGP-SSM consists of a GP dynamic layer and multiple GP emission layers,

$$\begin{aligned} p(\mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_0;\mu_0,\Sigma_0), \\ p(\mathbf{x}_t|\mathbf{x}_{t-1},f_{\text{dyn}},\theta_{\text{dyn}}) &= \mathcal{N}(\mathbf{x}_t;f_{\text{dyn}}(\mathbf{x}_{t-1}),\beta^{-1}_{\text{dyn}}), \\ p(h_t^{(l)}|h_t^{(l-1)},f^{(l)},\theta^{(l)}) &= \mathcal{N}(h_t^{(l)};f^{(l)}(h_t^{(l-1)}),[\beta^{(l)}]^{-1}), \end{aligned}$$

for $h_t^{(1)} = \mathbf{x}_t$, $h_t^{(L)} = y_t, t = 1:T$ and $l = 1:L$, where $\mathbf{h}_t^{(l)}$ is the hidden variable associated to the l-th emission layer and $f^{(l)}$ is the function of the l-th layer. The joint density of the latent variables,latent dynamics and the observed measurements is,

$$p(\mathbf{h}^{(1:L)}, f_{\text{dyn}}, f^{(1:L)}|\theta_{\text{dyn}}, \theta^{(1:L)}) = p(\mathbf{x}_0)p(f_{\text{dyn}}|\theta_{\text{dyn}}) \\ \times \left[\prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{x}_{t-1}, f_{\text{dyn}}, \theta_{\text{dyn}})\right] \left[\prod_{l=1}^{L} p(f^{(l)}|\theta^{(l)}) \prod_{t=1}^{T} p(\mathbf{h}_t^{(l)}|\mathbf{h}_t^{(l-1)}, f^{(l)}, \theta^{(l)})\right]. \quad (10)$$

The joint probability is analytically intractable, due to the non-liniarity of the data likelihoods and also the unknown intermediate outputs of each layer. So we make use of the SEP set for a tied factor constraint, see equation (9), in combination with GP sparse approximation to approximate the data likelihood of each layer,

$$q^*_{\text{SEP}}(\mathbf{x}_{0:T}, f_{\text{dyn}}, f^{(1:L)}|\theta_{\text{dyn}}, \theta^{(1:L)}) \\ = p(\mathbf{x}_0)\gamma(\mathbf{x}_0)\left[Z_{\text{dyn}} q_{\text{dyn}}(f_{\text{dyn}}|\theta_{\text{dyn}})\right]\left[Z_{\text{emiss}} \prod_{l=1}^{L} q^{(l)}(f^{(l)}|\theta^{(l)})\right] \\ = p(f_{\text{dyn}}|\theta_{\text{dyn}}) s(\mathbf{u}_{\text{dyn}})^T p(\mathbf{x}_0)\gamma(\mathbf{x}_0) \prod_{t=1}^{T} \gamma_{t-1}(\mathbf{x}_{t-1})\gamma_t(\mathbf{x}_t) \prod_{l=1}^{L} p(f^{(l)}|\theta^{(l)}) t(\mathbf{u}^{(l)})^T, \quad (11)$$

where $s(\mathbf{u}_{\text{dyn}})^T$ is the global factor of the dynamic layer, $\gamma_{t-1}(\mathbf{x}_{t-1})\gamma_t(\mathbf{x}_t)$ are the global transition factors with same parameterisation for similar factors, enforcing an identical contribution from each factor to the posterior, i.e. each common factor could be thought of as the average effect each factor has on the posterior Bui (2017). The factor graph of our model and its factorisations with global factor approximations is illustrated in figure 1.

The transition factors have to be optimized to obtain the approximate contribution towards the true posterior. Therefore the cavity for the dynamic layer can be computed as follows,

$$q^*_{\setminus 1}(\mathbf{X}_{1:T}, \mathbf{X}_{0:T-1}) = p(\mathbf{X}_{1:T}, \mathbf{X}_{0:T-1})/\gamma^{\alpha}(\mathbf{X}), \\ q^*_{\setminus 1}(f_{\text{dyn}}|\theta_{\text{dyn}}) = p(f_{\text{dyn}}|\theta_{\text{dyn}})/s^{(1-\alpha)}(\mathbf{u}_{\text{dyn}})^T.$$

Due to tied factor constraints the cavities above can be easily computed by taking out just a fraction of the posteriors. In combination with a fraction of the exact transition factors the tilted distribution takes the following form,

$$\tilde{p}(f_{\text{dyn}}, \mathbf{X}_{1:T}, \mathbf{X}_{0:T-1}|\theta) = q^*_{\setminus 1}(\mathbf{X}_{1:T}, \mathbf{X}_{0:T-1}) q^*_{\setminus 1}(f_{\text{dyn}}|\theta_{\text{dyn}}) p^{\alpha}(\mathbf{X}_{1:T}|\mathbf{X}_{0:T-1}, f_{\text{dyn}}, \theta_{\text{dyn}}).$$

The essential projection step of EP performs a minimization of the unnormalized KL divergence $\overline{\text{KL}}(\tilde{p}(f_{\text{dyn}}, \mathbf{X}_{1:T}, \mathbf{X}_{0:T-1}|\theta) \parallel q^*(f_{\text{dyn}}, \mathbf{X}_{1:T}, \mathbf{X}_{0:T-1}|\theta))$ by projecting the hybrid onto the approximate posterior $q^*(f_{\text{dyn}}, \mathbf{X}_{1:T}, \mathbf{X}_{0:T-1}|\theta) = proj(\tilde{p}(f_{\text{dyn}}, \mathbf{X}_{1:T}, \mathbf{X}_{0:T-1}|\theta))/(q^*_{\setminus 1}(\mathbf{X}_{1:T}, \mathbf{X}_{0:T-1}) q^*_{\setminus 1}(f_{\text{dyn}}|\theta_{\text{dyn}}))$, where the projection operator $proj(\cdot)$ returns the moments of the hybrid. This iterative moment matching refinement is achived by computing the log-partition of the hybrid $\log \tilde{Z}_{\text{dyn}} = \log \int \tilde{p}(f, \mathbf{X}_{1:T}, \mathbf{X}_{0:T-1}|\theta)\, \mathrm{d}\mathbf{X}_{1:T}\, \mathrm{d}\mathbf{X}_{0:T-1}\, \mathrm{d}f$ and its gradients w.r.t. cavity mean and covariance and leads to the following updates,

$$\boldsymbol{\mu} = \boldsymbol{\mu}_{\setminus 1} + \boldsymbol{\Sigma}_{\setminus 1} \frac{\mathrm{d} \log \tilde{Z}_{\text{dyn}}}{\mathrm{d}\boldsymbol{\mu}_{\setminus 1}}, \\ \boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{\setminus 1} - \boldsymbol{\Sigma}_{\setminus 1} \left(\frac{\mathrm{d} \log \tilde{Z}_{\text{dyn}}}{\mathrm{d}\boldsymbol{\mu}_{\setminus 1}} \left(\frac{\mathrm{d} \log \tilde{Z}_{\text{dyn}}}{\mathrm{d}\boldsymbol{\mu}_{\setminus 1}} \right)^{\mathrm{T}} - 2 \frac{\mathrm{d} \log \tilde{Z}_{\text{dyn}}}{\mathrm{d}\boldsymbol{\Sigma}_{\setminus 1}} \right) \boldsymbol{\Sigma}_{\setminus 1}.$$

With the new mean and covariance, the global factor can be easily updated by dividing the new approximate posterior $q^*(f_{\text{dyn}}, \mathbf{X}_{1:T}, \mathbf{X}_{0:T-1}|\theta_{\text{dyn}})$ by the cavity distributions, see step 4 in EP procedure.

The update rules for the emission layers include similar steps the ones of the dynamic layer. The cavity of each layer can be calculated by dividing layers posterior $p(f^{(l)}|\theta^{(l)})$ by a fraction of the corresponding global factor,

$$q^*_{\setminus 1}(f^{(l)}|\theta^{(l)}) = p(f^{(l)}|\theta^{(l)})/t^{(1-\alpha)}(\mathbf{u}^{(l)})^T. \quad (12)$$

For the emission model, the hybrid is composed of products of cavity distributions and fractions of exact factors over layers,

$$\tilde{p}(f^{(1:L)}, \mathbf{h}^{(1:L-1)}|\theta^{(1:L)}) \quad = \quad \left[\prod_{l=1}^{L} q^{*}_{\setminus 1}(f^{(l)}|\theta^{(l)})\right]\left[\prod_{l=1}^{L-1} p^{\alpha}(\mathbf{h}^{(l)}|\mathbf{h}^{(l-1)}, f^{(l)}, \theta^{(l)})\right]. \quad (13)$$

For the iterative moment matching is the log-partition extended over layers, $\log \tilde{Z}_{\text{emiss}} = \log \int \tilde{p}(f^{(1:L)}, \mathbf{h}^{(1:L-1)}|\theta^{(1:L)})\, \mathrm{d}\mathbf{h}^{(1:L-1)}\, \mathrm{d}f^{(1:L)}$, to perform the updates for new means and covariances per layer,

$$\begin{aligned}
\mathbf{m}^{(l)} &= \mathbf{m}^{(l)}_{\setminus 1} + \mathbf{S}^{(l)}_{\setminus 1} \frac{\mathrm{d} \log \tilde{Z}_{\text{emiss}}}{\mathrm{d}\mathbf{m}^{(l)}_{\setminus 1}}, \\
\mathbf{S}^{(l)} &= \mathbf{S}^{(l)}_{\setminus 1} - \mathbf{S}^{(l)}_{\setminus 1} \left(\frac{\mathrm{d} \log \tilde{Z}_{\text{emiss}}}{\mathrm{d}\mathbf{m}^{(l)}_{\setminus 1}} \left(\frac{\mathrm{d} \log \tilde{Z}_{\text{emiss}}}{\mathrm{d}\mathbf{m}^{(l)}_{\setminus 1}} \right)^{\mathrm{T}} - 2 \frac{\mathrm{d} \log \tilde{Z}_{\text{emiss}}}{\mathrm{d}\mathbf{S}^{(l)}_{\setminus 1}} \right) \mathbf{S}^{(l)}_{\setminus 1}.
\end{aligned}$$

Again, with the new means and covariances, the approximate factors can be trivially updated by dividing the cavity $q^{*}_{\setminus 1}(f^{(l)}|\theta^{(l)})$ distribution from the new approximate posterior $q^{*}(f^{(l)}, \mathbf{h}^{(l)}|\theta)$ of each layer.

4 Results

We tested our method by modeling the body movements of subjects performing a golf swing with 99 DOFs. The kinematic data (BVH format; from `http://mocap.cs.cmu.edu/`, subject 64) was converted to exponential maps Grassia (1998).Our DGP-SSM was trained on a small set of 779 data points (7 trajectories, downsampled from 120Hz to 30Hz) with two and three layers and different α-values (0.1, 0.25, 0.5, 0.75, 0.9). For comparison, we trained a usual one layer GPDM with the same training data. All models used FITC approximation with 90 pseudo-point inputs per model.

Learning was implemented on the CPU with an AMD Ryzen Threadripper 1950X 16 core processor 3.4 GHz with 64 GB RAM, and on GPU with a NVIDIA Quadro P600 graphics card with 24 GB RAM. Both implementations were realized in C++ using the ArrayFire framework Yalamanchili et al. (2015) which allows to implement a single code that can be run alternatively either on the CPU or the GPU.

For maximal 1000 iteration steps in each learning condition this took for the DGP-SSM in average 2.822h on CPU and 0.322h on GPU. The GPU implementation of DGP-SSM was almost nine times faster than the one on the CPU. The usual GPDM was the fastest of all CPU implementations (Table 1). One possible reason could be the different learning approach and code implementation of the usual GPDM. We also found a decrease in optimization time with increasing α-values (i.e. the approximation being closer to a Power EP than to VFE). Also, the optimization time decreases for models with more layers (due to the fact of computational complexity of $O(LNM^2)$).

To determine the prediction performance of our models we computed the *normalized mean squared error* (NMSE) and *mean absolute errors* (MAE), where the DGP-SSM performed overall better than the GPDM. We found also an increase of NMSE and MAE with higher α-values. An α-value of 0.5 seems to be the best trade-of between prediction performance and optimization time. Our probabilistic generative motion model is fully real-time capable for computer-animation applications with the GPU implementation, since the computation time is only 0.0035ms per frame.

5 Conclusion

Combining methods from Bayesian unsupervised learning and inference, we devised a real-time-capable method for the realization of deep Gaussian process state-space models (DGP-SSMs). Our method combines sparse GP approximations with Stochastic Expectation Propagation, and we provide a GPU implementation of the developed algorithms. We found that optimization using α-values around 0.5 results in the best trade-of between prediction accuracy of the model and optimization

	GPDM	DGP-SSM (2 Layer)					DGP-SSM (3 Layer)				
		$\alpha = 0.1$	$\alpha = 0.25$	$\alpha = 0.5$	$\alpha = 0.75$	$\alpha = 0.9$	$\alpha = 0.1$	$\alpha = 0.25$	$\alpha = 0.5$	$\alpha = 0.75$	$\alpha = 0.9$
NMSE	0.5868	0.2847	0.2685	0.3004	0.3125	0.3312	0.2980	0.2882	0.3102	0.3146	0.3289
MAE	9.8763	0.2068	0.1890	0.2035	0.2057	0.2325	0.2202	0.2069	0.2168	0.2132	0.2305
OT CPU	0.893h	4.2h	4.051h	3.991h	3,771h	2.418h	2.152h	2.131h	2.13h	2.129h	1.251h
OT GPU	-	0.483h	0.465h	0.459h	0.435h	0.278h	0.247h	0.245h	0.245h	0.245h	0.118h

Table 1: *Normalized mean square error (NMSE), mean absolute error (MAE) and optimization time (OT) for different learning conditions.* Decreasing optimization time with higher α-values and more layers comes at the cost of lower prediction performance.

speed. In spite of the sophisticated underlying probabilistic model, we demonstrated that the algorithm is real-time-capable when implemented on the GPU for applications in computer animation.

Future work will use such architectures as part of real-time engines for motion generation in computer graphics and VR. Further we plan to apply our algorithm on bigger data sets, exploiting batch learning on the GPU.

REFERENCES

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

Matthew Brand and Aaron Hertzmann. Style machines. In *Proc. SIGGRAPH'00*, pp. 183–192, 2000.

Thang Duc Bui. Efficient Deterministic Approximate Bayesian Inference for Gaussian Process models. (September), 2017.

Jinxiang Chai and Jessica K. Hodgins. Performance animation from low-dimensional control signals. *ACM Trans. Graph.*, 24(3):686–696, 2005.

Andreas Damianou and Neil Lawrence. Deep Gaussian Processes. volume 31 of *Proceedings of Machine Learning Research*, pp. 207–215, Scottsdale, Arizona, USA, 2013. PMLR.

Marc Deisenroth and Shakir Mohamed. Expectation propagation in gaussian process dynamical systems. volume 25. Curran Associates, Inc., 2012.

F. Sebastin Grassia. Practical parameterization of rotations using the exponential map. *J. Graph. Tools*, 3(3):29–48, March 1998. ISSN 1086-7651. doi: 10.1080/10867651.1998.10487493.

Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popovic. Style-based inverse kinematics. *ACM Trans. Graph.*, 23(3):522–531, 2004.

Félix G Harvey and Christopher Pal. Recurrent Transition Networks for Character Locomotion, 2019.

Daniel Hernandez-Lobato and Jose Miguel Hernandez-Lobato. Scalable Gaussian Process Classification via Expectation Propagation. In Arthur Gretton and Christian C Robert (eds.), *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pp. 168–176, Cadiz, Spain, 2016. PMLR.

Daniel Holden, Taku Komura, and Jun Saito. Phase-functioned neural networks for character control. *ACM Trans. Graph.*, 36(4), July 2017. ISSN 0730-0301. doi: 10.1145/3072959.3073663.

Leslie Ikemoto, Okan Arikan, and David A. Forsyth. Generalizing motion edits with Gaussian processes. *ACM Trans. Graph.*, 28(1), 2009.

Markus Kaiser, Clemens Otte, Thomas Runkler, and Carl Henrik Ek. Bayesian Alignments of Warped Multi-Output Gaussian Processes. In S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi, and R Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 6995–7004. Curran Associates, Inc., 2018.

N. D. Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of Machine Learning Research*, 6:1783–1816, 2005.

Sergey Levine, Jack M. Wang, Alexis Haraux, Zoran Popović, and Vladlen Koltun. Continuous character control with low-dimensional embeddings. *ACM Trans. Graph.*, 31(4):1–10, jul 2012. ISSN 07300301. doi: 10.1145/2185520.2335379.

Yingzhen Li, Jose Miguel Hernandez-Lobato, and Richard E Turner. Stochastic Expectation Propagation, 2015.

Thomas P Minka. Expectation Propagation for Approximate Bayesian Inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, pp. 362–369, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558608001.

Tom Minka. Power EP. Technical report, 2004.

Lucas Mourot, Ludovic Hoyet, François Le Clerc, François Schnitzler, and Pierre Hellier. A survey on deep learning for skeleton-based human animation. In *Computer Graphics Forum*, volume 41, pp. 122–157. Wiley Online Library, 2022.

Andrew Naish-guzman and Sean Holden. The Generalized FITC Approximation. In J Platt, D Koller, Y Singer, and S Roweis (eds.), *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2008.

R.M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, Dept. of Computer Science, University of Toronto, 1994.

Joaquin Quiñonero Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *J. Mach. Learn. Res.*, 6:1939–1959, 2005. ISSN 1532-4435.

Carl Edward Rasmussen and Christopher K. I. Williams. Gaussian processes for machine learning. *J. Am. Stat. Assoc.*, 103:429–429, 2008.

E Schmerling, K Leung, W Vollprecht, and M Pavone. Multimodal Probabilistic Model-Based Planning for Human-Robot Interaction. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3399–3406, may 2018. doi: 10.1109/ICRA.2018.8460766.

Anton Schwaighofer and Volker Tresp. Transductive and inductive methods for approximate gaussian process regression. In *Advances in Neural Information Processing Systems 15*, pp. 953–960. MIT Press, 2002.

Nick Taubert, Martin Löffler, Nicolas Ludolph, Andrea Christensen, Dominik Endres, and Martin A. Giese. A virtual reality setup for controllable, stylized real-time interactions between humans and avatars with sparse Gaussian process dynamical models. In *Proc. SAP'13*, pp. 41–44, 2013. doi: 10.1145/2492494.2492515.

M Titsias. Variational learning of inducing variables in sparse Gaussian processes. *Journal of Machine Learning Research - Proceedings Track*, 5:567–574, 2009.

Mark van der Wilk, Vincent Dutordoir, S T John, Artem Artemev, Vincent Adam, and James Hensman. A Framework for Interdomain and Multioutput Gaussian Processes, 2020.

Jack M. Wang, David J. Fleet, and Aaron Hertzmann. Gaussian process dynamical models for human motion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(2):283–298, 2008.

Pavan Yalamanchili, Umar Arshad, Zakiuddin Mohammed, Pradeep Garigipati, Peter Entschev, Brian Kloppenborg, James Malcolm, and John Melonakos. ArrayFire - A high performance software library for parallel computing with an easy-to-use API, 2015.

Yuting Ye and C. Karen Liu. Synthesis of responsive motion using a dynamic model. *Computer Graphics Forum*, 29(2):555–562, 2010.

Xingyu Zhao, Valentin Robu, D Flynn, Fateme Dinmohammadi, Michael Fisher, and Matt Webster. Probabilistic Model Checking of Robots Deployed in Extreme Environments, 2018.

Huaiyu Zhu and Richard Rohwer. Information Geometric Measurements of Generalisation. 1995.