Local and global sparse Gaussian process approximations

Edward Snelson
Gatsby Computational Neuroscience Unit
University College London, UK
snelson@gatsby.ucl.ac.uk

Zoubin Ghahramani
Department of Engineering
University of Cambridge, UK
zoubin@eng.cam.ac.uk

Abstract

Gaussian process (GP) models are flexible probabilistic nonparametric models for regression, classification and other tasks. Unfortunately they suffer from computational intractability for large data sets. Over the past decade there have been many different approximations developed to reduce this cost. Most of these can be termed global approximations, in that they try to summarize all the training data via a small set of support points. A different approach is that of local regression, where many local experts account for their own part of space. In this paper we start by investigating the regimes in which these different approaches work well or fail. We then proceed to develop a new sparse GP approximation which is a combination of both the global and local approaches. Theoretically we show that it is derived as a natural extension of the framework developed by Quiñonero Candela and Rasmussen [2005] for sparse GP approximations. We demonstrate the benefits of the combined approximation on some 1D examples for illustration, and on some large real-world data sets.

1 INTRODUCTION

Gaussian process (GP) models are popular nonparametric nonlinear Bayesian models for machine learning [see Rasmussen and Williams, 2006]. They can be used for a wide variety of tasks, but at their simplest they are nonlinear regression models. An attractive property of a GP is that predictions are made with an associated uncertainty. With a suitable choice of covariance function, these uncertainties grow large in regions away from observed data, and shrink close to observations. The nonparametric nature of a GP means that there are only a few hyperparameters of the covariance function that need to be learned.

For the case of regression with Gaussian noise, all computations necessary for GP prediction are analytical expressions. However, even in this case, the GP model becomes intractable for large data sets due to an $\mathcal{O}(N^3)$ cost for inverting the training data covariance matrix, and an $\mathcal{O}(N^2)$ cost per test case for prediction. In recent years there have been many sparse GP approximations developed that reduce the cost to $\mathcal{O}(NM^2)$ training time and $\mathcal{O}(M^2)$ prediction time [e.g. Csató, 2002, Seeger et al., 2003, Snelson and Ghahramani, 2006]. These are generally based on a small ($\ll N$) set of M support points. We refer to this type of approximation as *global*, because the M support points are essentially summarizing *all* N data points. Other approaches use iterative methods in combination with a fast matrix-vector product algorithm (e.g. IFGT [Yang et al., 2005]). These tend not to be feasible for very many input dimensions.

So far there has not been much work in assessing the regimes in which the global type of approximation is worthwhile. For example, some data sets may be best approached using a *local* type of approximation, where only training data points nearby to the test point are used to make a prediction. A very complex 'wiggly' data set may not be well summarized by a small number of support points, and a local regression scheme may be faster and more accurate. A natural question to ask is whether there is an approximation that combines the best of both worlds: a combination of a local and global approximation that will be suitable for all regimes. In this paper we develop such an approximation, and show how it can be derived from a natural extension of the approximation framework outlined by Quiñonero Candela and Rasmussen [2005].

Throughout this paper, 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.

2 GAUSSIAN PROCESS REVIEW

We denote a D dimensional input point as $\mathbf{x}$, and a scalar valued output as y. We have a training data set $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$, and a corresponding test data set $\mathcal{D}_T = \{\mathbf{x}_t, y_t\}_{t=1}^T$. In a Gaussian process regression model, we assume there is an underlying noise-free latent function $f(\mathbf{x})$ that we are modeling. At each observation in the training or test set there is therefore a latent function variable which we denote f_n or f_t. We refer collectively to groups of these points as: $(\mathbf{X}, \mathbf{f}, \mathbf{y}) = (\{\mathbf{x}_n\}, \{f_n\}, \{y_n\})_{n=1}^N$, and $(\mathbf{X}_T, \mathbf{f}_T, \mathbf{y}_T) = (\{\mathbf{x}_t\}, \{f_t\}, \{y_t\})_{t=1}^T$.

A Gaussian process places a distribution on functions $f(\mathbf{x})$ by defining a consistent multivariate Gaussian distribution on any collection of function variables. The function variables we need to consider are the ones corresponding to our training and test sets: $\mathbf{f}$ and $\mathbf{f}_T$. The Gaussian distribution on these $N+T$ variables is:

$$p(\mathbf{f}, \mathbf{f}_T) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{N+T}) \ , \quad \mathbf{K}_{N+T} = \begin{bmatrix} \mathbf{K}_N & \mathbf{K}_{NT} \\ \mathbf{K}_{TN} & \mathbf{K}_T \end{bmatrix} \ , \tag{1}$$

where $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$[1]. The covariance matrix of (1) is formed from a covariance function $K(\mathbf{x}, \mathbf{x}')$, which encodes the prior notion of smoothness. See appendix A for the covariance matrix notation we use. A typical covariance function is the squared exponential, which we use throughout:

$$K(\mathbf{x}, \mathbf{x}') = c \exp\left[-\textstyle\sum_{d=1}^D b_d (x_d - x_d')^2\right] \ . \tag{2}$$

$\mathbf{b}$ and c are hyperparameters controlling the lengthscales and size of the process respectively.

We assume Gaussian observation noise ϵ, of variance σ^2, such that $y = f + \epsilon$. The joint distribution on training and test outputs is:

$$p(\mathbf{y}, \mathbf{y}_T) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{N+T} + \sigma^2 \mathbf{I}) \ . \tag{3}$$

The predictive distribution is obtained by conditioning on the observed training outputs:

$$p(\mathbf{y}_T | \mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}_T, \boldsymbol{\Sigma}_T) \ , \tag{4a}$$

$$\begin{aligned} \boldsymbol{\mu}_T &= \mathbf{K}_{TN} [\mathbf{K}_N + \sigma^2 \mathbf{I}]^{-1} \mathbf{y} \\ \boldsymbol{\Sigma}_T &= \mathbf{K}_T - \mathbf{K}_{TN} [\mathbf{K}_N + \sigma^2 \mathbf{I}]^{-1} \mathbf{K}_{NT} + \sigma^2 \mathbf{I} \ . \end{aligned} \tag{4b}$$

It is important to appreciate that (4) is a correlated prediction. As well as giving you the mean and marginal variance at each test point, it tells you the predictive correlations between any pair of test outputs. Whilst these correlations are potentially useful for some applications, it is often just the marginal variances ($\operatorname{diag} \boldsymbol{\Sigma}_T$) that are computed and used as measures of predictive uncertainty. In this case it is sufficient to consider a single general test input $\mathbf{x}_*$, at which the predictive mean and variance is:

$$\begin{aligned} \mu_* &= \mathbf{K}_{*N} [\mathbf{K}_N + \sigma^2 \mathbf{I}]^{-1} \mathbf{y} \\ \sigma_*^2 &= K_* - \mathbf{K}_{*N} [\mathbf{K}_N + \sigma^2 \mathbf{I}]^{-1} \mathbf{K}_{N*} + \sigma^2 \ . \end{aligned} \tag{5}$$

Computing the inverse of $\mathbf{K}_N + \sigma^2 \mathbf{I}$ costs $\mathcal{O}(N^3)$. We see that the mean prediction is simply a weighted sum of N basis functions: $\mu_* = \mathbf{K}_{*N} \boldsymbol{\alpha}$, where $\boldsymbol{\alpha} = [\mathbf{K}_N + \sigma^2 \mathbf{I}]^{-1} \mathbf{y}$. Therefore, if we precompute $\boldsymbol{\alpha}$ the mean prediction per test case costs only $\mathcal{O}(N)$. For the variance no such precomputation can be done, and the cost is $\mathcal{O}(N^2)$ per test case.

3 SPARSE GP APPROXIMATIONS

Quiñonero Candela and Rasmussen [2005] showed how most previously considered sparse GP approximations could be constructed in the same approximation framework. We outline this framework here, and discuss the FIC and FITC approximations.

The starting point to any of the approximations is a set of *inducing inputs* $\bar{\mathbf{X}} = \{\bar{\mathbf{x}}_m\}_{m=1}^M$. If these points are selected as a subset of the data inputs then some authors call this the 'active set', or 'support set'. For the sparse pseudo-input Gaussian process (SPGP) [Snelson and Ghahramani, 2006] we relaxed this restriction by allowing 'pseudo-inputs' in arbitrary locations. Throughout this paper we use the blanket term (of Quiñonero Candela and Rasmussen [2005]) 'inducing inputs' to refer to either possibility.

Given a set of inducing inputs, the GP prior can be split into two parts:

$$p(\mathbf{f}, \mathbf{f}_T) = \int d\bar{\mathbf{f}} \, p(\mathbf{f}, \mathbf{f}_T | \bar{\mathbf{f}}) \, p(\bar{\mathbf{f}}) \ , \tag{6}$$

where the *inducing variables* $\bar{\mathbf{f}}$ are marginalized out. In the first stage to all the approximations, Quiñonero Candela and Rasmussen [2005] make the assumption that $\mathbf{f}$ and $\mathbf{f}_T$ are conditionally independent given $\bar{\mathbf{f}}$:

$$p(\mathbf{f}, \mathbf{f}_T) \approx q(\mathbf{f}, \mathbf{f}_T) = \int d\bar{\mathbf{f}} \, q(\mathbf{f}_T | \bar{\mathbf{f}}) \, q(\mathbf{f} | \bar{\mathbf{f}}) \, p(\bar{\mathbf{f}}) \ . \tag{7}$$

We will examine this assumption in more detail later on. Here the prior on the inducing variables is exact: $p(\bar{\mathbf{f}}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_M)$. The various different sparse approximations are then derived by making additional assumptions about the training and test conditionals $q(\mathbf{f}|\bar{\mathbf{f}})$ and $q(\mathbf{f}_T|\bar{\mathbf{f}})$.

[1] Strictly the distribution of (1) is conditioned on the inputs $\mathbf{X}$ and $\mathbf{X}_T$, but since the GP is a conditional model, every distribution is conditioned on the relevant inputs, and we omit them from the notation for brevity.

3.1 The fully independent (training) conditional (FI(T)C) approximation

FIC is the approximation we developed for the SPGP. To derive the FIC approximation within this framework, we make the assumption that the function variables in both the training and test conditionals of (7) are fully independent:

$$q(\mathbf{f}|\bar{\mathbf{f}}) = \prod_n p(f_n|\bar{\mathbf{f}}) \qquad q(\mathbf{f}_T|\bar{\mathbf{f}}) = \prod_t p(f_t|\bar{\mathbf{f}}) \;. \tag{8}$$

With these approximations we compute the integral of (7) to obtain the FIC approximate prior distribution:

$$q_{\text{FIC}}(\mathbf{f}, \mathbf{f}_T) = \mathcal{N}(\mathbf{0}, \tilde{\mathbf{K}}^{\text{FIC}}_{N+T}) \tag{9}$$

$$\tilde{\mathbf{K}}^{\text{FIC}}_{N+T} = \begin{bmatrix} \mathbf{Q}_N + \operatorname{diag}[\mathbf{K}_N - \mathbf{Q}_N] & \mathbf{Q}_{NT} \\ \mathbf{Q}_{TN} & \mathbf{Q}_T + \operatorname{diag}[\mathbf{K}_T - \mathbf{Q}_T] \end{bmatrix}$$

where $\mathbf{Q}$s are low-rank (rank M) matrices made from the covariance function $Q(\mathbf{x}, \mathbf{x}') = \mathbf{K}_{\mathbf{x}M}\mathbf{K}_M^{-1}\mathbf{K}_{M\mathbf{x}'}$ (see appendix A (23) for details). Notice that since the training and test variables are treated in exactly the same way, there was no need in this case to first separate them out as in (7). *All* function variables are conditionally independent given $\bar{\mathbf{f}}$. Therefore the FIC approximation corresponds to a standard GP model with a particular covariance function:

$$\tilde{K}^{\text{FIC}}(\mathbf{x}, \mathbf{x}') = Q(\mathbf{x}, \mathbf{x}') + \delta(\mathbf{x} - \mathbf{x}')[K(\mathbf{x}, \mathbf{x}) - Q(\mathbf{x}, \mathbf{x})] \;, \tag{10}$$

where δ is the Dirac delta function. The covariance matrix of (9) can be constructed directly from (10).

The FIC predictive distribution is formed from the blocks of (9) in the same way as (4) for the full GP:

$$\begin{aligned} \boldsymbol{\mu}^{\text{FIC}}_T &= \mathbf{Q}_{TN}[\tilde{\mathbf{K}}^{\text{FIC}}_N + \sigma^2\mathbf{I}]^{-1}\mathbf{y} \\ \boldsymbol{\Sigma}^{\text{FIC}}_T &= \tilde{\mathbf{K}}^{\text{FIC}}_T - \mathbf{Q}_{TN}[\tilde{\mathbf{K}}^{\text{FIC}}_N + \sigma^2\mathbf{I}]^{-1}\mathbf{Q}_{NT} + \sigma^2\mathbf{I} \;. \end{aligned} \tag{11}$$

$\tilde{\mathbf{K}}^{\text{FIC}}_N + \sigma^2\mathbf{I}$ can be inverted in $\mathcal{O}(NM^2)$ since it is the sum of the rank M matrix $\mathbf{Q}_N$ and a diagonal.

The FITC approximation differs slightly from FIC in that only the *training* conditional is factorized. The test conditional remains exact (c.f. (8)):

$$q(\mathbf{f}|\bar{\mathbf{f}}) = \prod_n p(f_n|\bar{\mathbf{f}}) \qquad q(\mathbf{f}_T|\bar{\mathbf{f}}) = p(\mathbf{f}_T|\bar{\mathbf{f}}) \;. \tag{12}$$

The FITC predictive distribution is therefore identical to FIC (11) apart from the approximate $\tilde{\mathbf{K}}^{\text{FIC}}_T$ being replaced with the exact $\mathbf{K}_T$ in the predictive covariance. However the difference is only apparent if you want to make *correlated* predictions. Since the diagonal of $\tilde{\mathbf{K}}^{\text{FIC}}_T$ is exact ($\operatorname{diag}\tilde{\mathbf{K}}^{\text{FIC}}_T = \operatorname{diag}\mathbf{K}_T$), the *marginal* variances of FITC and FIC are exactly the same. In either case the FI(T)C single test case predictive distribution is:

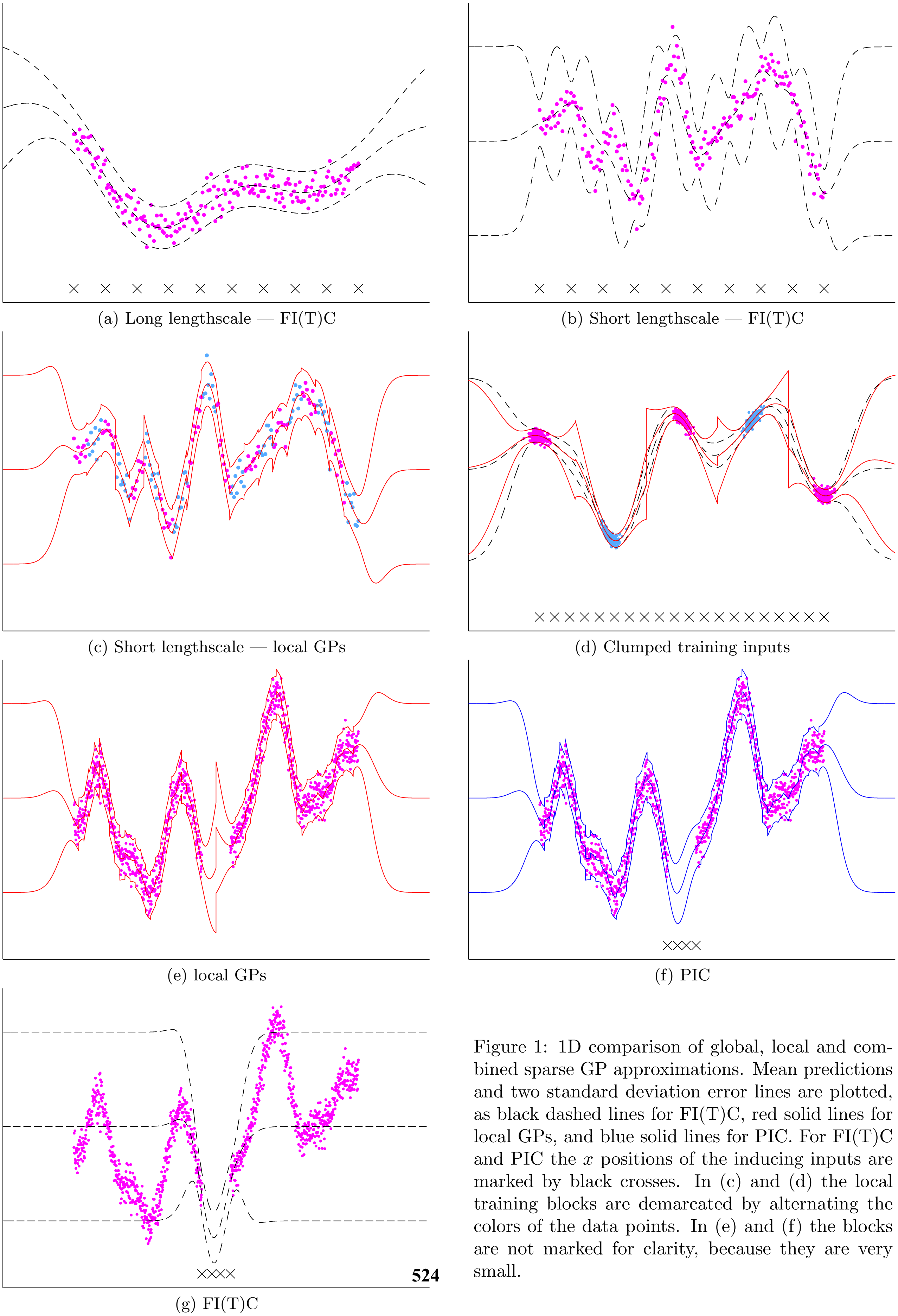
$$\begin{aligned} \mu^{\text{FIC}}_* &= \mathbf{Q}_{*N}[\tilde{\mathbf{K}}^{\text{FIC}}_N + \sigma^2\mathbf{I}]^{-1}\mathbf{y} \\ (\sigma^2_*)^{\text{FIC}} &= K_* - \mathbf{Q}_{*N}[\tilde{\mathbf{K}}^{\text{FIC}}_N + \sigma^2\mathbf{I}]^{-1}\mathbf{Q}_{N*} + \sigma^2 \;. \end{aligned} \tag{13}$$

Just as for the full GP we observe that the mean predictor of (13) is just a weighted sum of basis functions; however there are only M basis functions in the sum, rather than N: $\mu^{\text{FIC}}_* = \mathbf{K}_{*M}\boldsymbol{\alpha}$. Therefore, once precomputations of $\mathcal{O}(NM^2)$ have been done, the mean prediction per test case is only $\mathcal{O}(M)$. Similar reasoning shows that the variance predictions cost $\mathcal{O}(M^2)$ per test case.

4 LOCAL OR GLOBAL APPROXIMATIONS

To understand the regimes in which an approximation such as FI(T)C works well and not so well, it is simplest to look at an example. Figure 1a shows some sample data drawn from a GP with a fairly long lengthscale (relative to the input point sampling). The FI(T)C prediction is plotted, using just 10 evenly spaced inducing inputs. The approximation is clearly extremely good — a full GP prediction looks essentially identical. Figure 1b shows the same number of data points drawn from a GP with a much shorter lengthscale. The FI(T)C prediction is plotted again using only 10 inducing inputs, and is clearly much worse, particularly in the gaps between inducing inputs. The training and prediction costs for the examples in 1a and 1b are exactly the same. In this simple example, we could just increase the number of inducing inputs in 1b to take into account the extra complexity of the function. However, in a more realistic problem we may not be able to afford the extra cost to do this. For a very complex function we may find ourselves needing almost as many inducing inputs as data points to model the function well, and that takes us back towards $\mathcal{O}(N^3)$ complexity. Although each inducing input only affects predictions in a local region around itself, we refer to this type of approximation as global because *all* N data points contribute to each prediction made, via the inducing points.

An alternative type of approach to the data in figure 1b is to use a series of local GPs. This approach is shown in figure 1c. The training points are grouped into blocks of 10 points each, and independent GP predictors formed from each of the blocks. The nearest block's GP is used to predict at a given test point. This is a particular unsmoothed example of *local nonlinear regression*, similar in flavor to e.g. LOESS [Grosse, 1989]. It is also a trivial unsmoothed example of a mixture of GP experts [Rasmussen and Ghahramani,



(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

Figure 1: 1D comparison of global, local and combined sparse GP approximations. Mean predictions and two standard deviation error lines are plotted, as black dashed lines for FI(T)C, red solid lines for local GPs, and blue solid lines for PIC. For FI(T)C and PIC the x positions of the inducing inputs are marked by black crosses. In (c) and (d) the local training blocks are demarcated by alternating the colors of the data points. In (e) and (f) the blocks are not marked for clarity, because they are very small.

2002]. The independence between the blocks leads to the discontinuous nature of the prediction in 1c, but if we ignore the ugly aesthetics, the prediction is actually a much better fit than that of 1b. If as in this illustration we choose equal block sizes of size B, then the training cost is $\mathcal{O}(N/B \times B^3) = \mathcal{O}(NB^2)$, and prediction cost[2] per test case is $\mathcal{O}(B^2)$. For figures 1b and 1c we chose $B = M = 10$, so the costs are essentially equivalent. In this regime therefore the local type of approximation is the more efficient option.

Apart from the ugly discontinuities, the local GP approach would actually work pretty well for the longer lengthscale example of figure 1a. However there are certainly situations where the local approach can be poor. Figure 1d shows some data where the training inputs have been sampled in a non-uniform manner. Such a situation often happens in real world examples due to artifacts in the data collection process, and is more pronounced in high dimensions. In this situation, if we take the clusters as separate blocks and use the local GP approach the extrapolation between clusters is very poor, because the blocks are all independent from each other. The FI(T)C predictions are much better because they take into account the correlations between the clusters and extrapolate well.

5 A COMBINED LOCAL AND GLOBAL APPROXIMATION

With the discussion of the previous section in mind, it would be nice to have an approximation that combined the ideas of both the global and local approaches, so that it would be suitable to use in all regimes. In this section we develop such an approximation and show how it is naturally derived as an extension of the theoretical framework of section 3. To lead into this we briefly review one further sparse GP approximation that has some local aspects.

5.1 The partially independent training conditional (PITC) approximation

Quiñonero Candela and Rasmussen [2005] suggest a further improved approximation to FI(T)C[3]. Rather than assume complete training conditional independence as in FITC (12), PITC only assumes partial independence. The training points are grouped into 'blocks' or 'clusters' $\{\mathbf{X}_{B_s}, \mathbf{f}_{B_s}\}_{s=1}^{S}$, and conditional independence is only assumed between blocks:

$$q(\mathbf{f}|\bar{\mathbf{f}}) = \prod_s p(\mathbf{f}_{B_s}|\bar{\mathbf{f}}) \qquad q(\mathbf{f}_T|\bar{\mathbf{f}}) = p(\mathbf{f}_T|\bar{\mathbf{f}}) \;. \tag{14}$$

As in FITC, the test conditional of (14) remains exact. Assuming these approximate conditionals leads to the PITC training and test covariance:

$$\tilde{\mathbf{K}}_{N+T}^{\mathrm{PITC}} = \begin{bmatrix} \mathbf{Q}_N + \mathrm{bkdiag}[\mathbf{K}_N - \mathbf{Q}_N] & \mathbf{Q}_{NT} \\ \mathbf{Q}_{TN} & \mathbf{K}_T \end{bmatrix} \;. \tag{15}$$

The PITC single test point predictive distribution is:

$$\begin{aligned} \mu_*^{\mathrm{PITC}} &= \mathbf{Q}_{*N}[\tilde{\mathbf{K}}_N^{\mathrm{PITC}} + \sigma^2\mathbf{I}]^{-1}\mathbf{y} \qquad (16)\\ (\sigma_*^2)^{\mathrm{PITC}} &= K_* - \mathbf{Q}_{*N}[\tilde{\mathbf{K}}_N^{\mathrm{PITC}} + \sigma^2\mathbf{I}]^{-1}\mathbf{Q}_{N*} + \sigma^2 \;, \end{aligned}$$

where $\tilde{\mathbf{K}}_N^{\mathrm{PITC}} = \mathbf{Q}_N + \mathrm{bkdiag}[\mathbf{K}_N - \mathbf{Q}_N]$. As for FI(T)C the PITC mean predictor is simply a weighted sum of M basis functions. The cost per test case is therefore exactly the same: $\mathcal{O}(M)$ for the mean and $\mathcal{O}(M^2)$ for the variance. How about the precomputations? The cost to invert $\tilde{\mathbf{K}}_N^{\mathrm{PITC}} + \sigma^2\mathbf{I}$ depends on the sizes of the blocks $\{B_s\}$. There is no requirement for the blocks to be of equal size, but for simplicity suppose they all have size B. Then for the same reasons as in section 4, the extra precomputations cost $\mathcal{O}(NB^2)$.

The PITC approximation certainly has the flavour of trying to combine a local approximation with a global one. However, if we actually plot PITC predictions for the data in figure 1b, we find they are almost identical to FI(T)C. Why does the blocking not help us? The answer is easy to see by referring to the PITC predictive distribution of (16). Looking at the mean prediction: it is still just a weighted sum of basis functions centered on the same inducing inputs as in FI(T)C. The blocking has only altered the weights slightly. Fundamentally, when the basis functions are local such as the squared exponential, PITC is as incapable of modeling well away from inducing inputs as FI(T)C.

The PITC marginal likelihood $p(\mathbf{y}) = \mathcal{N}(\mathbf{0}, \tilde{\mathbf{K}}_N^{\mathrm{PITC}})$ is certainly a closer approximation to the full GP marginal likelihood than FI(T)C. As such, when optimizing the inducing inputs and hyperparameters as in the SPGP, the PITC approximation may yield better results. *given* a set of inducing inputs and hyperparameters, the PITC predictive distribution does not seem to offer much of an advantage over FI(T)C.

5.2 The partially independent conditional (PIC) approximation

In this section we develop a new approximation that successfully combines the ideas of the global and local approximations. Another way to understand why the PITC predictions are not much different to FI(T)C is

[2] ignoring the clustering cost (see section 5.3)

[3] The same type of approximation is also used by Tresp [2000] in the Bayesian committee machine (BCM), but in a less general context. In the BCM the equivalent of the inducing inputs turn out to be the test inputs [see Quiñonero Candela and Rasmussen, 2005].

to look again at the PITC prior covariance of (15). The structure of this covariance is such that the training inputs have been blocked separately from the test inputs — the test inputs have effectively been placed in a block of their own. This means that the PITC approximation cannot be considered a GP model with a particular covariance function, as the decision of which block in which to place an input depends on whether that input is a training input or test input. The consequence of this separation of training and test inputs into different blocks is that they only interact with each other via the M inducing inputs. This in turn leads to the predictive distribution being very similar to FI(T)C, and largely governed by the positioning of the inducing inputs.

The separation of the test points into their own block came about because of the first assumption about sparse GP approximations made by Quiñonero Candela and Rasmussen [2005]: the conditional independence of training and test points, denoted $\mathbf{f} \perp \mathbf{f}_T|\bar{\mathbf{f}}$, in (7). To derive a new approximation we relax this assumption and consider what happens if we block the *joint* training and test conditional. We treat the training and test inputs equivalently, and group them into blocks according only to their $\mathbf{x}$ positions. For ease of notation, and because we will only use the marginal predictive variance, we consider a single test input $\mathbf{x}_*$. Suppose that on the basis of its position this test input was grouped with training block B_S. Then the approximate conditional is:

$$p(\mathbf{f}, f_*|\bar{\mathbf{f}}) \approx q(\mathbf{f}, f_*|\bar{\mathbf{f}}) = p(\mathbf{f}_{B_S}, f_*|\bar{\mathbf{f}}) \prod_{s=1}^{S-1} p(\mathbf{f}_{B_s}|\bar{\mathbf{f}}) \,. \tag{17}$$

It seems logical to follow the naming convention introduced by Quiñonero Candela and Rasmussen [2005], and call this approximation the *partially independent conditional* (PIC) approximation. The PIC training and test covariance is:

$$\tilde{\mathbf{K}}^{\mathrm{PIC}}_{N+T} = \mathbf{Q}_{N+T} + \mathrm{bkdiag}[\mathbf{K}_{N+T} - \mathbf{Q}_{N+T}] \,. \tag{18}$$

Notice that unlike PITC, PIC *can* correspond to a standard GP with a particular covariance function. For example, suppose we divided the input space up into disjoint regions *before seeing any data.* Then if two points (training or test) fall into the same region they are placed in the same block. This corresponds to the following covariance function:

$$\begin{aligned} &\tilde{K}^{\mathrm{PIC}}(\mathbf{x},\mathbf{x}') = Q(\mathbf{x},\mathbf{x}') + \psi(\mathbf{x},\mathbf{x}')[K(\mathbf{x},\mathbf{x}') - Q(\mathbf{x},\mathbf{x}')] \\ &\quad \text{where} \quad \psi(\mathbf{x},\mathbf{x}') = \begin{cases} 1 & \text{if } \mathbf{x},\mathbf{x}' \in \text{same region} \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{19}$$

In practice typical clustering schemes we will use will rely on all the training data to define regions in input space, and so will not technically correspond to the covariance function of (19). At a high level though, (19) is a good description of the PIC covariance.

For ease of notation when discussing the predictive distribution, we refer to the training block that the test point $\mathbf{x}_*$ belongs to as B. As shorthand for all the *training* points excluding B, we use $\not{B}$. The PIC single test point predictive distribution is:

$$\begin{aligned} \mu_*^{\mathrm{PIC}} &= \tilde{\mathbf{K}}^{\mathrm{PIC}}_{*N}[\tilde{\mathbf{K}}^{\mathrm{PITC}}_{N} + \sigma^2\mathbf{I}]^{-1}\mathbf{y} \\ (\sigma_*^2)^{\mathrm{PIC}} &= K_* - \tilde{\mathbf{K}}^{\mathrm{PIC}}_{*N}[\tilde{\mathbf{K}}^{\mathrm{PITC}}_{N} + \sigma^2\mathbf{I}]^{-1}\tilde{\mathbf{K}}^{\mathrm{PIC}}_{N*} + \sigma^2 \,, \end{aligned} \tag{20}$$

where $\tilde{\mathbf{K}}^{\mathrm{PIC}}_{*N} = [\mathbf{Q}_{*\not{B}}\,, \mathbf{K}_{*B}]$. Let us look first at the mean predictor μ_*^{PIC}. Part of the mean predictor, which we define $\mathbf{p} = [\tilde{\mathbf{K}}^{\mathrm{PITC}}_{N} + \sigma^2\mathbf{I}]^{-1}\mathbf{y}$, is exactly the same as for PITC (16). We can expand μ_*^{PIC} further:

$$\begin{aligned} \mu_*^{\mathrm{PIC}} &= \mathbf{Q}_{*\not{B}}\mathbf{p}_{\not{B}} + \mathbf{K}_{*B}\mathbf{p}_B \\ &= \mathbf{K}_{*M}\boldsymbol{\beta} + \mathbf{K}_{*B}\mathbf{p}_B \,, \end{aligned} \tag{21}$$

where the weights $\boldsymbol{\beta} = \mathbf{K}_M^{-1}\mathbf{K}_{M\not{B}}\mathbf{p}_{\not{B}}$ (see (23)). We therefore see that the mean is a weighted sum of basis functions centered at the M inducing inputs *and* at the training inputs in block B. This has the desired feature of being a combination of a local and global predictor. The local information comes from the block B that the test point is assigned to, and the global information comes via the inducing points. A similar interpretation can be applied to the variance.

Referring to (18), we see that there are now two limiting processes that will return us to the full GP. If there are N inducing points placed exactly on the training points then $\mathbf{Q} = \mathbf{K}$, and therefore $\tilde{\mathbf{K}}^{\mathrm{PIC}}_{N+T} = \mathbf{K}_{N+T}$. Similarly if we decrease the number of blocks until we have only one block, then $\tilde{\mathbf{K}}^{\mathrm{PIC}}_{N+T} = \mathbf{K}_{N+T}$. In a practical situation we can push towards both these limits as far as our computational budget will allow.

Taking these limits in the opposite direction gives us some further insight. If we take all the block sizes to one then we recover FIC. If we take the number of inducing points to zero, we are left with the purely local GP predictor of section 4. We can also see this from (21), since $\mathbf{p}_B \to [\mathbf{K}_B + \sigma^2\mathbf{I}]^{-1}\mathbf{y}_B$.

FI(T)C and PITC both had $\mathcal{O}(M)$ and $\mathcal{O}(M^2)$ costs per test point for predicting the mean and variance respectively, after precomputations. How about PIC? Looking at (21), at first it seems that prediction will be too expensive because of the product $\mathbf{Q}_{*\not{B}}\mathbf{p}_{\not{B}}$. In general $\not{B}$ will be close in size to N, and so $\mathcal{O}(\not{B})$ will be too expensive. However, we can rewrite (21) again:

$$\begin{aligned} \mu_*^{\mathrm{PIC}} &= \mathbf{K}_{*M}\mathbf{K}_M^{-1}\mathbf{K}_{M\not{B}}\mathbf{p}_{\not{B}} + \mathbf{K}_{*B}\mathbf{p}_B \\ &= \mathbf{K}_{*M}\big(\underbrace{\mathbf{K}_M^{-1}\mathbf{K}_{MN}\mathbf{p}}_{\mathbf{w}_M} - \underbrace{\mathbf{K}_M^{-1}\mathbf{K}_{MB}\mathbf{p}_B}_{\mathbf{w}_M^B}\big) + \mathbf{K}_{*B}\mathbf{p}_B \,, \end{aligned} \tag{22}$$

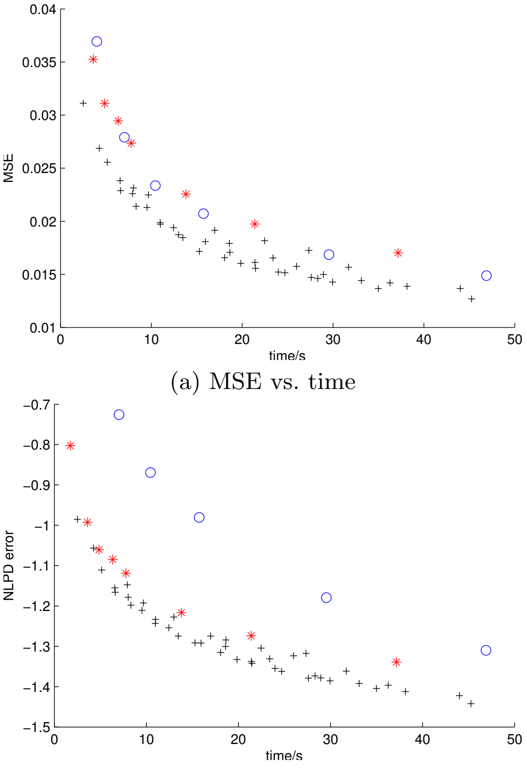
where $\mathbf{w}_M = \sum_{s=1}^{S} \mathbf{w}_M^{B_s}$. Hence we can precompute $\mathbf{p}$, then precompute $\mathbf{w}_M^{B_s}$ for each block, and finally precompute $\mathbf{w}_M$. Having done this the cost per test case at test time will be $\mathcal{O}(M+B)$ for the mean. We can play a similar trick for the variance, which then costs $\mathcal{O}\big((M+B)^2\big)$.

5.3 Clustering schemes

We need a scheme for clustering possibly high dimensional training inputs into blocks for the PIC approximation. We then need to be able to quickly assign a new test point to a block at test time. We suggest two simple schemes. The more complicated one is *farthest point clustering* [Gonzales, 1985]. The number of clusters S is chosen in advance. A random input point is picked as the first cluster center. The farthest point from this is chosen as the next center. The farthest point from both of these is chosen as the next, and so on, until we have S centers. Then each point in the training set is assigned to its nearest cluster center. At test time, a test point is simply assigned to the nearest cluster center. We also consider an even simpler algorithm which we call *random clustering*. It is exactly as above except that the cluster centers are picked randomly (without replacement) from the training input points. The naive costs for these algorithms are $\mathcal{O}(NS)$ training time and $\mathcal{O}(S)$ test time per test case. However with suitable data structures (e.g. KD-trees [Preparata and Shamos, 1985]) and implementation these costs can be reduced to $\mathcal{O}(N \log S)$ and $\mathcal{O}(\log S)$ [Feder and Greene, 1988].

6 RESULTS

The first thing to note is that PIC subsumes both the local GP approach and FI(T)C. By varying the number of inducing inputs and the size of the blocks we can obtain an approximation that is close to one or the other. The type of regimes in which the combined PIC approach is significantly advantageous than either one or the other can be seen by examining the failure modes as we did in section 4. Referring back to figures 1a–1d: FI(T)C fails for complex functions where we cannot afford to tile the space with inducing inputs; local GPs fail when we need to do extrapolation well. Figures 1e–1g show an illustrative 1D example where these problems are solved by the combined approach. In 1e the local GP approach works well apart from in the extrapolation between the two groups of data points. This can be completely fixed by the PIC approximation in 1f with the addition of a few well placed inducing points. The FI(T)C prediction based on these inducing points alone is shown in 1g. We see that the PIC predictor is a combination of the best parts of the local GP and FI(T)C predictors of 1e and 1g. In order to do well using FI(T)C alone we would need to tile the space densely with inducing points.



(a) MSE vs. time

(b) NLPD vs. time

Figure 2: Test set error vs. computation time for the *kin40k* data set. Blue circles – FI(T)C, red stars – local GPs, black crosses – PIC. Points obtained by varying number of inducing points, or number of blocks, or both (for PIC).

In a real world example, to obtain the maximum advantage from PIC, we will clearly need schemes to place the inducing points well. We can use approaches that attempt to maximize marginal likelihood as in the SPGP, or we can use simpler heuristics to place the inducing inputs well. We will leave a full experimental evaluation of such procedures to future work.

As a real world example we chose the *kin-40k* data set[4], a highly nonlinear robot arm control task. We then measured test set error as a function of computation time for the three methods: FI(T)C, local GPs, and PIC. Since it is not the goal of this paper to investigate hyperparameter learning and inducing point selection, we simply use hyperparameters obtained by training a GP on a smaller subset of the training data, and we use inducing points optimized as in the SPGP. The computation time reported is the precomputation and prediction time for all 30,000 test set examples.

[4] *kin-40k*: 10000 training, 30000 test, 9 attributes, see `http://ida.first.fraunhofer.de/~anton/data.html`

For local GPs and PIC, the time includes the extra clustering time, which was simply done by the random clustering method discussed in section 5.3. Points on the error/time plots of figure 2 were then obtained by varying the number of inducing points for FI(T)C, the number of blocks for local GPs, and both for PIC.

Figure 2a shows FI(T)C and local GPs performing very similarly in terms of MSE, with combined PIC approach giving a small but significant gain. Figure 2b shows PIC and local GPs performing well in terms of NLPD error[5], with FI(T)C performing much worse. This is probably because this data set is fairly complex with shortish lengthscales, meaning we are closer in regime to figure 1b than 1a; the local GPs and PIC therefore have tighter error bars and better NLPD.

We tried two other data sets for which the results are not plotted because they are very easy to describe. For *SARCOS*[6], which is another highly nonlinear task, local GPs performed much better than FI(T)C. The inducing inputs did not help improve PIC beyond the local GPs performance. For *Temp*[7], the opposite was the case. FI(T)C performed better than local GPs and the blocking did not improve PIC performance much beyond FI(T)C. Which approximation to use very much depends on the type of data. An advantage of PIC is that you are guarded against both failure modes. We might expect further advantages for PIC when we select inducing points in conjunction with the blocks, but this is beyond the scope of this paper.

7 CONCLUSIONS

In this paper we have developed a computationally efficient approximation that combines the advantages of the local regression/experts approach with the global inducing input based approach. From a theoretical point of view, PIC in some sense completes the sequence of approximations as set out in the framework of Quiñonero Candela and Rasmussen [2005].

From a practical point of view, we have explored the different types of regimes in which either the local or the global based approximations are more efficient, and we have demonstrated situations where the combined PIC method improves upon both of these. In practice the PIC approximation allows a user to vary the number of clusters and the number of inducing inputs to find the best performance. There are several interesting future directions to pursue, for example to try to learn inducing inputs in relation to the clustering, perhaps by the maximization of the PIC marginal likelihood. We are also extending PIC to a hierarchical version in which the inducing inputs are not coupled to all training points, potentially paving the way for GP applications on very large data sets.

A Covariance Notation

We construct covariance matrices and vectors from various combinations of training, test and inducing inputs, and the covariance function $K(\mathbf{x}, \mathbf{x}')$. Our notation uses $\mathbf{K}$ to represent any covariance matrix that is directly constructed from the covariance function, but with different indices to show which two sets of input points are involved. For example, the $N \times T$ rectangular covariance matrix between training points and test points is denoted $\mathbf{K}_{NT}$. It is constructed from the covariance function: $[\mathbf{K}_{NT}]_{nt} = K(\mathbf{x}_n, \mathbf{x}_t)$, or, with some abuse of notation, $\mathbf{K}_{NT} = K(\mathbf{X}, \mathbf{X}_T)$. Rather than use transpose symbols we simply swap the indices: $\mathbf{K}_{NM}^\top \equiv \mathbf{K}_{MN}$, since the covariance function is a symmetric function. To save further space, we contract the two indices of square or self covariances to one index, e.g. $\mathbf{K}_{NN} \equiv \mathbf{K}_N$. We denote a single general test point $(\mathbf{x}_*, f_*, y_*)$, in keeping with past GP references. Covariances that refer to this single test point have starred indices, e.g. $\mathbf{K}_{N*}$. A key part of all the sparse GP approximations is the 'low-rank' covariance function Q:

$$Q(\mathbf{x}, \mathbf{x}') = \mathbf{K}_{\mathbf{x}M} \mathbf{K}_M^{-1} \mathbf{K}_{M\mathbf{x}'} \;, \tag{23}$$

where $\mathbf{K}_{\mathbf{x}M}$ is shorthand for the vector *function* $[K(\mathbf{x}, \bar{\mathbf{x}}_1), \ldots, K(\mathbf{x}, \bar{\mathbf{x}}_M)]$. Any covariance matrix $\mathbf{Q}$ constructed from (23) will have maximum rank M.

References

L. Csató. Sparse online Gaussian processes. *Neural Comp.*, 14:641–668, 2002.

T. Feder and D. Greene. Optimal algorithms for approximate clustering. In *Proc. of the 20th ACM symp. on Theory of comp.*, pages 434–444. ACM Press, 1988.

T. Gonzales. Clustering to minimize the maximum intercluster distance. *Theor. Comp. Sci.*, 38:293–306, 1985.

E. Grosse. LOESS: Multivariate smoothing by moving least squares. In *Approximation Theory VI*, volume 1, pages 299–302. Academic Press, 1989.

F. P. Preparata and M. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

J. Quiñonero Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *JMLR*, 6:1939–1959, Dec 2005.

C. E. Rasmussen and Z. Ghahramani. Infinite mixtures of Gaussian process experts. In *NIPS 14*. MIT Press, 2002.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning.* MIT press, 2006.

M. Seeger, C. K. I. Williams, and N. D. Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In *AISTATS 9*, 2003.

E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *NIPS 18*, pages 1257–1264. MIT press, 2006.

V. Tresp. A Bayesian committee machine. *Neural Computation*, 12:2719–2741, 2000.

C. Yang, R. Duraiswami, and L. Davis. Efficient kernel machines using the improved fast gauss transform. In *NIPS 17*, pages 1561–1568. MIT Press, 2005.

[5] negative log predictive density

[6] `http://www.gaussianprocess.org/gpml/data/`

[7] `http://theoval.cmp.uea.ac.uk/~gcc/competition/`