A Greedy approximation scheme for Sparse Gaussian process regression

Vidhi Lalchand **A. C. Faul**
University of Cambridge, Cambridge, UK
The Alan Turing Institute, London, UK
{vr308, acf22}@cam.ac.uk

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. We demonstrate our method on selected one dimensional functions and show that the loss in accuracy due to using a subset of data points is marginal compared to the computational gains.

1 Introduction

Gaussian processes are nonparametric tools, allowing the complexity of the model to grow as more data is observed. Another attractive feature of GPs is the behaviour of the predictive variance 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. Quiñonero-Candela & Rasmussen (2005) provides a unifying summary of sparse approximations. A common theme in some of the earlier sparse methods involved developing a low-rank approximation to the covariance matrix, also called the Nyström approximation (Smola & Schölkopf, 2000; Seeger *et al.* , 2003). In these schemes the full covariance matrix of size N is replaced by the Nyström approximation requiring the inverse of a smaller matrix involving M data points. These class of methods are called *projected* process approximation schemes in Quiñonero-Candela & Rasmussen (2005). Most of the recent innovations in this field are driven by the variational approach in Titsias (2009).

In this short paper we focus on a greedy approximation scheme which results in a sequential construction of a subset of size M from the N training data points. The selection metric used to rank points for selection can be evaluated with low computational overhead. It might be worth noting that the task of selecting the active set in the context of regression is a general idea that can be coupled with different projected process approximation schemes to generate new methodologies.

Preprint. Work in progress.

2 GP Regression (GPR)

A GP is a collection of random variables $\{f(\mathbf{x})|\mathbf{x} \in X\}$, any finite number of which have a joint Gaussian distribution. A GP is fully specifed by the mean function $\mu(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$ which are user defined, the covarinace function typically depends on a set of hyperparameters $\boldsymbol{\theta}$. GPs can be used to define a distribution over functions $f(\mathbf{x}) \sim \mathcal{GP}(\mu, k)$ as they can be viewed as a collection of random variables, this means that any finite collection of function values $[f(\mathbf{x_1}), \ldots, f(\mathbf{x_N})]$ have a joint Gaussian distribution.

$$[f(\mathbf{x_1}), \ldots, f(\mathbf{x_N})] \sim \mathcal{N}(\boldsymbol{\mu}, K) \tag{1}$$

where $\boldsymbol{\mu}$ is the $N \times 1$ vector $\mu_i = \mu(\mathbf{x_i})$ and K is the $N \times N$ covariance matrix with $K_{ij} = k(\mathbf{x_i}, \mathbf{x_j})$.

Our training dataset consists of N pairs of data $(\mathbf{x_i}, y_i)_{i=1}^N$ where y_i are noisy realisations of some latent function f with Gaussian noise $y_i = f(\mathbf{x}_i) + \epsilon_i$, $\epsilon_i \in \mathcal{N}(0, \sigma^2)$. Let X, $\mathbf{y}$ denote the training inputs and noisy targets and $\mathbf{f}$ denote the vector of underlying latent function values. The liklihood of the data $\mathbf{y}|\mathbf{f} \sim \mathcal{N}(f, \sigma^2 I)$ and the prior $\mathbf{f} \sim \mathcal{N}(0, K)$ give the joint probability model $p(\mathbf{f}, \mathbf{y}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f})$. The predictive distribution at a set of test inputs X_* is given in closed form using properties of conditional Gaussians,

$$\begin{aligned} \mathbf{f}_*|\mathbf{y}, X, X_*, \boldsymbol{\theta}, \sigma^2 &\sim \mathcal{N}(\bar{\mathbf{f}_*}, cov(\mathbf{f}_*)) \\ \bar{\mathbf{f}_*} &= K_*(K + \sigma^2 I)^{-1}\mathbf{y} \\ cov(\mathbf{f}_*) &= K_{**} - K_*(K + \sigma^2 I)^{-1}K_*^T \end{aligned} \tag{2}$$

where K_{**} denotes the covariance matrix evaluated between the test inputs X_* and K_* denotes the covariance matrix evaluated between the test inputs X_* and training inputs X, if there are N_* test inputs the covariance matrix K_{**} is of size $N_* \times N_*$ and K_* is of size $N_* \times N$. The hyperparameters along with the noise variance $(\boldsymbol{\theta}, \sigma^2)$ are inferred through optimisation of the log marginal likelihood given by $\int p(\mathbf{y}|\mathbf{f})p(\mathbf{f})d\mathbf{f}$ which can be analytically derived through marginalising $\mathbf{f}$.

3 Greedy framework for Gaussian Process Regression

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 Seeger *et al.* (2003) 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 $\mathbf{y}_*$.

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

Table 1: Notation used in stagewise training

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.

- $K_{I_t} = K(X(I_t), X(I_t))$ denotes the $t \times t$ covariance matrix computed between the active inputs at the t^{th} stage of training.
- $K_{\setminus I_t} = K(X(R_t), X(I_t))$ 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.
- $\boldsymbol{\mu}_t$ and Σ_t denote the predictive posterior mean and covariance computed at stage t for the remainder inputs $X(R_t)$.

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 $\mathbf{y}(R_t) = \{r_i | i \in R_t\}$ and the predicted mean $\boldsymbol{\mu}_t$ provides the basis for convergence. If the decrease in $|\boldsymbol{\mu}_t - \mathbf{y}(R_t)|_2 = \sum_{i=1}^{N-t}(\mu_i - r_i)^2$ is under a threshold, we terminate.

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.
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 (see A.1 for a detailed discussion).

At stage t,

$$\mathbf{y}(R_t)|\mathbf{y}(I_t), X(I_t), X(R_t) \sim \mathcal{N}(\boldsymbol{\mu_t}, \Sigma_t) \tag{3}$$

$$\boldsymbol{\mu}_t = K_{\setminus I_t}(K_{I_t} + \sigma_n^2 \mathbb{I})^{-1}\mathbf{y}(I_t)$$

$$\Sigma_t = K_{R_t} - K_{\setminus I_t}(K_{I_t} + \sigma_n^2 \mathbb{I})^{-1}K_{\setminus I_t}^T \tag{4}$$

The above equations reflect the predictive posterior mean and covariance for a full GP introduced in eq. 2 where the active set $\mathbf{y}(I_t)$ plays the role of the target vector $\mathbf{y}$. The hyperparameters $(\boldsymbol{\theta}, \sigma^2)$ are kept fixed during the greedy training. They are estimated by optimising the marginal likelihood on a random subset of the training data (see section A.3 for a discussion).

3.1 The Algorithm

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

Algorithm 1 Greedy 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 $\boldsymbol{y}(R_t) = \boldsymbol{y} \setminus \boldsymbol{y}(I_t)$.
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 A.1 for sequential update rules.)
    Compute $\Delta_i \forall i \in R_t$
    Select i where $i = argmax_{i \in R_t} \Delta_i$
    $I_{t+1} \leftarrow I_t \cup \{i\}, R_{t+1} \leftarrow R_t \setminus \{i\}$
    If convergence is **true**:
        break;
end return I_T, R_T

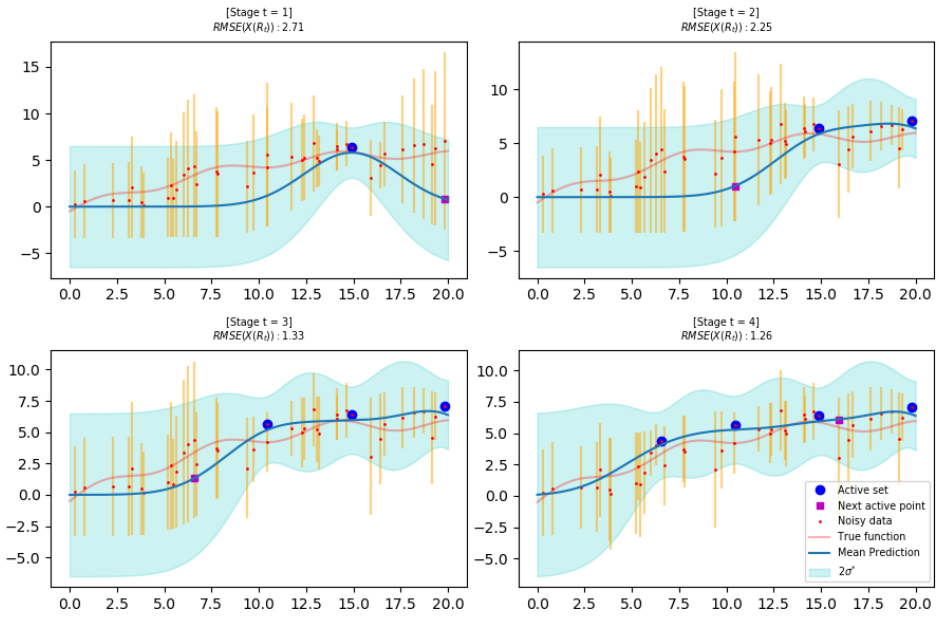

Figure 1: A demonstration of the greedy GP training (stages 1-4). The orange vertical lines denote the Δ selection metric. The algorithm terminates after 13 iterations; here we only depict the first 4 iterations owing to lack of space.

The algorithm advances in stages by selecting the next active point from the remainder set as the maximiser of the selection criteria Δ given by the additive term:

$$argmax_{u \in \boldsymbol{y}(R_t)} \Delta = \overbrace{\sqrt{diag(\Sigma_t)}}^{uncertainty} + \overbrace{|(\boldsymbol{\mu}_t - \boldsymbol{y}(R_t))|}^{underfit}$$

The component terms in Δ address the two-fold objective of targeting regions of high uncertianty captured by the term $\sqrt{diag(\Sigma_t)}$ which is the posterior predicted standard deviation at stage t and underfit captured by the error term $|(\boldsymbol{\mu}_t - \boldsymbol{y}(R_t))|$ which denotes the deviation of the remainder targets from the posterior predicted mean.

Note that both components of the addition are vectors of size $(N - t)$ as they are based on the remainder set. The metric Δ 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 greedy training for the function $x \sin x$ is shown in fig. 1. The computational complexity of the full greedy training algorithm is given in section A.2.

3.2 Experiments

We trained a GP using the greedy training approach by sampling noisy [1] 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 [2] training schemes. The squared exponential (SE) kernel was used in all the three schemes (see section A.3).

Function \ Data	Full GP	Random	Greedy 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%

Table 2: RMSE on Test data

[1] In order to have a systematic comparison the noise level for all the experiments was identical.

[2] The random subset scheme used targets sampled uniformly from the training data and were ensured to be the same size as that for the Greedy GP.

Acknowledgments

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

References

Quiñonero-Candela, Joaquin, & Rasmussen, Carl Edward. 2005. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, **6**(Dec), 1939–1959.

Seeger, Matthias, Williams, Christopher, & Lawrence, Neil. 2003. Fast forward selection to speed up sparse Gaussian process regression. *In: Artificial Intelligence and Statistics 9*.

Smola, Alex J, & Schölkopf, Bernhard. 2000. Sparse greedy matrix approximation for machine learning.

Snelson, Edward, & Ghahramani, Zoubin. 2006. Sparse Gaussian processes using pseudo-inputs. *Pages 1257–1264 of: Advances in neural information processing systems*.

Titsias, Michalis. 2009. Variational learning of inducing variables in sparse Gaussian processes. *Pages 567–574 of: Artificial Intelligence and Statistics*.

A Appendix

A.1 Greedy updates

In this section, we discuss in detail the stagewise updates to the posterior predicted mean and covariance given in eq. 4.

A.1.1 Mean

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

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,

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

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 A.2.

A.1.2 Covariance

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

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 $(N-(t+1)) \times (N-(t+1))$.

A.2 Complexity of Greedy 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 greedy 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.

$$\sum_{i=1}^{M} i^2 = 1^2 + 2^2 + \ldots + M^2 = \frac{M(M+1)(2M+1)}{6} = \mathcal{O}(M^3) \tag{5}$$

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 greedy 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 greedy GP approach.

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

A.3 Selection of hyperparameters

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

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_f^2 \exp\Big(-\frac{(\boldsymbol{x}_i - \boldsymbol{x}_j)}{2l^2} \Big) + \delta_{ij}\sigma_n^2 \tag{6}$$

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.

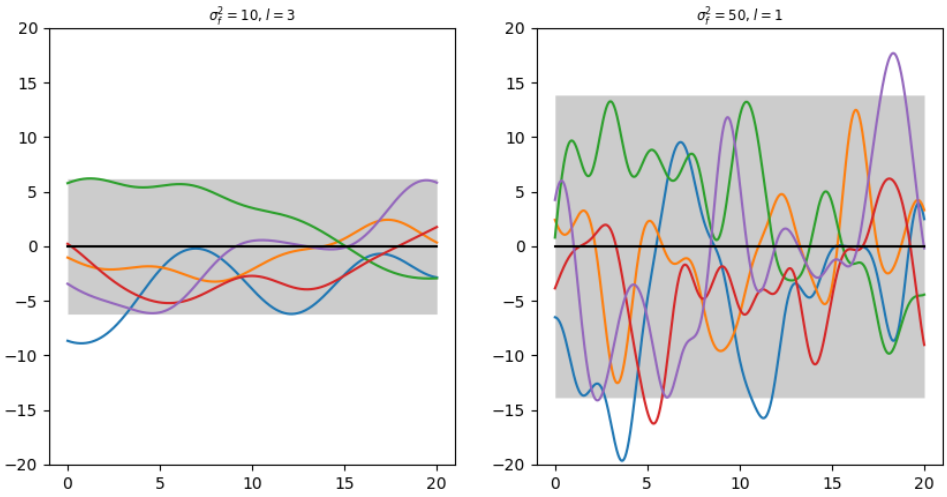

Figure 2: Samples from GP prior with SE covariance function and 95% confidence intervals $(\pm 1.96\sigma_f)$

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 greedy 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 Snelson & Ghahramani (2006) highlight that active selection causes non-smooth fluctuations in the marginal likelihood making the optimisation difficult. Hence, a different approach needs to be developed.