Gaussian Process Kernels
Term Project Report

Prince Mahesh Bharadwaj (17CY20023)

Adarsh Goyal (17CY20001)

Nwe Ni Kyaw (20CS91A01)

Amatya Sharma (17CS30042)

23.10.2020

Indian Institute of Technology, Kharagpur

TABLE OF CONTENTS

INTRODUCTION & PRELIMINARIES

The Gaussian process (GP) is a simple yet powerful probabilistic framework for various machine learning tasks. However, exact algorithms for learning and prediction are prohibitive to be applied to large datasets due to inherent computational complexity. To overcome this main limitation, various techniques have been proposed, and in particular we can classify approximations in scalable GPs into two main categories of local and global approximations. After studying these we focus on automatically constructing Gaussian process models as done in [19] where we try and search over sums and products of kernels and focus to optimize the approximate marginal log likelihood. In this term project we extensively focus ourselves on a rather deeper literature survey on Gaussian Processes and quote essential experimental results only if needed. The reason behind that is Gaussian Process kernels have been studied and experimented on a lot in the last 10 years but unfortunately it is still difficult to find all the compiled results at one place.

We first begin with defining the preliminaries and basics of Gaussian Processes.

Definition of Gaussian Process

A Gaussian process is a series of arbitrary variables such that there is a joint multivariate distribution in every finite subset of it. It is any distribution over functions such that any finite set of function values f(x1), f(x2), . . . f(xN) have a joint Gaussian distribution . A Gaussian process is specified by its mean and covariance functions.

Mean function,

$$\mathbf{E\,[f(x)] = \mu(x)}$$

Covariance function, it is also called kernel.

$$\mathbf{Cov\,[f(x), f(x')] = k(x, x')}$$

The mean function is zero everywhere because the uncertainty of the mean function is usually taken by adding an extra term to the kernel.

After considering the mean, the type of the structure that the GP model can capture depends entirely on its kernel. The kernel defines how new data is generalized or

extrapolated, by the model.

Model Selection

The essential property of GPs to build models automatically is that, provided a specific model, also known as the evidence, we can measure the marginal probability of a dataset. The marginal likelihood allows the comparison of the models, balancing a model's skill and its fit with the data. It can be used for model selection.

The Bayes rule for computing the posterior

$$\mathbf{P(\theta \setminus X,m) = (p(X, \theta \setminus m))/(p(X \setminus m)) = (p(X \setminus \theta ,m)\, p(\theta \setminus m))/ \left(\int p(X \setminus \theta ,m)\, p(\theta \setminus m) d\theta \right)}$$

$$\mathbf{= (Likelihood \times Prior)/(Marginal\ Likelihood)}$$

The denominator in the Bayes rule is the marginal likelihood.

Note that $p(X \setminus m) = E_{p(\theta \setminus m)}[p(X \setminus \theta ,m)]$ is the average /expected likelihood under model m.

Choose model m that has largest posterior probability

$$\mathbf{\hat{m} = \arg\max_m p(m \setminus X) = \arg\max_m = \arg\max_m p(X \setminus m)p(m)}$$

If all models are equality likely a priori then $\hat{m} = \arg\max_m p(X \setminus m)$. If m is a hyperparam, then $\arg\max_m p(X \setminus m)$ is MLE-II based hyperparameter estimation.

Predictive Distribution

Predictive distributions play a central role in statistics and in close areas such as *Machine Learning*. We realize that the uncertainty about the exact mean and variance to plug into the predictive formula is unclear.The distribution created by averaging future predictions over the posterior densities of all unknown parameters is called the "predictive density" in Bayesian analysis.

$$\mathbf{Predictive\ distribution(z) = Z\ f(z \mid \theta) f(\theta \mid x)\ d\theta}$$

where, x is the observed data and θ is the vector of unknown parameters. Thus $f(z \mid \theta)$ is the prediction we would make for future data if theta were known exactly, but since it is not, we average this quantity over $f(\theta \mid x)$, the posterior density of θ after observing the

data x.

The above formula is completely general, in that f can be any density or even an extremely complex model, and θ can represent one, two, or even thousands of unknown parameters.

Kernel

A kernel is a two input x, x′ positive-definite function. In this section, x and x′ are typically vectors in a Euclidean space, but graphs, images, discrete or categorical inputs, or even text may also be represented by kernels.

Gaussian process models use a kernel to define the prior covariance between any two function values:

$$\mathbf{Cov\,[f(x),\, f(x')] = k(x,\, x')}$$

Kernels are also used to specify the similarity between two items. In this case, this is somewhat misleading since the similarity between the two values of the function evaluated on each object. The kernel defines the functions are probable under the GP prior and then determines the general attributes of the model.

Characterization of Kernels

A symmetric function k is called positive semi-definite in X if: for every $N \in N$, and every choice $x_1, \cdots, x_N \in X$, the matrix $K = (k_{ij})$, where $k_{ij} = k(x_i, x_j)$ is positive semi-definite.

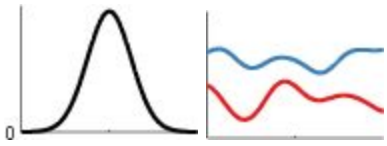
Theorem. k admits the existence of a map $\phi : X \to H$ s.t.

H is a Hilbert space and $k(x, x\,0) = \phi(x), \phi(x\,0)_H$

if and only if k is a positive semi-definite symmetric function in X.

Standard Kernels

Squared Exponential Kernel



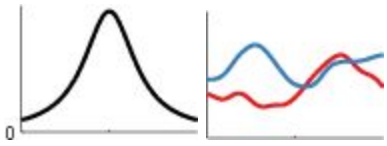
the Radial Basis Function kernel, the Gaussian kernel. It has the form:

$$k_{SE}(x,x') = \sigma^2 \exp\left(-(x-x')^2 / 2\ell^2\right)$$

The SE kernel has become the actual default kernel for GPs and SVMs. This may be because there are some good properties that it has. . It is universal, and we can integrate it into most of the functions that we want to. The prior of each function has an infinite number of derivatives . Also ,it only has two parameters:

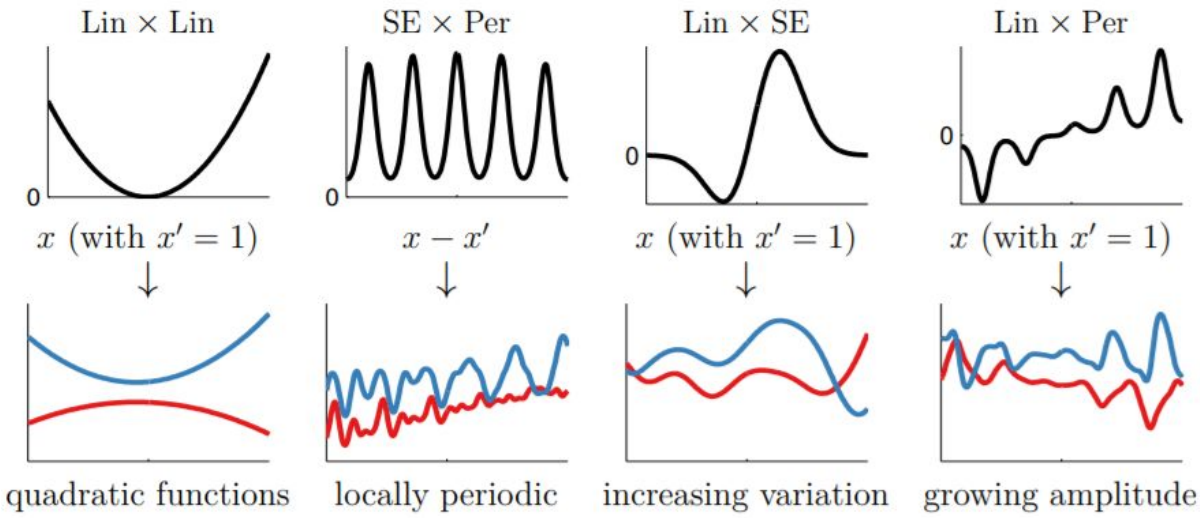
1. The length scale ℓ determines the length of the 'wiggles' in the function. Generally, you will not be able to extrapolate more than ℓ units away from the data.
2. The performance variance $\sigma 2$ determines the average distance the function deviates from its average value. Each kernel puts this parameter in front.This is just a scale factor.

Rational Quadratic Kernel



$$k_{RQ}(x,x') = \sigma^2 (1+(x-x')^2 2\alpha\ell 2)^{-\alpha}$$

This kernel is the equivalent of adding several SE kernels of different scales of length together. Therefore, GP priors expect this kernel to see features that differ smoothly over several scales of length. . The parameter α determines the proportional weighting of combinations of large and small scales.. When $\alpha \to \infty$, the RQ is identical to the SE.



In order to introduce a complex/higher level to the kernels, we can even try multiplying various kernels and visualizing them. The plots are obtained by drawing samples from different priors discussed so far including SE (Squared Exponential), Per (periodic), Lin (linear) kernels, and then multiplying these 1-D base kernels as in figure above.

APPROXIMATIONS

The Gaussian process (GP) is a simple yet powerful probabilistic framework for various machine learning tasks. However, exact algorithms for learning and prediction are prohibitive to be applied to large datasets due to inherent computational complexity. To overcome this main limitation, various techniques have been proposed, and in particular we can classify approximations in scalable GPs into two main categories:

1) **Local Approximations**: It employs the divide-and-conquer (D&C) technique to focus on the local subsets of training data. There are majorly following local approximation techniques:

a) **Naives Local Experts (NLE)** : It directly employs the pure local experts for prediction.
b) **Mixture of Experts (MoE)** : It "mixes" local and diverse experts owning individual parameters for improving the overall accuracy and reliability.
c) **Product of Experts (PoE)** : It multiplies these probability distributions, which is similar to an "AND" operation on the probability densities.

d) **Generalized Product of Experts (GPoE)** : It introduces a varying weight in PoE multiples.

e) **Bayesian committee machine (BCM)** : It provides an extension to PoE by aggregating the experts' predictions from another point of view by imposing a conditional independence assumption and explicitly introducing a common prior.

f) **Generalized Robust Bayesian committee machine (GRBCM)** : It introduces a global communication expert M_c and extends the basic robust BCM.

2) **Global Approximations**: It employs techniques that approximate the kernel matrix K_{nn} through global distillation. Three major techniques involved are:

a) **Subset of Data**
b) **Sparse Approximations**
c) **Sparse Kernels**

3) **Hybrid Approximation**: It employs a mixture of global and local approximation

techniques to replicate K_{nn} as closely as possible.

Throughout this section, we assume we have already obtained suitable hyperparameters for the covariance function, and we just concern ourselves with examining the nature of the approximations themselves.

Local Approximation

Local Approximation : It is a set of tools based on Divide & Conquer Approach to approximate the kernel K_{nn} requiring much lesser complexity than $O(n^3)$ of computing the original covariance matrix K_{nn}. It enables non-stationary features and uses localized experts to improve the scalability of GP.

1. Naives Local Expert (NLE)

NLE directly employs the pure local experts for prediction. The motivating idea for the concept is the fact that pairs of points far away from each other are lowly correlated. We are given a set $\{M_i\}_i$ where each M_i is a local expert completely deciding the subregion Ω_i defined by X_i. The goal is to predict at $x^* \in \Omega_i$ as

$$\mathbf{p(y^* | D, x^*) \approx pi(y^* | D_i, x^*)}$$

The algorithm partitions the input space, train the experts and chooses an appropriate one for predictions (not necessarily in the same order). This further gives us the following classification of the NLE:

1. **Transductive NLE [2]:** First selects a neighbourhood subset D* around x* and then trains the corresponding expert M*. Given the test set size n_t , it employs a dynamic partition to choose m_0 neighbor points around x*, resulting in $O(n_t m_0^3)$ complexity. For selecting D*, we can use the simple geometric closeness criteria or much more complex but accurate methods that sequentially select the neighbourhood set such as GP-based active learning methods.
2. **Inductive NLE [1]:** First trains all the experts $\{M_i\}$, partitions the space D and then decides corresponding prediction y*. For partitioning it employs a static partition of D using clustering techniques, e.g., the Voronoi tessellations [1] and trees. Given $m_0 = n/M$ is the training size for each expert, for training purposes the model trains independent local GP experts, resulting in $O(nm_0^2)$ time complexity.

Drawbacks NLE

Although NLE being naive provides a simple and faster implementation to locally approximate the kernel, the naiveness introduces certain drawbacks as well which we'll be discussing below:

- ➔ Poor **Generalizability** : Due a completely local approach the model lacks in generalizing the functionality across the whole kernel.
- ➔ Drops **Long-term spatial correlations** : The motivation towards building this model appears as its drawback since it ignores the correlation between far-away points which is not always true.
- ➔ **Discontinuous Predictions** on **Boundaries**.

Advancements and Developments:

Observing the aforementioned drawbacks there have been numerous advancements in NLE as follows:

- ➔ **Patched GPs [3] :** Using GPs with overlapping non-disjoint patches between two local GPs can improve the discontinuity issues by dealing with the limits near the merger of two local adjacent GPs.
- ➔ **Mixture or Product of GPs :** Pathed GPs although being able to resolve discontinuity and boundary issues is not scalable to higher dimensions and can sometimes result in negative variances as well. Thus a better alternative is provided by Mixture or Product of GPs as discussed in further sections.

2. Mixture of Experts (MoE) [4][5]

MoE "mixes" local and diverse experts owning individual parameters for improving the overall accuracy and reliability. It employs a Gaussian Mixture Model to express the mixture/combination of local and diverse experts. The mixture is represented using weights with the conditional probabilities $\mathbf{g_i(x)}$ formally termed as **gating function**. It manages the mixture with the means of a probabilistic partition of the input space and thus defining the subregions where the individual experts are responsible for themselves. It coheres with the term p(z=i) of the GMM which also refers to the weight of

x assigned to expert M_i. The choice of experts depends on the implementation and requirements and can be taken as a linear model or SVMs.

$$\mathbf{p(y|x) = \sum_{i=1 \text{ to } M} g_i(x) \, . \, p_i(y|x)}$$

The main assumption of the model is that data points are mutually independent and identical events. This assumption frees us to use negative **log likelihood maximization** as given below in order to learn $g_i(x)$ as well as the experts using standard **gradient-descent based optimizers** or **Expectation-Maximization algorithm.**

$$\textbf{Maximize} \quad \sum_{i=1 \text{ to } M} p(y_t|x_t) \qquad \text{[i.i.d assumption]}$$

This gives us the following expression for **predictive distribution**:

$$\mathbf{p(y^*|D, x^*) = \sum_{i=1 \text{ to } M} g_i(x^*|D) \, . \, p_i(y^*|D, x^*)}$$

--Predictive-- **--Posterior--**

Drawbacks of MoE

MoE despite being a significant improvement over previous model of NLE, it has some setbacks discussed as follows:

- **Model Selection Issue :** The decision of determining the number of experts has always been a question of interest as poses a drawback to the same as there are no certain measures whether to use fully generative models or others.
- **High model complexity :** The model is based on a multi-modal model, thus making the individual global experts responsible for all the data points. This leads to a higher complexity as the parametric gating function gi is not favored in the Bayesian nonparametric framework and the i.i.d. data assumption does not hold here since GP fits the data dependences through joint distribution.

Advancements and Developments:

Observing the aforementioned drawbacks there have been numerous advancements in MoE to recover from them:

- **Model Selection** : Various proposals to select the model such as **Akaike information criterion** and the synchronously balancing criterion have been pitched to choose over a set of candidate M values. Furthermore, researchers have been using stochastic techniques like the **Dirichlet process [6]**, **Polya urn**

distribution [7], **Pitman–Yor process [8]** to automatically infer the number of experts from data. Due to the complex prior and the infinite M, the model has to resort to a stick-breaking representation of Dirichlet Process.

➔ **Model Complexity** : There have been three major threads dealing with the issue:

- **Mixture of implicitly localized experts (MILE) [9]** : Model assigns the data dynamically according to the data property and the experts' performance.
  - **Combining Global and Sparse approximations** : This includes interpreting GP as the finite Bayesian linear model and using the FITC experts that factorize over f given the inducing set f_m. With m inducing points for each expert, the complexity is $O(nm^2M)$, which can be further reduced to $O(nm^2)$ with the Expectation Maximization technique.
- **Mixture of explicitly localized experts (MELE) [9]**: It involves pre partitioning the input space by clustering techniques and assigning points to the experts before training. This thread overcomes the main drawback of MILE i.e. MILE is a competitive learning process and some experts may fail due to the zero-coefficient problem caused by unreasonable initial parameters.

3. Product of Experts (PoE)[10]

Product of Experts (PoE) multiplies these probability distributions, which is similar to an "AND" operation on the probability densities as opposed to MoE which sums (weighted) over the probability distributions or the experts via an "OR" operation as follows:

$$\mathbf{p(y|x) = 1/Z \,.} \quad \prod_{\mathbf{i=1\ to\ M}} \mathbf{p_i(y|x)} \qquad [\ Z{:=}\ \text{normalizing constant}\ ]$$

Since pi(y|x) in (25) is a Gaussian distribution and using the fact that the product of multiple Gaussians is still a Gaussian distribution, we obtain a factorized marginal **likelihood** thus making it convenient to drop the 1/Z normalizing factor as follows:

$$\mathbf{p(y|X)} = \prod_{\mathbf{i=1\ to\ M}} \mathbf{p_i(y_i|x_i)}$$

where $p_i(y_i|X_i) \sim N(y_i|0, K_i + \sigma^2 I_{n\_i})$ with $K_i = k(X_i, X_i) \in R^{n\_i \times n\_i}$ and n_i being the training size of expert M_i. This factorization degenerates the full kernel matrix K_{nn} into a diagonal

block matrix diag[K_1,⋯ , K_M], leading to

$$\mathbf{K}_{nn}^{-1} \approx \mathbf{diag}[\mathbf{K}_1^{-1}, \cdots, \mathbf{K}_M^{-1}]$$

Hence, the **complexity** is substantially reduced to $\mathbf{O(nm_0^2)}$ given $n_i = m_0$.

MoE vs PoE and Drawbacks of PoE

- ➔ Due to the weighted sum form of the MoE, it will never be **sharper** than the sharpest expert; on the contrary, due to the product form of the PoE, it can be sharper than any of the experts.
- ➔ PoE produces poor prediction mean and overconfident prediction variance by aggregating the predictions from the independent experts, due to the inability of **suppressing poor experts**; on the contrary, the MoE provides the desirable predictions through gating functions.

Advancements and Developments:

Observing the aforementioned drawbacks there have been numerous advancements in PoE to recover from them:

- ➔ **Modifying the predicting process** : Various aggregation criteria have been proposed to weaken the votes of poor experts, in place of following the simple product rule to aggregate the experts' predictions. Generally the aggregated prediction is robust to weak experts providing a more accurate prediction. The following **modified product rule [11]** covers the necessities with weights β_i quantifying the contribution of $p_i(y_* | D_i, x_*)$ at x_* :

$$\mathbf{p(y_* | D, x_*)} = \prod_{i=1 \text{ to } M} \mathbf{p_i^{\beta_i}(y_* | D_*, x_*)} \qquad \text{[Eq General]}$$

4. Generalized Product of Experts (GPoE)[11]

GPoE introduces a varying weight β_i in equation [Eq General] of last section , which is defined as the difference in the differential entropy between the expert's prior and posterior, to increase or decrease the importance of experts based on their prediction uncertainty. However, with this flexible weight, the GPoE produces explosive prediction variance when leaving the training data. To address this issue, we can impose a

constraint $\sum_{i=1 \text{ to } M} \beta_i = 1$.

The **drawback** to GPoE is that it produces explosive prediction variance when leaving the training data and is thus inconsistent.

5. Bayesian committee machine (BCM)[12]

BCM aggregates the experts' predictions from another point of view by imposing a conditional independence assumption and explicitly introduces a common prior $p(y_* | \theta)$ for the experts :

$$p(y | y_*) \approx \prod_{i=1 \text{ to } M} p(y_i | y_*)$$

Thus from Bayes Rule we get the following **modified regression:**

$$p(y_* | D, x_*, \theta) = \left[\prod_{i=1 \text{ to } M} p_i^{\beta i}(y_* | D_*, x_*, \theta) \right] / \left[p_i^{\sum_{i=1 \text{ to } M} (\beta i-1)}(y_* | \theta) \right]$$

The **drawback** to BCM is that it produces unreliable prediction mean when leaving X, thus making it inconsistent.

6. Generalized Robust Bayesian committee machine (GRBCM)[13]

There have been advancements in BCM to overcome its inconsistency thus giving us generalized RBCM (GRBCM). To do so it introduces a global communication expert M_c rather than the fixed GP prior performing correction, i.e., acting as a base expert, and considers the covariance between global and local experts to support consistent predictions when $n \rightarrow \infty$.

Given $p_{+i}^{\beta i}(y_* | D_{+i}, x_*)$ as **predictive** distribution of the expert M_{+i} trained on the augmented data set $D_{+i} = \{D_i, D_c\}$, we get the following **generalized modified aggregation:**

$$p(y_* | D, x_*) = \left[\prod_{i=2 \text{to } M} p_{+i}^{\beta i}(y_* | D_{+i}, x_*) \right] / \left[p_i^{\sum_{i=2 \text{ to } M} (\beta i-1)}(y_* | D_c, x_*) \right]$$

The model **achieves** automatic regularization and eases the inference due to fewer hyperparameters and allows to temporarily ignore the noise term of GP in aggregation.

However there are certain **drawbacks** to this as well. One drawback of aggregations is the Kolmogorov inconsistency induced by the separation of training and predicting such that it is not a unifying probabilistic framework. Another drawback comes from shared

hyperparameters as those limit the capability of capturing nonstationary features, which is the superiority of local approximations.

Global Approximation

These approximations to the kernel matrix K_{nn} are made possible via global distillations. Following are the ways in which it can be achieved:

- **(a)** taking a subset of the training data with m (m<< n) points and reducing it into a smaller kernel matrix K_{mm}
- **(b)** By deleting uncorrelated entries in K_{nn} thereby creating a sparse kernel matrix $\tilde{K}_{nn}$ with many zero entries
- **(c)** m inducing points and n training points measure a low-rank representation which result in the Nyström approximation $K_{nn} \approx K_{nm} K^{-1}_{mm} K_{m}$

In order to achieve scalability, the sparsity of the full kernel matrix is necessary.

1. Using a subset of data

This is the simplest strategy to approximate the full GP by using a subset D_{sod} of the training data. The standard GP inference at a lower time complexity of $O(m^3)$ is retained through SoD because it operates on K_{mm}, comprising m (m<< n) data points. It produces reasonable prediction mean for the case with redundant data, but struggles in the case of overconfident prediction variance because of a limited subset. For selecting D_{sod} , one could randomly choose m points from D, use any clustering technique, like, k-means and KD tree, partitioning the data into m subsets and choosing their centroids as subset points, and employing active learning criteria, e.g., differential entropy, matching pursuit and information gain, to sequentially query data points with higher computing cost.

2. Using sparse Kernels

We want to get a sparse representation $\tilde{K}_{nn}$ of K_{nn} using the compactly supported (CS) kernel, imposing $k(xi, x j) = 0$ when $|x_i - x_j|$ exceeds a certain threshold. Therefore, only the non-zero terms in $\tilde{K}_{nn}$ will remain in the calculation. Subsequently, the GP using the CS kernel will be scaled as $O(\alpha n^3)$ with $0 < \alpha < 1$. The major challenge while constructing valid CS kernels would be to build a positive semidefinite (PSD) $\tilde{K}_{nn}$, i.e., $vT\tilde{K}_{nnv} \geq 0 \ \forall v \in R^n$.

3. Sparse Approximations

The Sherman–Morrison–Woodbury formula can be used to calculate the inversion as follows:

$$\left(\boldsymbol{K}_{nn}^{\epsilon}\right)^{-1} \approx \sigma_{\epsilon}^{-2}\boldsymbol{I}_n + \sigma_{\epsilon}^{-2}\boldsymbol{U}_{nm}\left(\sigma_{\epsilon}^{2}\boldsymbol{\Lambda}_{mm}^{-1} + \boldsymbol{U}_{nm}^{\mathsf{T}}\boldsymbol{U}_{nm}\right)^{-1}\boldsymbol{U}_{nm}^{\mathsf{T}}$$

and the Sylvester determinant theorem to calculate the determinant as

$$\left|\boldsymbol{K}_{nn}^{\epsilon}\right| \approx \left|\boldsymbol{\Lambda}_{mm}\right|\left|\sigma_{\epsilon}^{2}\boldsymbol{\Lambda}_{mm}^{-1} + \boldsymbol{U}_{nm}^{\mathsf{T}}\boldsymbol{U}_{nm}\right|$$

This results in the complexity of $\mathbf{O(nm^2)}$. The eigenfunctions of K_{nn} leading to the **Nyström approximation** as approximated as :

$$\mathbf{K_{nn} \approx Q_{nn} = K_{nm} K_{mm}^{-1} K_{nm}^{T}}$$

This enhances the large-scale kernel learning and the naive Nyström GP is allowed. Negative prediction variances may be produced by this scalable GP because it is not a complete generative probabilistic model. The Nyström approximation is only established on the training data and so does not warrant the PSD of the kernel matrix.

A set of inducing pairs (X_m, f_m) are introduced. The inducing variables f_m related to f follow the same GP prior $p(f_m) = N(0, K_{mm})$. Also, we assume f_m to be a sufficient statistic for f , i.e., for any variables z, $p(z \mid f, f_m) = p(z \mid f_m)$ holds true.

The sparse approximations have 3 main categories:

Prior Approximations:

We modify the joint prior using the independent assumption to get

$$\mathbf{p(f, f_*) = \int p(f \mid f_m)\, p(f_* \mid f_m)\, p(f_m)\, df_m}$$

where the training and test conditionals write, given a Nyström notation $Q_{ab} = K_{am} K_{mm}^{-1} K_{mb}$, and

$$\mathbf{p(f \mid f_m) = N(f \mid K_{nm}K_{mm}^{-1}f_m, K_{nn} - Q_{nn})}$$
$$\mathbf{p(f_* \mid f_m) = N(f_* \mid k_{*m}K_{mm}^{-1}f_m, k_{**} - Q_{**})}$$

where f_m is called inducing variables as the dependencies between f and f_* are induced by f_m. To obtain computational gains, we can modify the training and testing conditionals as follows

$$\mathbf{q(\, f \mid f_m\,) = N(\, f \mid K_{nm}K_{mm}^{-1}f_m,\ \tilde{Q}_{nn}\,)}$$
$$\mathbf{q(\, f_* \mid f_m\,) = N(\, f_* \mid k_{*m}K_{mm}^{-1}f_m,\ \tilde{Q}_{**}\,)}$$

Then, log p(y) is approximated by log q(y) as

$$\log q(\boldsymbol{y}) = -\frac{n}{2}\log 2\pi - \frac{1}{2}\log\left|\tilde{\boldsymbol{Q}}_{nn} + \boldsymbol{Q}_{nn} + \sigma_\epsilon^2 \boldsymbol{I}_n\right| - \frac{1}{2}\boldsymbol{y}^{\mathsf{T}}\left(\tilde{\boldsymbol{Q}}_{nn} + \boldsymbol{Q}_{nn} + \sigma_\epsilon^2 \boldsymbol{I}_n\right)^{-1}\boldsymbol{y}$$

It is found that with the help of specific selections of $\tilde{Q}_{nn}$, we can calculate $|\tilde{Q}_{nn} + Q^{nn} + \sigma^2 I_n|$ and $(\tilde{Q}_{nn} + Q_{nn} + \sigma^2 I_n)^{-1}$ with a complexity of $O(nm^2)$.
Deterministic training and test conditionals, i.e., $\tilde{Q}_{nn} = 0$ and $\tilde{Q}_{**} = 0$, are imposed by the subset of regressors (SoR), also known as deterministic-inducing conditional (DIC) as,

$$\mathbf{q_{SoR}(\, f \mid f_m\,) = N(\, f \mid K_{nm}K_{mm}^{-1}f_m,\ 0)}$$
$$\mathbf{q_{SoR}(\, f_* \mid f_m\,) = N(\, f_* \mid k_{*m}K_{mm}^{-1}f_m,\ 0)}$$

This is similar to applying the Nyström approximation to both training and testing data, which results in a degenerate GP with a rank of at most m kernel

$$\mathbf{k_{SoR}(\, x_i, x_j\,) = k(x_i, X_m)\, K_{mm}^{-1}\, k(X_m, x_j)}$$

The SoR can also be interpreted from a weight-space point. It is common knowledge that the GP using a kernel with an infinite expansion of the input x in the feature space defined by dense basis functions $\{\rho c(x)\} v\ c = 1$ is equivalent to a Bayesian linear model with infinite weights. Therefore, only m basis functions $\boldsymbol{\phi}_m(x) = [\boldsymbol{\phi}_1(x)\ ,\ \boldsymbol{\phi}_2(x)\ \ldots\ \boldsymbol{\phi}_m(x)]^T$ are used by the relevance vector machine (RVM) for approximation

$$\mathbf{p(f \mid w) = N(\, f \mid \Phi_{nm}\, w,\ K_{nn} - \Phi_{nm}\, \Sigma_{mm}\, \Phi_{nm}^{T}\,)}$$

Hence, the RVM is a GP.

Augmenting the basis functions at x to invert the behaviour of uncertainty$_*$ heals the RVM. However, this happens at a higher computing cost . On the other hand, the sparse

spectrum GP (SSGP) and its variational variants address the issue by reconstructing the Bayesian linear model from the spectral representation (Fourier features)

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\sigma_0^2}{m}\boldsymbol{\phi}_m^{\mathsf{T}}(\boldsymbol{x}_i)\boldsymbol{\phi}_m(\boldsymbol{x}_j) = \frac{\sigma_0^2}{m}\sum_{r=1}^{m}\cos\left(2\pi \boldsymbol{s}_r^{\mathsf{T}}(\boldsymbol{x}_i - \boldsymbol{x}_j)\right)$$

where $s_r \in R_d$ represents the spectral frequencies. Forcing greater informative assumptions to $Q\tilde{}_{nn}$ and $Q\tilde{}_{**}$ is another way. Consider for example, the deterministic training conditional (DTC) which uses this training conditional

$$q_{\mathrm{DTC}}(\boldsymbol{f}|\boldsymbol{f}_m) = \mathcal{N}\left(\boldsymbol{f}|\boldsymbol{K}_{nm}\boldsymbol{K}_{mm}^{-1}\boldsymbol{f}_m, \boldsymbol{0}\right)$$

The exact testing conditional is retained. The mean of prediction is equal to that of SoR, but the variance of prediction is always larger than that of SoR and it grows to the prior when inducing points are left out. Notably, the inconsistent conditionals make the DTC an inexact GP. Besides, the DTC and SoR often perform poorly due to restrictive prior assumption $Q\tilde{}_{nn} = 0$. The fully independent training conditional (FITC) [67] nullifies the dependence among { fi} n i=1 in such a way that given $V_{nn} = K_{nn} - Q_{nn}$, the training conditional qFITC(f | f_m) equals

$$\prod_{i=1}^{n} p(f_i|\boldsymbol{f}_m) = \mathcal{N}\left(\boldsymbol{f}|\boldsymbol{K}_{nm}\boldsymbol{K}_{mm}^{-1}\boldsymbol{f}_m, \mathrm{diag}[\boldsymbol{V}_{nn}]\right)$$

With the testing conditional retaining its exactness. The variances of (12) are identical to that of p(f | f_m) due to the correlation $Q\tilde{}_{nn} = \mathrm{diag}[V_{nn}]$. Hence, in comparison to SoR and DTC that throw away the uncertainty, FITC retains it to some extent, which causes a closer approximation of the prior p(f , f∗). Moreover, the full independence assumption extends to q(f_*| f_m). This derives the fully independent conditional (FIC) model, and makes it a nondegenerate GP with kernel:

$$k_{\mathrm{FIC}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = k_{\mathrm{SoR}}(\boldsymbol{x}_i, \boldsymbol{x}_j) + \delta_{ij}[k(\boldsymbol{x}_i, \boldsymbol{x}_j) - k_{\mathrm{SoR}}(\boldsymbol{x}_i, \boldsymbol{x}_j)]$$

where δ ij is the Kronecker delta. kFIC has a constant prior variance. However, it is not stationary. Alternatively, the approximation (12) can be derived from minimizing the Kullback–Leibler (KL) divergence

$$\mathrm{KL}(p(\boldsymbol{f}, \boldsymbol{f}_m)||q(\boldsymbol{f}_m)\textstyle\prod_{i=1}^{n} q(f_i|\boldsymbol{f}_m))$$

This quantifies the similarity between the exact and approximated joint prior. The partially independent training conditional (PITC) [16] has the training conditional qPITC(f | f_m) to improve FIT(C)

$$\prod_{i=1}^{M} p(\boldsymbol{f}_i|\boldsymbol{f}_m) = \mathcal{N}\left(\boldsymbol{f}|\boldsymbol{K}_{nm}\boldsymbol{K}_{mm}^{-1}\boldsymbol{f}_m, \text{blkdiag}[\boldsymbol{V}_{nn}]\right)$$

This is the same as partitioning training data D into M independent subsets (blocks) $\{D_i\}^M_{i=1}$ and considering the joint distribution of f_i in each subset. But, although it is a closer approximation to p(f | f_m), the blocking qPITC(f | f_m) brings only slight improvements over FITC [41]. This issue is addressed by the extended partially independent conditional (PIC) discussed in Section V-B.

Posterior Approximations:

These retain the exact prior but perform only an approximate inference. A highly familiar method, the elegant variational free energy (VFE) was proposed by Titsias by employing variational inference (VI) . VFE directly approximates the posterior p(f , f_m | y), the learning of which is a central task in statistical models, by introducing a variational distribution q(f , f_m | y). We now have their KL divergence KL(q(f , f_m | y) | | p(f , f_m | y))

$$\log p(\boldsymbol{y}) - \left\langle \log \frac{p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{f}_m)}{q(\boldsymbol{f}, \boldsymbol{f}_m|\boldsymbol{y})} \right\rangle_{q(\boldsymbol{f}, \boldsymbol{f}_m|\boldsymbol{y})} = \log p(\boldsymbol{y}) - F_q$$

$\langle . \rangle_{q(.)}$ representing the expectation over the distribution q(.). Minimizing the rigorously defined KL(q | | p) ≥ 0 is equivalent to maximizing F_q , since log p(y) is constant for q(f , f_m | y). F_q is therefore called evidence lower bound (ELBO) or VFE. Hence, this makes it possible to jointly optimize the variational parameters and hyperparameters. To derive a tighter bound, variational calculus finds the optimal variational distribution q_*(f_m | y) to remove the dependence of F_q on q(f_m | y) by taking the relevant derivative to be zero. This results in a collapsed bound as shown:

$$F_{\text{VFE}} = \log q_{\text{DTC}}(\boldsymbol{y}) - \frac{1}{2\sigma_\epsilon^2}\text{tr}[\boldsymbol{V}_{nn}] \geq F_q .$$

It is worth noting that F_{VFE} differs with log q_{DTC} only by a trace term, which, however, substantially improves the inference quality. We should decrease the trace tr[V_{nn}] ≥ 0 , which represents the total variance of predicting the latent variables f given fm, for maximizing F_{VFE}, . Specifically , tr[Vnn] = 0 implies f_m = f and the full GP is recovered. The trace term is a regularizer that:

1) guards against overfitting,
2) seeks to deliver a good inducing set, and
3) always improves F_q with increasing m (see the theoretical analysis.

The improvements of VFE were extended to continuous and discrete inputs. There was also an improvement in the estimation of inducing points. A mixture prior is assigned on X in the latent feature space, with some regularization bound for choosing good inducing points.
The FITC and VFE are interpreted jointly as

$$\log q_{\mathrm{PEP}}(\boldsymbol{y}) = \log q(\boldsymbol{y}) - \frac{1-\alpha}{2\alpha}\mathrm{tr}\left[\log\left(\boldsymbol{I}_n + \frac{\alpha}{\sigma_\epsilon^2}\boldsymbol{V}_{nn}\right)\right]$$

where log q(y) takes the form with $\tilde{Q}_{nn} = \alpha diag[V_{nn}]$. By changing $\alpha \in (0, 1]$, recovering FITC when $\alpha = 1$ and VFE when $\alpha \to 0$. Beside, better predictions can be made by using hybrid approximation with a moderate α, e.g., $\alpha = 0.5$. To further improve the scalability of VFE, Hensman et al. retained the variational distribution q(f_m | y) = N (f_m | m, S) in F_q to obtain a relaxed bound

$$F_q = \langle \log p(\boldsymbol{y}|\boldsymbol{f})\rangle_{p(\boldsymbol{f}|\boldsymbol{f}_m)q(\boldsymbol{f}_m|\boldsymbol{y})} - \mathrm{KL}(q(\boldsymbol{f}_m|\boldsymbol{y})||p(\boldsymbol{f}_m))$$

The first term in the right-hand side of F_q is the sum of n terms due to the i.i.d. observation noises, i.e., $p(\boldsymbol{y}|\boldsymbol{f}) = \prod_{i=1}^{n} p(y_i|f_i)$
Hence, the stochastic gradient descent (SGD) , which encourages large-scale learning, could be employed to obtain an unbiased estimation of Fq using a minibatch $\{X_b, y_b\}$ as

$$F_q \approx \frac{n}{|\boldsymbol{y}_b|}\sum_{y_i \in \boldsymbol{y}_b}\int q(\boldsymbol{f}_m|\boldsymbol{y})p(f_i|\boldsymbol{f}_m)\log p(y_i|f_i)df_i d\boldsymbol{f}_m - \mathrm{KL}(q(\boldsymbol{f}_m|\boldsymbol{y})||p(\boldsymbol{f}_m)).$$

Although the SVGP is highly scalable with desirable approximations, it has some drawbacks which have been mentioned below

1) the bound F_q is less tight than F_{VFE} because q(f_m | y) is not optimally eliminated.
2) it optimizes over q(f_m | y) with a huge number of variational parameters. This requires a lot of time to complete even one epoch of training.
3) the introduction of SVI brings the empirical requirement of carefully turning the parameters of SGD.

Inspired by the idea of Hensman, Peng derived the similar factorized variational bound for GPs by taking the weight-space augmentation. By deploying the variational model in a distributed machine learning platform PARAMETER SERVER , the authors first scaled GP up to billions of data points.

Similarly, Cheng and Boots also derived a stochastic variational framework from the weight space view with the difference being that the mean and variance of p(f |w), respectively, using decoupled basis function sets ϕ_a and ϕ_b, led to a more flexible inference.
Conducting a reverse VI in which "reverse" meant finding a prior p(f_m) = N (f_m| ν ,) such that the variational distribution q_*(f_m| y) = p(f_m| y) for FI(T)C and PI(T)C is the maximum of ELBO was the key. Introducing Kronecker structures for inducing points and the variance of q(f_m| y) further reduces the scalability of SVGP.

Titsias and Hensman's models have been further **improved** by using:

1) the Bayesian treatment of hyperparameters in place of traditional point estimation which risks overfitting when the number of hyperparameters is small.
2) the non-Gaussian likelihoods

Structured Sparse Approximations:

A direct speedup to solve $(K_{nn})^{-1}y$ in standard GP is achievable through fast matrix-vector multiplication, in which the linear system is solved iteratively using conjugate gradients (CGs) with $s(s << n)$ iterations, leading to a time complexity of $O(sn^2)$.
The kernel matrix decomposes to a Kronecker product $K_{nn} = K1....Kd,$ which eases the eigendecomposition with a greatly reduced time complexity of $O(d\overline{n}^{d+1})$ where $\overline{n} = \sqrt[d]{n},$ for d>1.

Grid constraint on the inducing points is introduced by the structured kernel interpolations. This is done to handle arbitrary data, retaining the efficient Kronecker structure at the same time.
For example by a local linear interpolation using the adjacent grid inducing points as

$$k(xi,\ u\,j) \approx wi\ k(ua,\ u\,j)\ +\ (1 - wi)k(ub,\ u\,j)$$

where u_a and u_b are two inducing points most closely bound x_i, and w_i is the interpolation weight. Inserting the approximation back into Q_{nn}, we have

$$Q_{nn} \approx W nm\, K_{mm}^{-1} W_{mm}^{T}$$

where the weight matrix W is extremely sparse since it only has two nonzero entries per row for local linear interpolation, leading to an impressive time complexity of $O(n + d\overline{m}^{d+1})$ with $\overline{m} = \sqrt[d]{m}$ for solving $(K_{nn})^{-1}y$. Also, the sparse W incurs the prediction mean with constant-time complexity O(1) and the prediction variance with complexity O(m) after precomputing.

There are two main drawbacks of the original SKI. Firstly, the number m of grid inducing points grows exponentially with dimensionality d. Thus, it is quite impractical for d > 5.

Secondly, the SKI might produce discontinuous predictions because of local weight interpolation and provide overconfident prediction variance when leaving the training data due to the restrictive SoR framework.
Due to the grid constraint, the structured sparse approximations use fixed inducing points, resort to dimensionality reduction for tackling high-dimensional tasks, and place the vast majority
of inducing points on the domain boundary with increasing d, which in turn may degenerate the model capability.

Choosing Inducing Points: for the inducing size, according to theoretical analysis the KL divergence between the variational approximation and the posterior can be arbitrarily small when m grows more slowly than n for regression with normally distributed inputs and the SE kernel.For the location of inducing points, clustering techniques could be used to select a finite set of space-filling inducing points. In a recent work, the first attempt to simultaneously determine the number and locations of inducing points in the Bayesian framework by placing a prior on Xm was shown.

AUTOMATIC LEARNING THE KERNEL ITSELF[19]

The choice of kernel which determines the right structure that can be learned by a GP model is still a black art. We solve this by defining an open-ended space of kernels and a procedure to search over this space and find a match according to the data structure.

1. Ingredients Of An Automatic Statistician

Currently Humans construct new models and check models. This subdivision tries to build an AI that does statistics which paves the road to the above goal.

1. **An open-ended language of models:**

Humans couldn't have reached this far in history without their sense of novelty. An automatic search through an open-ended class can achieve some of this flexibility, possible combining existing structures in novel ways.

2. **A search through model space:**

Humans researchers iteratively build and refine models using an iterative search procedure which starts from simple models. Thus, any search strategy capable of building arbitrarily complex models is likely to resemble an iterative model-building procedure.

3. **A model comparison procedure:**

An automatic statistician needs to somehow check the models it has constructed. This is done by penalising complexity using Bayesian information criterion as heuristics and approx. marginal likelihood to compare models.

4. **A model description procedure:**

The wholesome purpose is achieved when it flawlessly coordinates with the human brain thus a clear picture is presented. It helps a user to understand errors, structures, procedures, constraints and novel statistics that have been implemented.

Thus the ABCD (Automatic Bayesian Covariance Discovery) system is implemented.

2. A Language Of Regression Models

One can construct a wide variety of kernel structure just by adding and multiplying a small number of base kernels. Thus, a language of models is constructed using a language of kernels, such that they capture different properties of functions and set of rules which combine kernels to yield valid kernels. Hence, we use such base kernels as white noise (WN), constant (C), linear (Lin), squared-exponential (SE), rational-quadratic (RQ), sigmoidal (σ) and periodic (Per). We use a form of Per due to James Lloyd (personal communication) which has its constant component removed, and $\cos(x - x')$ as a special case.

3. A Model Search Procedure

Simple Greedy Search is used to span the space. We choose the highest scoring kernel and modify it by combining or replacing that part with another base kernel. The operations performed are:

Replacement: $k \rightarrow k'$

Addition: $k \rightarrow (k + k')$

Multiplication: $k \rightarrow (k \times k')$

In practice, extra operators are used which propose commonly-occurring structures, such as changepoints.

Parallels with strategies used by human researchers:

- Look for the structure in the residuals of a model, such as periodicity, and then extend the model thus, adding a new kernel to the existing structure.
- Starts with the structure which is assumed to hold globally, such as linearity, but finds that it only holds locally. This corresponds to multiplying a kernel structure by a local kernel such as SE.
- Incorporates input dimensions incrementally, analogous to algorithms like boosting, back-fitting, or forward selection. This corresponds to adding or multiplying with kernels on dimensions not yet included in the model.

Hyperparameter initialization:

Optimizing the marginal likelihood over parameters is not a convex optimization problem, and the space can have many local optima. We take advantage of our search procedure to provide reasonable initializations. All parameters which were part of the previous kernel are initialized to their previous values. All newly introduced parameters

are initialized randomly. Then all the parameters are optimized using conjugate gradients. This method is not guaranteed to find global optimum, but it implements the commonly used heuristic of iteratively modelling residuals.

4. A Model Comparison Procedure

Choosing a kernel requires a method for comparing models. We choose marginal likelihood as our criterion, since it balances the fit and complexity of a model (Rasmussen and Ghahramani, 2001).We could avoid overfitting by integrating the marginal likelihood over all free parameters, but this integral is difficult to do in general. Instead, we loosely approximate this integral using the Bayesian information criterion (BIC)

$$BIC(M) = \log_P(D \mid M) - \frac{1}{2}|M| \log N$$

where p(D|M) is the marginal likelihood of the data evaluated at the optimized kernel parameters, |M| is the number of kernel parameters, and N is the number of data points.

BIC simply penalizes the marginal likelihood in proportion to how many parameters the model has.The assumptions made by BIC are clearly inappropriate for the model class being considered.Other more sophisticated approximations are possible, such as Laplace's approximation and BIC is chosen because of its simplicity.

5. A Model Description Procedure

A GP whose kernel is a sum of kernels can be viewed as a sum of functions drawn from different GPs.

$$SE \times (RQ + Lin) = SE \times RQ + SE \times Lin$$

This decomposition into additive components provides a method of visualizing GP models which disentangles the different types of structure in the model

6. Structure discovery in time series

A) Mauna Loa atmospheric CO2:

First, our method analyzed records of carbon dioxide levels recorded at the Mauna Loa observatory (Tans and Keeling, accessed January 2012).On comparing our kernel with human experts' kernel it shows that the posterior mean and variance on this dataset as the search depth increases. The final model exhibits plausible extrapolation and interpretable components: a long-term trend, annual periodicity, and medium-term deviations.

B) Airline passenger counts:

On applying our method monthly to airline passenger counts we observe the similar trends of Mauna Loa data set and in addition to that we observe near linearity of long term trend and linearly growing amplitude of annual oscillations.

7. Related Work

Nonparametric regression in high dimensions

Nonparametric regression such as splines, locally weighted and GP regression cant generalize well in more than a few dimensions.Applying these methods in high dimensional spaces can require imposing additive structure.Generalized additive models assume the regression function is a transformed sum of functions defined on the individual dimensions. It is possible to extend additive models by adding more flexible interaction terms between dimensions. A related procedure is smoothing splines. This model is a weighted sum of splines along each input dimension, all pairs of dimensions, and possibly higher dimensional combinations. Because the number of terms to consider grows exponentially with the number of dimensions.

8. Kernel Learning

There is a large body of work attempting to construct rich kernels through a weighted sum of base kernels, called multiple kernel learning (MKL) .These approaches usually have a convex objective function. However the component kernels, as well as their parameters, must be specified in advance.

Salakhutdinov and Hinton (2008) use a deep neural network with unsupervised pre training to learn an embedding g(x) onto which a GP with an SE kernel is placed: Cov [f(x), f(x′)] = k(g(x), g(x′)). This is a flexible approach to kernel learning, but relies mainly on finding structure in the input density p(x). Instead, we focus on domains where most of the interesting structure is in f(x).

9. Changepoints

There is a wide body of work on changepoint modeling. Adams and MacKay (2007) developed a Bayesian online changepoint detection method which segments time-series into independent parts. This approach was extended by Saatçi et al. (2010) to Gaussian process models. Garnett et al. (2010) developed a family of kernels which modeled changepoints occurring abruptly at a single point. The changepoint kernel (CP) presented in this work is a straightforward extension to smooth changepoints.

10. Equation learning

Todorovski and Džeroski (1997), Washio et al. (1999) and Schmidt and Lipson (2009) learned parametric forms of functions, specifying time series or relations between quantities. In contrast, ABCD learns a parametric form for the covariance function, allowing it to model functions which do not have a simple parametric form but still have high level structure.

EXPERIMENTS

1. Experiments on Automatic Model Construction[19]:

1.1 Interpretability versus accuracy

BIC trades off model fit and complexity by penalizing the number of parameters in a kernel expression. This can result in ABCD favoring kernel expressions with nested products of sums, producing descriptions involving many additive components after expanding out all terms.While these models typically have good predictive performance, their large number of components can make them less interpretable.

The experiment was done without allowing parentheses during the search, discouraging nested expressions. This was done by distributing all products immediately after each search operator was applied. We call this procedure ABCD-interpretability, in contrast to the unrestricted version of the search, ABCD-accuracy.

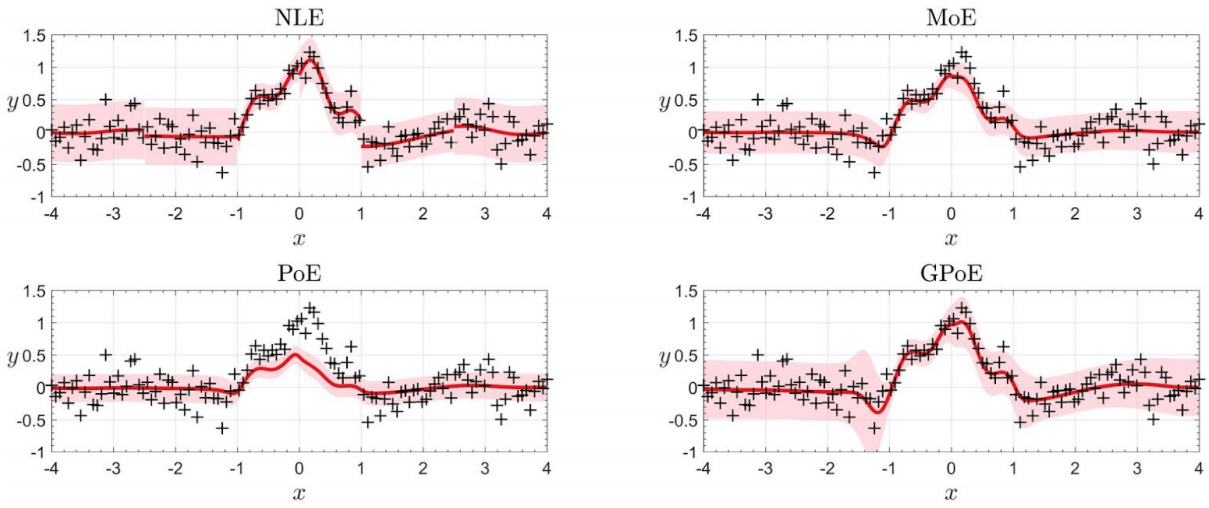
1.2 Structure recovery on synthetic data

We tested our method's ability to recover known structure on a set of synthetic datasets. For several composite kernel expressions, we constructed synthetic data by first sampling 300 locations uniformly at random, then sampling function values at those locations from a GP prior. We then added i.i.d. Gaussian noise to the functions at various signal-to-noise ratios (SNR). Table below : Kernels chosen by ABCD on synthetic data generated using known kernel structures. D denotes the dimension of the function being modeled. SNR indicates the signal-to-noise ratio. Dashes (–) indicate no structure was found. Each kernel implicitly has a WN kernel added to it.

True kernel	D	SNR = 10	SNR = 1	SNR = 0.1
$\mathrm{SE} + \mathrm{RQ}$	1	SE	$\mathrm{SE} \times \mathrm{Per}$	SE
$\mathrm{Lin} \times \mathrm{Per}$	1	$\mathrm{Lin} \times \mathrm{Per}$	$\mathrm{Lin} \times \mathrm{Per}$	SE
$\mathrm{SE}_1 + \mathrm{RQ}_2$	2	$\mathrm{SE}_1 + \mathrm{SE}_2$	$\mathrm{Lin}_1 + \mathrm{SE}_2$	Lin_1
$\mathrm{SE}_1 + \mathrm{SE}_2 \times \mathrm{Per}_1 + \mathrm{SE}_3$	3	$\mathrm{SE}_1 + \mathrm{SE}_2 \times \mathrm{Per}_1 + \mathrm{SE}_3$	$\mathrm{SE}_2 \times \mathrm{Per}_1 + \mathrm{SE}_3$	–
$\mathrm{SE}_1 \times \mathrm{SE}_2$	4	$\mathrm{SE}_1 \times \mathrm{SE}_2$	$\mathrm{Lin}_1 \times \mathrm{SE}_2$	Lin_2
$\mathrm{SE}_1 \times \mathrm{SE}_2 + \mathrm{SE}_2 \times \mathrm{SE}_3$	4	$\mathrm{SE}_1 \times \mathrm{SE}_2 + \mathrm{SE}_2 \times \mathrm{SE}_3$	$\mathrm{SE}_1 + \mathrm{SE}_2 \times \mathrm{SE}_3$	SE_1
$(\mathrm{SE}_1 + \mathrm{SE}_2) \times (\mathrm{SE}_3 + \mathrm{SE}_4)$	4	$(\mathrm{SE}_1 + \mathrm{SE}_2) \times \ldots$ $(\mathrm{SE}_3 \times \mathrm{Lin}_3 \times \mathrm{Lin}_1 + \mathrm{SE}_4)$	$(\mathrm{SE}_1 + \mathrm{SE}_2) \times \ldots$ $\mathrm{SE}_3 \times \mathrm{SE}_4$	–

The table above shows the results. For the highest signal-to-noise ratio, ABCD usually recovers the correct structure. The reported additional linear structure in the last row can be explained that the fact that functions sampled from SE kernels with long scales occasionally have near-linear trends. As the noise increases, our method generally backs off to simpler structures rather than reporting spurious structure.

2. Experiments on Local Approximations[17]:



Above are some experimental results carried out in [17]. The experiment considered 6-independent experts. The experimenters for the sake of a rattlefree implementation used the individual experts and the differential entropy as β_i in the softmax gating function.

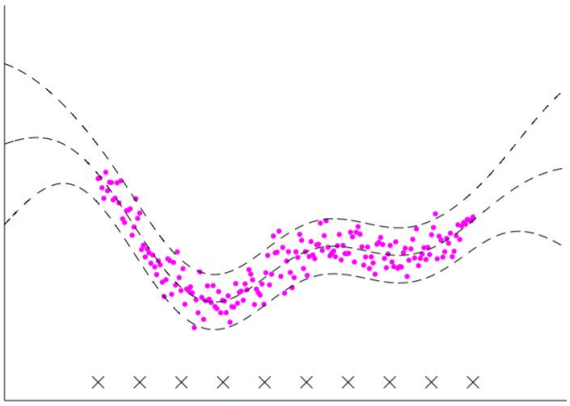
PoE : It was observed that PoE generates the worst predictions among all models being compared. It poorly estimated the mean along with an overconfident prediction of the kernel matrix and individual standard deviations. The theoretical reason is that it aggregates the predictions from six independent experts and is unable to hide weaker experts as already explained in previous sections on PoE.

MoE : Secondly it is observed that MoE provided better predictions that can be directly mapped to the fact that it introduces betterment in terms of gating functions.

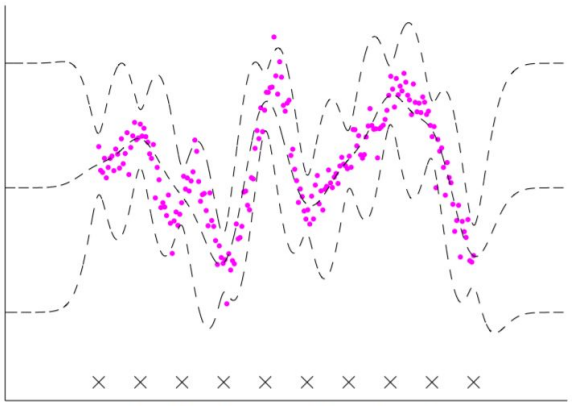
NLE : For NLE a general trend of improper generalizability and discontinuity was observed.

GPoE : The best predictions were observed with GPoE eliminating considerably the "boundary discontinuities" of NLE model.

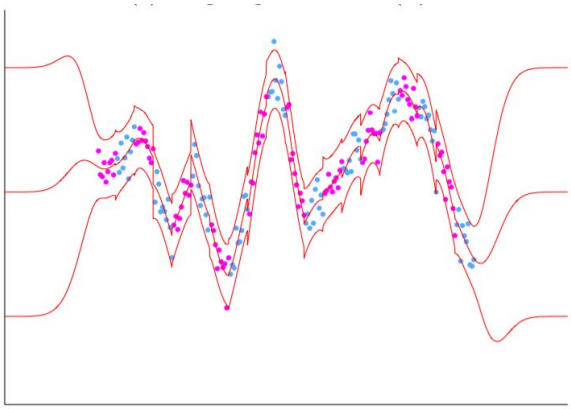
3. Experiments on Global - Local and Combined Approximations[18]



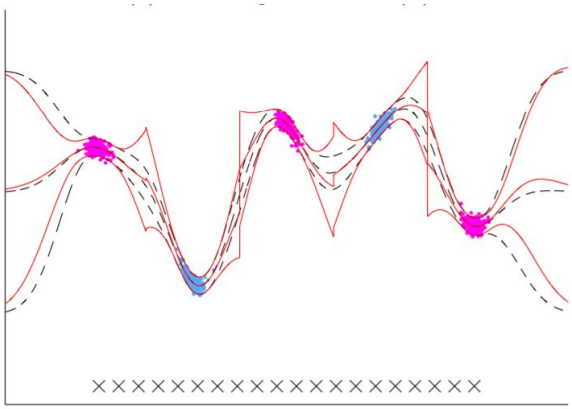
(a) Long lengthscale — FI(T)C



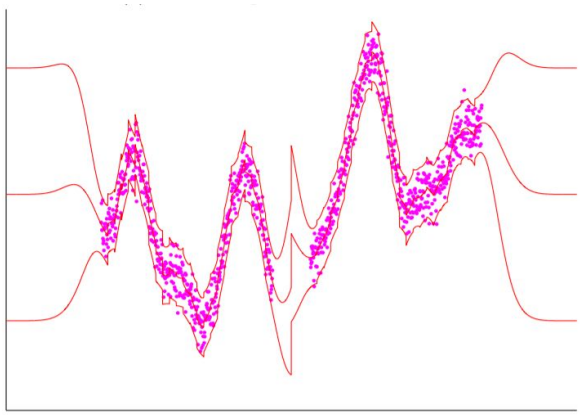
(b) Short lengthscale — FI(T)C



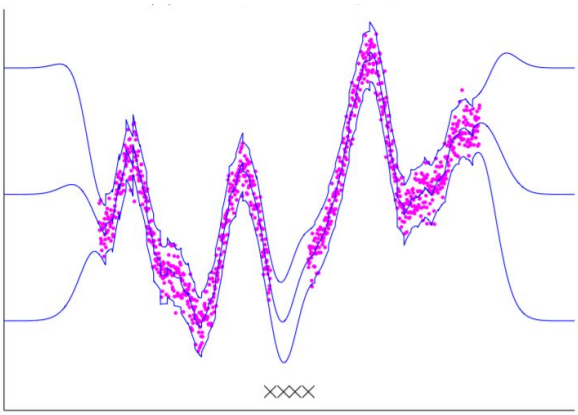
(c) Short lengthscale — local GPs



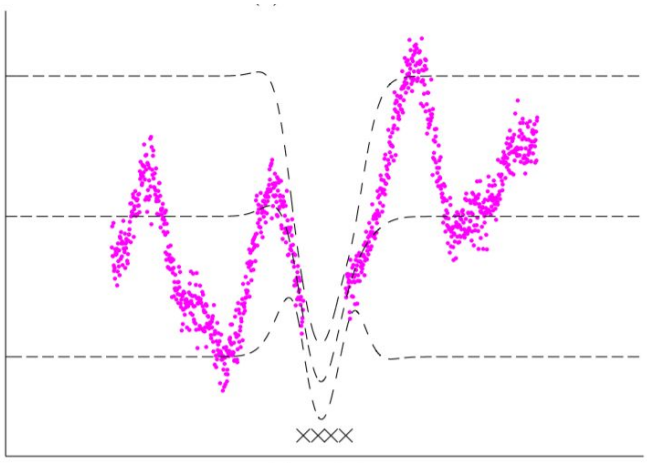
(d) Clumped training inputs



(e) local GPs



(f) PIC



(g) FI(T)C

One way of thinking of an improvement is to develop a middle way round the two approximations i.e. a combination of global and local approximations. This kind of approximation (partially independent conditional (PIC) approximation) might be able to deal with all the regimes handled individually by these two approximations. Although studying this completely seemed a bit extravagant for this term paper and thus we will be skipping through the literature and theoretical aspects of the model and directly present another experiment from [18] giving us a graphical analysis of global, local and combined sparse GP approximations. The graphs consider mean predictions and two standard deviation error lines as follows: [Please note local blocks in some figures are omitted due to minimal-unobservable size]

FI(T)C : represented using black dashed lines

Local Approximations : represented using red solid lines for local GPs

PIC : represented using blue solid lines

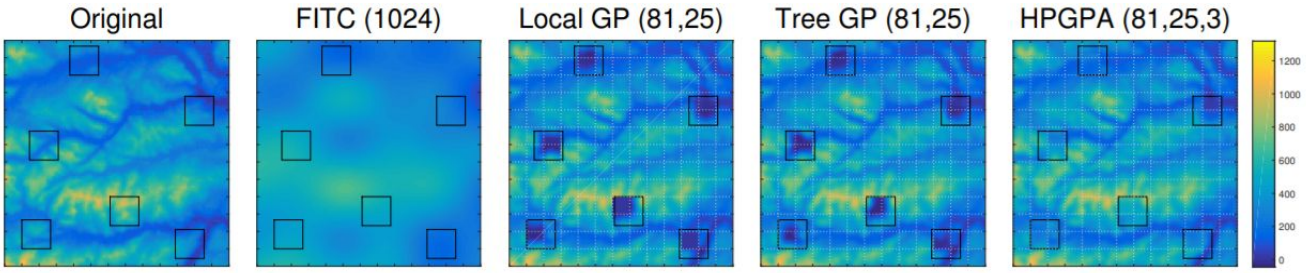
Black x pointers are used to represent the inducing inputs.

Note that FI(T)C predictions are plotted, using just 10 inputs.

Local training blocks are represented using different alternating color scheme i figures © and (d).

The local GP in fig a has a lot of discontinuity but apart from that it would actually work pretty well for such a longer length scale. But in fig d input data i.e the training inputs have been sampled in a non-uniform manner and local GP is unable to handle it. In this case FI(T)C predictions are much better as theoretically they are able to account for the correlations between different clusters and remove discontinuities in a better way. But again PIC outperforms both in this case.
It is observed that in fig e, local GP lacks in producing desired outputs as it couldn't handle boundary cases and predictions near boundaries of two groups. However, on the same data, using PIC gave pretty accurate predictions. Also, the best portions of fig e (local GP) and fig g (FI(T)C) can be observed in figure f (PIC) as well. Thus laying out the observation that combined approach PIC surpasses both the local GP approach and FI(T).

4. Experiments on Hierarchical-Partitioned Approximations[20]



The experiments carried out in [20] graphically differentiate between the performances of kernel approximations of FI (T)C, local GP, a tree-GP and a Hierarchical Partitioned Gaussian Process Approximation (HPGPA) designed by the authors themselves.. It is essentially done by employing 729 × 729 (531k) sized data (originally down-sampled from 3645 × 3645 data) and plotting the predictions corresponding to the goal of predicting the altitude of corresponding location . The data to be masked from the train set was probability proportional to the variance of altitudes with the aim to offer a challenging interpolation. The tree in HPGPA has every non-leaf node with either 9 or 81 furthus branches, and parallely author selected length-scales such that increase/decrease at each level proportional to the area of the region covered by the block.

As observed from above figure :
FITC because of a lack of inducing points predicts poorly giving a blurry output.
Local GP is unable to interpolate as the kernel is single scale (leads to formation of sink-holes). Same is the case for Tree-GP.
HPGPA predicts the output (which is of the form of an image) very closely to the original image as it is able to correctly predict on the basis of complex trends with the hierarchical multi-scale modeling.

: FITC does not scale well, local/tree GP makes inaccurate predictions although quite fast while HPGPA clearly makes better predictions than other methods.

5. Experiments on Comparing All Approximations:

So far, we have discussed and theoretically analyzed the various approximation schemes including various types of local, global, combined and hierarchical approximations. But we tried to realize the practical limit of these approximation schemes as all the theoretical analysis only gives us an asymptotic bound on the running time, whereas for practical purposes it may differ.

Our main focus was implementing a code similar to that described in paper on "A Framework for Evaluating Approximation Methods for Gaussian Process Regression" by Krzysztof Chalupka et al [22]. The framework given in [22] is suitable for SoD (subset of Data) and FI(T)C . We tried to extent at apply the same framework to GPoE and RBCM models following a similar concept where we the decompose the running time into two parts for analysis:

1. Time for training the models as well as the time taken for hyperparameter optimization.
2. Time required for the test cases i.e. the testing time.

This time-division allows us to do a task-specific analysis and comparison of various models. We used a smaller subset of the dataset SARCOS from http://www.gaussianprocess.org/ , the implementation had a few errors while extending to Local Approximations and we were unable to observe the predictions corresponding to the same.

6. Our Experiments: FI(T)C vs SE Kernel GP for COVID Predictions

We implemented the following two and compared their predictions:

1. Simple Squared Exponential Kernel with following specifications:
    a. Sigma_f (SE Kernel Variance) = 0.6
    b. l = 95
    c. Gaussian noise = .1
2. FI(T)C with following specifications:
    a. A Constant mean function with the peak as the constant value [We also experimented with a linear mean kernel but the predictions were more accurate in constant case.]
    b. We used an Exponential Quadratic Covariance Function with lengthscale = 0.1 and input_dimension as 1 (default for a scalar or PyMC3 random variable)
    c. We defined a set of median values of train data as Xu i.e. the set of inducing points

The main code snippet for FI(T)C approx is as shown below.
Note : For FI(T)C python implementation we used python's **PyMC3** library.

```
with pm.Model() as model:
    # Specify the covariance function.
    # using
    #The Exponentiated Quadratic kernel.  Also refered to as the Squared
    #Exponential, or Radial Basis Function kernel.
    #.. math::
    #   k(x, x') = \mathrm{exp}\left[ -\frac{(x - x')^2}{2 \ell^2} \right]

    #ls: Lengthscale.  If input_dim > 1, a list or array of scalars or PyMC3 random
    #variables.  If input_dim == 1, a scalar or PyMC3 random variable.
    cov_func = pm.gp.cov.ExpQuad(1, ls=1)

    figre, axs = plt.subplots()

    mean_func = pm.gp.mean.Constant(mn)

    gp = pm.gp.MarginalSparse(mean_func = mean_func, cov_func=cov_func, approx="FITC")

    # Place a GP prior over the function f.
    sigma = pm.HalfCauchy("sigma", beta=3)
    y_ = gp.marginal_likelihood("y", X=X, Xu=Xu, y=y, noise=sigma)

    mp = pm.find_MAP()
    trace = pm.sample(1000)
    print(trace, "\n")

...
```

We then obtain the prediction of test set Xs and plot it using matplotlib pyplot functions:

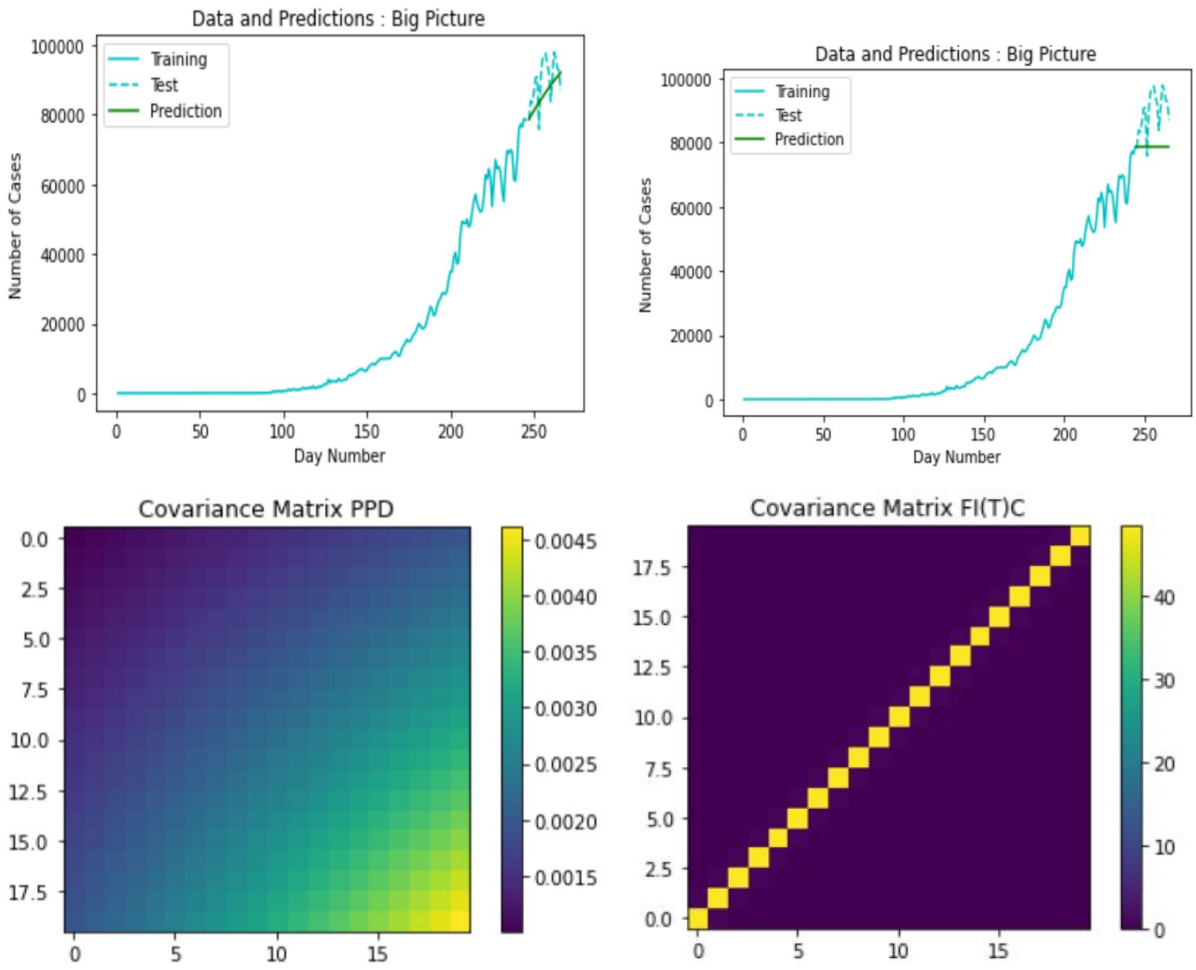
```
Xs = np.linspace(245, 265, 20)[:, None]
with model:
    samples = gp.predict(Xs, point=trace[-1], diag=False, pred_noise=0.1, given=None)# (trace,

figr, axs = plt.subplots()

plt.plot(Xtrain, Ytrain_India, 'c')
plt.plot(Xs, Ytest_India, 'c--')
plt.plot(Xs, samples[0], 'g')
plt.xlabel("Day Number")
plt.ylabel("Number of Cases")
plt.legend(["Training", "Test","Prediction"])
plt.title("Data and Predictions : Big Picture")

plt.show()
```

We also visualize the resulting covariance matrix corresponding to the predictions with the help of the following code:

```
figre, axs = plt.subplots()
plt.imshow(samples[1], interpolation='none')
plt.title("Covariance Matrix FI(T)C")
axs.set_ylim(axs.get_ylim()[::-1])
plt.colorbar()
plt.show()
```

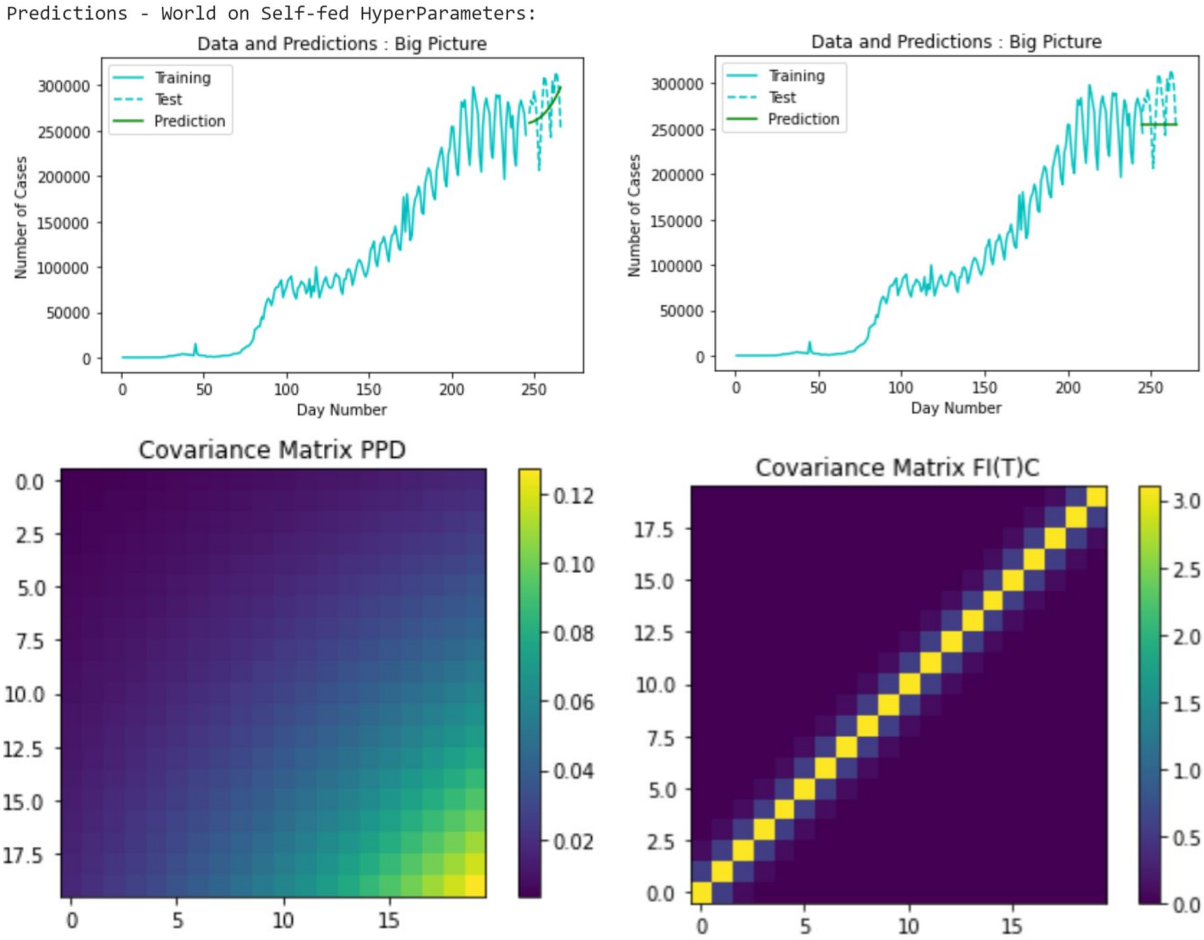
Observations India:

SE-Kernel GP | **FI(T)C**



Observations World:

SE-Kernel GP **FI(T)C**



Conclusions of Our Experiment...

The predictions of FI(T)C were observed to be poorer than the simple Squared Exponential Function GP. We observed this might have happened probably because of the following reasons:

1. The approximation hyperparameters were not optimized and user-fed, whereas we did a hyperparameter optimization for Simple SE-Kernel GP.
2. The choice of covariance function might have created poor predictions, as we chose exponential quadratic covariance function.
3. The output variance of predicted data appears very symmetric along the diagonal for FITC whereas for simple GP it is much concentrated along the lower right corner. We believe that choosing a more complex covariance function such as periodic, linear, cosine, polynomial, gibbs covariance functions.
4. A poor choice of induced points might have caused the predictions to go wrong in FITC. DataSet Source : https://ourworldindata.org/coronavirus-source-data

DISCUSSIONS AND FUTURE WORK

1. One way of thinking of an improvement is to develop a middle way round the two approximations i.e. a combination of global and local approximations. This kind of approximation might be able to deal with all the regimes handled individually by these two approximations. Although studying this completely seemed a bit extravagant for this term paper and thus we skipped through the literature and theoretical aspects of the model and directly presented another experiment from [18] giving us a graphical analysis of global, local and combined sparse GP approximations.
2. Another notable point while studying sparse approximations as well as local approximations was an inherent need of better, goal specific clustering techniques for the heuristics. For instance for a combined model a random clustering which picks; with no replacements; the cluster centers randomly from the training input points. The running time of this can go upto O(NS) training time and O(S) test time per test case. Thus there seems to be a need for betterment of these techniques to improve these algorithms from base.
3. Although PIC is a better approximation inculcating good features of both Local And Global Approximation, one way of thinking can be merge it with a more latest hierarchical version and thus allowing it to run on much larger datasets as compared to the conventional PICs in much faster time.

REFERENCES

[1] H.-M. Kim, B. K. Mallick, and C. C. Holmes, “Analyzing nonstationary spatial data using piecewise Gaussian processes,” J. Amer. Stat. Assoc., vol. 100, no. 470, pp. 653–668, 2005.

[2] R. Urtasun and T. Darrell, “Sparse probabilistic regression for activity independent human pose inference,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2008, pp. 1–8.

[3] C. Park and J. Huang, “Efficient computation of Gaussian process regression for large spatial data sets by patching local Gaussian processes,” J. Mach. Learn. Res., vol. 17, no. 174, pp. 1–29, 2016.

[4]] R. B. Gramacy et al., “LaGP: Large-scale spatial modeling via local approximate Gaussian processes in R,” J. Stat. Softw., vol. 72, no. 1, pp. 1–46, 2016.

[5] S. E. Yuksel, J. N. Wilson, and P. D. Gader, “Twenty years of mixture of experts,” IEEE Trans. Neural Netw. Learn. Syst., vol. 23, no. 8, pp. 1177–1193, Aug. 2012.

[6] C. E. Rasmussen and Z. Ghahramani, “Infinite mixtures of Gaussian process experts,” in Proc. Adv. Neural Inf. Process. Syst., 2002, pp. 881–888.

[7] E. Meeds and S. Osindero, “An alternative infinite mixture of Gaussian process experts,” in Proc. Adv. Neural Inf. Process. Syst., 2006, pp. 883–890.

[8] S. P. Chatzis and Y. Demiris, “Nonparametric mixtures of Gaussian processes with power-law behavior,” IEEE Trans. Neural Netw. Learn. Syst., vol. 23, no. 12, pp. 1862–1871, Dec. 2012.

[9] S. Masoudnia and R. Ebrahimpour, “Mixture of experts: A literature survey,” Artif. Intell. Rev., vol. 42, no. 2, pp. 275–293, 2014.

[10] G. E. Hinton, “Training products of experts by minimizing contrastive divergence,” Neural Comput., vol. 14, no. 8, pp. 1771–1800, 2002.

[11] Y. Cao and D. J. Fleet, “Generalized product of experts for automatic and principled fusion of Gaussian process predictions,” 2014, arXiv:1410.7827. [Online]. Available: https://arxiv.org/abs/1410.7827

[12] G. E. Hinton, “Training products of experts by minimizing contrastive divergence,” Neural Comput., vol. 14, no. 8, pp. 1771–1800, 2002

[13] H. Liu, J. Cai, Y. Wang, and Y.-S. Ong, “Generalized robust Bayesian committee machine for large-scale Gaussian process regression,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 1–10.

[14] Chris Williams, “Model Selection for Gaussian Process”, University of Edinburgh, UK, December 2006.

[15] David Kristjanson Duvenaud, “ Automatic Model Construction with Gaussian Process”, University of Cambridge, June 2014.

[16] David Duvenaud, “ The Kernel Cookbook : Advice on Covariance functions”, Cornell University, February 2013.

[17] When Gaussian Process Meets Big Data: A Review of Scalable GPs Haitao Liu , Yew-Soon Ong , Fellow, IEEE, Xiaobo Shen, and Jianfei Cai , Senior Member, IEEE

[18]Local and global sparse Gaussian process approximations by Edward Snelson Gatsby (Computational Neuroscience Unit University College London, UK) and Zoubin Ghahramani (Department of Engineering University of Cambridge, UK)

[19] Automatic Model Construction with Gaussian Processes by David Kristjanson Duvenaud University of Cambridge

[20] Hierarchically-partitioned Gaussian Process Approximation Byung-Jun Lee Jongmin Lee Kee-Eung Kim

[21] M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml

[22] A Framework for Evaluating Approximation Methods for Gaussian Process Regression by Krzysztof Chalupka et al.

[23] 10-708: Probabilistic Graphical Models, Spring 2015 21 : Advanced Gaussian Processes Lecturer: Eric P. Xing

[24] C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning

[25] DataSet Source : https://ourworldindata.org/coronavirus-source-data

[26] Our Experiment’s Implementation :

https://github.com/Sharamatya/AML/tree/main/Gaussian%20Approximation

[27] Kevin P. Murphy. Machine Learning, A Probabilistic Perspective, Chapters 4, 14 and 15.

[28] Christopher M. Bishop. Pattern Recognition and Machine Learning, Chapter 6.

[29] Carl Edward Rasmussen and Christopher K. I. Williams. Gaussian Processes for MachineLearning.

[30] Paper on Hands-on Experience with Gaussian Processes (GPs): Implementing GPs in Python - I by Kshitij Tiwaria (Intelligent Robotics Group, Department of Electrical Engg. & Automation, Aalto University, Espoo, 02150, Finland).
[Link-https://arxiv.org/pdf/1809.01913.pdf]

[31] PyMC3 Documentation : https://pymc3.readthedocs.io/en/latest/api/gp.html ,

https://docs.pymc.io/api/gp/implementations.html