# Gaussian Process Methods for Dimension Reduction

**Sai Satya Charan Malladi**
MSE Student
Department of Aerospace Engineering
University of Michigan, Ann Arbor
charanm@umich.edu

## Abstract

In this project, I have implemented a part of the paper "Gaussian Process Dynamical Models" by Wang et.al [6]. The paper proposes a dimension reduction method for high dimensional data. The authors test their method on human motion capture data. Specifically, I have learnt and compared Gaussian Process Dynamical Models(GPDM) to Gaussian Process Latent Variable Method(GPLVM) to test the claim that "latent maps obtained using GPDM are smoother than those using GPLVM ". I have also implemented an efficient online motion sampling method, mean-prediction proposed in the paper to sample motions form human "walking" data.

## 1 Introduction

My area of interest lies in learning based control. I have come across papers where Gaussian processes are used for learning the dynamics model [3], [4]. In the given time frame it is difficult to implement any of these papers. To better my understanding of Gaussian process I have chosen to implement a part of the paper "Gaussian Process Dynamics Models" by Wang et.al.[6]. The paper proposes a dimension reduction method for high dimensional data. The authors call this reduced space as latent space. My primary goal for the project is to implement Gaussian Process Dynamics Model (GPDM) and Gaussian Process Latent Variable Methods (GPLVM) algorithms for dimension reduction of human motion capture data and test the claim that "latent maps obtained using GPDM are smoother than those obtained through GPLVM". Smoother latent maps imply that the samples of latent motion generated would result in good reconstruction of human motion poses. The application of this can be found in video-based people tracking and data- driven animation fields where several samples of motion should be effectively generated. An individual human-pose is typically parameterized by more than 60 parameters, but the activity-specific human pose and motion has smaller intrinsic dimensionality. In the paper Wang et.al use a 3-dimensional latent space. Though the problem I am trying to solve is not directly related to the domain I am interested in, this project provides me a deeper understanding about Gaussian process applications in general which will be helpful for my future endeavours.

## 2 Dimension Reduction Methods

### 2.1 Probabilistic Principal Component Analysis (PPCA)

Let $\mathbf{Y} = [\mathbf{y_1} \dots \mathbf{y_T}]^\top$ be $T$ observations each $J$-dimensional and let $\mathbf{X} = [\mathbf{x_1} \dots \mathbf{x_T}]^\top$ be $D$-dimensional latent variables. In PPCA the assumption is that the $J$-dimensional observations are

linear function of $D$-dimensional latent variables

$$\mathbf{y}_t = \mathbf{M}\mathbf{x}_t + \varepsilon_t \tag{1}$$

where $D < J$ and $\varepsilon_t \sim \mathcal{N}\left(0, \sigma^2\right)$. The linear map $\mathbf{M}$ is $J \times D$ matrix relating $\mathbf{Y}, \mathbf{X}$. It is assumed that $\mathbf{x}_t$ is distributed as $D$-variate Gaussian and each observation $\mathbf{y}_t$ is i.i.d., this implies that the marginal distribution of $\mathbf{y}_t$ is a Gaussian.

$$\begin{aligned} \mathbf{x}_t &\sim \mathcal{N}_D(\mathbf{0}, \mathbf{I}) \\ p\left(\mathbf{y}_t \mid \mathbf{M}\right) &= \mathcal{N}_J\left(\mathbf{0}, \mathbf{C}\right) \end{aligned} \tag{2}$$

where $\mathbf{C} = \mathbf{M}\mathbf{M}^\top + \sigma^2\mathbf{I}$. PPCA seeks to minimize the negative of log-likelihood $p\left(\mathbf{y}_t \mid \mathbf{M}\right)$ to obtain the map $\mathbf{M}$.

$$\begin{aligned} \mathcal{L}(\mathbf{M}) &= -\log p(\mathbf{Y} \mid \mathbf{M}) \\ &= -\sum_{t=1}^{T} \log p\left(\mathbf{y}_t \mid \mathbf{W}\right) \\ &= -\sum_{t=1}^{T} \left[\log\left(\frac{1}{\sqrt{(2\pi)^J |\mathbf{C}|}}\right) - \frac{1}{2}\mathbf{y}_t^\top \mathbf{C}^{-1}\mathbf{y}_t\right] \\ &= -\sum_{n=1}^{N} \left[-\frac{J}{2}\log 2\pi - \frac{1}{2}\log|\mathbf{C}| - \frac{1}{2}\mathbf{y}_t^\top \mathbf{C}^{-1}\mathbf{y}_t\right] \\ &= \frac{TJ}{2}\log 2\pi + \frac{T}{2}\log|\mathbf{C}| + \frac{1}{2}\sum_{t=1}^{T} \mathbf{y}_t^\top \mathbf{C}^{-1}\mathbf{y}_t \\ &= \frac{TJ}{2}\log 2\pi + \frac{T}{2}\log|\mathbf{C}| + \frac{1}{2}\sum_{t=1}^{T} \operatorname{tr}\left(\mathbf{C}^{-1}\mathbf{y}_t\mathbf{y}_t^\top\right) \end{aligned} \tag{3}$$

In the derivation of Equation 3 we have used the trace identity,

$$\mathbf{Z}^T\Sigma^{-1}\mathbf{Z} = \operatorname{tr}\left(\Sigma^{-1}\mathbf{Z}\mathbf{Z}^T\right) \tag{4}$$

### 2.2 Gaussian Process Latent Variable Method (GPLVM) [5]

GPLVM can be looked as a generalization of PPCA. In PPCA we have a linear map that relates $\mathbf{X}$ to $\mathbf{Y}$. But in case of GPLVM the map can be non-linear, thus making PPCA a special case of GPLVM. In case of PPCA we have integrated out the latent variable $\mathbf{X}$ and optimized for the map $\mathbf{M}$, but in GPLVM this process is reversed i.e., we integrate out $\mathbf{M}$ and optimize for the latent variables $\mathbf{X}$. We can re-write Equation 1 as,

$$\mathbf{y}_t = \mathbf{X}\mathbf{m}_t + \varepsilon_t \tag{5}$$

Here the assumption is that each row $\mathbf{m}_t$ is i.i.d Gaussian, which allows us to write,

$$p(\mathbf{M}) = \prod_{j=1}^{J} \mathcal{N}_D\left(\mathbf{0}, \beta^{-1}\mathbf{I}\right) \tag{6}$$

$$p\left(\mathbf{y}_t \mid \mathbf{X}\right) = \mathcal{N}_T\left(\mathbf{0}, \mathbf{K}\right) \tag{7}$$

where $\mathbf{K} = \beta^{-1}\mathbf{X}\mathbf{X}^\top + \sigma^2\mathbf{I}$ in case of a linear kernel. Lawrence et.al [5] use a radial bias function (RBF) kerenel in the paper,

$$k\left(\mathbf{x}, \mathbf{x}'; \boldsymbol{\beta}\right) = \beta_1 \exp\left(-\frac{\beta_2}{2}\left\|\mathbf{x} - \mathbf{x}'\right\|_2^2\right) + \beta_3^{-1}\delta_{\mathbf{x},\mathbf{x}'} \tag{8}$$

For inference I have also put a prior on the kernel hyperparameters $\bar{\boldsymbol{\beta}} = (\beta_1 \dots \beta_3)$, $\left(\bar{\boldsymbol{\beta}} \propto \prod_i \beta_i^{-1}\right)$ . So the solution of GPLVM reduces to minimizing the negative of log-likelihood $p\left(\mathbf{y}_t \mid \mathbf{X}, \bar{\boldsymbol{\beta}}\right)$

$$\begin{aligned}
\mathcal{L}(\mathbf{X}, \bar{\boldsymbol{\beta}}) &= -p\left(\mathbf{y}_t \mid \mathbf{X}, \bar{\beta}\right) p\left(\bar{\boldsymbol{\beta}}\right) \\
&= \frac{JT}{2} \log 2\pi + \frac{J}{2} \log |\mathbf{K}| + \frac{1}{2} \sum_{j=1}^{J} \mathbf{y}_j^\top \mathbf{K}^{-1} \mathbf{y}_j + \sum_i \log \beta_i \\
&= \frac{JT}{2} \log 2\pi + \frac{J}{2} \log |\mathbf{K}| + \frac{1}{2} \sum_{j=1}^{J} \operatorname{tr}\left(\mathbf{K}^{-1} \mathbf{y}_j \mathbf{y}_j^\top\right) + \sum_i \log \beta_i
\end{aligned} \tag{9}$$

In the derivation of Equation 9 we have used the trace identity in Equation 4.

### 2.3 Gaussian Process Dynamical Models (GPDM) [6]

GPDM can be viewed as a GPLVM where the latent variable evolves according to its own Gaussian process, i.e., the latent variable evolves according to a smooth non-linear dynamics and the mapping from latent to observation space is also a smooth non-linear map.

#### 2.3.1 Gaussian Process Dynamics

Let $\mathbf{Y} = [\mathbf{y_1} \dots \mathbf{y_T}]^\top$ be $T$ observations, each $J$-dimensional, and indexed by discrete time $t$, and let $\mathbf{X} = [\mathbf{x_1} \dots \mathbf{x_T}]^\top$ be $D$-dimensional latent variables. Consider the following Markov dynamics,

$$\begin{aligned}
\mathbf{x}_t &= \boldsymbol{f}\left(\mathbf{x}_{t-1}; \mathbf{A}\right) + \mathbf{n}_{x,t} \\
\mathbf{y}_t &= \boldsymbol{g}\left(\mathbf{x}_t; \mathbf{B}\right) + \mathbf{n}_{y,t}
\end{aligned} \tag{10}$$

where $\boldsymbol{f}$ and $\boldsymbol{g}$ are mappings with parameters $\mathbf{A}$ and $\mathbf{B}$ respectively, and $\mathbf{n}_{x,t}$ and $\mathbf{n}_{y,t}$ are zero mean Gaussian noise. The author's propose a particular case in which $f$ and $g$ are linear combination of basis functions $\boldsymbol{\phi}(\boldsymbol{x})$ and $\boldsymbol{\psi}(\boldsymbol{x})$ respectively.

$$\begin{aligned}
\boldsymbol{f}(\mathbf{x}; \mathbf{A}) &= \sum_{k=1}^{K} \mathbf{a}_k \boldsymbol{\phi_k}(\mathbf{x}) \\
\boldsymbol{g}(\mathbf{x}; \mathbf{B}) &= \sum_{m=1}^{M} \mathbf{b}_m \boldsymbol{\psi_m}(\mathbf{x})
\end{aligned} \tag{11}$$

where $\mathbf{A} = [\mathbf{a}_1 \mid \dots \mid \mathbf{a}_K]$ and $\mathbf{B} = [\mathbf{b}_1 \mid \dots \mid \mathbf{b}_M]$ are $D \times K$ and $J \times M$ vectors respectively and where $\boldsymbol{\phi}(\boldsymbol{x}) = [\phi_1(\mathbf{x}) \dots \phi_K(\mathbf{x})]^\top$ and $\boldsymbol{\psi}(\boldsymbol{x}) = [\psi_1(\mathbf{x}) \dots \psi_M(\mathbf{x})]^\top$ are $K \times 1$ and $M \times 1$ vectors.

#### 2.3.2 Integrating Weights

Wang's modeling assumption is that each row of $\mathbf{B}$ and $\mathbf{A}$ i.e $\boldsymbol{b_j}$ ans $\boldsymbol{a_d}$ are i.i.d Gaussian. This idea is used to integrate out $\mathbf{B}$ and $\mathbf{A}$ in Equation 10 to give us expressions for $p(\mathbf{Y} \mid \mathbf{X})$ and $p(\mathbf{X})$.
First, lets integrate out $\mathbf{B}$. The assumption is that each row of $\mathbf{B}$ has isotropic Gaussian prior,

$$p(\mathbf{B}) = \prod_{j=1}^{J} \mathcal{N}(\mathbf{b}_j \mid \mathbf{0}, w_j^{-2} \mathbf{I}) \tag{12}$$

for some variance $w_j^{-2}$. Rewriting the model in terms of features of Y we get,

$$\mathbf{y}_j = \boldsymbol{\Psi} \mathbf{b}_j + \mathbf{n}_{y,j} \qquad \mathbf{n}_{y,j} \sim \mathcal{N}(\mathbf{0}, w_j^{-2} \sigma_Y^2 \mathbf{I}) \tag{13}$$

where, $\boldsymbol{\Psi} = [\boldsymbol{\psi}(\mathbf{x}_1) \dots \boldsymbol{\psi}(\mathbf{x}_T)]^\top$, or a $T \times M$ matrix. This implies that the distribution on $\mathbf{y}_j$, conditioned on $\mathbf{b_j}$ is a $T-$ variate normal,

$$\mathbf{y}_j \mid \mathbf{X}, \mathbf{b}_j \sim \mathcal{N}_T\left(\mathbf{y}_j \mid \boldsymbol{\Psi} \mathbf{b}_j, w_j^{-2} \sigma_Y^2 \mathbf{I}\right) \tag{14}$$

As both $p(\mathbf{b}_j)$ and $p(\mathbf{y}_j \mid \mathbf{b}_j)$ are independently Gaussian, $\mathbf{b}_j$ can be marginalised out to obtain,

$$p\left(\mathbf{y}_j \mid \mathbf{X}\right) = \mathcal{N}_T\left(\mathbf{y}_j \mid \mathbf{0}, \mathbf{\Psi} w_j^{-2} \mathbf{\Psi}^\top + w_j^{-2}\sigma_Y^2 \mathbf{I}\right) \tag{15}$$

The covariance term in Equation 15 has the inner product term term $\mathbf{\Psi}\mathbf{\Psi}^\top$ that can be kernelized,

$$\mathbf{\Psi} w_j^{-2} \mathbf{\Psi}^\top + w_j^{-2}\sigma_Y^2 \mathbf{I} = w_j^{-2}\left(\mathbf{\Psi}\mathbf{\Psi}^\top + \sigma_Y^2 \mathbf{I}\right) = w_j^{-2}\mathbf{K}_Y \tag{16}$$

where $\mathbf{K}_Y = \mathbf{\Psi}\mathbf{\Psi}^\top + \sigma_Y^2\mathbf{I}$. This allows us to write the joint density over all $J$ features as,

$$\begin{aligned}
p(\mathbf{Y} \mid \mathbf{X}) &= \prod_{j=1}^{J} p\left(\mathbf{y}_j \mid \mathbf{X}\right) \\
&= \prod_{j=1}^{J} \frac{1}{\sqrt{(2\pi)^T \left|w_j^{-2}\mathbf{K}_Y\right|}} \exp\left\{-\frac{1}{2}\mathbf{y}_j^\top w_j^{-2}\mathbf{K}_Y^{-1}\mathbf{y}_j\right\} \\
&= \frac{(w_j)^{TJ}}{\sqrt{(2\pi)^{TJ}\left|\mathbf{K}_Y\right|^J}} \exp\left\{-\frac{1}{2}\sum_{j=1}^{J}\mathbf{y}_j^\top w_j^{-2}\mathbf{K}_Y^{-1}\mathbf{y}_j\right\} \\
&= \frac{(w_j)^{TJ}}{\sqrt{(2\pi)^{TJ}\left|\mathbf{K}_Y\right|^J}} \exp\left\{-\frac{1}{2}\operatorname{tr}\left(\sum_{j=1}^{J}\mathbf{K}_Y^{-1}\mathbf{y}_j\mathbf{y}_j^\top w_j^{-2}\right)\right\} \\
&= \frac{|\mathbf{W}|^T}{\sqrt{(2\pi)^{TJ}\left|\mathbf{K}_Y\right|^J}} \exp\left\{-\frac{1}{2}\operatorname{tr}\left(\mathbf{K}_Y^{-1}\mathbf{Y}\mathbf{W}^2\mathbf{Y}^\top\right)\right\}
\end{aligned} \tag{17}$$

where $\mathbf{W} = \operatorname{diag}\left([w_1 \dots w_J]\right)$. In the derivation of Equation 17 we have used the trace identity in Equation 4.

Now let's integrate out $\mathbf{A}$. The assumption is that each row of $\mathbf{A}$ has isotropic Gaussian prior,

$$p(\mathbf{A}) = \prod_{d=1}^{D} \mathcal{N}_K\left(\mathbf{a}_d \mid \mathbf{0}, \mathbf{I}\right) \tag{18}$$

Rewriting the model in terms of columns of X and rows of $\mathbf{A}$,

$$\begin{aligned}
\mathbf{x}_d^{(2:T)} &= \mathbf{\Phi}^{1:(T-1)}\mathbf{a}_d + \mathbf{n}_{x,d} \qquad \mathbf{n}_{x,d} \sim \mathcal{N}(\mathbf{0}, \sigma_X^2\mathbf{I}) \\
\mathbf{x}_d^{(2:T)} \mid \mathbf{X}, \mathbf{a}_d &\sim \mathcal{N}_{T-1}\left(\mathbf{x}_d^{(2:T)} \mid \mathbf{\Phi}^{1:(T-1)}\mathbf{a}_d, \sigma_X^2\mathbf{I}\right)
\end{aligned} \tag{19}$$

where $\mathbf{x}_d^{(2:T)} = \left[x_d^{(2)} \dots x_d^{(T)}\right]^\top$ and $\Phi^{1:(T-1)} = \left[\phi\left(\mathbf{x}_1\right) \dots \phi\left(\mathbf{x}_{T-1}\right)\right]^\top$. As both $p(\mathbf{a}_d)$ and $p(\mathbf{x}_d \mid \mathbf{a}_d)$ are independently Gaussian, $\mathbf{a}_d$ can be marginalised out to obtain.

$$p\left(\mathbf{x}_d^{(2:T)} \mid \mathbf{X}\right) = \mathcal{N}_{T-1}\left(\mathbf{x}_d^{(2:T)} \mid \mathbf{0}, \mathbf{\Phi}^{1:(T-1)}\left(\mathbf{\Phi}^{1:(T-1)}\right)^\top + \sigma_X^2\mathbf{I}\right) \tag{20}$$

The covariance term in Equation 20 contains the inner product terms and thus it can be kernelized,

$$\mathbf{K}_X = \mathbf{\Phi}^{1:(T-1)}\left(\mathbf{\Phi}^{1:(T-1)}\right)^\top + \sigma_X^2\mathbf{I} \tag{21}$$

This allows us to marginalize out $\mathbf{A}$ from the model,

$$\begin{aligned} p(\mathbf{X}) &= \int p(\mathbf{X} \mid \mathbf{A}) p(\mathbf{A}) \mathrm{d}\mathbf{A} \\ &= \int p\left(\mathbf{x}_1\right) \prod_{d=1}^{D} p\left(\mathbf{x}_d \mid \mathbf{A}\right) p(\mathbf{A}) \mathrm{d}\mathbf{A} \\ &= p\left(\mathbf{x}_1\right) \prod_{d=1}^{D} \frac{1}{\sqrt{(2\pi)^{(T-1)} \left|\mathbf{K}_X\right|}} \exp\left\{-\frac{1}{2} \mathbf{x}_d^\top \mathbf{K}_X^{-1} \mathbf{x}_d\right\} \\ &= p\left(\mathbf{x}_1\right) \frac{1}{\sqrt{(2\pi)^{D(T-1)} \left|\mathbf{K}_X\right|^D}} \exp\left\{-\frac{1}{2} \sum_{d=1}^{D} \mathbf{x}_d^\top \mathbf{K}_X^{-1} \mathbf{x}_d\right\} \\ &= p\left(\mathbf{x}_1\right) \frac{1}{\sqrt{(2\pi)^{D(T-1)} \left|\mathbf{K}_X\right|^D}} \exp\left\{-\frac{1}{2} \operatorname{tr}\left(\mathbf{K}_X^{-1} \overline{\mathbf{X}} \overline{\mathbf{X}}^\top\right)\right\} \end{aligned} \tag{22}$$

where $\overline{\mathbf{X}} = \mathbf{X}^{(2:T)}$. Thus this model implies a Gaussian process prior on both the latent dynamics and the latent-to-observation maps when $\mathbf{K}_X$ and $\mathbf{K}_Y$ are defined using positive definite kernels. Wang et.al. used a a radial bias function (RBF) kernel for the latent mapping ($\mathbf{K}_Y$), $\mathbf{X} \to \mathbf{Y}$ and linear + RBF kernel for the latent dynamics ($\mathbf{K}_X$).

$$\begin{aligned} k_Y\left(\mathbf{x}, \mathbf{x}'; \boldsymbol{\beta}\right) &= \beta_1 \exp\left(-\frac{\beta_2}{2} \left\|\mathbf{x} - \mathbf{x}'\right\|_2^2\right) + \beta_3^{-1} \delta_{\mathbf{x},\mathbf{x}'} \\ k_X\left(\mathbf{x}, \mathbf{x}'; \boldsymbol{\alpha}\right) &= \alpha_1 \exp\left(-\frac{\alpha_2}{2} \left\|\mathbf{x} - \mathbf{x}'\right\|_2^2\right) + \alpha_3 \mathbf{x}^\top \mathbf{x}' + \alpha_4^{-1} \delta_{\mathbf{x},\mathbf{x}'} \end{aligned} \tag{23}$$

### 2.3.3 Inference

Wang et.al placed priors on kernel hyper parameters

$$\begin{aligned} \bar{\boldsymbol{\alpha}} &= (\alpha_1 \dots \alpha_4) \\ \bar{\boldsymbol{\beta}} &= (\beta_1 \dots \beta_3) \\ p(\bar{\boldsymbol{\alpha}}) \propto \prod_i \alpha_i^{-1}, \bar{\boldsymbol{\beta}} &\propto \prod_i \beta_i^{-1} \end{aligned}$$

Now we use Bayesian inference to obtain the posterior $p(\mathbf{X}, \bar{\boldsymbol{\beta}}, \bar{\boldsymbol{\alpha}} \mid \mathbf{Y})$,

$$p(\mathbf{X}, \bar{\boldsymbol{\beta}}, \bar{\boldsymbol{\alpha}} \mid \mathbf{Y}) \propto p(\mathbf{Y} \mid \mathbf{X}, \bar{\boldsymbol{\beta}}) p(\mathbf{X} \mid \bar{\boldsymbol{\alpha}}) p(\bar{\boldsymbol{\beta}}) p(\bar{\boldsymbol{\alpha}}) \tag{24}$$

Learning the GPDM from measurements $\mathbf{Y}$ reduces to minimizing the negative log posterior,

$$\begin{aligned} \mathcal{L}(\mathbf{X}, \bar{\boldsymbol{\beta}}, \bar{\boldsymbol{\alpha}}) = -\log p(\mathbf{X}, \bar{\boldsymbol{\beta}}, \bar{\boldsymbol{\alpha}} \mid \mathbf{Y}) = &\frac{J}{2} \log \left|\mathbf{K}_Y\right| + \frac{1}{2} \operatorname{tr}\left(\mathbf{K}_Y^{-1} \mathbf{Y} \mathbf{W} \mathbf{Y}^\top\right) \\ &+ \frac{D}{2} \log \left|\mathbf{K}_X\right| + \frac{1}{2} \operatorname{tr}\left(\mathbf{K}_X^{-1} \overline{\mathbf{X}} \overline{\mathbf{X}}^\top\right) \\ &+ \sum_i \log \alpha_i + \sum_i \log \beta_i. \end{aligned} \tag{25}$$

It is to be noted that all the constant terms are omitted in Equation 25 as they won't affect the optimization process.

## 3 Mean Prediction Sequences

One of the goals of the paper is to generate new motions for 3D people tracking and computer animation. The authors have used Hybrid Monte Carlo (HMC) to draw samples $\tilde{\mathbf{X}}_{1:T}^{(j)} \sim p\left(\tilde{\mathbf{X}}_{1:T} \mid \mathbf{x}_0, \mathbf{X}, \mathbf{Y}, \bar{\alpha}\right)$. A total of $3 \times T$ elements should be sampled to construct a new motion

trajectory. As the process is computationally intensive the authors also proposed a simple online method to generate new motion, called mean prediction. In mean prediction, the next time step $\tilde{\mathbf{x}}_t$ is conditioned on $\tilde{\mathbf{x}}_{t-1}$

$$\begin{gathered}\tilde{\mathbf{x}}_t \sim \mathcal{N}\left(\mu_X\left(\tilde{\mathbf{x}}_{t-1}\right); \sigma_X^2\left(\tilde{\mathbf{x}}_{t-1}\right)\mathbf{I}\right) \\ \mu_X(\mathbf{x}) = \bar{\mathbf{X}}^T\mathbf{K}_X^{-1}\mathbf{k}_X(\mathbf{x}), \quad \sigma_X^2(\mathbf{x}) = k_X(\mathbf{x},\mathbf{x}) - \mathbf{k}_X(\mathbf{x})^T\mathbf{K}_X^{-1}\mathbf{k}_X(\mathbf{x})\end{gathered} \tag{26}$$

where $\mathbf{k}_X$ is a vector with $k_X(x, x_i)$ in the $i-$th entry and $\mathbf{x}_i$ is the $i-$th training example. This method sets the latent position at each time step to be the most-likely (mean) point given the previous step. The authors report that mean prediction often generates motion that are like fair samples drawn from HMC.

## 4 Implementation Details

I have used CMU human motion capture data to test the hypothesis that "latent maps obtained through GPDM are smoother than GPLVM". The data is captured by a Vicon motion capture system consisting of 12 infrared MX-40 cameras, each of which is capable of recording at 120 Hz with images of 4 megapixel resolution. Motions are captured in a working volume of approximately 3m x 8m. The capture subject wears 41 markers and a stylish black garment. The data will be generated using a MATLAB software mocgui [2] created by Feng Zhou [8]. The data is 62 dimensional (56 joint angles, 3 body pose and 3 position) time series and the latent space is 3 dimensional. Wang et.al. used 3 velocity measurements instead of position in their implementation. The are several subjects available on the CMU motion capture database, the authors did not specify the subject and the particular trail used in the paper, so I have used subject-02 data in the project. The analysis is performed for three motions namely "walking", "running" and "jumping", which are $62 \times 298$, $62 \times 173$ and $62 \times 483$ time-frames respectively. The data is sub-sampled every 4 frames for implementation to promote faster optimization and is mean subtracted to be consistent with the derivation in Equation 13. Let $\mathbf{Y}$ denote the mean subtracted and sub-sampled data. For all considered motions, the matrix $\mathbf{W}$ in Equation 25 is set as as diagonal matrix with inverse of standard deviation of time-series of each of the 62 features of $\mathbf{Y}$ on it's diagonal.

$$\mathbf{W} = \text{diag}\left(\left[\sigma_1^{-1} \dots \sigma_{62}^{-1}\right]\right)$$

Some features in the data are found have 0 standard deviation, since division by 0 is a problem the standard-deviation for those features is set to a very small number $10^{-15}$. The hyperparameters for the kernels in Equation 23 are initialized as follows for the optimization,

$$\left(\alpha_1, \alpha_2, \alpha_3, \frac{1}{\alpha_4}\right)_{\text{initial}} = \left(0.9, 1, 0.1, 10^{-3}\right) \tag{27}$$

$$\left(\beta_1, \beta_2, \frac{1}{\beta_3}\right)_{\text{initial}} = \left(1, 1, 10^{-3}\right) \tag{28}$$

These initial values were not provided by Wang et.al. in the reference paper. I have referred to a later work by Wang [7] and used the information provided in that work. The latent variables $\mathbf{X}$ which are $3 \times T$ where $T$ is the time-frames depending on the motion are initialized using PCA solution for the data $\mathbf{Y}$ from the MATLAB function "pca". So the total number of variables to be optimized are $3 \times T$ latent variables and 7 parameters corresponding to kernels in Equation 23.A MATLAB optimization routine "fminunc" with "quasi-newton method" selected is used to minimize Equation 9, Equation 25 to obtain the latent maps for GPLVM and GPDM respectively. It is to be noted that the hyperparameters for kernel in Equation 8 are initialized as given in Equation 28.

## 5 Results

In Figure 1,

- The latent map obtained from GPDM is found to be smoother than that obtained through GPLVM for "walking" motion. The arrows in the subplot indicate areas where GPLVM is

Table 1: Hyper-parameters obtained through optimization

| Motion | Method | Hyperparameters | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | $\alpha_1$ | $\alpha_2$ | $\alpha_3$ | $\frac{1}{\alpha_4}$ | $\beta_1$ | $\beta_2$ | $\frac{1}{\beta_3}$ |
| Walking | GPDM | 0.8997 | 1.002 | 0.09824 | 1.967e-8 | 1.094 | 1.098 | 0.3969 |
| | GPLVM | - | - | - | - | 1.225 | 1.231 | 3.953 |
| Running | GPDM | 0.9011 | 1.003 | 0.09898 | 1.584e-8 | 1.035 | 1.038 | 0.39 |
| | GPLVM | - | - | - | - | 1.046 | 1.051 | 3.429 |
| Jumping | GPDM | 0.9 | 1.0 | 0.09999 | 0.0001677 | 1.002 | 1.003 | 0.546 |
| | GPLVM | - | - | - | - | 1.336 | 1.336 | 4.252 |

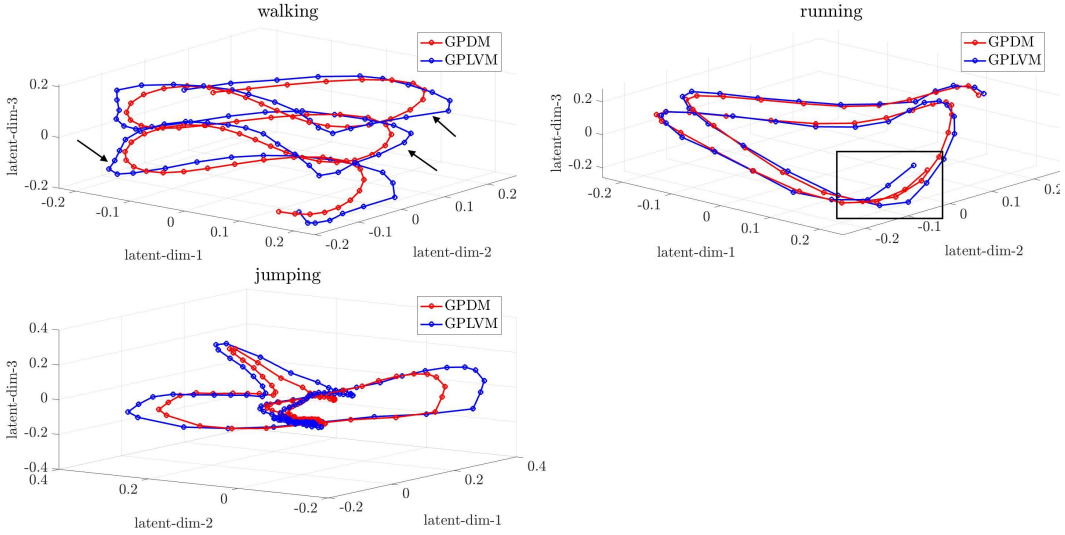


Figure 1: Latent maps for GPDM and GPLVM

non-smooth and the gap the between two consecutive latent points is large. Wang et.al [6] have also conducted their study on the "walking" data. The results obtained for this motion match with the pattern provided by Wang et.al in their paper.

- For the "running" and "jumping" motion the latent maps obtained from GPDM are found to be no better than GPLVM. In the "running" sub-plot the highlighted black box is where the GPLVM seems to have sharp transitions. For the "jumping" motion there is no visible difference in the maps indicating that GPLVM is good enough for few motions. The authors have not included these motions in their study so I don't have any reference to compare and evaluate my result.
- It was also found that the optimization is very sensitive to initial guess of hyperparameters, this behaviour was observed in project-2a also. So the initial guess for the hyperparameters is carefully set as mentioned in [7] for the "walking" motion. But for other motions like "jumping" and "running" the initial guess is not known, so even these motions are initialized with the hyperparameters used for "walking".

As I had a little more time, I have tried to implement the mean prediction sequences described in section 3 for one motion "walking" [Figure 2]. I have tried the same for "jumping" and "running" motions, but I was not able to get a good result, implying that the optimization for hyperparameters may not be correct for these motions. This may be because my optimization solver converges to a local minima and my initial guess for these motions might be off. I have tried using random initialization but as the parameters to optimize are many, $3 \times T + 7$, (3 latent dimensions, $T$ time

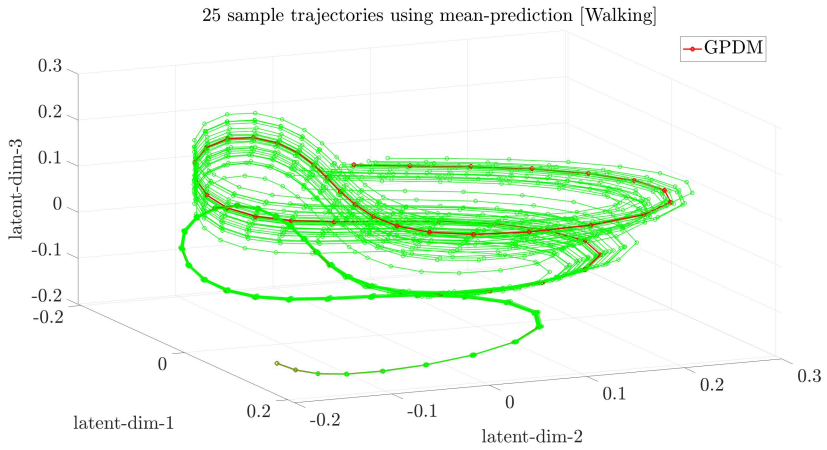


Figure 2: 25 sample trajectories generated using mean-prediction [Walking]

frames and 7 hyperparameters) the optimizer always gave a bad solution or never converged. So knowing a good initial guess might be crucial.

## 6 Conclusion

Through this project I have explored a new application of Gaussian process. In the course of the project I have found concepts like marginalisation, Bayesian inference, dynamical systems, maximisation of log-like-hood taught in class to be helpful for understanding math behind the method proposed in the paper. The results obtained from my implementation for "walking" motion are found to be inline with the trends reported in the paper. The study is also extented to "running" and "jumping" motions.

## Acknowledgements

I thank Gregory Gundersen whose blog post[1] helped me to understand math behind dimension reduction methods and structure the derivation in my project. I thank Prof. Alex Gorodetsky for being an amazing teacher.

## References

[1] Gregory gundersen. `https://gregorygundersen.com/blog/2020/07/24/gpdm/`.

[2] mocgui. `https://github.com/zhfe99/mocgui`.

[3] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472. Citeseer, 2011.

[4] Lukas Hewing, Juraj Kabzan, and Melanie N Zeilinger. Cautious model predictive control using gaussian process regression. *IEEE Transactions on Control Systems Technology*, 28(6):2736–2743, 2019.

[5] Neil D Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Nips*, volume 2, page 5. Citeseer, 2003.

[6] Jack M Wang, David J Fleet, and Aaron Hertzmann. Gaussian process dynamical models. In *NIPS*, volume 18, page 3. Citeseer, 2005.

[7] Jack M Wang, David J Fleet, and Aaron Hertzmann. Gaussian process dynamical models for human motion. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):283–298, 2007.

[8] Feng Zhou, Fernando De la Torre, and Jessica K. Hodgins. Hierarchical aligned cluster analysis for temporal clustering of human motion. *IEEE Transactions Pattern Analysis and Machine Intelligence (PAMI)*, 35(3):582–596, 2013.

Sai Satya Charan Malladi
charanm@umich.edu

# Project 4

AEROSP 567
Dec 16, 2021

# Code

## Pre-Process Data

```
% Sai Satya Charan Malladi
% AEROSP 567 Fall 21
% Final Project

% data_loader.m
% file to pre-process data

%% Begin

% load the walking data of subject 02
load('02_02_moc.mat')
data_dense = wsMoc.Dof;

% sample every 4 frames to generate sparse data
sample_rate = 4;
data_sparse = data_dense(:,1:sample_rate:length(data_dense));

% subtract mean from the sparse data
data_sparse = data_sparse - mean(data_sparse,2);
save('data_walking_sparse.mat','data_sparse')


% load the running data of subject 02
load('02_03_moc.mat')
data_dense = wsMoc.Dof;

% sample every 4 frames to generate sparse data
sample_rate = 4;
data_sparse = data_dense(:,1:sample_rate:length(data_dense));

% subtract mean from the sparse data
data_sparse = data_sparse - mean(data_sparse,2);
save('data_running_sparse.mat','data_sparse')


% load the jumping data of subject 02
load('02_04_moc.mat')
data_dense = wsMoc.Dof;

% sample every 4 frames to generate sparse data
sample_rate = 4;
data_sparse = data_dense(:,1:sample_rate:length(data_dense));

% subtract mean from the sparse data
data_sparse = data_sparse - mean(data_sparse,2);
save('data_jumping_sparse.mat','data_sparse')
```

## RBF Kernel

```
function val = rbf_kernel(x,xp,beta1,beta2,beta3)
% rbf kernel

    val =  beta1*exp(-beta2/2*norm(x-xp)^2);
    if x == xp
        val = val + beta3;
    end

end
```

## Linear+RBF kernel

```
function val = linear_rbf_kernel(x,xp,alpha1,alpha2,alpha3,alpha4)
% linear + rbf kernel

    val =  alpha1*exp(-alpha2/2*norm(x-xp)^2) + alpha3*x*xp';
    if x == xp
        val = val + alpha4;
    end

end
```

## GPLVM

```
function cost = gplvm_objective(z,Y,Xdim,Ydim,W,betalen,kernel_Y)
%

% unpack z
X = reshape(z(1:end-betalen,1),Xdim);
beta = z(end-betalen+1:end);

%
time_steps = Xdim(1);
obs_dim = Ydim(2);
K_Y = zeros(time_steps);

for ii = 1:time_steps
    for jj = 1:time_steps
        K_Y(ii,jj) = kernel_Y(X(ii,:),X(jj,:),beta(1),beta(2),beta(3));
    end
end

% K_Y = K_Y + beta(3)*eye(time_steps);
det_term = obs_dim/2*logdet(K_Y);
trace_term = 1/2*trace(K_Y\Y*Y');
log_likelihood = det_term + trace_term + sum(log(beta));

cost = (log_likelihood);

```

```
end
```

## GPDM

```
function cost = gpdm_objective(z,Y,Xdim,Ydim,W,alphalen,betalen,...
                                    kernel_X,kernel_Y)
%

% unpack z
X = reshape(z(1:end-(alphalen+betalen),1),Xdim);
alpha = z(end-(alphalen+betalen)+1:  end-betalen);
beta = z(end-betalen+1:end);

%
Xout = X(2:end,:);
time_steps = Xdim(1);
lnt_dim = Xdim(2);
obs_dim = Ydim(2);

K_Y = zeros(time_steps);
K_X = zeros(time_steps-1);

for ii = 1:time_steps
    for jj = 1:time_steps
        K_Y(ii,jj) = kernel_Y(X(ii,:),X(jj,:),beta(1),beta(2),beta(3));
        if ii ≠ time_steps && jj ≠ time_steps
            K_X(ii,jj) = ...
               kernel_X(X(ii,:),X(jj,:),alpha(1),alpha(2),alpha(3),alpha(4));
        end
    end
end

% K_Y = K_Y + beta(3)*eye(time_steps);
det_term = obs_dim/2*logdet(K_Y);
trace_term = 1/2*trace(K_Y\Y*W^2*Y');
log_likelihood = det_term + trace_term;

% K_X = K_X + alpha(4)*eye(time_steps-1);
det_term = lnt_dim/2*logdet(K_X);
trace_term = 1/2*trace(K_X\Xout*Xout');
log_prior =  det_term + trace_term + sum(log(alpha)) + sum(log(beta));

cost = (log_prior+log_likelihood);

end
```

## Main

```
% Sai Satya Charan Malladi
```

```
% AEROSP 567 Fall 21
% Final Project

% main.m
% GPLVM and GDPM on the data

clc; clear all; close all;

%% Begin

X_pca = cell(3,1);
X_gpdm = cell(3,1);
alpha_gpdm = cell(3,1);
beta_gpdm = cell(3,1);
X_gplvm = cell(3,1);
beta_gplvm = cell(3,1);

% % kernels
kernel_X = @(x,xp,alpha1,alpha2,alpha3,alpha4) ...
    linear_rbf_kernel(x,xp,alpha1,alpha2,alpha3,alpha4);
kernel_Y = @(x,xp,beta1,beta2,beta3) rbf_kernel(x,xp,beta1,beta2,beta3);

for ii = 1:3
    %%%% load data
    switch ii
        case 1
            load('data_walking_sparse');
            motion = 'walking';
        case 2
            load('data_running_sparse');
            motion = 'running';
        case 3
            load('data_jumping_sparse');
            motion = 'jumping';
    end

    %%%% PCA Initialization
    latent_dim = 3;
    coeff_pca = pca(data_sparse);
    Y = data_sparse';
    Ydim = size(Y);
    vararr = var(Y);
    vararr(vararr == 0) = 1e-15;
    W = diag(1./sqrt(vararr));
    % W = diag(sqrt(vararr));

    X_pca{ii} = coeff_pca(:,1:3);
    X0 = X_pca{ii};
    Xdim = size(X0);

    % initialize
    alpha0 = [0.9; 1; 0.1; 1/exp(1)];
    alphalen = length(alpha0);
    beta0 = [1; 1; 1/exp(1)];
    betalen = length(beta0);

```

```
57      %%%% GDPM
58      % inital guess
59      z0 = [X0(:); alpha0; beta0];
60      % test
61      test_gdpm = ...
            gpdm_objective(z0,Y,Xdim,Ydim,W,alphalen,betalen,kernel_X,kernel_Y);
62      options = ...
            optimoptions('fminunc','Algorithm','quasi-newton','Display','iter','MaxFunEvals',15
63  %     options = optimoptions('fminunc','Display','iter','MaxFunEvals',15e4);
64      z_gpdm = fminunc(@(z) gpdm_objective(z,Y,Xdim,Ydim,W,alphalen,betalen,...
65                                                  kernel_X,kernel_Y),z0,options);
66      X_gpdm{ii} = reshape(z_gpdm(1:end-(alphalen+betalen),1),Xdim);
67      alpha_gpdm{ii} = z_gpdm(end-(alphalen+betalen)+1:  end-betalen);
68      beta_gpdm{ii} = z_gpdm(end-betalen+1:end);
69
70      %%%% GPLVM
71      % inital guess
72      z0 = [X0(:); beta0];
73      % test
74      test_gplvm = gplvm_objective(z0,Y,Xdim,Ydim,W,betalen,kernel_Y);
75      options = ...
            optimoptions('fminunc','Algorithm','quasi-newton','Display','iter','MaxFunEvals',15
76  %     options = optimoptions('fminunc','Display','iter','MaxFunEvals',15e4);
77      z_gplvm = fminunc(@(z) ...
            gplvm_objective(z,Y,Xdim,Ydim,W,betalen,kernel_Y),z0,options);
78      X_gplvm{ii} = reshape(z_gplvm(1:end-betalen,1),Xdim);
79      beta_gplvm{ii} = z_gplvm(end-betalen+1:end);
80
81  end
82
83
84  % %% plot setup
85  load('optim_result.mat')
86  figa = figure('Position', get(0, 'Screensize'));
87  figatile = tiledlayout(2,3,'TileSpacing','tight','Padding','tight');
88
89  % plot
90  for ii = 1:3
91      switch ii
92          case 1
93              motion = 'walking';
94          case 2
95              motion = 'running';
96          case 3
97              motion = 'jumping';
98      end
99
100     figure(figa)
101     nexttile(ii)
102     plot3(X_gpdm{ii}(:,1), X_gpdm{ii}(:,2), ...
            X_gpdm{ii}(:,3),'ro-','LineWidth',2,'DisplayName','GPDM')
103 %     hold on
104 %     plot3(X_gplvm{ii}(:,1), X_gplvm{ii}(:,2), ...
        X_gplvm{ii}(:,3),'bo-','LineWidth',2,'DisplayName','GPLVM')
105 %     plot3(X_pca{ii}(:,1), X_pca{ii}(:,2), ...
        X_pca{ii}(:,3),'ko-','LineWidth',2,'DisplayName','PCA')
```

```
    set(gca,'FontSize',20)
    set(gca,'TickLabelInterpreter','latex');
    xlabel('latent-dim-1','fontsize',20,'interpreter','latex')
    ylabel('latent-dim-2','fontsize',20,'interpreter','latex')
    zlabel('latent-dim-3','fontsize',20,'interpreter','latex')
    title(motion,'fontsize',25,'interpreter','latex')
    legend('location','best','fontsize',20,'interpreter','latex')
    grid on

    figure(figa)
    nexttile(ii+3)
%     plot3(X_gpdm{ii}(:,1), X_gpdm{ii}(:,2), ...
    X_gpdm{ii}(:,3),'ro-','LineWidth',2,'DisplayName','GPDM')
%     hold on
    plot3(X_gplvm{ii}(:,1), X_gplvm{ii}(:,2), ...
        X_gplvm{ii}(:,3),'bo-','LineWidth',2,'DisplayName','GPLVM')
%     plot3(X_pca{ii}(:,1), X_pca{ii}(:,2), ...
    X_pca{ii}(:,3),'ko-','LineWidth',2,'DisplayName','PCA')
    set(gca,'FontSize',20)
    set(gca,'TickLabelInterpreter','latex');
    xlabel('latent-dim-1','fontsize',20,'interpreter','latex')
    ylabel('latent-dim-2','fontsize',20,'interpreter','latex')
    zlabel('latent-dim-3','fontsize',20,'interpreter','latex')
    title(motion,'fontsize',25,'interpreter','latex')
    legend('location','best','fontsize',20,'interpreter','latex')
    grid on

 end


%% mean prediction sequences
load('optim_result.mat')

% just for walking
X = X_gpdm{1};
alpha = alpha_gpdm{1};

%
Xout = X(2:end,:);
Xdim = size(X);
time_steps = Xdim(1);
lnt_dim = Xdim(2);

% compute K_X
K_X = zeros(time_steps-1);
for ii = 1:time_steps
    for jj = 1:time_steps
        if ii ≠ time_steps && jj ≠ time_steps
            K_X(ii,jj) = ...
                kernel_X(X(ii,:),X(jj,:),alpha(1),alpha(2),alpha(3),alpha(4));
        end
    end
end


% sample sequences
```

```
num_sequences = 25;
sample_sequence = zeros(time_steps,lnt_dim,num_sequences);


for ii = 1:num_sequences
    %
    x_tt =  X(1,:);
    sample_sequence(1,:,ii) = x_tt;

    % get k_x
    for tt = 2:time_steps
        k_x = zeros(time_steps-1,1);
        for jj = 1:time_steps-1
            k_x(jj,1) = ...
               kernel_X(x_tt,X(jj,:),alpha(1),alpha(2),alpha(3),alpha(4));
        end

         % calculate mu and sigma
        mu_tt = Xout'/K_X*k_x;
        sigma2_tt = kernel_X(x_tt,x_tt,alpha(1),alpha(2),alpha(3),alpha(4)) ...
           - k_x'/K_X*k_x;

        % next state
        x_tt = (mu_tt + sqrt(sigma2_tt)*randn(lnt_dim,1))';
        sample_sequence(tt,:,ii) = x_tt;
    end
end

% plot
figb = figure('Position', get(0, 'Screensize'));

% plot
figure(figb)
plot3(X(:,1), X(:,2), X(:,3),'ro-','LineWidth',2,'DisplayName','GPDM')
hold on
for ii = 1:num_sequences
    plot3(sample_sequence(:,1,ii), sample_sequence(:,2,ii), ...
        sample_sequence(:,3,ii),'go-','LineWidth',1,'HandleVisibility','off')
end
set(gca,'FontSize',30)
set(gca,'TickLabelInterpreter','latex');
xlabel('latent-dim-1','fontsize',30,'interpreter','latex')
ylabel('latent-dim-2','fontsize',30,'interpreter','latex')
zlabel('latent-dim-3','fontsize',30,'interpreter','latex')
title('25 sample trajectories using mean-prediction ...
    [Walking]','fontsize',30,'interpreter','latex')
legend('location','best','fontsize',30,'interpreter','latex')
grid on
```