Proceedings

A Progressive Framework for Sparse Gaussian Process Regression

V.R. Lalchand [1,2], A.C. Faul [1,2]

[1] University of Cambridge
[2] The Alan Turing Institute
* Correspondence: vr308@cam.ac.uk

Version July 27, 2018 submitted to Proceedings

Abstract: In their standard form Gaussian processes (GPs) provide a powerful non-parametric framework for regression and classificaton tasks. Their one limiting property is their $\mathcal{O}(N^3)$ scaling where N is the number of training data points. In this paper we present a framework for GP training with sequential selection of training data points using an intuitive selection metric. The greedy forward selection strategy is devised to target two factors - regions of high predictive uncertainty and underfit. Under this technique the complexity of GP training is reduced to $\mathcal{O}(M^3)$ where $(M \ll N)$ if M data points (out of N) are eventually selected. The sequential nature of the algorithm circumvents the need to invert the covariance matrix of dimension $N \times N$ and enables the use of favourable matrix inverse update identities. We outline the algorithm and sequential updates to the posterior mean and variance at each stage of training. We demonstrate our method on selected one dimensional functions to aid visualisation and show that the loss in accuracy due to using a subset of data points is marginal compared to the computational gains.

Keywords: gaussian processes; bayesian; on-line training; greedy approach; sequential training

1. Introduction

A GP model is nonparametric, this means that the complexity of the model grows as more data is observed. Another attractive feature of GP models is the behaviour of the predictive variance (also referred to in this context as uncertainty), which is naturally higher in regions away from the training data. This is intuitive as in regions where there is no training data there is higher uncertainty about the interpolating function. The application of GPs to the regression task involves the computation of a matrix inverse, this leads to the $\mathcal{O}(N^3)$ scaling. Further, $\mathcal{O}(N^2)$ space is required to store a dense covariance matrix in memory.

There has been significant interest in finding sparse approximations to the full Gaussian process in order to speed up training and prediction times to $\mathcal{O}(NM^2)$ where M is the size of an auxiliary set, typically a subset of the training data. [1] provides an excellent overview of sparse approximations. A common theme of the sparse methods involves developing a low-rank approximation to the covariance matrix. We describe the most influential approaches in section 3 but to aid in understanding a brief introduction to GPs and regression follows first.

2. Gaussian Processes for Regression

Here we provide a concise summary of GPs for regression, see Chapter 2 of [2] for a detailed account. We can think of a mathematical function $f(\boldsymbol{x})$ as a collection of random variables defined at locations $\boldsymbol{x} \in X$. A GP can be thought of as defining a probability distribution (prior) over function values $f(\boldsymbol{x})$. Notationally, we say f is distributed as a GP with mean $m(\boldsymbol{x})$ and covariance function $k(\boldsymbol{x}, \boldsymbol{x}^{'})$. The

covariance function typically depends on some *hyperparameters* that control various properties of the underlying f.

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

In the context of regression, we are given a training set $\{(\boldsymbol{x_i}, y_i), i = 1, \ldots, N\}$ where $\mathbf{y} = \{y_i\}_{i=1}^N \in \mathbb{R}$ are noisy realisations of a latent function f and $X = \{\boldsymbol{x}_i\}_{i=1}^N \in \mathbb{R}^d$ are d-dimensional inputs, $y_i = f(\boldsymbol{x}_i) + \mathcal{N}(0, \sigma_n^2)$. The objective is to predict f_* for a new unseen point x_*.

Assuming a zero mean GP prior on the latent function $f \sim \mathcal{GP}(0, k(\cdot, \cdot))$ we can write any finite collection of f values as a multivariate Gaussian. In particular, evaluations of f at X, $\mathbf{f} = \{f(\boldsymbol{x}_i)\}_{i=1}^N$ are distributed as a multivariate Gaussian,

$$p(\mathbf{f}|X)^1 = \mathcal{N}(0, K(X, X)) \tag{2}$$

where $K(X, X)$ is the covariance matrix constructed by applying the covariance function to the X input points. In our experiments, we use the *squared exponential* (SE) covariance function; section A of the appendix encloses a discussion on covariance functions.

According to the prior, the targets/outputs $\mathbf{y}$ are a sample from a multivariate Gaussian distribution $\mathbf{y} \sim \mathcal{N}(0, K + \sigma_n^2 \mathbb{I})$ (where we use the shorthand K for $K(X, X)$). Hence, the joint prior distribution of the training targets $\mathbf{y}$ and test output f_* is,

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K + \sigma_n^2 I & K_*^T \\ K_* & K_{**} \end{bmatrix}\right) \tag{3}$$

where T denotes matrix transposition and the mean is a $(N+1)$ zero vector and the covariance matrix has dimension $(N+1) \times (N+1)$. For a new input point $\boldsymbol{x}_*$, the covariance between the input point and all training data points is summarised by the row vector $K_* = \begin{bmatrix} k(\boldsymbol{x}_*, \boldsymbol{x}_1) & k(\boldsymbol{x}_*, \boldsymbol{x}_2) \ldots k(\boldsymbol{x}_*, \boldsymbol{x}_N) \end{bmatrix}$ and the covariance with itself (variance) is denoted as $K_{**} = k(\boldsymbol{x}_*, \boldsymbol{x}_*)$.

Once we have specified the joint prior of the training data points along with the test data point, we can derive the posterior distribution by conditioning the joint prior in eq. 3 on the training data points $\mathbf{y}$. The posterior defined as $p(f_*|\mathbf{y})$ can be derived in closed form using properties of conditional Gaussian distributions (see Appendix A.1 of [2]).

$$\begin{aligned} f_*|\mathbf{y} &\sim \mathcal{N}(\overline{f_*}, var(f_*)) \\ \overline{f_*} &= K_*(K + \sigma_n^2 I)^{-1}\mathbf{y} \\ var(f_*) &= K_{**} - K_*(K + \sigma_n^2 I)^{-1}K_*^T \end{aligned} \tag{4}$$

The hyperparametes of the covariance function $k(\cdot, \cdot)$ used to generate the covariance matrix K are discovered through optimisation of the marginal likelihood. Gradient based optimizers can be used to minimize the negative log of the marginal likelihood given by,

$$log\ \ p(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^T(K + \sigma_n^2 I)^{-1}\mathbf{y} - \frac{1}{2}log|K + \sigma_n^2| - \frac{N}{2}log(2\pi) \tag{5}$$

which follows directly from the fact that, $\mathbf{y} \sim \mathcal{N}(0, K + \sigma_n^2 I)$.

[1] We omit the explicit conditioning on X for the sake of brevity

Also, the prior $p(\mathbf{f})$ times the likelihood given by $p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{f}, \sigma_n^2)$ gives the marginal likelihood in 5 in closed form as the distirbutions in the integral are both Gaussian.

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

3. Summary of sparse approximation approaches

The mean and variance of the predictive posterior distribution in eq. 4 involves inversion of matrix $K + \sigma_n^2\mathbb{I}$. The predictive posterior $p(f_*|\mathbf{y})$ is directly derived from the joint prior in eq. 3. Hence, an approximation to the joint prior $p(\mathbf{y}, f_*)$ using a lower rank matrix $Q_{M\times M}$ where $M \ll N$ instead of the full covariance matrix K will reduce the complexity of predictive inference.

Note that,

$$p(f_*|\mathbf{y}) = \int p(\mathbf{f}, f_*|\mathbf{y})d\mathbf{f} \tag{7}$$

and

$$p(\mathbf{f}, f_*|\mathbf{y}) = \frac{p(\mathbf{f}, f_*)p(\mathbf{y}|\mathbf{f})}{p(\mathbf{y})} \tag{8}$$

Essentially, the joint prior, the likelihood and the posterior are interlinked through the Bayes' rule. Most approaches proposed in literature start with a modified likelihood $p(\mathbf{y}|\mathbf{f}) \approx q(\mathbf{y}|\mathbf{u})$ where $\mathbf{u} = [u_1, \ldots, u_m]$ are some arbitrary inducing function values, typically $\mathbf{u} \in \mathbf{y}$. The approaches mainly differ in terms of the theoretical approach in deriving the approximations and how the inducing values are selected.

[3], [4] and [5] all use the likelihood approximation,

$$p(\mathbf{y}|\mathbf{f}) \approx q(\mathbf{y}|\mathbf{u}) = \mathcal{N}(K_{\mathbf{f},\mathbf{u}}K_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}, \sigma_n^2) \tag{9}$$

[2] in chapter 8 provides a useful interpretation for this approximation calling it, the *projected process approximation* (PPA) as it uses the projection $\mathbf{f} = K_{\mathbf{f},\mathbf{u}}K_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}$. The PPA construction uses a likelihood involving all N data points while the posterior predictive distribution involves inverting only a $M \times M$ matrix. The key feature being that information contained in the N points leaves an imprint on the posterior predictive distribution.

Methods proposed in [6] [5] integrate the idea of online training and sparse representation providing update equations for the posterior mean and covariance. [1] reinterprets all the existing sparse representations and derives the joint prior $p(\mathbf{f}, f_*) \approx q(\mathbf{f}, f_*)$ implied by the different likelihood approximation schemes in literature. Once the approximate joint prior is specified, the predictive distribution $q(f_*|\mathbf{y})$ is derived the same way as for the full GP. The authors point out that this standardisation enables one to compare the different representations.

The likelihood approximation schemes still require the specification of the choice of inducing variables $\mathbf{u}$. Let us refer to them as the active set I. While one can select the active set through random sampling, studies have shown that the approach does not work as well as actively selecting the points as optimisers of a selection metric. [7] proposes using a differential entropy score which is a function of the predictive variance to systematically include points in the active set, while [3] proposes using an information gain criteria which can be computed cheaply.

[8] propose a sparse GP regression scheme using *pseudo inputs* in which active set inputs ($X_\mathbf{u}$) and the hyperparameters (θ) for the covariance function are selected through joint optimization of the marginal

likelihood. Thus it converts a discrete optimization problem of subset selection to a continuous paradigm where the active set is not constrained to lie within the training set. The method addresses the dual selection problems of the active set I and the kernel hyperparameters θ. Some authors refer to this technique as the *Fully Independent Training Conditional* (FITC), (see section 6 of [1] for a detailed discussion).

4. A Progressive framework for Gaussian Process Regression

In this section we describe a framework for training GPs progressively. We assume we have N pairs of data in the form of inputs X and targets $\mathbf{y}$, the aim is to select a smaller informative subset $\mathbf{u} \in \mathbf{y}$ (the *active* set) which play an active role in the inference. All other training points $\mathbf{y} \backslash \mathbf{u}$ belong to the *remainder* set. The active set is constructed incrementally, at each iteration exactly one training data point is selected. Following the notation from [3] let I denote indices of the active set and $R = \{1, \ldots, N\} \backslash I$ denote the indices of remainder points. Training happens in stages we index by t. Hence, I_t and R_t denote the index sets at stage t of training.

We denote the active set $\mathbf{u}$ by $\mathbf{y}(I_t)$ from here on to clearly incorporate the stage of training. Similarly, the remainder set is denoted as $\mathbf{y}(R_t)$. The active and remainder input locations are denoted as $X(I_t)$ and $X(R_t)$ respectively. For the purposes of testing we have a hold-out set with N_* test inputs X_* and targets $\mathbf{y}_*$.

Table 1. Notation used in stagewise training

At stage t				
	Index set	Inputs	Targets	Size
Active points	I_t	$X(I_t)$	$\mathbf{y}(I_t)$	t
Remainder points	R_t	$X(R_t)$	$\mathbf{y}(R_t)$	$N-t$

At stage t the active set has exactly t data points as at each stage exactly one point is added to the active set. We have a fixed set of N training pairs, the active set grows in size as more points are added to it and the remainder set shrinks in size as points are removed from it. Essentially, we start with all the training data points in the remainder set and points move from the remainder set to the active set in each iteration based on a selection criteria.

Further,

- $K = K(X, X)$ denotes the full covariance matrix computed on N inputs X.
- $K_{I_t} = K(X(I_t), X(I_t))$ denotes the $t \times t$ covariance matrix computed between the active inputs at the t^{th} stage of training.
- $K_{\backslash I_t} = K(X(R_t), X(I_t))$ denotes the $(N-t) \times t$ covariance matrix computed between the t active inputs selected so far and the $(N-t)$ remainder inputs in R_t.
- $K_{R_t} = K(X(R_t), X(R_t))$ denotes the $(N-t) \times (N-t)$ covariance matrix computed between the remainder inputs at stage t.
- $K_{**} = K(X_*, X_*)$ denotes the covariance matrix computed on the test inputs X_*.
- $K_{*I_t} = K(X_*, X(I_t))$ denotes the $N_* \times t$ covariance matrix computed between the test inputs and active inputs at stage t.
- $\boldsymbol{\mu}_t$ and Σ_t denote the predictive posterior mean and covariance computed at stage t for the remainder inputs $X(R_t)$.
- $\boldsymbol{\mu}^*$ and Σ^* denote the predictive posterior mean and covariance for the test inputs X_*.

The algorithm starts with a single training point $\mathbf{y}(I_1) = [u_1]$ in the active set which is selected at random from $\mathbf{y}$, the predictive posterior mean and covariance denoted by $\boldsymbol{\mu_1}$ and Σ_1 at stage 1 are computed by conditioning on the active set (of 1 point) while the goodness of fit

is assessed by predicting on the remainder inputs ($N-1$ points). In short, we predict at each stage the mean and covariance of the remainder inputs $X(R_t)$ and compare them to the true remainder targets $\mathbf{y}(R_t)$, the mean squared error computed between the true remainder targets and the predicted remainder targets provides the basis for convergence (see section D of the Appendix).

The main reason for this stagewise iterative training approach is two-fold:

1. The selection criteria for the active training target at each stage is tied to the predictive posterior mean and variance computed on the remainder inputs. We discuss this in more detail in the next section.

2. Since the active set is grown one point at a time, we can take advantage of favourable matrix inverse update identities in order to update the predictive posterior mean and variance at each stage.

In general, at stage t,

$$p(\mathbf{y}(R_t)|\mathbf{y}(I_t)) = \mathcal{N}(\boldsymbol{\mu_t}, \Sigma_t)$$

$$\begin{aligned} \boldsymbol{\mu}_t &= K_{\backslash I_t}(K_{I_t} + \sigma_n^2\mathbb{I})^{-1}\mathbf{y}(I_t) \\ \Sigma_t &= K_{R_t} - K_{\backslash I_t}(K_{I_t} + \sigma_n^2\mathbb{I})^{-1}K_{\backslash I_t}^T \end{aligned} \tag{10}$$

where $\boldsymbol{\mu}_t$ and Σ_t denote the predicted posterior mean and covariance.

The above equations reflect the predictive posterior mean and covariance for a full GP introduced in eq. 4 where the active set $\mathbf{y}(I_t)$ plays the role of the target vector $\mathbf{y}$.

4.1. The Algorithm

Below we give the general algorithm for active set selection using a general selection metric we denote as Δ.

Algorithm 1 Progressive framework for GPR

Initialisation: Pick a random target $u_1 \in \mathbf{y}$.
Convergence Condition: $(RMSE_{t-1} - RMSE_t) < \delta$ calculated on the remainder set $\boldsymbol{y}(R_t) = \boldsymbol{y}\backslash\boldsymbol{y}(I_t)$.
for each stage t:
    GP Train on $(\boldsymbol{X}(I_t), \boldsymbol{y}(I_t))$.
    GP Predict on $\boldsymbol{X}(R_t)$.
    Update posterior mean μ_t and covariance Σ_t. (see section C for sequential update rules.)
    Compute $\Delta_i \forall i \in R_t$
    Select i where $i = argmax_{i \in R_t}\Delta_i$
    $I_{t+1} \leftarrow I_t \cup \{i\}, R_{t+1} \leftarrow R_t \backslash \{i\}$
    If convergence is **true**:
        break;
end
return I_T, R_T

4.2. Selection of active points

In this section we propose a selection criteria Δ that is used to advance the algorithm in the greedy search. One of the most attractive features of using a GP for regression is the way in which uncertainty

is propogated around predictions. The posterior predicted variance contracts surrounding the training data and expands back to the prior variance away from the training data (see fig. 1).

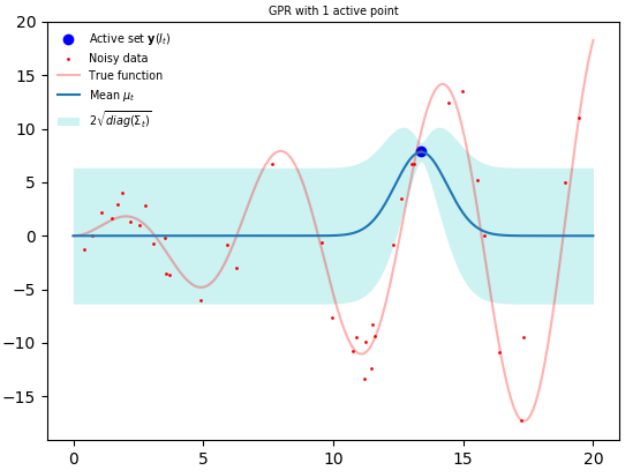


Figure 1. GPR with 1 active point

Hence, a simple rule which selects the next active target where the posterior predictive variance is maximised poses a problem of non-uniqueness. There are usually several remainder targets with the same posterior predicted variance (captured by the diagonal of the matrix Σ_t). The prior variance σ_f^2 serves as an upper bound for the posterior predicted variance. Let us term the posterior predicted variance captured in $diag(\Sigma_t)$ as Δ_1.

An alternative metric that serves as a reasonable choice is the absolute error or deviation from the posterior predicted mean at each stage t to the true remainder targets. At the end of each stage t we have, the posterior predicted mean $\boldsymbol{\mu}_t = [\mu_1, \ldots, \mu_{N-t}]$ and the true remainder set of targets $\mathbf{y}(R_t) = \{r_i | i \in R_t\}$. For each remainder target $r_i \in \mathbf{y}(R_t)$ we can compute its absolute deviation from the mean, let us call this metric Δ_2. See fig. 2.

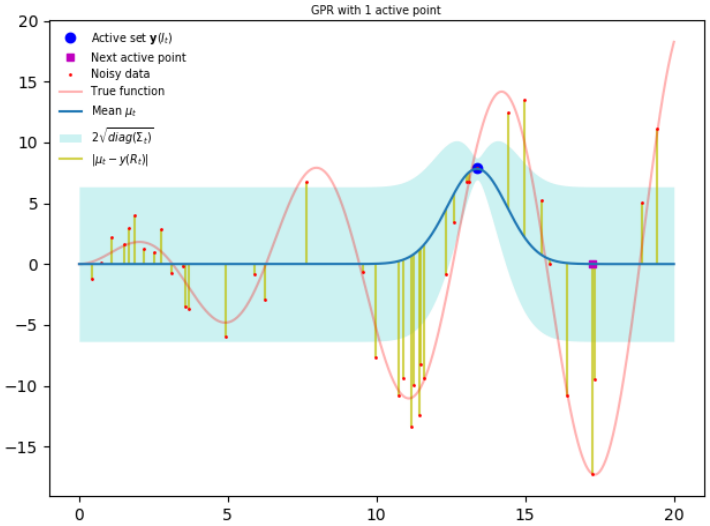


Figure 2. GPR with 1 active point using metric Δ_2 (depicted by the yellow vertical lines). According to this selection rule the next remainder point to be chosen for the active set is the red dot (noisy target) corresponding to the magenta point on the predicted mean.

The selection rule can be described as, $argmax_{u \in \boldsymbol{y}(R_t)} \Delta_2 = |\boldsymbol{\mu}_t - \boldsymbol{y}(R_t)|$ where $|\cdot|$ denotes absolute value. This metric is unique for each remainder target value, as seen in fig. 2. However, a drawback of Δ_2 is that it could end up selecting points in regions where the uncertainty is low (due to a nearby selected active point) versus points where the uncertainty is high along with a higher deviation from the mean.

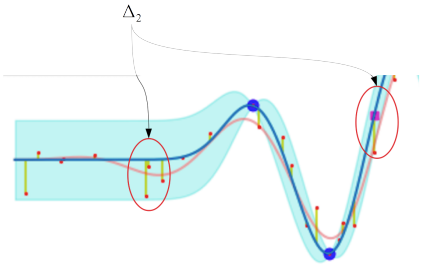


Figure 3. Snapshot of Progressive regression using Δ_2 (shown by the vertical yellow lines), the blue dots depict the active points chosen so far, the magenta square dot is the remainder point that would be chosen next if Δ_2 is used. Selecting a point in the left red circle could be a better choice as there is no active point in that region and could lead to a bigger reduction in overall uncertainty.

An ideal metric would penalise both high uncertainty captured by $diag(\Sigma_t)$ and deviation from the posterior predicted mean $|(\boldsymbol{\mu}_t - \boldsymbol{y}(R_t))|$. The metric Δ_3 is motivated by the desire to capture the two factors of high uncertainty and underfit. It is given by,

$$\Delta_3 = \sqrt{diag(\Sigma_t)} + |(\boldsymbol{\mu}_t - \boldsymbol{y}(R_t))|$$

Rule 3: $argmax_{u \in \boldsymbol{y}(R_t)} \Delta_3 = \sqrt{\Delta_1} + \Delta_2$

Note that both components of the addition are vectors of size $(N - t)$ as they are based on the remainder set. Further, we use $\sqrt{diag(\Sigma_t)}$ as the standard deviation is comparable in scale to the deviation from posterior mean. The metric Δ_3 can be evaluated in $\mathcal{O}(1)$ as Σ_t and $\boldsymbol{\mu}_t$ are obtained directly from the training step. A visual depiction of the evolution of progressive training for the function $x \sin x$ is shown in fig. A3.

4.3. Complexity of Progressive training

The cost of computing an updated inverse for a square matrix of size M grown by 1 row and 1 column is quadratic $\mathcal{O}(M^2)$ if we use block inversion (Schur complement) and assuming we know the inverse of the matrix of size M. In the progressive GP framework, in each stage the covariance matrix computed on active points grows by 1 row and 1 column. If we end up with an active set of M points after M stages of training we have conducted the matrix update operation M times and the cost each time is quadratic in the dimension of the matrix we are updating. This give us the following computational cost in terms of operations.

$$\sum_{i=1}^{M} i^2 = 1^2 + 2^2 + \ldots + M^2 = \frac{M(M+1)(2M+1)}{6} = \mathcal{O}(M^3) \tag{11}$$

Hence, while the order of complexity is quadratic per iteration, the overall complexity is $\mathcal{O}(M^3)$ if M is the size of the final active set. It is important to note that if the inverse if calculated directly in each iteration, the complexity is $\mathcal{O}(M^4)$; hence, the update with the Schur complement is essential.

The table below highlights the computational complexity of the full GP and the progressive approach to training. Note that, we still need to compute and store the full covariance matrix in order to speed up the matrix updates in the progressive GP approach.

Task	Full GP	Progressive GP
Training	$\mathcal{O}(N^3)$	$\mathcal{O}(M^3)$
Prediction	$\mathcal{O}(N^2)$	$\mathcal{O}(M^2N)$
Storage (for K)	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$

4.4. Selection of hyperparameters

The hyperparamters for the SE kernel used in the covariance matrix $\theta = (\sigma_f^2, l, \sigma_n^2)$ are pre-selected through optimization of the log marginal likelihood (LML) using a random subset of the training data in a pre-processing step. During the running of the algorithm, the hyperparameters remain fixed. In this paper, we mainly focus our efforts at providing a framework for selecting active targets and inputs from the training points while simultaneously training the GP. A framework that weaves together the hyperparameter selection and active set selection in the context of progressive training of GPs is being researched. Preliminary experiments where we varied the hyperparameters during stagewise training using marginal likelihood optimisation lead to instability. The authors of [8] highlight that active selection causes non-smooth fluctuations in the marginal likelihood making the optimisation difficult. Hence, a different way for concomitant hyperparameter optimisation needs to be developed.

5. Experiments

We trained a GP using the progressive training approach by sampling noisy (in order to have a systematic comparison the noise level for all the experiments was identical, further the random subset was ensured to be the same size as that for the progressive GP) values from a host of $1d$ functions; we then predict on a hold out unseen test set X_*. We report the generalisation error (RMSE) under three training schemes. The full GP, the random subset and the progressive GP algorithm.

Table 2. RMSE on Test data

Function \ Data	Full GP	Random	Progressive GP	% of full dataset
$x^2 sin(x)$	32.24	91.62	39.29	22%
$x sin(x)$	2.36	5.95	2.82	18%
$0.5 sin(x) + 0.5x - 0.02(x-5)^2$	1.14	2.17	1.96	31%

6. Discussion and Further Work

Experimentation with some standardised high dimensional datasets is a natural avenue to explore next. Further, we need to investigate the dynamics of choosing hyperparameters along with the active points from the data. It is evident that the hyperparameters chosen during initialisation do effect the choice of active points selected during the greedy search. The mechanics of this interplay between hyperparameters and active points needs to be investigated. The main criticism of this approach is that remainder points in $\mathbf{y}(R_t)$ don't play a role in the inference except in a weak sense of guiding the algorithm to the next stage. This information loss can be justified in situations where there is known to be redundancy in the data and the prediction based on the subset is accurate enough for the task. An important caveat is that the progressive approach which prescribes a method to select M points can be combined with a likelihood approximation scheme, some of which were discussed in section 3 to build a more powerful sparse GP algorithm; and one that makes use of the $N - M$ remainder points. For instance, the reduced rank approximation of the martix $K_{NN} \sim \tilde{K} = K_{NM} K_{MM}^{-1} K_{MN}$ can be used instead of the full covariance matrix K_{NN}. In conclusion, we have proposed a simple criteria to progressively train GPs along with update equations for the mean and covariance. The progressive GP approach yields sparse approximations to the full GP with marginal loss in prediction accuracy relative to training with the full dataset.

Acknowledgments: This work was supported by the Alan Turing Institute through the Doctoral Studentship for International Students.

Abbreviations

The following abbreviations are used in this manuscript:

GP	Gaussian processes
GPR	Gaussian process regression

Appendix A. The covariance function

The covariance function controls the properties of functions a GP can model, for example, smoothness, stationarity and shape; and can typically have multiple parameters called *hyperparameters*. The covariance function k takes as input, two input locations and hence can be viewed as representing some form of a similarity between the two data points. The process of learning the hyperparameters of the covariance function is an important part of GP training.

The *squared exponential*(SE) kernel is defined as,

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_f^2 \exp\Big(-\frac{(\boldsymbol{x}_i - \boldsymbol{x}_j)}{2l^2}\Big) + \delta_{ij}\sigma_n^2 \tag{A1}$$

where $\theta = (\sigma_f^2, l, \sigma_n^2)$ is the set of hyperparameters, comprising the signal variance $\sigma_f^2 > 0$ which controls the variation of function values from their mean, the lengthscale $l > 0$ which controls how smooth a function is and $\sigma_n^2 >= 0$ is the noise variance which allows for noise to be present in the data. The noise variance applies only when $i = j$. In the noiseless case, we just drop the additive noise variance term.

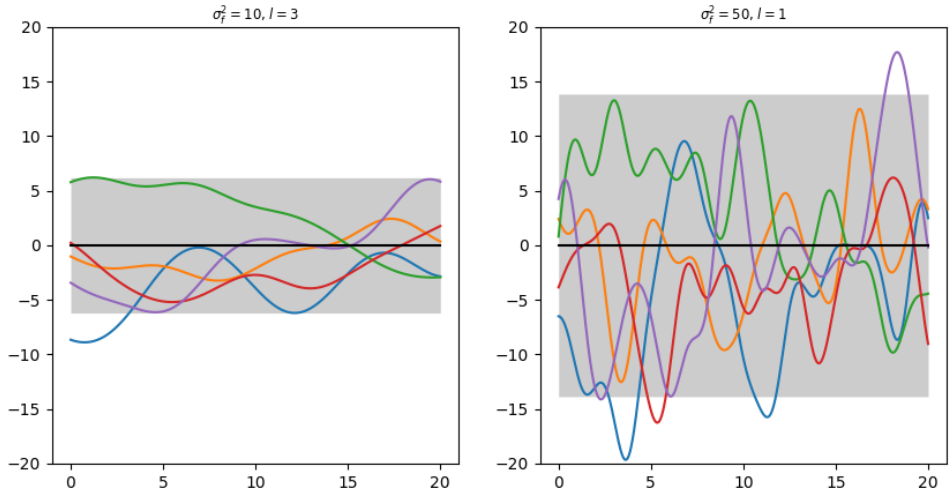


Figure A1. Samples from GP prior with SE covariance function and 95% confidence intervals ($\pm 1.96\sigma_f$)

Before performing GPR we calculate the covariance matrix by applying the covariance function to all possible combinations of the input points,

$$K(X,X) = \begin{bmatrix} k(\boldsymbol{x}_1,\boldsymbol{x}_1) & k(\boldsymbol{x}_1,\boldsymbol{x}_2) & \dots & k(\boldsymbol{x}_1,\boldsymbol{x}_N) \\ k(\boldsymbol{x}_2,\boldsymbol{x}_1) & k(\boldsymbol{x}_2,\boldsymbol{x}_2) & \dots & k(\boldsymbol{x}_2,\boldsymbol{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_N,\boldsymbol{x}_1) & k(\boldsymbol{x}_N,\boldsymbol{x}_2) & \dots & k(\boldsymbol{x}_N,\boldsymbol{x}_N) \end{bmatrix} \tag{A2}$$

For the sake of notational convenience we denote $K(X,X)$ as K.

Appendix B. GP Regression (GPR)

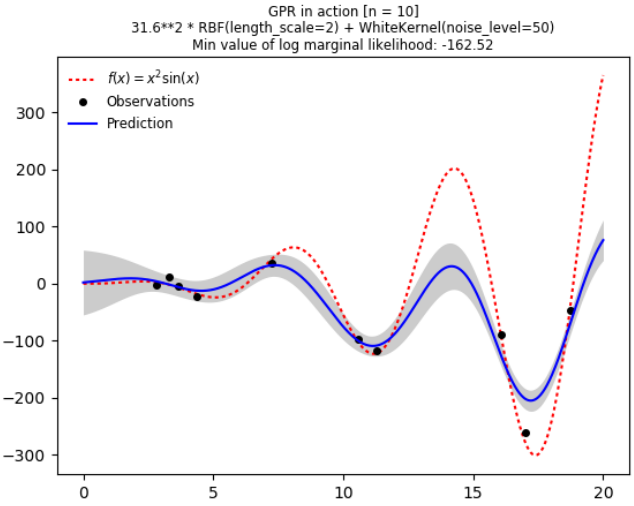


Figure A2. GPR in action where the kernel hyperparameters have been optimized through minimization of the negative log marginal likelihood

An important aspect of GPR is the selection of hyperparameters for the covariance function. The SE kernel shown in A1 has three free parameters. There are several approaches to setting the values of these hyperparameters based on training data, the joint problem of selecting an appropriate kernel function and its hyperparameters is called *model selection* [2].

Appendix C. Progressive updates

In this section, we discuss in detail the stagewise updates to the posterior predicted mean and covariance given in eq. 10.

Appendix C.0.1. Mean

$$\begin{gathered} \text{At stage t: } \mu_t = K_{\setminus I_t}(K_{I_t} + \sigma^2\mathbb{I})^{-1}\boldsymbol{y}(I_t) \\ \downarrow \\ \text{At stage t+1: } \mu_{t+1} = K_{\setminus I_{t+1}}(K_{I_{t+1}} + \sigma^2\mathbb{I})^{-1}\boldsymbol{y}(I_{t+1}) \end{gathered}$$

Notice that the mean update from stage $t \to t+1$ involves updating two matrices.

First,

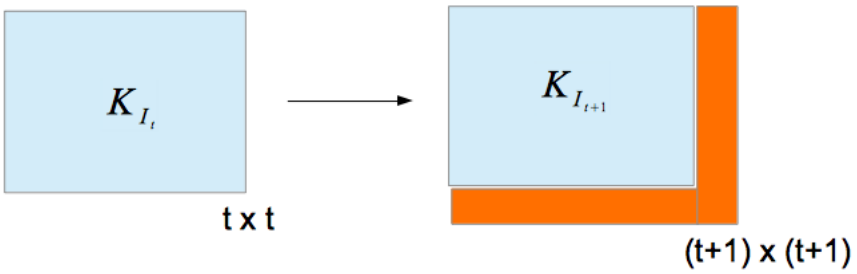
$$K_{\setminus I_t} \to K_{\setminus I_{t+1}}$$

In this update we evolve a $(N-t)\times t$ matrix into a $(N-(t+1))\times(t+1)$. We are dropping a row and adding a column. The newly added column contains the covariances computed between the newly selected active point, say s and all the $(N-(t+1))$ points in the remainder set $(k(s,r_i)|i \in R_{t+1})$. If we assume that the full covariance matrix $K(X,X)$ is computed at the start then this update just requires selecting the corresponding entries from the matrix $K(X,X)$. The complexity of this operation is just $\mathcal{O}(1)$.

Second,

$$K_{I_t} \rightarrow K_{I_{t+1}}$$

This is the covariance matrix computed on the active points, it grows by 1 row and 1 column in each stage.



Further, its inverse is required in each stage. We make use of block inversion to update the inverse. The complexity of this operation is quadratic per iteration, this is shown in section 4.3.

Appendix C.0.2. Covariance

$$\begin{aligned}\text{At stage t: } \Sigma_t &= K_{R_t} - K_{\setminus I_t}(K_{I_t}+\sigma^2\mathbb{I}_t)^{-1}K_{\setminus I_t}^T \\ &\downarrow \\ \text{At stage t+1: } \Sigma_{t+1} &= K_{R_{t+1}} - K_{\setminus I_{t+1}}(K_{I_{t+1}}+\sigma^2\mathbb{I}_t)^{-1}K_{\setminus I_{t+1}}^T\end{aligned}$$

The updates of $K_{\setminus I_t} \rightarrow K_{\setminus I_{t+1}}$ and $K_{I_t} \rightarrow K_{I_{t+1}}$ were already discussed in the previous section. The only new matrix update involved here is,

$$K_{R_t} \rightarrow K_{R_{t+1}}$$

Since, K_{R_t} is the covariance matrix computed on the remainder inputs and the size of the remainder inputs shrinks as the training progresses, $K_{R_t} \rightarrow K_{R_{t+1}}$ involves dropping a row and a column as the matrix shrinks from size $(N-t)\times(N-t)$ to $(N-(t+1))\times(N-(t+1))$.

Appendix D. Convergence of progressive training

At each stage of progressive training, the algorithm tracks the generalisation error by computing the posterior predictive mean and covariance on the remainder set $X(R_t)$. We define generalisation error as the square root of the sum of squared errors (abbreviated as RMSE) between the mean $\boldsymbol{\mu}_t$ and the true remainder targets $\mathbf{y}(R_t) = \{r_i|i \in R_t\}$.

The RMSE is given by the l^2 norm,

$$|\boldsymbol{\mu}_t - \mathbf{y}(R_t)|_2 = \sum_{i=1}^{N-t} (\mu_i - r_i)^2 \tag{A3}$$

where the mean prediction for test inputs at stage t is given by eq. 10.

If we denote the RMSE at stage t as ξ_t the algorithm terminates when,

$$\xi_{t-1} - \xi_t < \delta \tag{A4}$$

where δ denotes a threshold value usually tied to the range of the target values $\mathbf{y}$.

References

1. Quiñonero-Candela, J.; Rasmussen, C.E. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research* **2005**, *6*, 1939–1959.
2. Rasmussen, C.E. Gaussian processes in machine learning. In *Advanced lectures on machine learning*; Springer, 2004; pp. 63–71.
3. Seeger, M.; Williams, C.; Lawrence, N. Fast forward selection to speed up sparse Gaussian process regression. Artificial Intelligence and Statistics 9, 2003, number EPFL-CONF-161318.
4. Smola, A.J.; Bartlett, P.L. Sparse greedy Gaussian process regression. Advances in neural information processing systems, 2001, pp. 619–625.
5. Csató, L.; Opper, M. Sparse on-line Gaussian processes. *Neural computation* **2002**, *14*, 641–668.
6. Huber, M.F. Recursive Gaussian process: On-line regression and learning. *Pattern Recognition Letters* **2014**, *45*, 85–91.
7. Lawrence, N.D.; Seeger, M.; Herbrich, R. Fast sparse gaussian process methods: The informative vector machine. Advances in Neural Information Processing Systems 15. Citeseer, 2003.
8. Snelson, E.; Ghahramani, Z. Sparse Gaussian processes using pseudo-inputs. Advances in neural information processing systems, 2006, pp. 1257–1264.

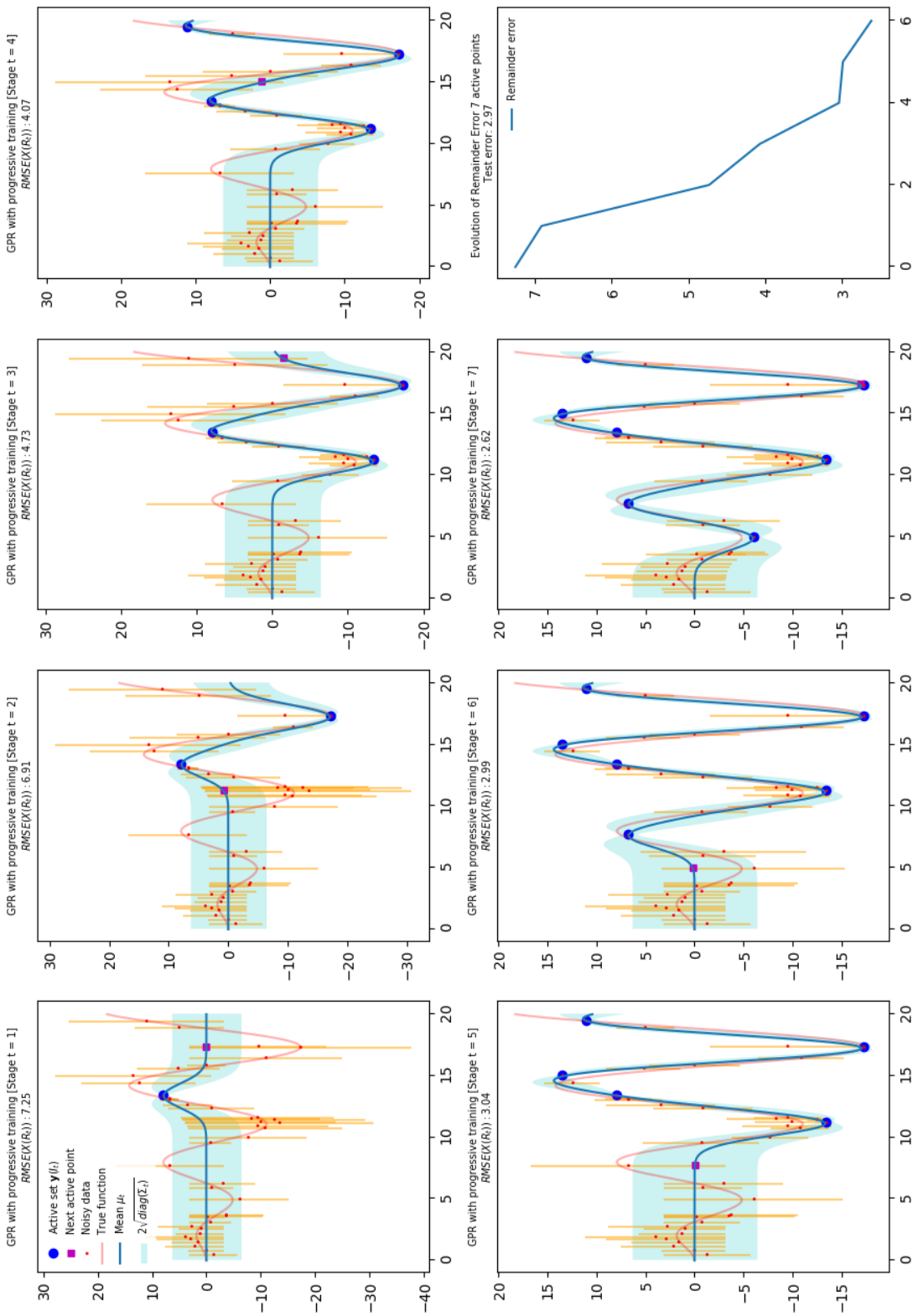


Figure A3. The progressive GP framework in action. The orange vertical bars show the Δ_3 metric on the basis of which the next active point is selected. The algorithm terminates after 7 iterations when the reduction in the RMSE on the remainder set is below a preset threshold.