Sequential Inference for Deep Gaussian Process

Yali Wang, Marcus Brubaker, Brahim Chaib-draa, Raquel Urtasun

Chinese Academy of Sciences
University of Toronto
Laval University

AISTATS 2016

Presented by Lei Yu

Sept 9, 2016
An efficient **sequential** inference framework for **Deep Gaussian Process (DGP)** is proposed.

Two DGP extensions:

- **heteroscedasticity** (non-stationary noise)
- **multi-task** learning.
Motivation

- GPs can not handle with non-stationary functions, e.g. functions with sudden jumps
- Variants of GP to address non-stationarity:
  - warped Gaussian Process [1]
  - product models [2]
  - Deep GPs
- Inference of DGP:
  - Variational approximations are commonly used for inference
  - Matrix inversion involved in inference leads to $O(N^3)$ complexity
Sequential Inference for Deep GP
Deep Gaussian Process

The output is assumed to be multi-dimensional, $y \in \mathbb{R}^{D_y}$ and to be generated as follows:

$$
\begin{align*}
    h_0 &= x \\
    h_i &= f_i(h_{i-1}) + v_i, \ i = 1, \ldots, L \\
    y &= f_y(h_L) + v_y
\end{align*}
$$

with

$$
    f_i \sim \mathcal{GP}(0, k_{\theta_i}) \quad f_y \sim \mathcal{GP}(0, k_{\theta_y})
$$

(a) Deep GP

Figure 1: The graphical model for DGP.
Sequential Inference for Deep GP

Sequential inference means, for each training pair \((x^n, y^n)\), the inference of latent variables is composed of two steps:

- **State Estimation:**
  - *sequential Monte Carlo* [3] for the latent states \(S^n = \{h^n_{1:L}\}\)

- **Model Updating:**
  - *sparse online GP* (GP\(_{so}\)) [4] or *Kernel Recursive Least-Square method* [5] for model parameters \(M = \{M_1, ..., M_L, M_y\}\), with

\[ M^{n-1} \equiv \{ \tilde{X}^{n-1}, \mu^{n-1}, \Sigma^{n-1}, Q^{n-1} \} \]

Active training set  Inverse covariance matrix of \(X\)  Posterior over \(X\): \(N(\mu, \Sigma)\)
State Estimation

The latent states $S^n$ are estimated via the posterior expectation

$$\mathbb{E}[S^n] = \int S^n p(S^n|x^n, y^n, M^{n-1}, \Theta_{dgp})dS^n$$

- intractable due to the deep architecture

**Sequential Monte Carlo:** for $k = 1, \ldots, N_p$

1. *sampling* $k$th sample (particle) of $S^n$ from current DGP model

   $$h^n_1(k) \sim p(h^n_1|x^n, M^{n-1}_1, \Theta_{dgp})$$
   $$h^n_i(k) \sim p(h^n_i|h^n_{i-1}(k), M^{n-1}_i, \Theta_{dgp}) \quad i = 2, \ldots, L$$

2. *weighting*

   $$\hat{h}_i^n = \sum_{k=1}^{N_p} \hat{w}^n(k)h^n_i(k), \text{ with } \hat{w}^n(k) = \frac{w^n(k)}{\sum_{k=1}^{N_p} w^n(k)}$$

   where $w^n(k)$ is determined by likelihood in the observation layer

   $$w^n(k) = p(y^n|h^n_L(k), M^{n-1}_y, \Theta_{dgp})$$
Model Updating

Once the states of $i$-th layer are ready, the input-output training data $(h^n_{i-1}, h^n_i)$ for single GP is available and online updating from $M^n_{i-1}$ to $M^n_i$ is required (layer index is omit)

\[
M^{n-1} \triangleq \{\hat{X}^{n-1}, \mu^{n-1}, \Sigma^{n-1}, Q^{n-1}\}
\]

\[
M^n \triangleq \{\hat{X}^n, \mu^n, \Sigma^n, Q^n\}
\]

And the task is fulfilled in two steps:

1. updating the posterior distribution $p(f^n|\tilde{X}^n)$ based on

\[
p(f^{n-1}|\tilde{X}^{n-1}) = \mathcal{N}(f^{n-1}|\mu^{n-1}, \Sigma^{n-1})
\]

2. updating the active set of training data $\tilde{X}$.

*Kernel Recursive Least-Square method is exploited [5]*
KRLS: Updating posterior (1)

When $n$-th data is available, update $\tilde{X}_l = \tilde{X}_{l-1} \cup (h^n_{l-1}, h^n_l)$.
Then according to Bayes rule:

$$p(f^n|\tilde{X}^n) = p(f^{n-1}, f^n|\tilde{X}^{n-1}, h^n) = \frac{p(h^n|f^n)p(f^n|f^{n-1})p(f^{n-1}|\tilde{X}^{n-1})}{p(h^n|\tilde{X}^{n-1})}$$

- Observation prior

$$p(h^n|f^n) = \mathcal{N}(h^n|f^n, \sigma^2)$$

- GP and Conditioned GP prior

$$p(f^{n-1}|\tilde{X}^{n-1}) = \mathcal{N}(f^{n-1}|\mu^{n-1}, \Sigma^{n-1}), \quad p(f^n|f^{n-1}) = \mathcal{N}(f^n|q_n^T f^{n-1}, \gamma_n^2)$$

- Evidence

$$p(h^n|\tilde{X}^{n-1}) = \int p(h^n|f^n)p(f^n|f^{n-1})p(f^{n-1}|\tilde{X}^{n-1})df^{n-1}df^n$$

$$= \mathcal{N}(h^n|q_n^T f^{n-1}, \sigma^2 + \gamma_n^2 + q_n^T \Sigma^{n-1} q_n)$$
KRLS: Updating posterior (2)

\[ p(f^n | \tilde{X}^n) = \mathcal{N}(f^n | \mu^n, \Sigma^n) \] (1)

where

\[ \mu^n = \left[ \begin{array}{c} \mu^{n-1} \\ q_n \mu^{n-1} \end{array} \right] + \frac{h^n - q_n^T \mu^{n-1}}{\eta_n^2} \psi_n \]

\[ \Sigma^n = \left[ \begin{array}{cc} \Sigma^{n-1} & \Sigma^{n-1} q_n \\ q_n^T \Sigma^{n-1} & \gamma_n^2 + q_n^T \Sigma^{n-1} q_n \end{array} \right] - \frac{\psi_n \psi_n^T}{\eta_n^2} \]

\[ Q^n = \left[ \begin{array}{cc} Q^{n-1} & 0 \\ 0 & 0 \end{array} \right] + \frac{1}{\gamma_n^2} \left[ \begin{array}{c} q_n \\ -1 \end{array} \right] \left[ \begin{array}{c} q_n \\ -1 \end{array} \right]^T \]

\[ q_n = Q^{n-1} k_{n-1}(h^n) \]

\[ \gamma_n^2 = k(h^n, h^n) - k_{n-1}^T(h^n) q_n \]

\[ \psi_n = \left[ \begin{array}{c} \Sigma^{n-1} q_n \\ \gamma_n^2 + q_n^T \Sigma^{n-1} q_n \end{array} \right] \]
KRLS: Updating active dataset without info loss

- Consider the posterior of $f^n$ given the data set $X^n$,

$$p(f^n|X^n) = p(f^{n-1}, f^n|X^n) = p(f^{n-1}|X^n) p(f^n|f^{n-1}, X^n)$$ related to $f^n$

- $f^n$ can be removed $\Leftrightarrow$ $f^n$ is non-informative, i.e.

$$p(f^n|f^{n-1}, X^n) = \frac{p(y^n|f^n)}{p(y^n|f^{n-1})} p(f^n|f^{n-1}) = p(f^n|f^{n-1})$$

$$p(y^n|f^n) \sim \mathcal{N}(f^n, \sigma^2 I) \quad p(y^n|f^{n-1}) \sim \mathcal{N}(q_n^T f^{n-1}, (\sigma^2 + \gamma^2)I)$$

- the second equation verified when $\gamma^2_n = 0$ and $f^n = q_n^T f^{n-1}$.
- If $\gamma^2_n = 0$, basis $f^n$ can be removed without any loss of information, and update active set $\hat{X}_l^n = \hat{X}_l^n \setminus (h^n_{l-1}, h^n_l)$
KRLS: Updating active dataset with info loss

Once the amount of bases in the dictionary grows larger than a predefined budget $N_{AC}$, remove one basis from the dictionary.

- Considering $i$-th basis, removing $f^n_i$ will effect the approximation results, which is measured by the KL distance

$$\text{KL}(p(f^n|X^n)||p([f^{n-1}]_{-i}|X^n) p(f^n_i|[f^{n-1}]_{-i}))$$

- for the exact distribution, mean of $f^n$ is $\mu^n$
- for the approximate distribution, mean of $[f^{n-1}]_{-i}$ remains unchanged, i.e. $[\mu^n]_{-i}$, while mean of $f^n_i$ is $q^T_n [\mu^n]_{-i}$

$$\mathcal{E}_i = \mu^n_i - q^T_n [\mu^n]_{-i} = \frac{[Q^n \mu^n]_i}{Q^n_{i,i}}$$

- Select the removed basis via

$$k = \arg \min_i (\mathcal{E}_i^2)$$
KRLS: Re-Updating after Removing Basis

- Model parameters should be updated simply via

$$
\begin{align*}
\mu^n &\leftarrow [\mu^n]_{-k} \\
\Sigma^n &\leftarrow [\Sigma^n]_{-k,-k} \\
Q^n &\leftarrow [Q^n]_{-k,-k} - \frac{[Q^n]_{-k,k}[Q^n]^T_{-k,k}}{[Q^n]_{k,k}}
\end{align*}
$$

- Update active set $$\tilde{X}_n = \tilde{X}_n \setminus (h_{l-1}^k, h_l^k)$$
Overall algorithm

**Algorithm 1** Sequential inference for DGP

1: **Input:**
2: The $n$-th data pair: $(x^n, y^n)$
3: The DGP model parameters at the $(n-1)$-th step: $M^{n-1} = \{M_1^{n-1}, \ldots, M_L^{n-1}, y^{n-1}\}$
4: **Output:**
5: Estimates of the latent state at the $n$-th step: $\hat{S}^n = \{\hat{\mathbf{h}}^n_{1:L}\}$
6: DGP model parameters at the $n$-th step: $M^n = \{M^n_1, \ldots, M^n_L, y^n\}$
7: **State Estimation:**
8: Draw $N_p$ samples of $S^n$ from (7) & (8)
9: Weight samples by (9) & (11)
10: Estimate $\hat{S}^n$ by (10)
11: **Model Update:**
12: Update $M^{n-1}$ to $M^n$ by using GP $\hat{s}_o$ with $(x^n, \hat{\mathbf{h}}^n_1), \ldots, (\hat{\mathbf{h}}^n_{L-1}, \hat{\mathbf{h}}^n_L), (\hat{\mathbf{h}}^n_L, y^n)$.

- For $n$-th training data, the complexity (including sampling and model updating) is $O(N_{AC}^2)$
- The overall complexity is $O(N_{AC}^2 N)$, which is much less than that of full GP $O(N^3)$. 
Prediction and Hyperparameter Learning

- Given a trained DGP model $M = \{M_1, ..., M_L, M_y\}$ and a new input $x^*$,
  
  \[
  h_1^*(k) \sim p(h_1^*|x^*, M_1, \Theta_{dgp}) \\
  h_i^*(k) \sim p(h_i^*|h_{i-1}^*(k), M_i, \Theta_{dgp})
  \]

- Predictive distribution of $y^*$ is Gaussian mixture
  
  \[
  \hat{y}^* \sim \frac{1}{N_p} \sum_{k=1}^{N_p} p(y^*|h_L^*(k), M_y, \Theta_{dgp})
  \]

- Hyperparameter $\Theta_{dgp}$ for each layer can be learned by minimizing the negative log likelihood.
Sequential Inference for Deep GP

Sequential DGP Extensions

Sequential DGP Extensions
Heteroscedastic Learning

Homoscedastic noise, i.e. $\sigma_y$ is a constant

$$y = f_y(h_L) + v_y$$

Heteroscedastic noise, i.e. $\sigma_y$ is dependent to input

$$s_\alpha = f_\alpha(h_L) + v_\alpha, \ s_\beta = f_\beta(h_L) + v_\beta$$

$$p(f|s_\alpha, s_\beta) = \mathcal{N}(s_\alpha, \exp(s_\beta))$$

Figure 2: Heteroscedastic extension of DGP.
Multi-Task Learning

- Deep network structure of DGP allows correlations between outputs to be represented by sharing the latent layers
- **Multi-Task learning** is straightforward in sequential inference scheme,
  - Unnormalized weight is computed by using only observed dimensions of the output $y_{\text{obs}}^n$
    
    $$ w^n(k) = p(y_{\text{obs}}^n | h_L^k, M_y^{n-1}, \Theta_{dgp}) $$
  
  - Missing dimensions are estimated using the same procedure as latent states.
  - Model parameters $M_y^n$ are estimated using both observed and estimated outputs.
Further possible extensions

A novel and efficient inference framework for DGP is proposed. Further possible extensions might be:

1. Exploiting deep learning techniques such as ReLU activation functions and dropout in sided the framework

2. Sequential DGP with other network structures e.g. CNNs, RNNs, LSTMs.
Experiments
Experiments

• ARD kernal:

\[ k(x, x') = \sigma^2_{ker} \exp \left[ -0.5 \sum_{i=1}^{d} c_i (x_i - x'_i)^2 \right] \]

• Evaluation:
  - Root mean squared error (RMSE)
  - mean negative log probability (MNLP)
  - Average training time (second)

• Abbreviations:
  - Sequential DGP - DGP_{seq}
  - Heteroscedastic sequential DGP - HDGP_{seq}
  - Variational inference DGP - DGP_{var}
  - Sparse online GP - GP_{so}
DGP as Deep Dynamical Prior: unsupervised learning

Input is the time step and output is the multi-dimensional observation.

- Datasets (Time sequences): motion, flu and stock
- Training size 1500/300/1000 for motion/flu/stock datasets
- Layers $L = 2$
- Latent dimension $D = 5$
- Number of samples $N_p = 100$
- Budget size $N_{AC} = 100/100/200$ respectively for 3 datasets
Performance w.r.t. Latent Layer Number $L$

Figure 1: DGP as Deep Dynamical Prior (Robustness to Complexity of the Deep Model): RMSE/MNLP/Training Time (Column 1/2/3) as a function of the number of latent layers $L$. Row1/2/3: motion/flu/stock. The error bar is mean±standard deviation.
Performance w.r.t. Layer Dimensionality $D$

Figure 2: DGP as Deep Dynamical Prior (Robustness to Complexity of the Deep Model): RMSE/MNLP/Training Time (Column 1/2/3) as a function of the dimensionality of latent layers $D$. Row1/2/3: motion/flu/stock. The error bar is mean±standard deviation.
Performance w.r.t. Number of Particles $N_p$

Figure 3: DGP as Deep Dynamical Prior (Properties of Our Inference Framework): RMSE/MNLP/Training Time (Column 1/2/3) as a function of the number of particles $N_p$. Row1/2/3: motion/flu/stock. The error bar is mean±standard deviation. Note that there is no $N_p$ in GP$_{so}$, hence GP$_{so}$ is a straight line for the plot of $N_p$. 


Performance w.r.t. Size of Active Set $N_{AC}$

Figure 4: DGP as Deep Dynamical Prior (Properties of Our Inference Framework): RMSE/MNLP/Training Time (Column 1/2/3) as a function of the size of active set $N_{AC}$. Row1/2/3: motion/flu/stock. The error bar is mean±standard deviation.
Qualitative Results on *motion* dataset

Figure 3: DGP as Deep Dynamical Prior (the motion dataset). Particle Estimation of \( y \).

\[
\sum_{N_p=1}^{100} \text{Weights of Particles} \times \text{Particles} = \text{Estimation}_{140} \\
\sum_{N_p=1}^{100} \text{Weights of Particles} \times \text{Particles} = \text{Estimation}_{700} \\
\sum_{N_p=1}^{100} \text{Weights of Particles} \times \text{Particles} = \text{Estimation}_{1200} \\
\sum_{N_p=1}^{100} \text{Weights of Particles} \times \text{Particles} = \text{Estimation}_{1570} \\
\sum_{N_p=1}^{100} \text{Weights of Particles} \times \text{Particles} = \text{Estimation}_{2070}
\]
Dimensionality Reduction

Low-dimensional latent representations on *Frey* face dataset: input and output are the noisy face images.

![Graph showing reconstruction error as a function of the number of episodes.](image)

**Figure 8:** Dimensionality Reduction for Image Reconstruction (Face Data): the reconstruction error as a function of the number of episodes. Note that there is no sequential inference & hyperparameter learning episode in DGP\(_{\text{var}}\). The results of DGP\(_{\text{var}}\) are lines. The error bar is mean ± standard deviation.

**Figure 9:** Dimensionality Reduction for Image Reconstruction (Face Data): the reconstruction image comparison at the time step 50:50:1900.

**Figure 10:** Heteroscedastic GP Modeling (Posterior of Motorcycle Data). In Plot10(a), the observations are black crosses, the mean of our HGP\(_{\text{seq}}\)/GP are pink / blue lines, 95% confidence interval of our HGP\(_{\text{seq}}\)/GP are pink / blue dashed lines.
Heteroscedastic Learning Result on Motercycle

Dynamical prior estimation on *Motorcycle* dataset: 94 time/acceleration pairs.

![Graphs showing comparison between methods](image)

<table>
<thead>
<tr>
<th>Methods</th>
<th>RMSE</th>
<th>MNLP</th>
<th>Train. T(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GP</td>
<td>22.81±3.96</td>
<td>9.22±0.34</td>
<td>0.24±0.07</td>
</tr>
<tr>
<td>NGP</td>
<td>22.80±3.95</td>
<td>9.22±0.33</td>
<td>0.45±0.09</td>
</tr>
<tr>
<td>WGP</td>
<td>22.88±4.18</td>
<td>8.73±0.90</td>
<td>1.79±0.19</td>
</tr>
<tr>
<td>MLHGP</td>
<td>22.89±3.60</td>
<td>8.44±1.07</td>
<td>1.03±0.22</td>
</tr>
<tr>
<td>VHGP</td>
<td>22.88±3.76</td>
<td>8.45±0.41</td>
<td>1.95±0.28</td>
</tr>
<tr>
<td>our HGP$_{seq}$</td>
<td>22.79±3.98</td>
<td>8.25±0.52</td>
<td>0.93±0.05</td>
</tr>
</tbody>
</table>

Table 4: Comparison with State-of-the-art Heteroscedastic GP Approaches (Motorcycle Data).
Missing Data in Multi-task Learning (1)

Dynamical prior estimation on motion, flu and stock (small portion of observations is missing)

<table>
<thead>
<tr>
<th>Data</th>
<th>Methods</th>
<th>RMSE(%)</th>
<th>MNLP</th>
<th>Train_T(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motion</td>
<td>GP_{so}</td>
<td>1.45</td>
<td>-4.81</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>our DGP_{seq}</td>
<td>1.0±0.1</td>
<td>-4.95±0.75</td>
<td>202±2</td>
</tr>
<tr>
<td></td>
<td>our HDGP_{seq}</td>
<td>1.1±0.3</td>
<td>-5.15±0.20</td>
<td>236±8</td>
</tr>
<tr>
<td>Flu</td>
<td>GP_{so}</td>
<td>5.3</td>
<td>-2.42</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>our DGP_{seq}</td>
<td>4.0±0.6</td>
<td>-3.09±0.32</td>
<td>50±2</td>
</tr>
<tr>
<td></td>
<td>our HDGP_{seq}</td>
<td>3.0±0.7</td>
<td>-3.16±0.11</td>
<td>108±6</td>
</tr>
<tr>
<td>Stock</td>
<td>GP_{so}</td>
<td>6.4</td>
<td>-1.95</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>our DGP_{seq}</td>
<td>6.4±0.1</td>
<td>-2.18±0.07</td>
<td>305±2</td>
</tr>
<tr>
<td></td>
<td>our HDGP_{seq}</td>
<td>6.6±0.3</td>
<td>-2.41±0.04</td>
<td>413±4</td>
</tr>
</tbody>
</table>

Table 1: Missing Data in Multi-Task Learning (Motion/Flu/Stock).
Missing Data in Multi-task Learning (2)

Dynamical prior estimation on *Jura* 359/100 total/missing dataset, with input 2D location and output the measurement of cadmium, nickel and zinc concentrations. ¹

<table>
<thead>
<tr>
<th>Methods</th>
<th>MAE</th>
<th>Train_T(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GP</td>
<td>0.5739±0.0003</td>
<td>74</td>
</tr>
<tr>
<td>CMOGP</td>
<td>0.4552±0.0013</td>
<td>784</td>
</tr>
<tr>
<td>SLFM</td>
<td>0.4578±0.0025</td>
<td>792</td>
</tr>
<tr>
<td>GPRN</td>
<td>0.4040±0.0006</td>
<td>1040</td>
</tr>
<tr>
<td>GPRN-NPV1</td>
<td>0.4147±0.0001</td>
<td>130</td>
</tr>
<tr>
<td>our DGP&lt;sub&gt;seq&lt;/sub&gt;</td>
<td>0.4150±0.0061</td>
<td>21</td>
</tr>
</tbody>
</table>

Table 5: Comparison with State-of-the-art Multi-task GPs (the Jura dataset). The results of GP, CMOGP, SLFM, GPRN are from [12].

¹MAE - Mean absolute error; CMOGP - convolved multiple outputs GP; SLFM - semiparametric latent factor model; GPRN - GP regression networks; GPRN-NPV1 - GPRN with nonparametric variational inference (mode 1).
Regresion on *Parkinsons* elemonitoring dataset, a 6 months biomedical voice recording from 42 people with early-stage Parkinson’s disease for a total of 5875 input/output pairs.

- 5000 for training and 875 for testing
- **Parameter setting:** $L = 2$, $D = 2$, $N_p = 500$, $N_{AC} = 100$

<table>
<thead>
<tr>
<th>Data</th>
<th>Methods</th>
<th>RMSE</th>
<th>MNLP</th>
<th>Train T(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parkinsons</td>
<td>GP$_{so}$</td>
<td>9.33 ± 0.15</td>
<td>7.45 ± 0.02</td>
<td>37 ± 2</td>
</tr>
<tr>
<td></td>
<td>DGP$_{var}$</td>
<td>9.26 ± 0.41</td>
<td>8.51 ± 1.35</td>
<td>33486 ± 1370</td>
</tr>
<tr>
<td></td>
<td>our DGP$_{seq}$</td>
<td><strong>8.86 ± 0.23</strong></td>
<td><strong>7.23 ± 0.08</strong></td>
<td>546 ± 18</td>
</tr>
</tbody>
</table>

Table 2: DGP as Regressor (Parkinsons Data): Prediction Error & Training Efficiency.
Classification on *Tumor* gene expression dataset and *MNIST* digits dataset. ²

- **Tumor**: $L = 3, D = 12, N_p = 200, N_{AC} = 144$
- **MNIST**: $L = 1, D = 400, N_p = 100, N_{AC} = 1000$

<table>
<thead>
<tr>
<th>Data</th>
<th>Methods</th>
<th>Acc</th>
<th>Train T(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tumor</strong></td>
<td>GP&lt;sub&gt;so&lt;/sub&gt;</td>
<td>0.65</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>DGP&lt;sub&gt;var&lt;/sub&gt;</td>
<td>0.50</td>
<td>2256</td>
</tr>
<tr>
<td></td>
<td>our DGP&lt;sub&gt;seq&lt;/sub&gt;</td>
<td><strong>0.73</strong></td>
<td>53</td>
</tr>
<tr>
<td><strong>MNIST</strong></td>
<td>GP&lt;sub&gt;so&lt;/sub&gt;</td>
<td>0.9500</td>
<td>60mins</td>
</tr>
<tr>
<td></td>
<td>DGP&lt;sub&gt;var&lt;/sub&gt;</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>DVI-GPLVM</td>
<td>0.9405</td>
<td>20mins*</td>
</tr>
<tr>
<td></td>
<td>our DGP&lt;sub&gt;seq&lt;/sub&gt;</td>
<td>0.9424</td>
<td>180mins</td>
</tr>
</tbody>
</table>

²DVI-GPLVM - GPLVM with distributed variational inference.
References