Sparse Gaussian Process for Missing Heart Rate Time Series Value Imputation

Jiaming Hu **Hanyuan Zhang** **Yirong Bian**

Abstract

Missing data is very common in many datasets but traditional machine learning methods often can not solve the missing data problem greatly. In this project, we try to apply Gaussian Process (GP) to solve this problem. GP is a non-parametric method with uncertainty estimation. However, full GP is slow for large datasets. Hence, we combine the GP with sparse approximation to get some sparse algorithms that can make the computation tractable. In this project, We will carry out an empirical study to investigate the performance of full GP as well as different sparse GPs to a heart time series dataset from MIT-BIH. We find that RBF is much better than the Periodic kernel in our task. Also, GP regression can give satisfactory results to fill in missing values in small intervals, even with limited training size. However, the performance of GP deteriorates fast as we increase the interval between each of the training points in interpolation and extrapolation tasks.

1 Introduction

Machine learning is playing an increasingly important role in healthcare, and many algorithms are successfully assisting healthcare workers (1).

An important application of machine learning in healthcare is digital diagnosis (2). Heart rate time series are associated with many heart-related pathologies such as atrial fibrillation or congestive heart failure. Therefore, heart rate is often seen as an important indicator of an individual's health. The prediction of pathological events from heartbeat time series data has been intensively studied, especially in the last decade (3): its relatively low cost makes it particularly promising.

However, as a kind of clinical data, heart rate data often contain missing data in the real healthcare field. It is common in datasets from other domains as well. Some examples include those related to laboratory measurements or shopping platforms (4). Machine learning models can learn the data structure and capture the complex relationships between different data. Therefore, people can estimate the true distribution of the data and make predictions about previous data (5). We can make better decisions, create high-quality clusters, and make more accurate predictions by utilizing machine learning tools. However, ignoring the effects of missing data when people use machine learning methods may lead to sub-optimal models that do not have good enough generalization performance.

Therefore, considering instances of missing values is a key step in machine learning methods. Gaussian process regression is a good way to find missing values. It is meaningful to the Gaussian process to do the imputation of the missing data of heart rate. There are already people who have gotten good results from their previous work (6). In our experiments, we tried regression prediction of heart rate time series using the Gaussian process, and we found that the prediction results were very good in small intervals.

Preprint. Under review.

2 Background

Gaussian processes (GPs) are fully probabilistic models which can naturally estimate predictive uncertainty through posterior variances. These uncertainties play a pivotal role in many application domains. Gaussian Process is completely specified by its mean and covariance function, denoted as

$$f(x) \sim \mathcal{GP}(m(\mathbf{x}), k^{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}')) \tag{1}$$

where $m(\cdot), k^{\boldsymbol{\theta}}(\cdot,\cdot)$ are mean and covariance function correspondingly(5). It is worth noting that the covariance function often depends on a set of hyper-parameters $\boldsymbol{\theta}$, and we denote this dependence by the superscript. A GP can be used as a prior over a regression problem. Namely, for a training dataset consisting of N noisy observations $\{(x_i, y_i)\}_{i=1}^N, x_i \in \mathbb{R}^d, y \in \mathbb{R}$, we assume that each scalar $y_i = f_i + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ is the noise. We denote the matrix composed with all training points x_i as X and y_i as $\mathbf{y}$, and the testing points as x_i^*. Then the data induce a posterior GP, specified by a posterior mean function and a posterior covariance function:

$$\begin{aligned} m_{\mathbf{y}}(x_i^*) &= K_{x_i^* X}(\sigma^2 I + K_{XX})^{-1}\mathbf{y} \\ k_{\mathbf{y}}(x_i^*, x_j^*) &= k(x_i^*, x_j^*) - K_{x_i^* X}(K_{XX} + \sigma^2 I)^{-1} K_{X x_j^*} \end{aligned} \tag{2}$$

where $K_{x_i^* X}$ is a row vector of kernel function between x_i^* and training points $\{x_i\}_{i=1}^n$, and K_{XX} is a covariance matrix between the training points. Then, any query in the inference process can be answered by the above two functions.

Now, in the training time, as the covariance function is dependent on the hyper-parameter, we develop a naive training strategy to maximize the likelihood and find the local optimum of hyper-parameters. The objective function we are maximizing is

$$\begin{aligned} \log p(\mathbf{y}|X;\boldsymbol{\theta},\sigma) &= \log\left[N\left(\mathbf{y} \mid \mathbf{0}, \sigma^2 I + K_{XX}^{\boldsymbol{\theta}}\right)\right] \\ &= -\frac{1}{2}\mathbf{y}^T[K_{XX}^{\boldsymbol{\theta}} + \sigma^2]^{-1}\mathbf{y} - \frac{1}{2}\log[K_{XX}^{\boldsymbol{\theta}} + \sigma^2] - \frac{N}{2}\log 2\pi \end{aligned} \tag{3}$$

We compute the differential with respect to $\boldsymbol{\theta}$ and then use the gradient ascent method to optimize the objective function (7).

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\theta}}\log p(\mathbf{y}|X;\boldsymbol{\theta}) &= \frac{1}{2}\mathbf{y}^T K_{XX}(\boldsymbol{\theta})^{-1}\frac{\partial K_{XX}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}K_{XX}(\boldsymbol{\theta})^{-1}\mathbf{y} - \frac{1}{2}\operatorname{tr}\left(K_{XX}(\boldsymbol{\theta})^{-1}\frac{\partial K_{XX}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right) \\ \frac{\partial}{\partial \boldsymbol{\theta}}\log p(\mathbf{y}|X;\boldsymbol{\theta}) &= \frac{1}{2}\operatorname{tr}\left(\left(\boldsymbol{\alpha}\boldsymbol{\alpha}^T - K_{XX}(\boldsymbol{\theta})^{-1}\right)\frac{\partial K_{XX}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right), \quad \boldsymbol{\alpha} = K_{XX}(\boldsymbol{\theta})^{-1}\mathbf{y} \end{aligned} \tag{4}$$

The gradient ascent step with learning rate η is as follows:

$$\begin{aligned} \boldsymbol{\theta}^{(i+1)} &= \boldsymbol{\theta}^{(i)} + \eta\nabla_{\boldsymbol{\theta}^{(i)}}\log p(\mathbf{y}|X;\boldsymbol{\theta}^{(i)}) \\ \boldsymbol{\theta}^{(i+1)} &= \boldsymbol{\theta}^{(i)} + \frac{\eta}{2}\operatorname{tr}\left(\left(\boldsymbol{\alpha}\boldsymbol{\alpha}^T - K_{XX}(\boldsymbol{\theta})^{-1}\right)\frac{\partial K_{XX}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^{(i)}}\right) \end{aligned} \tag{5}$$

Here, we explicitly write $K_{nn}(\boldsymbol{\theta})$ to illustrate that such covariance matrix is dependent on the hyper-parameter $\boldsymbol{\theta}$. Similarly, we can take the derivative with respect to σ and optimize it using gradient ascent.

To motivate the following chapters and algorithms we implement to the real-world dataset. We begin by observing that the optimization involves computing the inversion of the $n \times n$ matrix K_{nn} at each step, which costs $\mathcal{O}(n^3)$. When the training set is large, this operation is intractable. Therefore, it is important to design the sparse Gaussian process to make the calculation feasible. The following three sparse Gaussian process algorithms will approximate the full GP algorithm introduced in this chapter by using Lanczos matrix factorization tricks (8), reduction of training points, and structured kernel interpolation.

3 Problem definition and algorithms

3.1 Task

We intend to use Gaussian process regression to interpolate a set of heart rate time series data from the MIT-BIH dataset. Moreover, we are going to try to predict the heart rate using three

sparse Gaussian process methods: SKI/KISS GP, variational learning of inducing variables in sparse Gaussian processes(SGPR), and constant time predictive distributions(LOVE) on the same dataset. We have two experiment sets, for Experiment Set A, we used 1000 data points to test the performance of the different combinations of GP methods and kernel choices such as RBF, Periodic, and RBF + Periodic kernels. For Experiment Set B, we used 600 data points with RBF kernel to test the performance of sparse GP on different test size proportions. For Experiment Set C, we generate the time series data from a known function, fix the RBF kernel, and compare different sparse GP methods. We mainly focus on the result of training loss, training speed, and model accuracy on the testing set.

3.2 Algorithms

3.2.1 Structured Kernel Interpolation for Scalable Mixture of Gaussian Processes

In this section, we will introduce the structure kernel interpolation (SKI) algorithm proposed by Guo et al. (9). With kernel interpolation, SKI method gives kernel approximations for fast computations. SKI builds based on Inducing point methods and Structure exploiting approaches. Inducing point methods do not require the data to have special structures but do not work with a large number of inducing points, which leads to its degenerated performance for expressive kernel learning like the spectral mixture kernel. Structure exploiting approaches works on larger datasets and flexible kernel learning, with great gains in scalability. However, it has a very reactive requirement on the datasets, that the data has to be placed on a Cartesian product grid. SKI maintain the advantages of the above two methods and is able to overcome their disadvantages of them. It lifts the grid restriction, which gives the model ability to use inputs in arbitrary locations. It also works with large numbers of inducing points, which guarantees the performance of expressive kernel learning. Also, SKI can work with various inducing point methods.

SKI estimates the kernel evaluated at training and inducing inputs by the weighted average of kernels evaluated at the two closest inducing inputs that bound the training inputs. Here we take the prominent subset of regressors (SoR) as an example to illustrate.

For SoR, we approximate the kernel by:

$$\tilde{K}_{SoR}(X, Z) = K_{XU} K_{UU}^{-1} K_{UZ}$$

To approximate K_{XU}, take input point x_i and inducing point u_j as an example, we start with finding two closest inducing points u_m and u_n that bound x_i. Then we have our estimation on $K(x_i, u_j)$:

$$\tilde{K}(x_i, u_j) = w_i k(u_m, u_j) + (1 - w_i) k(u_n, u_j)$$

More generally, we have

$$K_{XU} \approx W K_{UU}$$

where W is the interpolation weights. Now we can write our K_{SKI}:

$$K_{XX} \overset{\text{SoR}}{\approx} K_{XU} K_{UU}^{-1} K_{UX} \overset{\text{SKI}}{\approx} W K_{UU} K_{UU}^{-1} K_{UU} W^T = W K_{UU} W^T = K_{SKI}$$

Note that SKI can be applied to every inducing point method. Finally, when we combine the SKI with Gaussian Process, sparse interpolation, and Kronecker or Toeplitz algebra, we call this method KISS-GP.

3.2.2 Sparse Gaussian Process Regression

In this section, we will introduce a Sparse Gaussian Process Regression (SPGR) algorithm proposed by Titsias using variational learning of inducing variables(10). The SPGR method uses the variational method to directly approximate the posterior in equation 2 and systematically choose the m inducing variables U_m to speed up the training process. It accomplishes this by augmenting the predictive model:

$$p(\mathbf{z}|\mathbf{y}) = \int p(\mathbf{z}|\mathbf{f}, \mathbf{f}_m) p(\mathbf{f}|\mathbf{f}_m, \mathbf{y}) p(\mathbf{f}_m|\mathbf{y}) d\mathbf{f} d\mathbf{f}_m \tag{6}$$

where $\mathbf{f}_m$ is the function evaluated at the inducing points U_m, $\mathbf{f}$ is the function evaluated at the training points X. Suppose the $\mathbf{f}_m$ is a sufficient statistic, then it means $p(\mathbf{z}|\mathbf{f}_m, \mathbf{f}) = p(\mathbf{z}|\mathbf{f}_m)$ the

above can be written as

$$\begin{aligned} q(\mathbf{z}) &= \int p(\mathbf{z}|\mathbf{f}_m)p(\mathbf{f}|\mathbf{f}_m)\phi(\mathbf{f}_m)d\mathbf{f}d\mathbf{f}_m \\ &= \int p(\mathbf{z}|\mathbf{f}_m)\phi(\mathbf{f}_m)d\mathbf{f}_m \\ &= \int q(\mathbf{z},\mathbf{f}_m)d\mathbf{f}_m \end{aligned} \tag{7}$$

where $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{y})$, $\phi(\mathbf{f}_m) = p(\mathbf{f}_m|\mathbf{y})$. And $p(\mathbf{f}|\mathbf{f}_m,\mathbf{y}) = p(\mathbf{f}|\mathbf{f}_m)$ is true since $\mathbf{y}$ is a noisy observation of $\mathbf{f}$ and any $\mathbf{z}$ is conditionally independent from $\mathbf{f}$ given $\mathbf{f}_m$. Then according to equation 7, we can write the approximate posterior as:

$$\begin{aligned} m^q_{\mathbf{y}}(x^*_i) &= K_{x^*_i U}K^{-1}_{UU}\boldsymbol{\mu} \\ k^q_{\mathbf{y}}(x^*_i,x^*_j) &= k\left(x^*_i,x^*_j\right) - K_{x^*_i U}K^{-1}_{UU}K_{Ux^*_j} + K_{x^*_i U}BK_{Ux^*_j}, \quad B = K^{-1}_{UU}AK^{-1}_{UU} \end{aligned} \tag{8}$$

Here, we treat $\phi(\mathbf{f}_m)$ as a variational Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix A. Note that by using this sparse posterior GP, the time complexity is reduced to $\mathcal{O}(nm^2)$. To specify the distribution ϕ and the inducing points X_m, we treat X_m as a variational parameter and select it by minimizing the KL divergence between the true posterior $p(\mathbf{f},\mathbf{f}_m|\mathbf{y})$ and $q(\mathbf{f},\mathbf{f}_m) = p(\mathbf{f}|\mathbf{f}_m)\phi(\mathbf{f}_m)$. It is equivalent to maximizing the variational lower bound of the true log marginal likelihood:

$$F_V\left(U_m,\phi\right) = \int p\left(\mathbf{f}|\mathbf{f}_m\right)\phi\left(\mathbf{f}_m\right)\log\frac{p(\mathbf{y}|\mathbf{f})p\left(\mathbf{f}_m\right)}{\phi\left(\mathbf{f}_m\right)}d\mathbf{f}d\mathbf{f}_m \tag{9}$$

by analytically solving the optimal choice by differentiation for

$$\begin{aligned} \phi^*(\mathbf{f}_m) &= \mathcal{N}(\mathbf{f}_m|\boldsymbol{\mu},A) \\ \boldsymbol{\mu} &= \sigma^{-2}K_{UU}\Sigma K_{UX}\mathbf{y}, \quad A = K_{UU}\Sigma K_{UU} \\ \Sigma &= (K_{UU}+\sigma^{-2}K_{UX}K_{XU})^{-1} \end{aligned} \tag{10}$$

we obtain:

$$F_V\left(U_m\right) = \log\left[N\left(\mathbf{y}\mid\mathbf{0},\sigma^2 I + Q_{XX}\right)\right] - \frac{1}{2\sigma^2}\operatorname{Tr}(\tilde{K}) \tag{11}$$

where $Q_{XX} = K_{XU}K^{-1}_{UU}K_{UX}, \tilde{K} = \operatorname{Cov}(\mathbf{f}|\mathbf{f}_m) = K_{XX} - K_{XU}K^{-1}_{UU}K_{UX}$. By the gradient ascent method, we can easily maximize the above function and it provides us with a rigorous framework for selecting inducing points. We will apply it to the real heart rate dataset to illustrate its speed and accuracy.

3.2.3 Lanczos Variance Estimates (LOVE)

The LOVE algorithm is based on the Lanczos algorithm, which could decompose a symmetric matrix $B \in R^{n\times n}$ as QTQ^T, where $T \in R^{n\times n}$ is symmetric tridiagonal and $Q \in R^{n\times n}$ is orthonormal and the mean computation of KISS-GP, by which we could use $w^T(x^*_i)Cw(x^*_j)$ to represent the predictive covariance between x^*_i and x^*_j. C is an $m \times m$ matrix, which depends on training set. Nevertheless, with C providing fast matrix-vector multiplication (MVM), we could avoid explicit matrix computation. With the Lanczos algorithm, we can factorize C into two matrices R^T and R' with rank k in almost linear time. After pre-computing for once, because of the special structure of R^T and R', for every covariance entry, all the variances can be computed in constant time $\mathcal{O}(k)$. For the predictive covariance

$$k_{\mathbf{y}}(x^*_i,x^*_j) = k(x^*_i,x^*_j) - K_{x^*_i X}(K_{XX}+\sigma^2 I)^{-1}K_{Xx^*_j} \tag{12}$$

where X^* is the set of test points and X is the set of training points. By using the KISS approximations, for the second term in (12), we could get

$$w^T_{x^*_i}K_{UU}W(K_{SKI}+\sigma^2 I)^{-1}W^T K_{UU}w_{x^*_j} \tag{13}$$

where W is the sparse interpolation matrix of training set and $w_{x^*_j}, w_{x^*_i}$ are sparse interpolation of x^*_j, x^*_i respectively. Notice that

$$C = K_{UU}W(K_{SKI}+\sigma^2 I)^{-1}W^T K_{UU} \approx R^T R' \tag{14}$$

By using Lanczos algorithm, we could approximate $(K_{XX} + \sigma^2 I)^{-1} \approx Q_k T_k^{-1} Q_k^T$. We write the subscript of Q_k, T_k to emphasize the number of iterations k in the algorithm. By plugging back to the equation 14, we have:

$$C \approx K_{UU} W Q_k T_k^{-1} Q_k^T W^T K_{UU} \tag{15}$$

where $R^T = K_{UU} W Q_k$ and $R' = T_k^{-1} Q_k^T W^T K_{UU}$. The computation of R^T and R' takes $\mathcal{O}(kn + km \log m)$ total time.

4 Experiments

4.1 Data and Basic Settings

We collect the heart rate time series data from the MIT-BIH dataset. The series contains 1800 evenly-spaced measurements of instantaneous heart rate from a single subject. The measurements (in units of beats per minute) occur at 0.5-second intervals (11). We randomly split the data into train and test subsets. Besides, we normalize the data to 0 mean to enhance the performance of our algorithm. With prior knowledge and information from the training set, viewing the test set as missing data, we apply the full Gaussian process and three sparse Gaussian process methods to do the data imputation. In this work, we conduct two sets of experiments to investigate how the different factors influence the performance of the posterior inference means to the missing data, as well as their training speed. Note in the below paragraphs we only selected a few interesting experiment results. For more details, please visit our GitHub repository.

4.2 Experiment Set A

In this set of experiments, we are going to discuss how below the 3 factors influence the performance and training speed of our Gaussian process models. They are 1. kernel choice 2. inducing model choice 3. whether to include LOVE. With 1000 data points in total and test size = 0.2, below are the 18 models we devise for this section:

Models in Experiment Set A			
A.1.1.1	RBF + Full GP + No LOVE	A.1.1.2	RBF + Full GP + LOVE
A.1.2.1	RBF + SKI + No LOVE	A.1.2.2	RBF + SKI + LOVE
A.1.3.1	RBF + SGPR + No LOVE	A.1.3.2	RBF + SGPR + LOVE
A.2.1.1	Periodic + Full GP + No LOVE	A.2.1.2	Periodic + No Full GP + LOVE
A.2.2.1	Periodic + SGPR + No LOVE	A.2.2.2	Periodic + SKI + LOVE
A.2.3.1	Periodic + SGPR + No LOVE	A.2.3.2	Periodic + SGPR + LOVE
A.3.1.1	RBF + Periodic + Full GP + No LOVE	A.3.1.2	RBF + Periodic + Full GP + LOVE
A.3.2.1	RBF + Periodic + SKI + No LOVE	A.3.2.2	RBF + Periodic + SKI + LOVE
A.3.3.1	RBF + Periodic + SGPR + No LOVE	A.3.3.2	RBF + Periodic + SGPR + LOVE

In particular, we number our models A.I.J.K in the following way: I represents kernel choice - 1: RBF 2: Periodic 3. RBF + Period. J represents the inducing model choice - 1: full GP 2:SKI 3: SGPR. K represents whether to Include LOVE - 1:No LOVE 2:with LOVE.

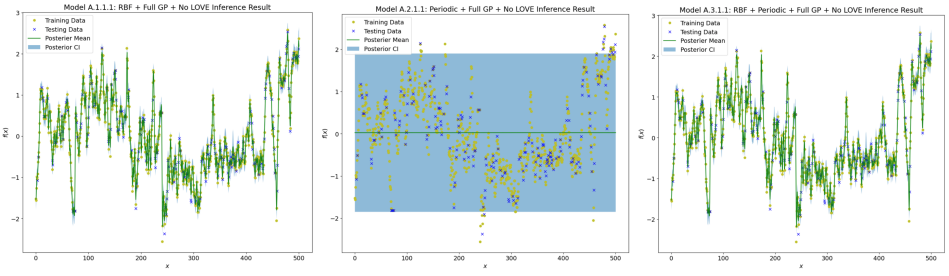

Figure 1: Experiment set A: RBF vs Periodic vs RBF + Periodic Inference performance

There we are going to present a few interesting findings. First, in terms of kernel choice, the periodic kernel has the worst performance, while RBF and RBF + Periodic present a similar performance. As

we can see from figure 1, model A.2.1.1 just gives a zero mean for the posterior inference, while model A.1.1.1 and model A.3.1.1 give a very similar inference. Hence, it is reasonable to deduce that all credit should be contributed to the RBF kernel and the periodic kernel doesn't play a role in this task.

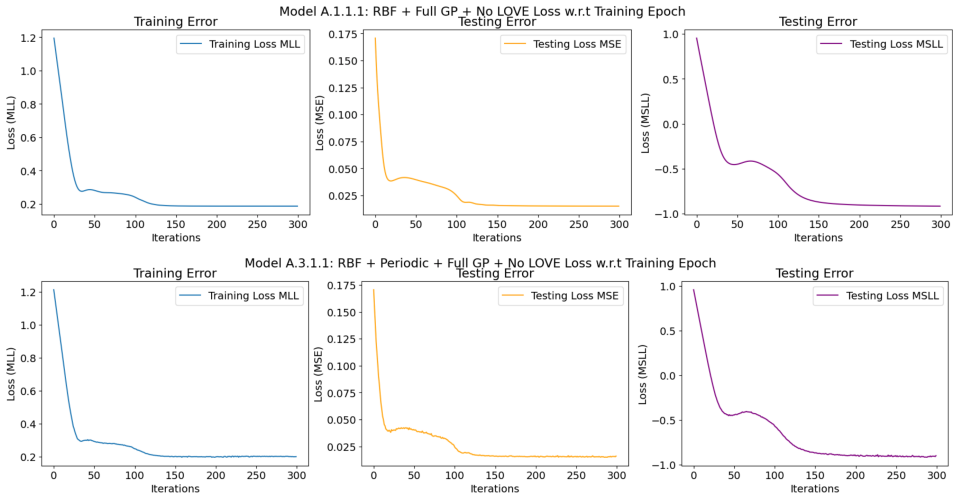

Figure 2: Experiment set A: Loss Log(RBF vs RBF + Periodic)

This argument could be more tenable if we take a look at the loss log of these models. As we can see from figure 2, Model A.1.1.1 and Model A.3.1.1 share a very alike loss lag, except that the latter one is noisier. A meaningful interpretation is that the Periodic kernel just adds some noise in the training and inference process. The second interesting finding is about training speed with respect to inducing

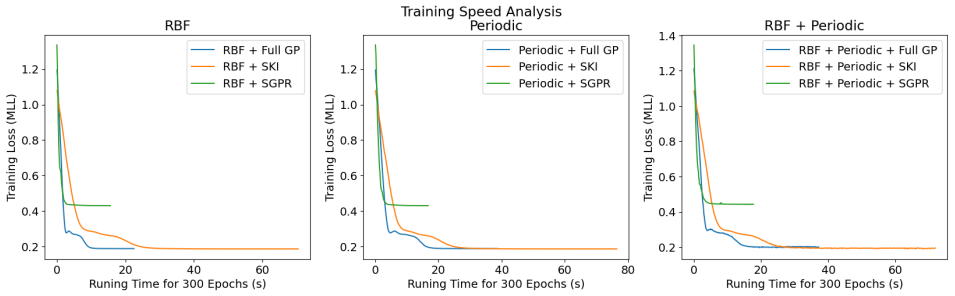

Figure 3: Experiment set A: Training Speed Analysis

models. As we can see from figure 3, regardless of the kernel choices, the SKI is always the slowest one. Also, the SGPR beat Full GP only by a very small margin. This is quite counter-intuitive since SKI and SGPR should speed up the training process. We guess that this is because our dataset is not large enough to show the advantage of SKI and SGPR. In our experiments, we find that the training speed of SKI is approaching to full GP model as we increase the data points included. Also, it is noteworthy that SGPR has a higher training loss at the end. There is a trade-off between accuracy and training speed for SGPR. This can also be shown in figure 4. As we can see, SGPR has the largest confidence interval. This is also true for other settings.

4.3 Experiment Set B

In this set of experiments, we are conducting experiments to explore the train size and test size going to influence the performance and training speed of our Gaussian process models. Fixing the kernel choice as RBF or RBF + Periodic, inducing model as full GP, we range the testing size from 0.1 to 0.9 (No LOVE). To speed up the training process, we only use 600 data points in this set of experiments. below are the 18 models we devise for this section:

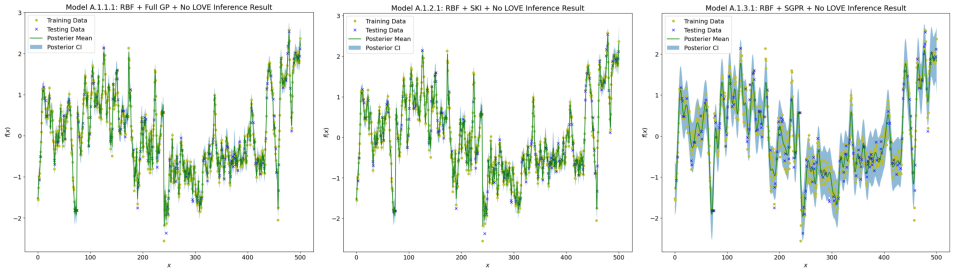

Figure 4: Experiment set A: Inducing Model Comparison

Models in Experiment Set B				
Experiment B.1	B.1.1	RBF (test size = 0.1)	B.1.1	RBF + Periodic (test size = 0.1)
Experiment B.2	B.2.1	RBF (test size = 0.2)	B.2.1	RBF + Periodic (test size = 0.2)
Experiment B.3	B.3.1	RBF (test size = 0.3)	B.3.1	RBF + Periodic (test size = 0.3)
Experiment B.4	B.4.1	RBF (test size = 0.4)	B.4.1	RBF + Periodic (test size = 0.4)
Experiment B.5	B.5.1	RBF (test size = 0.5)	B.5.1	RBF + Periodic (test size = 0.5)
Experiment B.6	B.6.1	RBF (test size = 0.6)	B.6.1	RBF + Periodic (test size = 0.6)
Experiment B.7	B.7.1	RBF (test size = 0.7)	B.7.1	RBF + Periodic (test size = 0.7)
Experiment B.8	B.8.1	RBF (test size = 0.8)	B.8.1	RBF + Periodic (test size = 0.8)
Experiment B.9	B.9.1	RBF (test size = 0.9)	B.9.1	RBF + Periodic (test size = 0.9)

In particular, we number our models B.I.J in the following way: I represents test size and J represents kernel choice - 1: RBF 2: RBF + Period.

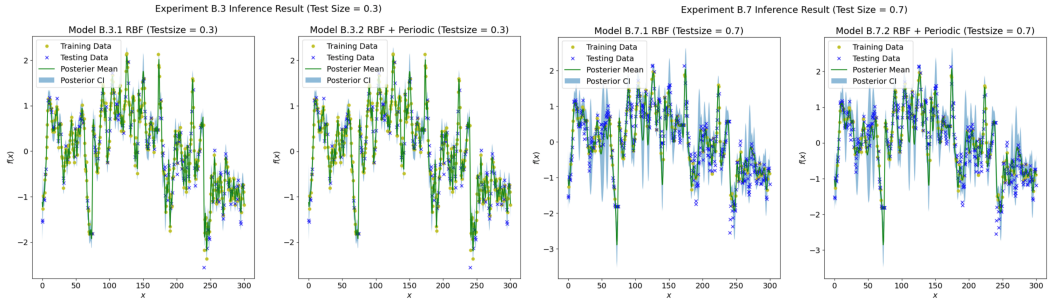

Figure 5: Inference Result with Different Test Size

In this set of experiments, our most exciting experiment is that our Gaussian models do not need a large amount of data to fill in the missing value well. As we can see from figure 5, compared with the left two inference results where the training size is 0.7, the right two inference result where the training size is 0.3 is still at a satisfactory level. This shows that GP regression is powerful when filling in missing values in small intervals, even with a limited training size. However, GP regression does not work well when filling in missing values in large intervals. As we can see from figure 6, the result is not satisfactory even with a deep feature extractor. The detailed experiment setting is in the 'Experiment with Deep GP.ipynb' file in our GitHub repository.

4.4 Experiment Set C

In this set of experiments, we intend to further experiment with different sparse Gaussian methods and analyze their speed and accuracy. We generate 2000 data points based on the function:

$$f(x) = \sin(2\pi x) + \cos(\frac{4}{5}\pi x) + 0.2\omega, \quad \omega \sim \mathcal{N}(0, 1)$$

where ω is the white noise. We fix to the RBF kernel, and similar to the previous two experiments, we randomly split the data into train and test sets with a test size of 0.2. Moreover, we configure 20 inducing points for the SPGR method and set up two different grid sizes 0.001 and 0.1 of the

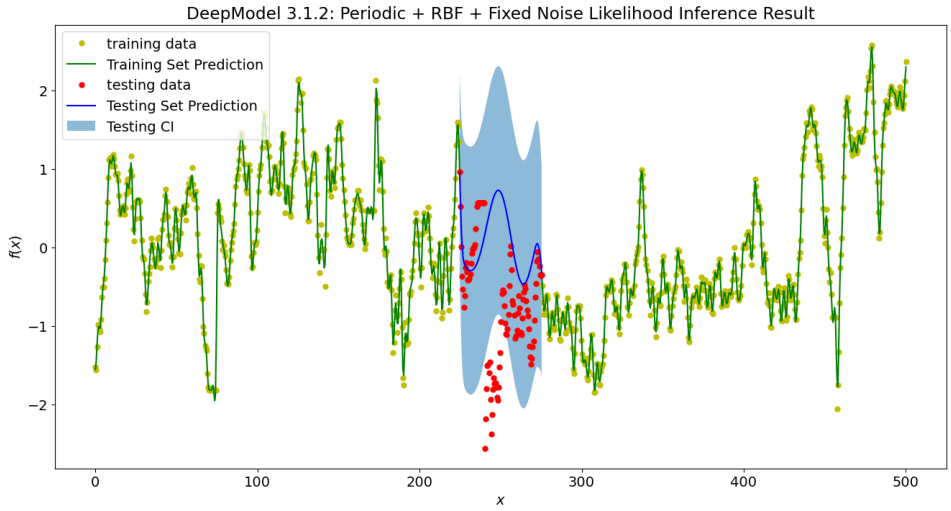

Figure 6: GP with Deep Feature Extractor Inference Result

training set for the SKI method. The speed of different models is illustrated in Figure 7. The left plot is when we use 0.001 grid size of the training set and train for 300 epochs and the right plot is when we use 0.1 grid size and train for 100 epochs. It is worth noting that in the left graph, the SKI is fast to converge but remains a large training loss while on the right, SKI's speed is similar to the model of Full GP with almost the same accuracy. We conclude that there is a trade-off between the accuracy and the speed of SKI, dependent on the grid size. On the other hand, SGPR is extremely fast in speed at the cost of terminal training loss. It outperforms the other two methods not only in the speed of convergence but also in the time of each iteration. In this setting, the SGPR method is the most scalable one if the fitting does not need very high accuracy.

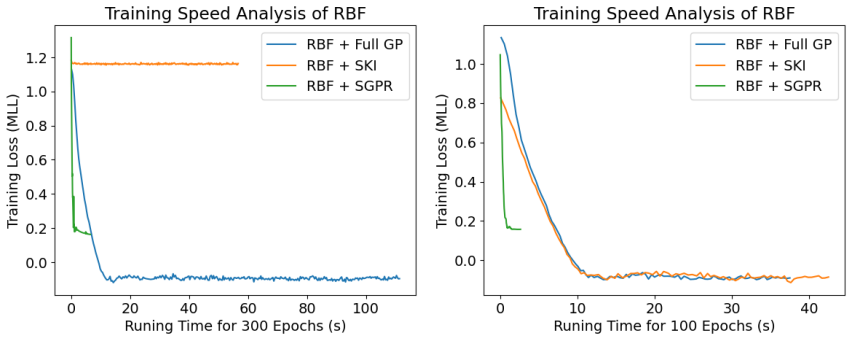

Figure 7: Different Sparse GP Speed Result

5 Conclusion and Discussion

In conclusion, we found out that kernel choice is important for different datasets. In our experiment, the RBF kernel is a much better choice than the periodical kernel for the heart rate time series data. Moreover, GP regression is powerful when filling in missing values in small intervals, even with limited training size but for interpolation and extrapolation of large intervals, the performance of GP is not as expected. The training speed and the advantage of sparse GP are significant when the training data size has more than thousands of rows. For different sparse Gaussian methods, to obtain the best balance between the training speed and the accuracy of the model, the tuning of the hyper-parameter is rather important. Nonetheless, we verified that it is possible for the GP model to do missing value imputation and the choice of sparse GP model and the kernel varies from case to case.

The code we used to train and evaluate our models is available at `https://github.com/hanyuanz2000/Sparse-Gaussian-Process-for-Missing-Heart-Rate-Data-Imputation`

6 Contribution

We worked together on the basic Gaussian process regression part and wrote the report. Each one is responsible for one implementation of the sparse Gaussian process method. All team members contribute equally.

References

[1] S. Saria, A. Butte, and A. Sheikh, "Better medicine through machine learning: What's real, and what's artificial?," 2018.

[2] B. Heinrichs and S. B. Eickhoff, "Your evidence? machine learning algorithms for medical diagnosis and prediction," *Human brain mapping*, vol. 41, no. 6, pp. 1435–1444, 2020.

[3] Y. Ma, C.-w. Wu, C.-K. Peng, A. Ahn, S. M. Bertisch, L. A. Lipsitz, G. Y. Yeh, B. Manor, V. Novak, J. M. Hausdorff, *et al.*, "Complexity-based measures of heart rate dynamics in older adults following long-and short-term tai chi training: cross-sectional and randomized trial studies," *Scientific reports*, vol. 9, no. 1, pp. 1–10, 2019.

[4] R. J. Little and D. B. Rubin, "Statistical analysis with missing data (vol. 793)," 2019.

[5] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[6] O. Stegle, S. V. Fallert, D. J. MacKay, and S. Brage, "Gaussian process robust regression for noisy heart rate data," *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 9, pp. 2143–2151, 2008.

[7] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, vol. 2. MIT press Cambridge, MA, 2006.

[8] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2013.

[9] Y. Guo, N. Xu, and Z. Jin, "Structured kernel interpolation for scalable mixture of gaussian processes," in *2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP)*, pp. 61–65, 2022.

[10] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes," in *Artificial intelligence and statistics*, pp. 567–574, PMLR, 2009.

[11] G. Moody, "Heart rate time series," Jul 2005.