Efficient Variational Gaussian Processes Initialization via Kernel-based Least Squares Fitting

Xinran Zhu\textsuperscript{1}, Jacob R. Gardner\textsuperscript{2}, David Bindel\textsuperscript{1}
\textsuperscript{1}Cornell University, (xz584, bindel)@cornell.edu
\textsuperscript{2}University of Pennsylvania, jacobrg@seas.upenn.edu

Abstract

Stochastic variational Gaussian processes (SVGP) scale Gaussian process inference to large datasets through inducing points and stochastic training. However, the training process involves hard multimodal optimization, and often suffers from slow and suboptimal convergence when initializing inducing points directly from training data. We provide a better initialization of inducing points from kernel-based least squares fitting. We show empirically that our approach consistently reaches better prediction performance. The total time cost of our method, including initialization, is comparable to the standard SVGP training.

1 Introduction

Gaussian processes (GP) are a popular probabilistic learning framework, especially when inference with uncertainty estimation is necessary \cite{12, 17, 26, 27, 31}. However, exact GP inference costs $O(n^3)$ in computation and $O(n^2)$ in storage for $n$ training points. Stochastic variational Gaussian processes (SVGP) have recently shown success in scaling up GP inference on large datasets \cite{8}; the fundamental idea lies in variational GP inference \cite{28}. Variational GPs introduce a small set of inducing points and corresponding inducing values as a "fake" training dataset, and assume that the inducing data is sufficient for inference. The variational GP model is then trained by minimizing the distance between the true GP posterior and the approximate GP posterior, which involves optimizing the variational Evidence Lower Bound (ELBO) \cite{9, 10}. Furthermore, the SVGP model decomposes the log likelihood term of ELBO into a sum over training labels, thereby enabling stochastic optimization. However, the stochastic training of variational ELBO, which contains model hyperparameters, inducing parameters, and variational distribution parameters, involves hard multimodal optimization problems which have many local minima, and often suffers from slow and suboptimal convergence.

We propose a better initialization method of the inducing points, and get a better initialization of the variational parameters as well as kernel hyperparameters as by-products. The key observation is that the kernel-based least squares function approximation shares the same mean predictor formulation as variational GPs. In the kernel-based least squares function approximation setting, the regularized least squares error can be viewed as a function of the inducing points via variable projection. Therefore, optimizing inducing points boils down to solving a nonlinear least squares problem, which can be solved by standard numerical methods. We demonstrate the efficiency of our initialization in SVGP training by comparing to standard initialization methods, such as random subsampling and K-means initialization, on a variety of real datasets. With our initialization, we show better prediction performance consistently on various real datasets. The total time cost, including initialization computation, is comparable to the standard SVGP training.
2 Background

A Gaussian process (GP) is a distribution over function values, denoted as \( f \sim \mathcal{GP}(\mu(x), k(x, x')) \), where \( \mu \) is the mean function and \( k \) is the covariance function \([22]\). We assume familiarity with GPs and briefly introduce them for notational clarity. Given data points \( X = \{x_i\}_{i=1}^n \) and function observations \( f = \{f(x_i)\} \in \mathbb{R}^n \), a GP prior assumes a multivariate normal distribution \( f \sim \mathcal{N}(\mu_X, K_{XX}) \), where \( \mu_X, K_{XX} \) are the mean values and covariance matrix at data \( X \). Conditioning on noisy observations \( y \) = \( f + \epsilon \), where \( \epsilon \sim \mathcal{N}(0, \sigma^2 I) \), the posterior distribution at a new data point \( x^* \) is \( p(f^*[y]) = \mathcal{N}(\mu(x^*), \Sigma^*) \), where \( \mu(x^*) = \mu(x^*) + K_{x^*x}(K_{XX} + \sigma^2 I)^{-1}(y - \mu_X) \) and \( \Sigma^* = K(x^*, x^*) - K_{x^*x}(K_{XX} + \sigma^2 I)^{-1}K_{xx}^* \). If we assume a prior mean of zero, the posterior mean \( \mu^* \) becomes \( \mu^* = K_{x^*x}(K_{XX} + \sigma^2 I)^{-1}y = K_{x^*x}c = \sum_{i=1}^n c_i k(x^*, x_i) \), where \( c \) = \( (K_{XX} + \sigma^2 I)^{-1}y \). We generally estimate model hyperparameters such as kernel lengthscale \( l \) and noise \( \sigma \) by Maximum Likelihood. The log marginal likelihood function \([22]\) can be optimized by standard numerical solvers such as LBFGS \([20]\) with a cost of \( \mathcal{O}(\mu^3) \) flops for each evaluation.

2.1 Stochastic Variational GP (SVGP)

Variational GPs improve scalability of standard GPs by introducing a set of inducing points \( U = \{u_i\}_{i=1}^m \) with associated inducing values \( g = \{g_i\}_{i=1}^m \) to represent function values at \( U \) under the same GP prior assumption \([23, 9, 8]\). Therefore, the inference at a new point \( x^* \) becomes \( p(f^*[y]) = \int p(f^*[g])p(g|y)dg \). If we assume the inducing data is sufficient for inference, i.e., \( f^* \) and \( f \) are independent conditioning on \( g \), we have \( p(f^*[y]) \approx \int p(f^*[g])p(g|y)dg \). Further assuming a variational distribution \( q(g) \sim \mathcal{N}(\mu, \Sigma) \) that approximates the posterior \( p(g|y) \), the inference for the function value \( f^* \) at \( x^* \) is

\[
p(f^*[y]) \approx \int p(f^*[g])q(g|y)dg \text{ and KL}[q(g)||p(g)],
\]

where \( q(f(x_i)) , \sigma(f(x_i)^2) \) are the predicted mean and variance from Eq.\([1]\) respectively. To achieve heteroscedastic modeling and improve predictive variances, the Parametric Gaussian Process Regressor (PPGPR) \([11]\) proposes a similar stochastic ELBO loss for SVGP:

\[
\begin{align*}
\text{ELBO}_{\text{SVGP}} &= \sum_{i=1}^n \left\{ \log \mathcal{N}(y_i|\mu_f(x_i), \sigma_f(x_i)^2) - \frac{\sigma_f(x_i)^2}{2\sigma^2} \right\} - \text{KL}[q(g)||p(g)],
\end{align*}
\]

We empirically apply our initialization method on both ELBO_{PPGPR} (SVGP) and ELBO_{PPGPR} (PPGPR) training in Section\([4]\). Given inducing point locations, the analytical optimal variational distribution can be obtained by differentiating Eq.\([2]\)\([23]\):

\[
q^*(g) = \mathcal{N}(\bar{m}, \bar{S}) = \mathcal{N}(\bar{m}, \bar{S}) = \mathcal{N}(\sigma^{-2} K_{UU} \Sigma K_{UX} y, \bar{K}_{UU} \Sigma K_{UU}).
\]

3 Kernel-based Least Squares

Given data points \( X = \{x_i\}_{i=1}^n \in \mathbb{R}^{n \times d} \) and noisy observations \( y = \{y_i\} \in \mathbb{R}^n \), a kernel-based function approximation can be formed as \( s(x) = \sum_{i=1}^m c_i k(x, u_i) \) \([30]\), where \( k(\cdot, \cdot) \) is a positive definite kernel function, and the centers of the kernel basis functions \( U = \{u_i\}_{i=1}^m \) is a smaller set of points than the training data \( X \) \([11, 30]\), same as inducing points. For a given set of inducing points \( U \), the coefficients \( c \) are solved from a least squares problem:

\[
\min_{c \in \mathbb{R}^m} \mathcal{L}(c) = \min_{c \in \mathbb{R}^m} \{ \|K_U c - y\|^2 + \sigma^2 \|c\|^2_{K_{UU}} \},
\]
We consider eight univariate regression UCI datasets [5], with the number of training data ranging from 11250 to 47706 and input dimensions ranging from 8 to 20. We use 500 or 800 inducing points based on training data size. We use an Adam optimizer [13] and separately tune the learning rate for each method on each dataset.

Results were averaged over 10 random train/test/validation splits. As shown in Table 1, SVGP-lm yields significantly lower RMSE than all baselines on all datasets, and so does PPGPR-lm. As
Table 1: Testing RMSE on eight univariate regression datasets (lower is better). Results are averaged over 10 random train/validation/test splits.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Pol</th>
<th>Elevators</th>
<th>Bike</th>
<th>Kin40k</th>
<th>Protein</th>
<th>Keggdir</th>
<th>Slice</th>
<th>Keggundir</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVGP-random</td>
<td>0.3009</td>
<td>0.3806</td>
<td>0.4421</td>
<td>0.1853</td>
<td>0.5168</td>
<td>0.08832</td>
<td>0.1412</td>
<td>0.1218</td>
</tr>
<tr>
<td>SVGP-kmeans</td>
<td>0.2995</td>
<td>0.3823</td>
<td>0.4403</td>
<td>0.1836</td>
<td>0.514</td>
<td>0.0886</td>
<td>0.1405</td>
<td>0.1217</td>
</tr>
<tr>
<td>SVGP-pivchol</td>
<td>0.3012</td>
<td>0.3808</td>
<td>0.4393</td>
<td>0.1802</td>
<td>0.5177</td>
<td>0.08872</td>
<td>0.1412</td>
<td>0.1217</td>
</tr>
<tr>
<td>SVGP-lm</td>
<td><strong>0.2187</strong></td>
<td><strong>0.3703</strong></td>
<td><strong>0.3336</strong></td>
<td><strong>0.1526</strong></td>
<td><strong>0.486</strong></td>
<td>0.08909</td>
<td><strong>0.1239</strong></td>
<td><strong>0.12</strong></td>
</tr>
<tr>
<td>PPGPR-random</td>
<td>0.3201</td>
<td>0.3931</td>
<td>0.6243</td>
<td>0.2886</td>
<td>0.5102</td>
<td>0.09304</td>
<td>0.229</td>
<td>0.1249</td>
</tr>
<tr>
<td>PPGPR-kmeans</td>
<td>0.3229</td>
<td>0.3925</td>
<td>0.6246</td>
<td>0.286</td>
<td>0.5108</td>
<td><strong>0.08997</strong></td>
<td>0.2287</td>
<td>0.1245</td>
</tr>
<tr>
<td>PPGPR-pivchol</td>
<td>0.3323</td>
<td>0.393</td>
<td>0.6286</td>
<td>0.2892</td>
<td>0.5208</td>
<td>0.09009</td>
<td>0.234</td>
<td>0.1248</td>
</tr>
<tr>
<td>PPGPR-lm</td>
<td><strong>0.2947</strong></td>
<td><strong>0.3747</strong></td>
<td><strong>0.4499</strong></td>
<td><strong>0.2441</strong></td>
<td><strong>0.4907</strong></td>
<td>0.09014</td>
<td><strong>0.2123</strong></td>
<td><strong>0.124</strong></td>
</tr>
</tbody>
</table>

Table 2: Testing NLL on eight univariate regression datasets (lower is better). Results are averaged over 10 random train/validation/test splits.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Pol</th>
<th>Elevators</th>
<th>Bike</th>
<th>Kin40k</th>
<th>Protein</th>
<th>Keggdir</th>
<th>Slice</th>
<th>Keggundir</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVGP-random</td>
<td>0.2971</td>
<td>0.4538</td>
<td>0.6112</td>
<td>-0.1945</td>
<td>0.7652</td>
<td>-1.028</td>
<td>-0.4666</td>
<td>-0.6884</td>
</tr>
<tr>
<td>SVGP-kmeans</td>
<td>0.2869</td>
<td>0.4577</td>
<td>0.607</td>
<td>-0.197</td>
<td>0.7584</td>
<td>-1.026</td>
<td>-0.4743</td>
<td>-0.6869</td>
</tr>
<tr>
<td>SVGP-pivchol</td>
<td>0.2875</td>
<td>0.4538</td>
<td>0.6039</td>
<td>-0.229</td>
<td>0.7639</td>
<td>-1.025</td>
<td>-0.4692</td>
<td>-0.6894</td>
</tr>
<tr>
<td>SVGP-lm</td>
<td><strong>0.1211</strong></td>
<td><strong>0.4289</strong></td>
<td><strong>0.358</strong></td>
<td><strong>-0.3675</strong></td>
<td><strong>0.7959</strong></td>
<td><strong>-1.031</strong></td>
<td><strong>-0.4843</strong></td>
<td><strong>-0.6986</strong></td>
</tr>
<tr>
<td>PPGPR-random</td>
<td>-0.1474</td>
<td>0.3785</td>
<td>-0.5554</td>
<td>-0.8402</td>
<td><strong>0.5654</strong></td>
<td>-1.628</td>
<td>-1.126</td>
<td>-1.899</td>
</tr>
<tr>
<td>PPGPR-kmeans</td>
<td>-0.1596</td>
<td>0.3778</td>
<td>-0.5517</td>
<td>-0.8395</td>
<td>0.5578</td>
<td><strong>-1.642</strong></td>
<td><strong>-1.132</strong></td>
<td><strong>-1.908</strong></td>
</tr>
<tr>
<td>PPGPR-pivchol</td>
<td><strong>-0.1629</strong></td>
<td>0.3745</td>
<td>-0.5194</td>
<td><strong>-0.8515</strong></td>
<td>0.5777</td>
<td>-1.633</td>
<td>-1.113</td>
<td>-1.897</td>
</tr>
<tr>
<td>PPGPR-lm</td>
<td>-0.1384</td>
<td><strong>0.3572</strong></td>
<td><strong>-0.6666</strong></td>
<td>-0.8178</td>
<td>0.5784</td>
<td>-1.617</td>
<td>-1.111</td>
<td>-1.902</td>
</tr>
</tbody>
</table>

shown in Table 2. SVGP-lm consistently shows the best NLL on seven out of eight datasets; while PPGPR-lm only shows comparable results compared to baselines. This shows that our initialization helps finding a better local minimum for the multimodal optimization problem during training. The total time cost of SVGP-lm (including LM and training) is comparable to the standard SVGP training with other baseline initialization methods. See Appendix C for more details such as error bars on regression results, time cost comparison and ablation study.

5 Related Work

The selection and optimization of inducing points is one of the most crucial and hardest parts in variational GP training. The original work of Titsias [28] treated the inducing points as variational parameters to avoid overfitting. Many works have explored better ways of optimizing inducing points during the training. Rossi et al. [25] treated inducing points as model parameters with priors and posteriors and proposed a fully Bayesian treatment of both inducing points and GP hyperparameters. Similarly, Uhrenholt et al. [29] placed a point process prior on the inducing points to select a good number of inducing points in a probabilistic way. Burt et al. [2] provided an asymptotic bound on the KL divergence, which helps with selecting number of inducing points. Our work, combining the ideas of kernel-based least squares fitting and variational GP inference, however, is orthogonal to those works and can be applied to any training methods as an initialization.

6 Conclusion

SVGP enables GP inference on large datasets by combining inducing points methods and stochastic training. However, the stochastic ELBO loss function is highly multimodal with many local minima and training often suffers from slow and suboptimal convergence. Our initialization through kernel-based least squares fitting solves for a good set of inducing points, and yields good initial values for the variational parameters as well as kernel hyperparameters as a by-product. Empirically, our initialization helps SVGP and PPGPR find better local minima and obtain better prediction performance under a comparable total time cost. Our initialization approach can be combined with any other training improvement of SVGP.
References

A Details of the Initialization Method

In this section, we discuss other details of our initialization method, including the initialization of the variational covariance matrix as a by-product and treatment of the noise variance parameter.

A.1 Uncertainty estimation

One of the strengths of GPs is the uncertainty estimation, which the kernel-based function approximation is not capable of. However, a given set of inducing points $U$ and kernel hyperparameters $\theta$ are sufficient to compute the optimal variational covariance $\tilde{S}$ in Eq. 3 as a by-product:

$$\tilde{S} = K_{UU}\Sigma K_{UU},$$

where $\Sigma = (K_{UU} + \sigma^{-2}K_{UX}K_{XU})^{-1}$. Unlike the variational mean, which could then guarantee an equally good mean prediction of variational GP, there is no guarantee of how the $\tilde{S}$ would perform in uncertainty estimation using a good set of $U$ and $\theta$ from kernel-base approximation. However, in our empirical study, initializing the variational covariance by the computed optimal covariance $\tilde{S}$ helps reduce both the training and testing NLL a lot. So we also incorporate the computed $\tilde{S}$ in our initialization scheme.

A.2 The $\sigma$ parameter

The $\sigma$ parameter plays an important role in matching the mean predictor. In the kernel-based least squares function approximation setting, the $\sigma$ is a regularization parameter in the least squares problem in Eq. 5. While in the variational GP setting, the $\sigma$ parameter stands for the standard deviation of the observation noise $\epsilon$, where the noise $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$ is assumed to be Gaussian. Though mathematically equivalent in the two mean predictor formulations, the treatment of the $\sigma$ parameter is usually different in these two settings. In a least squares setting, the regularization parameter $\sigma$ is not solved within the minimization problem in Eq. 5 but there are various standard ways of selecting a good value for regularization parameter $\sigma$. While in a GP setting, the noise variance $\sigma^2$ is usually optimized together with all model hyperparameters such as kernel hyperparameters $\theta$ via Maximum Likelihood [23]. In our initialization, we use fixed $\sigma$ in the kernel-based least squares fitting setting, and then use the same $\sigma$ as an initialization of the variational GP.

B Jacobian of the projected problem

In this section we derive the Jacobian $J_{r}(p)$ of the residual $r(p)$ in the projected problem

$$\minimize_{p} \mathcal{L}_p(p) = \minimize_{p} \| (I - AA^\dagger) \hat{y} \|^2 := \| r(p) \|^2.$$

Note that $r(p) = (I - AA^\dagger) \hat{y}$, where only the matrix $A$ contains $p$, so it boils down to finding the Jacobian of the residual projector $P = I - AA^\dagger$ with respect to $p$. 


Figure 1: A performance comparison of using different number of LM iterations on the Elevators dataset. LM-X indicates that the initialization uses X LM iteration.

Table 3: Standard error of testing RMSE, averaged over 10 random train/validation/test splits.

<table>
<thead>
<tr>
<th></th>
<th>Pol (1e-3)</th>
<th>Elevators (1e-3)</th>
<th>Bike (1e-3)</th>
<th>Kin40k (1e-3)</th>
<th>Protein (1e-3)</th>
<th>Keggdir (1e-3)</th>
<th>Slice (1e-3)</th>
<th>Keggundir (1e-3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVGP-random</td>
<td>1.39</td>
<td>1.59</td>
<td>3.47</td>
<td>1.63</td>
<td>1.27</td>
<td>1.01</td>
<td>1.42</td>
<td>0.87</td>
</tr>
<tr>
<td>SVGP-kmeans</td>
<td>1.30</td>
<td>1.74</td>
<td>3.42</td>
<td>1.93</td>
<td>1.49</td>
<td>0.97</td>
<td>1.25</td>
<td>0.90</td>
</tr>
<tr>
<td>SVGP-pivchol</td>
<td>1.62</td>
<td>1.68</td>
<td>3.13</td>
<td>1.67</td>
<td>1.40</td>
<td>0.98</td>
<td>1.40</td>
<td>0.89</td>
</tr>
<tr>
<td>SVGP-lm</td>
<td>1.56</td>
<td>1.89</td>
<td>2.90</td>
<td>1.15</td>
<td>1.06</td>
<td>1.00</td>
<td>0.96</td>
<td></td>
</tr>
<tr>
<td>PPGPR-random</td>
<td>2.13</td>
<td>2.13</td>
<td>4.37</td>
<td>4.90</td>
<td>1.20</td>
<td>1.32</td>
<td>1.81</td>
<td>0.89</td>
</tr>
<tr>
<td>PPGPR-kmeans</td>
<td>2.22</td>
<td>2.17</td>
<td>4.59</td>
<td>4.71</td>
<td>1.35</td>
<td>0.96</td>
<td>3.10</td>
<td>0.91</td>
</tr>
<tr>
<td>PPGPR-pivchol</td>
<td>2.44</td>
<td>2.20</td>
<td>4.36</td>
<td>5.02</td>
<td>1.39</td>
<td>0.90</td>
<td>1.94</td>
<td>0.95</td>
</tr>
<tr>
<td>PPGPR-lm</td>
<td>4.17</td>
<td>2.23</td>
<td>12.0</td>
<td>0.61</td>
<td>1.29</td>
<td>8.85</td>
<td>2.05</td>
<td>0.88</td>
</tr>
</tbody>
</table>

Here we use the variational notation to denote derivatives of matrices. For example, $\delta S$ represent infinitesimal changes to a vector (or matrix) $S$, and the Jacobian of $S$ can be readily obtained from $\delta S$ by applying $\delta S$ to each single dimension of the variables that we differentiate with respect to.

Using the definition of pseudoinverse, we can derive

$$
\delta P = - (\delta A) A^\dagger - (A^\dagger)^T (\delta A)^T + (A^\dagger)^T (A^\dagger A + (\delta A)^T A) A^\dagger
= - P(\delta A) A^\dagger - (A^\dagger)^T (\delta A)^T P,
$$

Therefore, for the residual we have

$$
\delta r = (\delta A) \bar{y} = - P(\delta A) c - (A^\dagger)^T (\delta A)^T r_0,
$$

where $c = A^\dagger \bar{y}$ and $r_0 = \bar{y} - Ac$. Given the economy QR decomposition of $A = QR$ and observe that $(A^\dagger)^T = QR^{-T}$ we have

$$
\delta r = ( I - QQ^T )(\delta A) c - QR^{-T} (\delta A)^T r_0
= -(\delta A) c + QQ^T (\delta A) c - QR^{-T} (\delta A)^T r_0.
$$

C Experiment Details

Regression Details In this section, we provide more detailed results of our empirical evaluation. Table 3 includes standard errors of RMSE results, and Table 4 includes standard errors of NLL results. Table 5 presents the total time cost comparison on SVGP. The time cost of SVGP-lm includes both LM initialization and the subsequent SVGP training. As shown in Table 5, the total time cost of SVGP-lm is comparable to, and sometimes less than, the standard SVGP-random training. We only compare SVGP-lm with SVGP-random because: a) Other baselines share similar time cost as SVGP-random and b) PPGPR shares similar time cost as the SVGP counterparts.
Table 4: Standard error of testing NLL, averaged over 10 random train/validation/test splits.

<table>
<thead>
<tr>
<th></th>
<th>Pol (1e-3)</th>
<th>Elevators (1e-3)</th>
<th>Bike (1e-3)</th>
<th>Kin40k (1e-3)</th>
<th>Protein (1e-3)</th>
<th>Kegdirdir (1e-3)</th>
<th>Slice (1e-3)</th>
<th>Keggundir (1e-3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVGP-random</td>
<td>3.11</td>
<td>3.90</td>
<td>6.59</td>
<td>0.39</td>
<td>2.27</td>
<td>8.09</td>
<td>7.14</td>
<td>6.29</td>
</tr>
<tr>
<td>SVGP-kmeans</td>
<td>2.89</td>
<td>4.43</td>
<td>6.50</td>
<td>3.89</td>
<td>2.57</td>
<td>9.35</td>
<td>6.22</td>
<td>6.60</td>
</tr>
<tr>
<td>SVGP-piv chol</td>
<td>3.61</td>
<td>4.30</td>
<td>6.08</td>
<td>3.82</td>
<td>2.48</td>
<td>9.24</td>
<td>7.02</td>
<td>6.62</td>
</tr>
<tr>
<td>SVGP-lm</td>
<td>7.21</td>
<td>4.68</td>
<td>18.5</td>
<td>3.57</td>
<td>4.13</td>
<td>9.21</td>
<td>4.67</td>
<td>6.18</td>
</tr>
<tr>
<td>PPGPR-random</td>
<td>4.49</td>
<td>5.83</td>
<td>12.8</td>
<td>11.8</td>
<td>7.26</td>
<td>29.4</td>
<td>2.89</td>
<td>19.9</td>
</tr>
<tr>
<td>PPGPR-kmeans</td>
<td>4.65</td>
<td>6.22</td>
<td>10.4</td>
<td>14.1</td>
<td>6.55</td>
<td>21.5</td>
<td>2.71</td>
<td>16.2</td>
</tr>
<tr>
<td>PPGPR-piv chol</td>
<td>5.74</td>
<td>5.64</td>
<td>12.0</td>
<td>5.78</td>
<td>6.56</td>
<td>16.1</td>
<td>3.20</td>
<td>22.1</td>
</tr>
<tr>
<td>PPGPR-lm</td>
<td>9.88</td>
<td>5.05</td>
<td>23.2</td>
<td>2.92</td>
<td>3.00</td>
<td>21.7</td>
<td>3.14</td>
<td>27.9</td>
</tr>
</tbody>
</table>

Table 5: Total time cost (min) comparison on SVGP. PPGPR has similar time cost results.

<table>
<thead>
<tr>
<th></th>
<th>Pol</th>
<th>Elevators</th>
<th>Bike</th>
<th>Kin40k</th>
<th>Protein</th>
<th>Kegdirdir</th>
<th>Slice</th>
<th>Keggundir</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVGP-random</td>
<td>3.5</td>
<td>2.5</td>
<td>2.5</td>
<td>15.4</td>
<td>18.2</td>
<td>19.0</td>
<td>21.2</td>
<td>25.1</td>
</tr>
<tr>
<td>SVGP-lm</td>
<td>3.5</td>
<td>1.3</td>
<td>5.1</td>
<td>11.2</td>
<td>8.00</td>
<td>10.9</td>
<td>19.6</td>
<td>26.4</td>
</tr>
</tbody>
</table>

Ablation Study on LM Iterations  We perform ablation study on different number of LM iterations, and compare how it affects the performance of SVGP-lm and PPGPR-lm. Figure shows the training on the Elevators dataset using results from different LM iterations. On this dataset, we see that using 10 LM iterations is sufficient and increasing the number of LM iterations to 50 or 150 does not significantly increase the training performance.