Analysis of Some Methods for Reduced Rank Gaussian Process Regression

Joaquin Quiñonero-Candela[1,2] and Carl Edward Rasmussen[2]

[1] Informatics and Mathematical Modelling, Technical University of Denmark, Richard Petersens Plads, B321, 2800 Kongens Lyngby, Denmark
jqc@imm.dtu.dk
[2] Max Planck Institute for Biological Cybernetics, Spemann straße 38, 72076 Tübingen, Germany
carl@tuebingen.mpg.de

Abstract. While there is strong motivation for using Gaussian Processes (GPs) due to their excellent performance in regression and classification problems, their computational complexity makes them impractical when the size of the training set exceeds a few thousand cases. This has motivated the recent proliferation of a number of cost-effective approximations to GPs, both for classification and for regression. In this paper we analyze one popular approximation to GPs for regression: the reduced rank approximation. While generally GPs are equivalent to infinite linear models, we show that Reduced Rank Gaussian Processes (RRGPs) are equivalent to finite sparse linear models. We also introduce the concept of degenerate GPs and show that they correspond to inappropriate priors. We show how to modify the RRGP to prevent it from being degenerate at test time. Training RRGPs consists both in learning the covariance function hyperparameters and the support set. We propose a method for learning hyperparameters for a given support set. We also review the Sparse Greedy GP (SGGP) approximation (Smola and Bartlett, 2001), which is a way of learning the support set for given hyperparameters based on approximating the posterior. We propose an alternative method to the SGGP that has better generalization capabilities. Finally we make experiments to compare the different ways of training a RRGP. We provide some Matlab code for learning RRGPs.

1 Motivation and Organization of the Paper

Gaussian Processes (GPs) have state of the art performance in regression and classification problems, but they suffer from high computational cost for learning and predictions. For a training set containing n cases, the complexity of training is $\mathcal{O}(n^3)$ and that of making a prediction is $\mathcal{O}(n)$ for computing the predictive mean, and $\mathcal{O}(n^2)$ for computing the predictive variance.

A few computationally effective approximations to GPs have recently been proposed. These include the sparse iterative schemes of Csató (2002), Csató and Opper (2002), Seeger (2003), and Lawrence et al. (2003), all based

R. Murray-Smith, R. Shorten (Eds.): Switching and Learning, LNCS 3355, pp. 98–127, 2005.

on minimizing KL divergences between approximating and true posterior; Smola and Schölkopf (2000) and Smola and Bartlett (2001) based on low rank approximate posterior, Gibbs and MacKay (1997) and Williams and Seeger (2001) on matrix approximations and Tresp (2000) on neglecting correlations. Subsets of regressors (Wahba et al., 1999) and the Relevance Vector Machine (Tipping, 2001) can also be cast as sparse linear approximations to GPs. Schwaighofer and Tresp (2003) provide a very interesting yet brief comparison of some of these approximations to GPs. They only address the quality of the approximations in terms of the predictive mean, ignoring the predictive uncertainties, and leaving some theoretical questions unanswered, like the goodness of approximating the maximum of the posterior.

In this paper we analyze sparse linear or equivalently reduced rank approximations to GPs that we will call Reduced Rank Gaussian Processes (RRGPs). We introduce the concept of degenerate Gaussian Processes and explain that they correspond to inappropriate priors over functions (for example, the predictive variance shrinks as the test points move far from the training set). We show that if not used with care at prediction time, RRGP approximations result in degenerate GPs. We give a solution to this problem, consisting in augmenting the finite linear model at test time. This guarantees that the RRGP approach corresponds to an appropriate prior. Our analysis of RRGPs should be of interest in general for better understanding the infinite nature of Gaussian Processes and the limitations of diverse approximations (in particular of those based solely on the posterior distribution).

Learning RRGPs implies both selecting a support set, and learning the hyperparameters of the covariance function. Doing both simultaneously proves to be difficult in practice and questionable theoretically. Smola and Bartlett (2001) proposed the Sparse Greedy Gaussian Process (SGGP), a method for learning the support set for given hyperparameters of the covariance function based on approximating the posterior. We show that approximating the posterior is unsatisfactory, since it fails to guarantee generalization, and propose a theoretically more sound greedy algorithm for support set selection based on maximizing the marginal likelihood. We show that the SGGP relates to our method in that approximating the posterior reduces to partially maximizing the marginal likelihood. We illustrate our analysis with an example. We propose an approach for learning the hyperparameters of the covariance function of RRGPs for a given support set, originally introduced by Rasmussen (2002). We also provide Matlab code in Appendix B for this method.

We make experiments where we compare learning based on selecting the support set to learning based on inferring the hyperparameters. We give special importance to evaluating the quality of the different approximations to computing predictive variances.

The paper is organized as follows. We give a brief introduction to GPs in Sect. 2. In Sect. 3 we establish the equivalence between GPs and linear models, showing that in the general case GPs are equivalent to *infinite* linear models. We also present degenerate GPs. In Sect. 4 introduce RRGPs and address the

issue of training them. In Sect. 5 we present the experiments we conducted. We give some discussion in Sect. 6.

2 Introduction to Gaussian Processes

In inference with parametric models prior distributions are often imposed over the model parameters, which can be seen as a means of imposing regularity and improving generalization. The form of the parametric model, together with the form of the prior distribution on the parameters result in a (often implicit) prior assumption on the joint distribution of the function values. At prediction time the quality of the predictive uncertainty will depend on the prior over functions. Unfortunately, for probabilistic parametric models this prior is defined in an indirect way, and this in many cases results in priors with undesired properties. An example of a model with a peculiar prior over functions is the Relevance Vector Machine introduced by Tipping (2001) for which the predictive variance shrinks for a query point far away from the training inputs. If this property of the predictive variance is undesired, then one concludes that the prior over functions was undesirable in the first place, and one would have been happy to be able to directly define a prior over functions.

Gaussian Processes (GPs) are non-parametric models where a Gaussian process[3] prior is directly defined over function values. The direct use of Gaussian Processes as priors over functions was motivated by Neal (1996) as he was studying priors over weights for artificial neural networks. A model equivalent to GPs, *kriging*, has since long been used for analysis of spatial data in Geostatistics (Cressie, 1993). In a more formal way, in a GP the function outputs $f(\boldsymbol{x}_i)$ are a collection random variables indexed by the inputs $\boldsymbol{x}_i$. Any finite subset of outputs has a joint multivariate Gaussian distribution (for an introduction on GPs, and thorough comparison with Neural Networks see (Rasmussen, 1996)). Given a set of training inputs $\{\boldsymbol{x}_i | i = 1, \ldots, n\} \subset \mathbb{R}^D$ (organized as rows in matrix X), the joint prior distribution of the corresponding function outputs $\mathbf{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n)]^\top$ is Gaussian $p(\mathbf{f}|X, \theta) \sim \mathcal{N}(0, K)$, with zero mean (this is a common and arbitrary choice) and *covariance matrix* $K_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The GP is entirely determined by the *covariance function* $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ with parameters θ.

An example of covariance function that is very commonly used is the squared exponential:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \theta_{D+1}^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{D}\frac{1}{\theta_d^2}\left(X_{id} - X_{jd}\right)^2\right) . \tag{1}$$

θ_{D+1} relates to the amplitude of the functions generated by the GP, and θ_d is a lengthscale in the d-th dimension that allows for Automatic Relevance Determination (ARD) (MacKay, 1994; Neal, 1996): if some input dimensions are

[3] We will use the expression "Gaussian Process" (both with capital first letter) or "GP" to designate the non-parametric model where a Gaussian process prior is defined over function values

un-informative about the covariance between observed training targets, their associated θ_d will be made large (or effectively infinite) and the corresponding input dimension will be effectively pruned from the model. We will call the parameters of the covariance function *hyperparameters*, since they are the parameters of the prior.

In general, inference requires choosing a parametric form of the covariance function, and either estimating the corresponding parameters θ (which is named by some Maximum Likelihood II, or second level of inference) or integrating them out (often through MCMC). We will make the common assumption of Gaussian independent identically distributed output noise, of variance σ^2. The training outputs $\mathbf{y} = [y_1, \ldots, y_n]^\top$ (or targets) are thus related to the function evaluated at the training inputs by a likelihood distribution[4] $p(\mathbf{y}|\mathbf{f}, \sigma^2) \sim \mathcal{N}(\mathbf{f}, \sigma^2 I)$, where I is the identity matrix. The posterior distribution over function values is useful for making predictions. It is obtained by applying Bayes' rule:[5]

$$\begin{aligned} p(\mathbf{f}|\mathbf{y}, X, \theta, \sigma^2) &= \frac{p(\mathbf{y}|\mathbf{f}, \sigma^2)\, p(\mathbf{f}|X, \theta)}{p(\mathbf{y}|X, \theta, \sigma^2)} \\ &\sim \mathcal{N}\left(K^\top \left(K + \sigma^2 I\right)^{-1} \mathbf{y}, K - K^\top \left(K + \sigma^2 I\right)^{-1} K\right) \quad . \end{aligned} \tag{2}$$

The mean of the posterior does not need to coincide with the training targets. This would be the case however, if the estimated noise variance happened to be zero, in which case the posterior at the training cases would be a delta function centered on the targets.

Consider now that we observe a new input $\boldsymbol{x}_*$ and would like to know the distribution of $f(\boldsymbol{x}_*)$ (that we will write as f_* for convenience) conditioned on the observed data, and on a particular value of the hyperparameters and of the output noise variance. The first thing to do is to write the augmented prior over the function values at the training inputs and the new function value at the new test input:

$$p\left(\begin{bmatrix}\mathbf{f} \\ f_*\end{bmatrix} \middle| \boldsymbol{x}_*, X, \theta\right) \sim \mathcal{N}\left(0, \begin{bmatrix} K & \mathbf{k}_* \\ \mathbf{k}_*^\top & k_{**}\end{bmatrix}\right) \quad , \tag{3}$$

where $\mathbf{k}_* = [K(\boldsymbol{x}_*, \boldsymbol{x}_1), \ldots, K(\boldsymbol{x}_*, \boldsymbol{x}_n)]^\top$ and $k_{**} = K(\boldsymbol{x}_*, \boldsymbol{x}_*)$. Then we can write the distribution of f_* conditioned on the training function outputs:

$$p(f_*|\mathbf{f}, \boldsymbol{x}_*, X, \theta) \sim \mathcal{N}\left(\mathbf{k}*^\top K^{-1}\mathbf{f}, k_{**} - \mathbf{k}*^\top K^{-1}\mathbf{k}_*\right) \quad . \tag{4}$$

The predictive distribution of f_* is obtained by integrating out the training function values $\mathbf{f}$ from (4) over the posterior distribution (2). The predictive distribution is Gaussian:

$$\begin{aligned} p(f_*|\mathbf{y}, \boldsymbol{x}_*, X, \theta, \sigma^2) &= \int p(f_*|\mathbf{f}, \boldsymbol{x}_*, X, \theta)\, p(\mathbf{f}|\mathbf{y}, X, \theta, \sigma^2)\, \mathrm{d}\mathbf{f} \\ &\sim \mathcal{N}\left(m(\boldsymbol{x}_*), v(\boldsymbol{x}_*)\right) \quad , \end{aligned} \tag{5}$$

[4] Notice that learning cannot be achieved from the likelihood alone: defining a prior over function values is essential to learning.

[5] In Sect. A.2 some algebra useful for deriving (2) is given: notice that the likelihood $p(\mathbf{y}|\mathbf{f}, \sigma^2)$ is also Gaussian in $\mathbf{f}$ with mean $\mathbf{y}$.

with mean and variance given by:

$$m(\boldsymbol{x}_*) = \mathbf{k}_*^\top \left(K + \sigma^2 I\right)^{-1} \mathbf{y} \ , \quad v(\boldsymbol{x}_*) = k_{**} - \mathbf{k}_*^\top \left(K + \sigma^2 I\right)^{-1} \mathbf{k}_* \ . \tag{6}$$

Another way of obtaining the predictive distribution of f_* is to augment the evidence with a new element y_* corresponding to the noisy version of f_* and to then write the conditional distribution of y_* given the training targets $\mathbf{y}$. The variance of the predictive distribution of y_* is equal to that of the predictive distribution of f_* (6) plus the noise variance σ^2, while the means are identical (the noise has zero mean).

Both if one chooses to learn the hyperparameters or to be Bayesian and do integration, the marginal likelihood of the hyperparameters (or evidence of the observed targets)[6] must be computed. In the first case this quantity will be maximized with respect to the hyperparameters, and in the second case it will be part of the posterior distribution from which the hyperparameters will be sampled. The evidence is obtained by averaging the likelihood over the prior distribution on the function values:

$$p(\mathbf{y}|X, \theta, \sigma^2) = \int p(\mathbf{y}|\mathbf{f})\, p(\mathbf{f}|X, \theta)\, \mathrm{d}\mathbf{f} \sim \mathcal{N}\left(0, K + \sigma^2 I\right) \ . \tag{7}$$

Notice that the evidence only differs from the prior over function values in a "ridge" term added to the covariance, that corresponds to the additive Gaussian i.i.d. output noise. Maximum likelihood II learning involves estimating the hyperparameters θ and the noise variance σ^2 by minimizing (usually for convenience) the negative log evidence. Let $Q \equiv \left(K + \sigma^2 I\right)$. The cost function and its derivatives are given by:

$$\begin{aligned} \mathcal{L} &= \frac{1}{2} \log |Q| + \frac{1}{2} \mathbf{y}^\top Q^{-1} \mathbf{y} \ , \\ \frac{\partial \mathcal{L}}{\partial \theta_i} &= \frac{1}{2} \mathrm{Tr}\left(Q^{-1} \frac{\partial Q}{\partial \theta_i}\right) - \mathbf{y}^\top Q^{-1} \frac{\partial Q}{\partial \theta_i} Q^{-1} \mathbf{y} \ , \\ \frac{\partial \mathcal{L}}{\partial \sigma^2} &= \frac{1}{2} \mathrm{Tr}\left(Q^{-1}\right) - \mathbf{y}^\top Q^{-1} Q^{-1} \mathbf{y} \ , \end{aligned} \tag{8}$$

and one can use some gradient descent algorithm to minimize $\mathcal{L}$ (conjugate gradient gives good results, Rasmussen, 1996).

For Gaussian processes, the computational cost of learning is marked by the need to invert matrix Q and therefore scales with the cube of the number of training cases ($\mathcal{O}(n^3)$). If Q^{-1} is known (obtained from the learning process), the computational cost of making predictions is $\mathcal{O}(n)$ for computing the predictive mean, and $\mathcal{O}(n^2)$ for the predictive variance for each test case. There is a need for approximations that simplify the computational cost if Gaussian Processes are to be used with large training datasets.

[6] We will from now on use indistinctly "marginal likelihood" or "evidence" to refer to this distribution.

3 Gaussian Processes as Linear Models

Gaussian Processes correspond to parametric models with an infinite number of parameters. Williams (1997a) showed that infinite neural networks with certain transfer functions and the appropriate priors on the weights are equivalent to Gaussian Processes with a particular "neural network" covariance function. Conversely, any Gaussian Process is equivalent to a parametric model, that can be infinite.

In Sect(s). 3.1 and 3.2 we establish the equivalence between GPs and linear models. For the common case of GPs with covariance functions that cannot be expressed as a finite expansion, the equivalent linear models are infinite. However, it might still be interesting to approximate such GPs by a finite linear model, which results in *degenerate* Gaussian Processes. In Sect. 3.3 we introduce degenerate GPs and explain that they often correspond to inappropriate priors over functions, implying counterintuitive predictive variances. We then show how to modify these degenerate GPs at test time to obtain more appropriate priors over functions.

3.1 From Linear Models to GPs

Consider the following extended linear model, where the model outputs are a linear combination of the response of a set of basis functions $\{\phi_j(\boldsymbol{x})|j = 1, \ldots, m\} \subset [\mathbb{R}^D \to \mathbb{R}]$:

$$f(\boldsymbol{x}_i) \;=\; \sum_{j=1}^{m} \phi_j(\boldsymbol{x}_i)\, \alpha_j = \boldsymbol{\phi}(\boldsymbol{x}_i)\, \boldsymbol{\alpha} \;, \qquad\qquad \mathbf{f} \;=\; \Phi\, \boldsymbol{\alpha} \;, \tag{9}$$

where as earlier $\mathbf{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n)]^\top$ are the function outputs. The weights are organized in a vector $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_M]^\top$, and $\phi_j(\boldsymbol{x}_i)$ is the response of the j-th basis function to input x_i. $\boldsymbol{\phi}(\boldsymbol{x}_i) = [\phi_1(\boldsymbol{x}_i), \ldots, \phi_m(\boldsymbol{x}_i)]$ is a row vector that contains the response of all m basis functions to input $\boldsymbol{x}_i$ and matrix Φ (sometimes called *design matrix*) has as its i-th row vector $\boldsymbol{\phi}(\boldsymbol{x}_i)$. Let us define a Gaussian prior over the weights, of the form $p(\boldsymbol{\alpha}|A) \sim \mathcal{N}(0, A)$. Since $\mathbf{f}$ is a linear function of $\boldsymbol{\alpha}$ it has a Gaussian distribution under the prior on $\boldsymbol{\alpha}$, with mean zero. The prior distribution of $\mathbf{f}$ is:

$$p(\mathbf{f}|A, \Phi) \sim \mathcal{N}(0, C) \;, \qquad C = \Phi\, A\, \Phi^\top \;. \tag{10}$$

The model we have defined corresponds to a Gaussian Process. Now, if the number of basis functions m is smaller than the number of training points n, then C will not have full rank and the probability distribution of $\mathbf{f}$ will be an elliptical pancake confined to an m-dimensional subspace in the n-dimensional space where $\mathbf{f}$ lives (Mackay, 1997).

Let again $\mathbf{y}$ be the vector of observed training targets, and assume that the output noise is additive Gaussian i.i.d. of mean zero and variance σ^2. The likelihood of the weights is then Gaussian (in $\mathbf{y}$ and in $\boldsymbol{\alpha}$) given by

$p(\mathbf{y}|\boldsymbol{\alpha}, \Phi, \sigma^2) \sim \mathcal{N}(\Phi\,\boldsymbol{\alpha}, \sigma^2\, I)$. The prior over the training targets is then given by

$$p(\mathbf{y}|A, \Phi, \sigma^2) \sim (0, \sigma^2\, I + C) \ , \tag{11}$$

and has a full rank covariance, even if C is rank deficient.

To make predictions, one option is to build the joint distribution of the training targets and the new test function value and then condition on the targets. The other option is to compute the posterior distribution over the weights from the likelihood and the prior. Williams (1997b) refers to the first option as the "function-space view" and to the second as the "weight-space view". This distinction has inspired us for writing the next two sections.

The Parameter Space View. Using Bayes' rule, we find that the posterior is the product of two Gaussians in $\boldsymbol{\alpha}$, and is therefore a Gaussian distribution:

$$\begin{aligned} p(\boldsymbol{\alpha}|\mathbf{y}, A, \Phi, \sigma^2) &= \frac{p(\mathbf{y}|\boldsymbol{\alpha}, \Phi, \sigma^2)\, p(\boldsymbol{\alpha}|A)}{p(\mathbf{y}|A, \Phi, \sigma^2)} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma) \ , \\ \boldsymbol{\mu} = \sigma^{-2}\, \Sigma\, \Phi^\top\, \mathbf{y} \ , &\qquad \Sigma = \left[\sigma^{-2}\, \Phi^\top \Phi + A^{-1}\right]^{-1} \ . \end{aligned} \tag{12}$$

The maximum a posteriori (MAP) estimate of the model weights is given by $\boldsymbol{\mu}$. If we rewrite this quantity as $\boldsymbol{\mu} = [\Phi^\top \Phi + \sigma^2\, A]^{-1} \Phi^\top \mathbf{y}$, we can see that the Gaussian assumption on the prior over the weights and on the output noise results in $\boldsymbol{\mu}$ being given by a regularized version of the normal equations. For a new test point $\boldsymbol{x}_*$, the corresponding function value is $f_* = \boldsymbol{\phi}(\boldsymbol{x}_*)\, \boldsymbol{\alpha}$; for making predictions the α's are drawn from the posterior. Since f_* is linear in $\boldsymbol{\alpha}$, it is quite clear that the predictive distribution $p(f_*|\mathbf{y}, A, \Phi, \sigma^2)$ is Gaussian, with mean and variance given by:

$$m(\boldsymbol{x}_*) = \boldsymbol{\phi}(\boldsymbol{x}_*)^\top \boldsymbol{\mu} \ , \qquad v(\boldsymbol{x}_*) = \boldsymbol{\phi}(\boldsymbol{x}_*)^\top\, \Sigma\, \boldsymbol{\phi}(\boldsymbol{x}_*) \ . \tag{13}$$

We can rewrite the posterior covariance using the matrix inversion lemma (see Appendix A.1) as $\Sigma = A - A[\sigma^2\, I + \Phi\, A\, \Phi^\top]^{-1}\, A$. This expression allows us to rewrite the predictive mean and variance as:

$$\begin{aligned} m(\boldsymbol{x}_*) &= \boldsymbol{\phi}(\boldsymbol{x}_*)^\top\, A\, \Phi^\top [\sigma^2\, I + \Phi\, A\, \Phi^\top]^{-1} \mathbf{y} \ , \\ v(\boldsymbol{x}_*) &= \boldsymbol{\phi}(\boldsymbol{x}_*)^\top\, A\, \boldsymbol{\phi}(\boldsymbol{x}_*) - \boldsymbol{\phi}(\boldsymbol{x}_*)^\top\, A\, \Phi^\top [\sigma^2\, I + \Phi\, A\, \Phi^\top]^{-1} \Phi\, A\, \boldsymbol{\phi}(\boldsymbol{x}_*) \ , \end{aligned} \tag{14}$$

which will be useful for relating the parameter space view to the GP view.

The Gaussian Process View. There exists a Gaussian Process that is equivalent to our linear model with Gaussian priors on the weights given by (9). The covariance function of the equivalent GP is given by:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_i)^\top\, A\, \boldsymbol{\phi}(\boldsymbol{x}_j) = \sum_{k=1}^{m} \sum_{l=1}^{m} A_{kl}\, \phi_k(\boldsymbol{x}_i)\, \phi_l(\boldsymbol{x}_j) \ . \tag{15}$$

The covariance matrix of the prior over training function values is given by $K = \Phi\, A\, \Phi^\top$ and we recover the same prior as in (10). Taking the same noise model as previously, the prior over targets is identical to (11).

Given a new test input $\boldsymbol{x}_*$, the vector of covariances between $\mathbf{f}_*$ and the training function values is given by $\mathbf{k}_* = \Phi\, A\, \boldsymbol{\phi}(\boldsymbol{x}_*)$ and the prior variance of f_* is $k_{**} = \boldsymbol{\phi}(\boldsymbol{x}_*)\, A\, \boldsymbol{\phi}(\boldsymbol{x}_*)$. Plugging these expressions into the equations for the predictive mean and variance of a GP (6) one recovers the expressions given by (14) and (13). The predictive mean and variance of a GP with covariance function given by (15) are therefore identical to the predictive mean and variance of the linear model.

A fundamental property of the GP view of a linear model is that the set of m basis functions appear *exclusively* as inner products. Linear models where m is infinite are thus tractable under the GP view, provided that the basis functions and the prior over the weights are appropriately chosen. By appropriately chosen we mean such that a generalized dot product exists in feature space, that allows for the use of the *"kernel trick"*. Schölkopf and Smola (2002) provide with extensive background on kernels and the *"kernel trick"*.

Let us reproduce here an example given by Mackay (1997). Consider a one-dimensional input space, and let us use squared exponential basis functions $\phi_c(x_i) = \exp(-(x_i - c)/(2\lambda^2))$, where c is a given center in input space and λ is a known lengthscale. Let us also define an isotropic prior over the weights, of the form $A = \sigma_\alpha^2\, I$. We want to make m go to infinity, and assume for simplicity uniformly spaced basis functions. To make sure that the integral converges, we set variance of the prior over the weights to $\sigma_\alpha^2 = s/\Delta m$, where Δm is the density of basis functions in the input space. The covariance function is given by:

$$\begin{aligned} k(x_i, x_j) &= s \int_{c_{min}}^{c_{max}} \phi_c(x_i)\, \phi_c(x_j)\, \mathrm{d}c \ , \\ &= s \int_{c_{min}}^{c_{max}} \exp\left[-\frac{(x_i - c)^2}{2\lambda^2}\right] \exp\left[-\frac{(x_j - c)^2}{2\lambda^2}\right] \mathrm{d}c \ . \end{aligned} \tag{16}$$

Letting the limits of the integral go to infinity, we obtain the integral of the product of two Gaussians (but for a normalization factor), and we can use the algebra from Sect. A.2 to obtain:

$$k(x_i, x_j) = s\sqrt{\pi\lambda^2}\, \exp\left[-\frac{(x_i - x_j)^2}{4\lambda^2}\right] \ , \tag{17}$$

which is the squared exponential covariance function that we presented in (1). We now see that a GP with this particular covariance function is equivalent to a linear model with infinitely many squared exponential basis functions.

In the following we will show that for any valid covariance function, a GP has an equivalent linear model. The equivalent linear model will have infinitely many weights if the GP has a covariance function that has no finite expansion.

3.2 From GPs to Linear Models

We have just seen how to go from any linear model, finite or infinite, to an equivalent GP. We will now see how to go the opposite way, from an arbitrary GP to an equivalent linear model, which will in general be infinite and will be finite only for particular choices of the covariance function.

We start by building a linear model where all the function values considered (training *and* test inputs) are equal to a linear combination of the rows of the corresponding covariance matrix of the GP we wish to approximate, computed with the corresponding covariance function $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. As in Sect. 2, the covariance function is parametrized by the hyperparameters θ. A Gaussian prior distribution is defined on the model weights, with zero mean and covariance equal to the inverse of the covariance matrix:

$$\begin{bmatrix}\mathbf{f}\\ f_*\end{bmatrix} = \begin{bmatrix}K & \mathbf{k}_*\\ \mathbf{k}_*^\top & k_{**}\end{bmatrix} \cdot \begin{bmatrix}\boldsymbol{\alpha}\\ \alpha_*\end{bmatrix} \quad , \qquad p\left(\begin{bmatrix}\boldsymbol{\alpha}\\ \alpha_*\end{bmatrix}\middle| \boldsymbol{x}_*, X, \theta\right) \sim \mathcal{N}\left(0, \begin{bmatrix}K & \mathbf{k}_*\\ \mathbf{k}_*^\top & k_{**}\end{bmatrix}^{-1}\right) \quad . \tag{18}$$

To compute the corresponding prior over function values we need to integrate out the weights $[\boldsymbol{\alpha}, \alpha_*]^\top$ from the left expression in (18) by averaging over the prior (right expression in (18)):

$$\begin{aligned} p\left(\begin{bmatrix}\mathbf{f}\\ f_*\end{bmatrix}\middle| \boldsymbol{x}_*, X, \theta\right) &= \int \delta\left(\begin{bmatrix}\mathbf{f}\\ f_*\end{bmatrix} - \begin{bmatrix}K & \mathbf{k}_*\\ \mathbf{k}_*^\top & k_{**}\end{bmatrix} \cdot \begin{bmatrix}\boldsymbol{\alpha}\\ \alpha_*\end{bmatrix}\right) p\left(\begin{bmatrix}\boldsymbol{\alpha}\\ \alpha_*\end{bmatrix}\middle| \boldsymbol{x}_*, X, \theta\right) \mathrm{d}\boldsymbol{\alpha} \\ &\sim \mathcal{N}\left(0, \begin{bmatrix}K & \mathbf{k}_*\\ \mathbf{k}_*^\top & k_{**}\end{bmatrix}\right) \quad , \end{aligned} \tag{19}$$

and we recover exactly the same prior over function values as for the Gaussian Process, see (3).

Notice that for the linear model to correspond to the full GP two requirements need to be fulfilled:

1. There must be a weight associated to each training input.
2. There must be a weight associated to each possible test input.

Since there are as many weights as input instances, we consider that there is an infinite number of weights of which we only use as many as needed and qualify such a linear model of *infinite*.

Of course, for covariance functions that have a finite expansion in terms of m basis functions, the rank of the covariance matrix will never be greater than m and the equivalent linear model can be readily seen to be finite, with m basis functions. A trivial example is the case where the covariance function is built from a finite linear model with Gaussian priors on the weights. The linear model equivalent to a GP is only infinite if the covariance function of the GP has no finite expansion. In that case, independently of the number of training and test cases considered, the covariance matrix of the prior (independently of its size) will always have full rank.[7]

[7] The covariance matrix can always be made rank deficient by replicating a function value in the joint prior, but we do not see any reason to do this in practice.

It becomes evident how one should deal with GPs that have an equivalent finite linear model. If there are more training cases than basis functions, $n > m$, then the finite linear model should be used. In the case where there are less training cases than basis functions, $m > n$, it is computationally more interesting to use the GP.

One strong motivation for the use of Gaussian Processes is the freedom to directly specify the covariance function. In practice, common choices of GP priors imply covariance functions that do not have a finite expansion. For large datasets, this motivates the equivalent infinite linear model by a finite or sparse one. The approximated GP is called Reduced Rank GP since its covariance matrix has a maximum rank equal to the number of weights in the finite linear model.

We will see later in Sect. 4 that the finite linear approximation is built by relaxing the requirement of a weight being associated to each training input, resulting in training inputs with no associated weight. This relaxation should only be done at training time. In the next Section we show the importance of maintaining the requirement of having a weight associated to each test input.

3.3 "Can I Skip α_*?" or Degenerate Gaussian Processes

One may think that having just "as many weights as training cases" with no additional weight α_* associated to each test case gives the same prior as a full GP. It does only for the function evaluated at the training inputs, but it does not anymore for any additional function value considered. Indeed, if we posed $\mathbf{f} = K\,\boldsymbol{\alpha}$ with a prior over the weights given by $p(\boldsymbol{\alpha}|X,\theta) \sim \mathcal{N}(0, K^{-1})$, we would obtain that the corresponding prior over the training function values is $p(\mathbf{f}|X,\theta,\sigma^2) \sim \mathcal{N}(0, K)$. It is true that the linear model would be equivalent to the GP, but only when the function values considered are in $\mathbf{f}$. Without addition of α_*, the linear model and prior over function values are respectively given by:

$$\begin{bmatrix}\mathbf{f}\\ f_*\end{bmatrix} = \begin{bmatrix}K\\ \mathbf{k}_*^\top\end{bmatrix}\cdot\boldsymbol{\alpha}\ , \qquad p\left(\begin{bmatrix}\mathbf{f}\\ f_*\end{bmatrix}\middle|\,\boldsymbol{x}_*, X, \theta\right) \sim \mathcal{N}\left(0, \begin{bmatrix}K & \mathbf{k}_*\\ \mathbf{k}_*^\top & \mathbf{k}_*^\top K^{-1}\mathbf{k}_*\end{bmatrix}\right)\ . \tag{20}$$

The prior over the new function values f_* differs now from that of the full GP. Notice that the *prior* variance of f_* *depends* on the training inputs: for the common choice of an RBF-type covariance function, if $\boldsymbol{x}_*$ is far from the training inputs, then there is *a priori* no signal, that is f_* is zero without uncertainty! Furthermore, the distribution of $\mathbf{f}_*$ conditioned on the training function outputs, which for the full GP is given by (4), has now become:

$$p(f_*|\mathbf{f}, \boldsymbol{x}_*, X, \theta) \sim \mathcal{N}\left(\mathbf{k}_*^\top K^{-1}\mathbf{f}, 0\right)\ . \tag{21}$$

Given $\mathbf{f}$, any additional function value f_* is not a random variable anymore, since its conditional distribution has zero variance: f_* is fully determined by $\mathbf{f}$.

If $\boldsymbol{\alpha}$ has a fixed finite size, the prior over functions implied by the linear model ceases to correspond to the GP prior. The joint prior over sets of function values is still Gaussian, which raises the question "is this still a GP?". We choose to call such a *degenerate* process a "degenerate Gaussian Process".

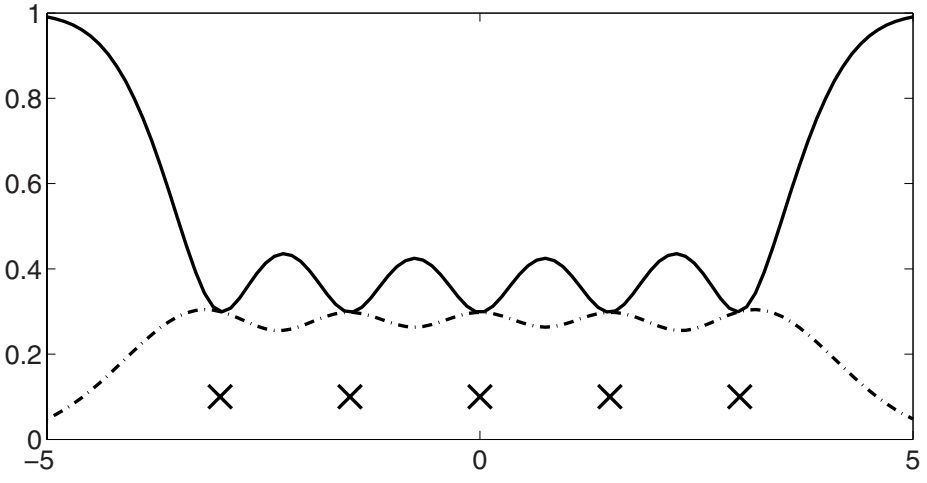

Fig. 1. Predictive standard deviation for a full GP (solid line) and for a degenerate GP (slash-dotted line). The hyperparameters θ_i are all set to 1. The crosses indicate the horizontal location of the 5 training inputs.

Degenerate GPs produce a predictive distribution that has maximal variability around the training inputs, while the predictive variance fades to the noise level as one moves away from them. We illustrate this effect on Fig. 1. We plot the predictive standard deviation of a full GP and its degenerate counterpart for various test points. The training set consists of 5 points: both models have thus 5 weights associated to the training set. The full GP has an additional weight, associated to each test point one at a time. Though it might be a reasonable prior in particular contexts, we believe that it is in general *inappropriate* to have smaller predictive variance far away from the observed data. We believe that appropriate priors are those under which the predictive variance is reduced when the test inputs approach training inputs. In the remaining of the paper we will consider that appropriate priors are desirable, and qualify the prior corresponding to a degenerate GP of inappropriate.

4 Finite Linear Approximations

As we have discussed in Sect. 3.1, a weight must be associated to each test case to avoid inappropriate priors that produce inappropriate predictive error bars. However, the requirement of each training case having a weight associated to it can be relaxed. For computational reasons it might be interesting to approximate, *at training time*, a GP by a finite linear model with less weights than training cases. The model and the prior on the weights are respectively given by:

$$\mathbf{f} = K_{nm}\,\boldsymbol{\alpha}_m \ , \qquad p(\boldsymbol{\alpha}_m|X,\theta) \sim \mathcal{N}(0, K_{mm}^{-1}) \ , \tag{22}$$

The subscripts m and n are used to indicate the dimensions: α_m is of size $n \times 1$ and K_{mn} of size $m \times n$; in the following we will omit these subscripts where unnecessary or cumbersome. Sparseness arises when $m < n$: the induced prior over training function values is $p(\mathbf{f}|X, \theta) \sim \mathcal{N}\left(0, K_{nm}\, K_{mm}^{-1}\, K_{nm}^{\top}\right)$, and rank of the covariance matrix is at most m. We call such an approximation a Reduced Rank Gaussian Process (RRGP).

The m inputs associated to the weights in $\boldsymbol{\alpha}_m$ do not need to correspond to training inputs. They can indeed be any set of arbitrary points in input space. We will call such points *support inputs* (in recognition to the large amount of work on sparse models done by the Support Vector Machines community). In this paper we will adopt the common restriction of selecting the support set from the training inputs. We discuss ways of selecting the support points in Sect. 4.4.

Learning an RRGP consists both in learning the hyperparameters of the covariance function and in selecting the support set. In practice however, it is hard to do both simultaneously. Besides the technical difficulties of the optimization process (observed for example by Csató (2002)), there is the fundamental issue of having an excessive amount of flexibility that may lead to overfitting (observed for example by Rasmussen (2002) and Seeger et al. (2003)). Smola and Bartlett (2001) address the issue of selecting the support set (Sect. 4.5), assuming that the covariance hyperparameters are given. However, we show that they do this in a way that does not guarantee generalization and we propose an alternative theoretically more sound approach in Sect. 4.4. In the next Section we show how to learn the hyperparameters of the covariance function for the RRGP for a fixed support set. We also show how to make predictions under a degenerate GP, that is, without an additional weight for the new test case, and with the inclusion of a new weight that ensures appropriate predictive variances.

4.1 Learning a Reduced Rank Gaussian Process

The likelihood of the weights is Gaussian in $\mathbf{y}$ and is a linear combination of α_m, given by $p(\mathbf{y}|X, \theta, \boldsymbol{\alpha}_m, \sigma^2) \sim \mathcal{N}(K_{nm}\, \boldsymbol{\alpha}_m, \sigma^2\, I)$, where σ^2 is again the white noise variance. The marginal likelihood of the hyperparameters of the full GP is given by (7). For the sparse finite linear approximation, the marginal likelihood is obtained by averaging the weights out of the likelihood over their prior:

$$\begin{aligned} p(\mathbf{y}|X, \theta, \sigma^2) &= \int p(\mathbf{y}|X, \theta, \boldsymbol{\alpha}_m, \sigma^2)\, p(\boldsymbol{\alpha}_m|X, \theta)\, \mathrm{d}\boldsymbol{\alpha}_m \\ &\sim \mathcal{N}\left(0, \sigma^2\, I + K_{nm}\, K_{mm}^{-1} K_{nm}^{\top}\right) \quad . \end{aligned} \tag{23}$$

As expected, for the case where the support set comprises all training inputs and $m = n$, we recover the full Gaussian Process.

Let us define $\tilde{Q} \equiv \left[\sigma^2\, I + K_{nm}\, K_{mm}^{-1} K_{nm}^{\top}\right]$, the covariance of the RRGP evidence. Maximum likelihood learning of the hyperparameters can be achieved by minimizing the negative log evidence. The cost function and its derivatives are given by (8) where Q is replaced by $\tilde{Q}$. Since the simple linear algebra involved

can be tedious, we give here the explicit expression of the different terms. For the terms involving $\log|\tilde{Q}|$ we have:

$$\begin{aligned}
\log|\tilde{Q}| &= (n-m)\,log(\sigma^2) + \log\left|K_{nm}^\top K_{nm} + \sigma^2 K_{mm}\right| \ , \\
\frac{\partial \log|\tilde{Q}|}{\partial \theta_i} &= \mathrm{Tr}\left[\tilde{Q}^{-1}\frac{\partial \tilde{Q}}{\partial \theta_i}\right] = 2\,\mathrm{Tr}\left[\frac{\partial K_{nm}}{\partial \theta_i} Z^\top\right] - \mathrm{Tr}\left[K_{mm}^{-1} K_{nm}^\top Z \frac{\partial K_{nm}}{\partial \theta_i}\right] \ , \\
\frac{\partial \log|\tilde{Q}|}{\partial \sigma^2} &= \frac{n-m}{\sigma^2} + \mathrm{Tr}\left[Z_{mm}\right] \ ,
\end{aligned} \tag{24}$$

where we have introduced $Z \equiv K_{nm}\left[K_{nm}^\top K_{nm} + \sigma^2 K_{mm}\right]^{-1}$. For the terms involving $\tilde{Q}^{-1}$ we have:

$$\begin{aligned}
\mathbf{y}^\top \tilde{Q}^{-1}\mathbf{y} &= \left(\mathbf{y}^\top\mathbf{y} - \mathbf{y}^\top Z K_{nm}^\top \mathbf{y}\right)/\sigma^2 \ , \\
\frac{\partial \mathbf{y}^\top \tilde{Q}^{-1}\mathbf{y}}{\partial \theta_i} &= \mathbf{y}^\top Z \frac{\partial K_{mm}}{\partial \theta_i} Z^\top \mathbf{y} - 2\,\mathbf{y}^\top\left(I - Z K_{nm}^\top\right)\frac{\partial K_{nm}}{\partial \theta_i} Z^\top \mathbf{y}/\sigma^2 \ , \\
\frac{\partial \mathbf{y}^\top \tilde{Q}^{-1}\mathbf{y}}{\partial \sigma^2} &= -\mathbf{y}^\top\mathbf{y}/\sigma^4 + \mathbf{y}^\top Z K_{nm}^\top \mathbf{y}/\sigma^4 + \mathbf{y}^\top Z K_{mm} Z^\top \mathbf{y}/\sigma^2 \ .
\end{aligned} \tag{25}$$

The hyperparameters and the output noise variance can be learnt by using the expressions we have given for the negative log marginal likelihood and its derivatives in conjunction with some gradient descent algorithm. The computational complexity of evaluating the evidence and its derivatives is $\mathcal{O}(nm^2 + nDm)$, which is to be compared with the corresponding cost of $\mathcal{O}(n^3)$ for the full GP model.

4.2 Making Predictions Without $\boldsymbol{\alpha}_*$

The posterior over the weights associated to the training function values is $p(\boldsymbol{\alpha}_m|\mathbf{y}, K_{mn}, \sigma^2) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with:

$$\boldsymbol{\mu} = \sigma^{-2}\boldsymbol{\Sigma}\, K_{mn}^\top \mathbf{y} \ , \qquad \boldsymbol{\Sigma} = \left[\sigma^{-2} K_{mn}^\top K_{mn} + K_{mm}\right]^{-1} \ . \tag{26}$$

At this point one can choose to make predictions right now, based on the posterior of $\boldsymbol{\alpha}_m$ and without adding an additional weight α_* associated to the new test point $\boldsymbol{x}_*$. As discussed in Sect. 3.3, this would correspond to a degenerate GP, leading to inappropriate predictive variance. The predictive mean on the other hand can still be a reasonable approximation to that of the GP: Smola and Bartlett (2001) approximate the predictive mean exactly in this way. The expressions for the predictive mean and variance, when not including α_*, are respectively given by:

$$m(\boldsymbol{x}_*) = \mathbf{k}(\boldsymbol{x}_*)^\top \boldsymbol{\mu}, \qquad v(\boldsymbol{x}_*) = \sigma^2 + \mathbf{k}(\boldsymbol{x}_*)^\top \boldsymbol{\Sigma}\, \mathbf{k}(\boldsymbol{x}_*). \tag{27}$$

$\mathbf{k}(\boldsymbol{x}_*)$ denotes the $m \times 1$ vector $[K(\boldsymbol{x}_*, \boldsymbol{x}_1), \ldots, K(\boldsymbol{x}_*, \boldsymbol{x}_m)]^\top$ of covariances between $\boldsymbol{x}_*$ and at the m support inputs (as opposed to $\mathbf{k}_*$ which is the $n \times 1$

vector of covariances between $\boldsymbol{x}_*$ and at the n training inputs). Note that if no sparseness is enforced, $(m = n)$, then $\boldsymbol{\mu} = (K_{nn} + \sigma^2 I)^{-1}\mathbf{y}$ and the predictive mean $m(\boldsymbol{x}_*)$ becomes identical to that of the full GP. Also, note that for decaying covariance functions,[8] if $\boldsymbol{x}_*$ is far away from the selected training inputs, the predictive variance collapses to the output noise level, which we have defined as an inappropriate prior.

The computational cost of predicting without α_* is an initial $\mathcal{O}(nm^2)$ to compute $\boldsymbol{\Sigma}$, and then an additional $\mathcal{O}(m)$ for the predictive mean and $\mathcal{O}(m^2)$ for the predictive variance per test case.

4.3 Making Predictions with $\boldsymbol{\alpha_*}$

To obtain a better approximation to the full GP, especially in terms of the predictive variance, we add an extra weight α_* to the model for each test input $\boldsymbol{x}_*$. Unless we are interested in the predictive covariance for a set of test inputs, it is enough to add one single α_* at a time. The total number of weights is therefore only augmented by one for any test case.

For a new test point, the mean and covariance matrix of the new posterior over the augmented weights vector are given by:

$$\begin{aligned} \boldsymbol{\mu}_* &= \sigma^{-2}\boldsymbol{\Sigma}_* \begin{bmatrix} K_{mn}^\top \\ \mathbf{k}_*^\top \end{bmatrix} \mathbf{y} \ , \\ \boldsymbol{\Sigma}_* &= \begin{bmatrix} \boldsymbol{\Sigma}^{-1} & \mathbf{k}(\boldsymbol{x}_*) + \sigma^{-2} K_{nm}^\top \mathbf{k}_* \\ \mathbf{k}(\boldsymbol{x}_*)^\top + \sigma^{-2}\mathbf{k}_*^\top K_{nm} & k_{**} + \sigma^{-2}\mathbf{k}_*^\top \mathbf{k}_* \end{bmatrix}^{-1} . \end{aligned} \tag{28}$$

and the computational cost of updating the posterior and computing the predictive mean and variance is $\mathcal{O}(nm)$ for each test point. The most expensive operation is computing $K_{nm}^\top \mathbf{k}_*$ with $\mathcal{O}(nm)$ operations. Once this is done and given that we have previously computed $\boldsymbol{\Sigma}$, computing $\boldsymbol{\Sigma}_*$ can be efficiently done using inversion by partitioning in $\mathcal{O}(m^2)$ (see Sect. A.1 for the details). The predictive mean and variance can be computed by plugging the updated posterior parameters (28) into (27), or alternatively by building the updated joint prior over the training and new test function values. We describe in detail the algebra involved in the second option in App. A.5. The predictive mean and variance when including α_* are respectively given by:

$$\begin{aligned} m_*(\boldsymbol{x}_*) &= \mathbf{k}_*^\top \left[K_{nm}\, K_{mm}^{-1}\, K_{nm}^\top + \sigma^2 I + \mathbf{v}_*\mathbf{v}_*^\top / c_*\right]^{-1} \mathbf{y} \ , \\ v_*(\boldsymbol{x}_*) &= \sigma^2 + k_{**} + \mathbf{k}_*^\top \left[K_{nm}\, K_{mm}^{-1}\, K_{nm}^\top + \sigma^2 I + \mathbf{v}_*\mathbf{v}_*^\top / c_*\right]^{-1} \mathbf{k}_* \ . \end{aligned} \tag{29}$$

where $\mathbf{v}_* \equiv \mathbf{k}_* - K_{nm}\, K_{mm}^{-1}\, \mathbf{k}(\boldsymbol{x}_*)$ is the difference between the actual and the approximated covariance of f_* and $\mathbf{f}$, and $c_* \equiv k_{**} - \mathbf{k}(\boldsymbol{x}_*)^\top K_{mm}^{-1}\, \mathbf{k}(\boldsymbol{x}_*)$ is the predictive variance at $\boldsymbol{x}_*$ of a full GP with the support inputs as training inputs.

[8] Covariance functions whose value decays with the distance between the two arguments. One example is the squared exponential covariance function described in Sect. 2. Decaying covariance functions are very commonly encountered in practice.

4.4 Selecting the Support Points

One way of addressing the problem of selecting the m support inputs is to select them from among the n training inputs. The number of possible sets of support inputs is combinatorial, C_n^m.[9] Since we will typically be interested in support sets much smaller than the training sets ($m < n$), this implies that the number of possible support sets is roughly exponential in m. Ideally one would like to evaluate the evidence for the finite linear model approximation (23), for each possible support input set, and then select the set that yields a higher evidence. In most cases however, this is impractical due to computational limitations. One suboptimal solution is to opt for a greedy method: starting with an empty subset, one includes the input that results in a maximal increase in evidence. The greedy method exploits the fact that the evidence can be computed efficiently when a case is added (or deleted) to the support set.

Suppose that a candidate input $\boldsymbol{x}_i$ from the training set is considered for inclusion in the support set. The new marginal likelihood is given by:

$$\mathcal{L}_i = \frac{1}{2}\log|\tilde{Q}_i| + \frac{1}{2}\,\mathbf{y}^\top \tilde{Q}_i^{-1}\,\mathbf{y} \;\;, \qquad \tilde{Q}_i \equiv \sigma^2\, I + K_{n\tilde{m}}\, K_{\tilde{m}\tilde{m}}^{-1}\, K_{n\tilde{m}}^\top \;\;, \tag{30}$$

where $\tilde{m}$ is the set of $m+1$ elements containing the m elements in the current support set plus the new case $\boldsymbol{x}_i$. $\tilde{Q}_i$ is the updated covariance of the evidence of the RRGP augmented with $\boldsymbol{x}_i$. Let us deal separately with the two terms in the evidence. The matrix inversion lemma allows us to rewrite $\tilde{Q}_i$ as:

$$\tilde{Q}_i = \sigma^{-2}\, I - \sigma^{-4}\, K_{n\tilde{m}}\, \boldsymbol{\Sigma}_i\, K_{n\tilde{m}}^\top \;\;, \qquad \boldsymbol{\Sigma}_i = \left[K_{n\tilde{m}}^\top K_{n\tilde{m}}/\sigma^2 + K_{\tilde{m}\tilde{m}}\right]^{-1} \;\;, \tag{31}$$

where $\boldsymbol{\Sigma}_i$ is the covariance of the posterior over the weights augmented in α_i, the weight associated to $\boldsymbol{x}_i$. Notice that $\boldsymbol{\Sigma}_i$ is the same expression as $\boldsymbol{\Sigma}_*$ in (28) if one replaces the index $*$ by i. In both cases we augment the posterior in the same way. Computing $\boldsymbol{\Sigma}_i$ from $\boldsymbol{\Sigma}$ costs therefore only $\mathcal{O}(nm)$.

The term of $\mathcal{L}$ quadratic in $\mathbf{y}$ can be rewritten as:

$$\mathcal{Q}_i = \frac{1}{2\sigma^2}\,\mathbf{y}^\top\mathbf{y} - \frac{1}{2\sigma^4}\,\mathbf{y}^\top K_{n\tilde{m}}\,\boldsymbol{\Sigma}_i\,K_{n\tilde{m}}^\top\mathbf{y} \;\;, \tag{32}$$

and can be computed efficiently in $\mathcal{O}(nm)$ if $\boldsymbol{\Sigma}$ and $K_{nm}^\top\mathbf{y}$ are known. In Sect. A.3 we provide the expressions necessary for computing $\mathcal{Q}_i$ incrementally in a robust manner from the Cholesky decomposition of $\boldsymbol{\Sigma}$. In Sect. 4.5 we describe Smola and Bartlett's Sparse Greedy Gaussian Process (SGGP) Regression which uses $\mathcal{Q}_i$ solely as objective function for selecting the support set in a greedy manner.

The term of $\mathcal{L}$ that depends on $\log|\tilde{Q}_i|$ can be expressed as:

$$\mathcal{G}_i = \frac{1}{2}\,\left[\log|\boldsymbol{\Sigma}_i| - \log|K_{\tilde{m}\tilde{m}}| + n\log\sigma^2\right] \;\;, \tag{33}$$

[9] C_n^m is "n choose m": the number of combinations of m elements out of n without replacement and where the order does not matter.

and computed at a cost of $\mathcal{O}(nm)$ (the cost of computing $K_{nm}^\top \mathbf{k}_i$). The algebra in Sect. A.3 can be used to update the determinants from the incremental Cholesky decompositions at no additional cost.

The overall cost of evaluating the evidence for each candidate point for the support set is $\mathcal{O}(nm)$. In practice, we may not want to explore the whole training set in search for the best candidate, since this would be too costly. We may restrict ourselves to exploring some reduced random subset.

4.5 Sparse Greedy Gaussian Process Regression

Smola and Bartlett (2001) and Schölkopf and Smola (2002) present a method to speed up the prediction stage for Gaussian processes. They propose a sparse greedy techniques to approximate the Maximum a Posteriori (MAP) predictions, treating separately the approximation of the predictive mean and that of the predictive variance.

For the predictive mean, Smola and Bartlett adopt a finite linear approximation of the form given by (22), where no extra weight α_* associated to the test input is added. Since this is a degenerate GP, it is understandable that they only use it for approximating the predictive mean: we now know that the predictive uncertainties of degenerate GPs are inappropriate.

The main contribution of their paper is to propose a method for selecting the m inputs in the support set from the n training inputs. Starting from a full posterior distribution (as many weights as training inputs), they aim at finding a sparse weight vector (with only m non-zero entries) with the requirement that *the posterior probability at the approximate solution be close to the maximum of the posterior probability* (quoted from (Schölkopf and Smola, 2002, Sect. 16.4.3)). Since the optimal strategy has again a prohibitive cost, they propose a greedy method where the objective function is the full posterior evaluated at the optimal weights vector with only m non-zeros weighs, those corresponding to the inputs in the support set.

The posterior on $\boldsymbol{\alpha}_n$ (full posterior) is given by (26), where $m = n$, i.e. matrix K_{nm} is replaced by the full $n \times n$ matrix K. The objective function used in (Smola and Bartlett, 2001; Schölkopf and Smola, 2002) is the part of the negative log posterior that depends on $\boldsymbol{\alpha}_n$, which is the following quadratic form:

$$-\frac{1}{2}\mathbf{y}^\top K_{nm}\,\boldsymbol{\alpha}_m + \frac{1}{2}\,\boldsymbol{\alpha}_m^\top \left[K_{nm}^\top K_{nm} + \sigma^2\, K_{mm}\right]\,\boldsymbol{\alpha}_m \ , \tag{34}$$

where as usual $\boldsymbol{\alpha}_m$ denotes the part of $\boldsymbol{\alpha}_n$ that hasn't been clamped to zero. Notice that it is essential for the objective function to be the full posterior evaluated at a sparse $\boldsymbol{\alpha}_n$, rather than the posterior on $\boldsymbol{\alpha}_m$ (given by (26) with indeed $m \neq n$). In the latter case, only the log determinant of the covariance would play a rôle in the posterior, since $\boldsymbol{\alpha}_m$ would have been made equal to the posterior mean, and we would have a completely different objective function from that in (Smola and Bartlett, 2001; Schölkopf and Smola, 2002).

Given two candidates to the support set, the one resulting in a support set for which the minimum of (34) is smaller is chosen. The minimum of (34) is given by:

$$-\frac{1}{2}\mathbf{y}^\top K_{nm} \left[K_{nm}^\top K_{nm} + \sigma^2 K_{mm}\right] K_{nm}^\top \mathbf{y} \ , \tag{35}$$

and it is in fact this quantity that is minimized with respect to the m elements in the support set in a greedy manner. The expression given in (35) with $m \neq n$ is in fact an upper bound to the same expression with $m = n$, which corresponds to selecting the whole training set as active set. Smola and Bartlett (2001); Schölkopf and Smola (2002) also provide a lower bound to the latter, which allows them to give a stop criterion to the greedy method based on the relative difference between upper and lower bound. The computational cost of evaluating the expression given in (35) for each candidate to the support set is $\mathcal{O}(nm)$, and use can be made of an incremental Cholesky factorization for numerical stability. The expressions in Sect. A.3 can be used. The computational cost is therefore the same for the SGGP method as for the greedy approach based on maximizing the evidence that we propose in Sect. 4.4.

Why Does It Work? One might at this point make abstraction from the algorithmic details, and ask oneself the fair question of why obtaining a sparse weight vector that evaluated under the posterior over the full weight vector yields a probability close to that of the non-sparse solution is a good approximation. Along the same lines, one may wonder whether the stopping criterion proposed relates in any way with good generalization.

It turns out that the method often works well in practice, in a very similar way as our proposed greedy criterion based on maximizing the evidence. One explanation for the SGGP method to select meaningful active sets is that it is in fact minimizing a part of the negative log evidence $\mathcal{L}_i$, given by (30). Indeed, notice that minimizing the objective function given by (35) is exactly equivalent to minimizing the part of the negative log evidence quadratic in $\mathbf{y}$ given by (32). So why would the method work if it only maximizes $\mathcal{Q}_i$ (32), the part of $\mathcal{L}_i$ that has to do with fitting the data, and ignores $\mathcal{G}_i$ (33), the part that enforces regularization? We believe that overfitting will seldom happen because m is typically significantly smaller than n, and that therefore we are selecting from a family of models that are all very simple. In other words, it is the sparsity itself that guarantees some amount of regularization, and therefore $\mathcal{G}_i$ can be often safely omitted from the negative log evidence. However, as we will see in what follows, the SGGP can fail and indeed overfit. The problem is that the SGGP fails to provide a valid stopping criterion for the process of adding elements to the support set.

But, How Much Sparsity? If sparsity seemingly ensures generalization, then it would also seem that a criterion is needed to know *the minimum sparsity level required*. In other words, we need to know how many inputs it is safe to include in the support set. (Smola and Bartlett, 2001; Schölkopf and Smola, 2002) use

a measure they call the "gap", which is the relative difference between the upper and lower bound on the negative log posterior. They choose an arbitrary threshold below which they consider that the approximate posterior has been maximized to a value close enough to the maximum of the full posterior. Once again we fail to see what such a criterion has to do with ensuring generalization, and we are not the only ones: Schwaighofer and Tresp (2003) report "we did not observe any correlation between the gap and the generalization performance in our experiments". It might be that for well chosen hyperparameters of the covariance, or for datasets that do not lend themselves to sparse approximations, keeping on adding cases to the support set cannot be harmful. Yet the SGGP does not allow learning the hyperparameters, and those must be somehow guessed (at least not in a direct way).

We provide a simple toy example (Fig. 2) in which the value of minimizing the negative log evidence becomes apparent. We generate 100 one-dimensional training inputs, equally spaced from -10 to 10. We generate the corresponding training inputs by applying the function $\sin(x)/x$ to the inputs, and adding Gaussian noise of variance 0.01. We generate the test data from 1000 test inputs equally spaced between -12 and 12. We use a squared exponential covariance function as given by (1), and we set the hyperparameters in the following way: the lengthscale is $\theta_1 = 1$, the prior standard deviation of the output signal is $\theta_2 = 1$ and the noise variance is $\sigma^2 = \theta_3 = 0.01$. Note that we provide the model with the *actual* variance of the noise. We apply the greedy strategy for selecting the support set by minimizing in one case the negative log evidence and in the other case the negative log posterior. Interesting things happen. We plot the test squared error as a function of m, the size of the support set for both greedy strategies. Both have a minimum for support sets of size around 8 to 10 elements, and increase again as for larger support sets. Additionally, we compute the negative log evidence as a function of m, and we see that it has a minimum around the region where the test error is minimal. This means that we can actually use the evidence to determine good levels of sparsity. We also plot the "gap" as a function of m, and indicate the location of the arbitrary threshold of 0.025 used by Smola and Bartlett (2001); Schölkopf and Smola (2002). The gap cannot provide us with useful information in any case, since it is *always* a monotonically decreasing function of m! The threshold is absolutely arbitrary, and has no relation to the expected generalization of the model.

Approximating Predictive Variances. Obtaining the predictive variance based on the posterior of the weights associated to the support set is a bad idea, since those will be smaller the further away the test input is from the inputs in the support set. An explicit approximation to the predictive variance of a full GP, given in (6) is proposed instead. For a given test input $\boldsymbol{x}_*$, Smola and Bartlett (2001); Schölkopf and Smola (2002) propose to approximate the term:

$$-\mathbf{k}_*^\top \left[K + \sigma^2 I\right]^{-1} \mathbf{k}_* \ , \tag{36}$$

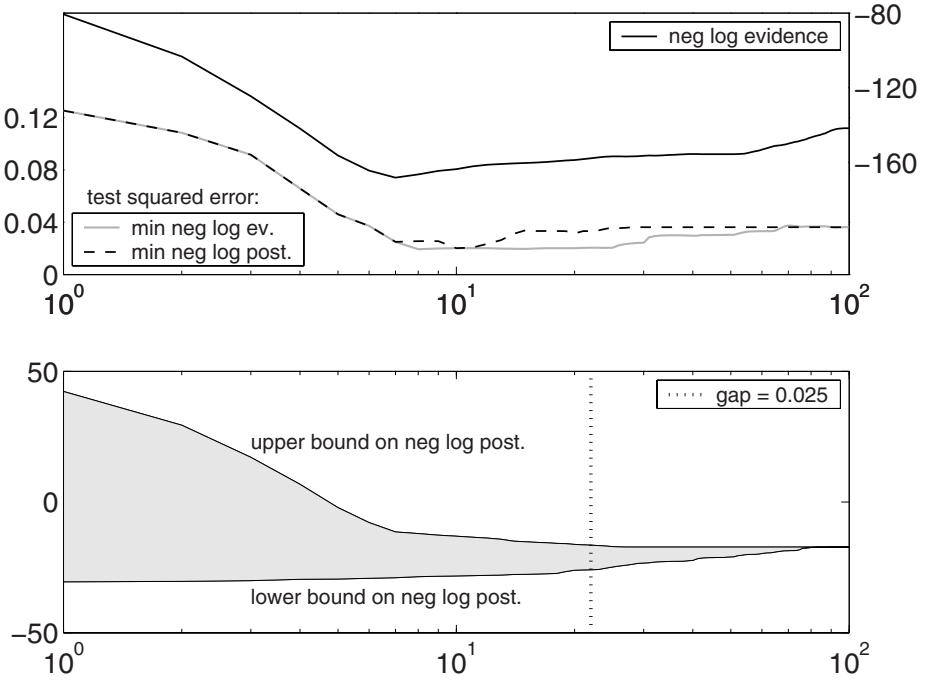

Fig. 2. Comparison between a sparse greedy approximation based on minimizing the negative log evidence, and one based on minimizing the negative log posterior. In both figures the horizontal axis indicates the size of the support set. *Top:* the solid black curve is the negative log evidence, with values given by the right vertical axis, the other two curves are the test squared error of the greedy methods based on minimizing the negative log evidence (solid gray) and the negative log posterior (dashed black), with values given on the left vertical axis. *Bottom:* for the SGGP approach the upper and lower bounds on the negative lower posterior are given, and the vertical dotted line shows the minimum size of the support set for which the "gap" is smaller that 0.025.

using the fact that it is the minimum (with respect to the $n \times 1$ weights vector $\boldsymbol{\beta}$, one weight associated to each training input) of the quadratic form:

$$-2\,\mathbf{k}_*^\top \boldsymbol{\beta} + \boldsymbol{\beta}^\top \left[K + \sigma^2\, I\right]\, \boldsymbol{\beta}\ . \tag{37}$$

They then go on to propose finding a sparse version $\boldsymbol{\beta}_m$ of $\boldsymbol{\beta}$ with only m non-zero elements.[10] The method is again a greedy incremental minimization of the

[10] This m does not have anything to do with the number of inputs in the support set of our previous discussion. It corresponds to a *new* support set, this time for approximating the predictive variance at $\boldsymbol{x}_*$. We insist on using the same symbol though because it still corresponds to a support set with $m < n$.

expression in (37). For a given choice of active elements (non-zero) in $\boldsymbol{\beta}$, the minimum of the objective function is given by:

$$-\mathbf{k}(\boldsymbol{x}_*)^\top \left[K_{mm} + \sigma^2 I\right]^{-1} \mathbf{k}(\boldsymbol{x}_*) \ , \tag{38}$$

where here again $\mathbf{k}(\boldsymbol{x}_*)$ represents an $m \times 1$ vector containing the covariance function evaluated at $\boldsymbol{x}_*$ and at the m inputs in the support set. Again, the support set yielding a minimal value of the expression in (38) will be chosen. The expression in (38) is also an upper bound on the (36), which means that bad approximations only mean an overestimate of the predictive variance, which is less bad than an underestimate. For each candidate to the support set, (38) can be evaluated in $\mathcal{O}(m^2)$ (this cost includes updating $\left[K_{mm} + \sigma^2 I\right]^{-1}$). Luckily, in practice the typical size of the support sets for approximating predictive variances is around one order of magnitude smaller than the size of the support set for approximating predictive means. Smola and Bartlett (2001); Schölkopf and Smola (2002) also provide a lower bound to (36), which allows to use a similar stop criterion as in the approximation of the predictive means.

Limitations Though it does work in practice and for the datasets on which we have tried it, there is no fundamental guarantee that SGGP will always work, since it does not maximize the whole of the evidence: it ignores the term in $\log |\tilde{Q}|$.

The hyperparameters of the covariance function need to be known: they cannot be learned by maximizing the posterior, since this would lead to overfitting.

While for approximating the predictive means one needs to find a unique support set, a specific support set needs to be estimated for *each* different test input if one wants to obtain good approximations to the predictive variance. The computational cost becomes then $\mathcal{O}(knm^2)$ per training case, where k is the size of a reduced random search set (Smola and Bartlett (2001) suggest using $k = 59$).

5 Experiments

We use the KIN40K dataset (for more details see Rasmussen, 1996, Chap. 5). This dataset represents the forward dynamics of an 8 link all-revolve robot arm. The dataset contains 40000 examples, the input space is 8-dimensional, and the 1-dimensional output represents the distance of an end-point of the robot arm from a fixed point. The mapping to be learned is low noise and highly nonlinear. This is of importance, since it means that the predictions can be improved by training on more data, and sparse solutions do not arise trivially.

We divide the dataset into 10 disjoint subsets of 4000 elements, that we then further split into training and test sets of 2000 elements each. The size of the support set is set to 512 elements in all cases. For each method we perform then 10 experiments, and compute the following losses: the Mean Absolute Error (MAE), the Mean Squared Error (MSE) and the Negative Test

Log-density (NTL). We also compute the training negative log likelihood per training case. Averaged results over the 10 disjoint sub-datasets are shown in the upper part of Table 1. SGGP is the sparse support set selection method proposed by Smola and Bartlett (2001); to compute predictive uncertainties, we do not use the sparse greedy approximation they suggest, since it has a too high computational cost of $\mathcal{O}(knm^2)$ per test case, with $k = 59$ and $m \approx 250$ in our case to reach $gap < 0.025$. As an alternative, they suggest to use the predictive uncertainties given by a reduced GP trained only on the support set obtained for approximating the predictive mean; the computational cost is low, $\mathcal{O}(m^2)$ per test case, but the performance is too poor to be worth reporting (NTL of the order of 0.3). To compute predictive uncertainties with the SGGP method we use the expressions given by (27) and (29). SGEV is our alternative greedy support set selection method based on maximizing the evidence. The HPEV-rand method selects a support set at random and learns the covariance hyperparameters by maximizing the evidence of the approximate model, as described in Sect. 4.1. The HPEV-SGEV and HPEV-SGGP methods select the support set for fixed hyperparameters according to the SGEV and and SGGP methods respectively, and then for that selected support set learn the hyperparameters by using HPEV. This procedure is iterated 10 times for both algorithms, which is enough for the likelihood to apparently converge. For all algorithms we present the results for the naïve non-augmented degenerate prediction model, and for the augmented non-degenerate one.

The experimental results show that the performance is systematically superior when using the augmented non-degenerate RRGP with an additional weight α_*. This superiority is expressed in all three losses, mean absolute, mean squared and negative test predictive density (which takes into account the predictive uncertainties). We believe that the relevant loss is the last one, since it reflects the fundamental theoretical improvement of the non-degenerate RRGP. The fact that the losses related to the predictive mean are also better can be explained by the model being slightly more flexible. We performed paired t-tests that confirmed that under all losses and algorithms considered, the augmented RRGP is significantly superior than the non-augmented one, with p-values always smaller than 1%. We found that for the dataset considered SGGP, SGEV and HPEV-rand are not significantly different. It would then seem that learning the hyperparameters for a random support set, or learning the support set for (carefully selected) hyperparameters by maximizing the posterior or the evidence are methods with equivalent performance. We found that both for the augmented and the non-augmented case, HPEV-SGEV and HPEV-SGGP are significantly superior to the other three methods, under all losses, again with p-values below 1%. On the other hand, HPEV-SGEV and HPEV-SGGP are not significantly different from each other under any of the losses.

The lower part of Table 1 shows the results of an additional experiment we made, where we compare SGEV to HPEV-rand on a larger training set. We generate this time 10 disjoint test sets of 4000 cases, and 10 corresponding training sets of 36000 elements. The size of the support sets remains 512. We

		non-augmented			*augmented*		
method	tr. neg log lik	MAE	MSE	NTL	MAE	MSE	NTL
SGGP	–	0.0481	0.0048	−0.3525	0.0460	0.0045	−0.4613
SGEV	−1.1555	0.0484	0.0049	−0.3446	0.0463	0.0045	−0.4562
HPEV-rand	−1.0978	0.0503	0.0047	−0.3694	0.0486	0.0045	−0.4269
HPEV-SGEV	−1.3234	0.0425	0.0036	−0.4218	0.0404	0.0033	−0.5918
HPEV-SGGP	−1.3274	0.0425	0.0036	−0.4217	0.0405	0.0033	−0.5920
2000 *training* - 2000 *test*							
SGEV	−1.4932	0.0371	0.0028	−0.6223	0.0346	0.0024	−0.6672
HPEV-rand	−1.5378	0.0363	0.0026	−0.6417	0.0340	0.0023	−0.7004
36000 *training* - 4000 *test*							

Table 1. Comparison of different learning methods for RRGPs on the KIN40K dataset, for 2000 training and test cases (*upper subtable*) and for 36000 training and 4000 test cases (*lower subtable*). The support set size is set to 512 for all methods. For each method the training negative log marginal likelihood per case is given, together with the Mean Absolute Error (MAE), Mean Squared Error (MSE) and Negative Test Log-likelihood (NTL) losses. SGGP (Smola and Bartlett, 2001) and SGEV (our alternative to SGGP based on maximizing the evidence) are based on learning the support set for fixed hyperparameters. HPEV-random learns the hyperparameters for a random subset, and HPEV-SGEV and HPEV-SGGP are methods where SGEV and SGGP are respectively interleaved with HPEV, for 10 repetitions.

compute the same losses as earlier, and consider also the augmented and the non-augmented RRGPs for making predictions. Paired t-tests[11] confirm once again the superiority of the augmented model to the non-augmented one for both models and all losses, with p-values below 1%.

6 Discussion

We have proposed to augment RRGPs at test time, by adding an additional weight α_* associated to the new test input $\boldsymbol{x}_*$. The computational cost for the predictive mean increases to $\mathcal{O}(nm)$ per case, i.e. $\mathcal{O}(n)$ more expensive than the non-augmented case. It might seem surprising that this is more expensive than the $\mathcal{O}(n)$ cost per case of the full GP! Of course, the full GP has has an initial cost of $\mathcal{O}(n^2)$ provided that the covariance matrix has been inverted, which costs $\mathcal{O}(n^3)$. Computing predictive variances has an initial cost of $\mathcal{O}(nm^2)$ like for the non-augmented case, and then a cost per case of $\mathcal{O}(nm)$ which is more expensive than the $\mathcal{O}(m^2)$ for the non-augmented case, and below the $\mathcal{O}(n^2)$ of the full GP. It may be argued that the major improvement brought by augmenting the RRGP is in terms of the predictive variance, and that one might therefore

[11] Due to dependencies between the training sets, assumptions of independence needed for the t-test could be compromised, but this is probably not a major effect.

consider computing the predictive mean from the non-augmented model, and the predictive variance from the augmented. However, the experiments we have conducted show that the augmented RRGP is systematically superior to the non-augmented, for all losses and learning schemes considered. The mean predictions are also better, probably due to the gain in flexibility by having an additional basis function.

Which method should be used for computing predictive variances? We have shown that using the degenerate RRGP, (27), has a computational cost of $\mathcal{O}(m^2)$ per test case. Using the augmented non-degenerate RRGP is preferable though because it gives higher quality predictive uncertainties, but the cost augments to $\mathcal{O}(nm)$ per test case. Smola and Bartlett (2001) propose two possibilities. A cost efficient option, $\mathcal{O}(m^2)$ per test case, is to base the calculation of all test predictive variances on the support set selected by approximating the posterior, which is in fact equivalent to computing predictive variances from a small full GP trained only on the support set. They show that the predictive variances obtained will always be an upper bound on the ones given by the full GP, and argue that inaccuracy (over estimation) is for that reason benign. We found experimentally that the errorbars from a small full GP trained only on the support set are very poor. The more accurate, yet more costly option consists is selecting a new support set for each test point. While they argue that the typical size of such test sets is very small (of the order of 25 for reasonable hyperparameters for the abalone dataset, but of the order of 250 for the KIN40K dataset), the computational cost per test case rises to $\mathcal{O}(knm^2)$. As we have explained, k is the size of a reduced random search set that can be fixed to 59 (see Smola and Bartlett, 2001). For their method to be computationally cheaper than our augmented RRGP, the support set that our method selects should contain more than $59 \times 25^2 = 36875$ elements. This is two orders of magnitude above the reasonable size of support sets that we would choose. In the experiments, we ended up computing the predictive variances for the SGGP from our expressions (27) and (29).

We found that none of the two possible "one-shot" approaches to training a RRGP is significantly superior to the other. In other words, selecting support sets at random and optimizing the hyperparameters does not provide significantly different performance than fixing the hyperparameters and selecting the support set in a supervised manner. Furthermore, on the dataset we did our experiments SGGP and SGEV did not prove to be significantly different either. We expect SGEV to perform better than SGGP on datasets where for the given hyperparameters the learning curve saturates, or even deteriorates as the support set is increased, as is the case in the example we give in Fig. 2. Interleaving support set selection and hyperparameter learning schemes proves on the other hand to be promising. The experiments on KIN40K show that this scheme gives much superior performance to the two isolated learning schemes.

It is interesting to note the relation between the RRGP and the Nyström approximation proposed by Williams and Seeger (2001). In that approach the predictive mean and variance are respectively given by:

$$
\begin{aligned}
m(\boldsymbol{x}_*) &= \mathbf{k}_*^\top \left[K_{nm}\, K_{mm}^{-1}\, K_{nm}^\top + \sigma^2\, I\right]^{-1} \mathbf{y} \ , \\
v(\boldsymbol{x}_*) &= \sigma^2 + k_{**} + \mathbf{k}_*^\top \left[K_{nm}\, K_{mm}^{-1}\, K_{nm}^\top + \sigma^2\, I\right]^{-1} \mathbf{k}_* \ .
\end{aligned}
\tag{39}
$$

These expressions are very similar to those obtained for the augmented RRGP, given by (29). However, the additional term in the approximate covariance for the augmented RRGP ensures that it is positive definite, see (Williams et al., 2002), and that therefore our approach does not suffer from negative predictive variances as is the case for the Nyström approximation for GPs.

Future work will involve the theoretical study of other sparse approximations to GPs that have been recently proposed, which we enumerate in Sect. 1, and the experimental comparison of these methods to those presented in this paper.

A Useful Algebra

A.1 Matrix Identities

The matrix inversion lemma, also known as the Woodbury, Sherman & Morrison formula states that:

$$
(Z + UWV^\top)^{-1} = Z^{-1} - Z^{-1}U(W^{-1} + V^\top Z^{-1}U)^{-1}V^\top Z^{-1}, \tag{A-40}
$$

assuming the relevant inverses all exist. Here Z is $n \times n$, W is $m \times m$ and U and V are both of size $n \times m$; consequently if Z^{-1} is known, and a low rank (ie. $m < n$) perturbation are made to Z as in left hand side of eq. (A-40), considerable speedup can be achieved. A similar equation exists for determinants:

$$
|Z + UWV^\top| = |Z|\ |W|\ |W^{-1} + V^\top Z^{-1}U| \ . \tag{A-41}
$$

Let the symmetric $n \times n$ matrix A and its inverse A^{-1} be partitioned into:

$$
A = \begin{pmatrix} P & Q \\ Q^T & S \end{pmatrix} \ , \qquad A^{-1} = \begin{pmatrix} \tilde{P} & \tilde{Q} \\ \tilde{Q}^T & \tilde{S} \end{pmatrix} \ , \tag{A-42}
$$

where P and $\tilde{P}$ are $n_1 \times n_1$ matrices and S and $\tilde{S}$ are $n_2 \times n_2$ matrices with $n = n_1 + n_2$. The submatrices in A^{-1} are given in Press et al. (1992, p. 77):

$$
\begin{aligned}
\tilde{P} &= P^{-1} + P^{-1}QM^{-1}Q^TP^{-1}, \\
\tilde{Q} &= -P^{-1}QM^{-1}, \qquad\qquad \text{where } M = S - Q^TP^{-1}Q \\
\tilde{S} &= M^{-1} \ .
\end{aligned}
\tag{A-43}
$$

There are also equivalent formulae

$$
\begin{aligned}
\tilde{P} &= N^{-1}, \\
\tilde{Q} &= -N^{-1}QS^{-1}, \qquad\qquad \text{where } N = P - QS^{-1}Q^T \\
\tilde{S} &= S^{-1} + S^{-1}Q^TN^{-1}QS^{-1} \ .
\end{aligned}
\tag{A-44}
$$

A.2 Product of Gaussians

When using linear models with Gaussian priors, the likelihood and the prior are both Gaussian. Their product is proportional to the posterior (also Gaussian), and their integral is equal to the marginal likelihood (or evidence). Consider the random vector $\boldsymbol{x}$ of size $n \times 1$ and the following product:

$$\mathcal{N}(\boldsymbol{x}|\mathbf{a}, A)\,\mathcal{N}(P\,\boldsymbol{x}|\mathbf{b}, B) = z_c\,\mathcal{N}(\boldsymbol{x}|\mathbf{c}, C) \ , \tag{A-45}$$

where $\mathcal{N}(\boldsymbol{x}|\mathbf{a}, A)$ denotes the probability of $\boldsymbol{x}$ under a Gaussian distribution centered on $\mathbf{a}$ (of size $n \times 1$) and with covariance matrix A (of size $n \times n$). P is a matrix of size $n \times m$ and vectors $\mathbf{b}$ and $\mathbf{c}$ are of size $m \times 1$, and matrices B and C of size $m \times m$. The product of two Gaussians is proportional to a new Gaussian with covariance and mean given by:

$$C = \left(A^{-1} + P\,B^{-1}P^\top\right)^{-1} \ , \qquad c = C\,\left(A^{-1}\mathbf{a} + P\,B\,\mathbf{b}\right) \ .$$

The normalizing constant z_c is gaussian in the means $\mathbf{a}$ and $\mathbf{b}$ of the two Gaussians that form the product on the right side of (A-45):

$$z_c = (2\,\pi)^{-\frac{m}{2}}|B + P^\top A^{-1}P| \times \exp\left(-\frac{1}{2}(\mathbf{b} - P\,\mathbf{a})^\top\left(B + P^\top A^{-1}P\right)^{-1}(\mathbf{b} - P\,\mathbf{a})\right) \ .$$

A.3 Incremental Cholesky Factorization for SGQM

Consider the quadratic form:

$$Q(\boldsymbol{\alpha}) = -\mathbf{v}^\top\boldsymbol{\alpha} + \frac{1}{2}\,\boldsymbol{\alpha}^\top A\,\boldsymbol{\alpha} \ , \tag{A-46}$$

where A is a symmetric positive definite matrix of size $n \times n$ and $\mathbf{v}$ is a vector of size $n \times 1$. Suppose we have already obtained the minimum and the minimizer of $Q(\boldsymbol{\alpha})$, given by:

$$Q_{opt} = -\frac{1}{2}\,\mathbf{v}^\top A^{-1}\,\mathbf{v} \ , \qquad \boldsymbol{\alpha}_{opt} = A^{-1}\,\mathbf{v} \ . \tag{A-47}$$

We now want to minimize an augmented quadratic form $Q_i(\boldsymbol{\alpha})$, where $\boldsymbol{\alpha}$ is now of size $n \times 1$ and A and $\mathbf{v}$ are replaced by A_i and $\mathbf{v}_i$ of size $n+1 \times n+1$ and $n+1 \times 1$ respectively, given by:

$$A_i = \begin{bmatrix} A & \mathbf{b}_i \\ \mathbf{b}_i^\top & c_i \end{bmatrix} \ , \qquad \mathbf{v}_i = \begin{bmatrix} \mathbf{v} \\ v_i \end{bmatrix} \ .$$

Assume that vector $\mathbf{b}_i$ of size $n \times 1$ and scalars c_i and v_i are somehow obtained. We want to exploit the incremental nature of A_i and $\mathbf{v}_i$ to reduce the number of operations necessary to minimize $Q_i(\boldsymbol{\alpha})$. One option would be to compute A_i^{-1} using inversion by partitioning, with cost $\mathcal{O}\left((n+1)^2\right)$ if A^{-1} is known.

For iterated incremental computations, using the Cholesky decomposition of A_i is numerically more stable. Knowing L, the Cholesky decomposition of A, the Cholesky decomposition L_i of A_i can be computed as:

$$L_i = \begin{bmatrix} L & 0 \\ \mathbf{z}_i^\top & d_i \end{bmatrix} \ , \qquad L\,\mathbf{z}_i = \mathbf{b}_i, \qquad d_i^2 = c_i - \mathbf{z}_i^\top \mathbf{z}_i \ . \tag{A-48}$$

The computational cost is $\mathcal{O}(n^2/2)$, corresponding to the computation of $\mathbf{z}_i$ by back-substitution. Q_i^{min} can be computed as:

$$Q_i^{min} = Q^{min} - \frac{1}{2}\, u_i^2 \ , \qquad u_i = \frac{1}{d_i}\,(v_i - \mathbf{z}_i^\top \mathbf{u}) \ , \qquad L\,\mathbf{u} = \mathbf{v} \ , \tag{A-49}$$

and the minimizer $\boldsymbol{\alpha}^{opt}$ is given by:

$$L^\top \boldsymbol{\alpha}^{opt} = \mathbf{u}_i \ , \qquad \mathbf{u}_i = \begin{bmatrix} \mathbf{u} \\ u_i \end{bmatrix} \ . \tag{A-50}$$

Notice that knowing $\mathbf{u}$ from the previous iteration, computing Q_i^{min} has a cost of $\mathcal{O}(n)$. This is interesting if many different i's need to be explored, for which only the minimum of Q_i is of interest, and not the minimizer. Once the optimal i has been found, computing the minimizer $\boldsymbol{\alpha}^{opt}$ requires a back-substitution, with a cost of $\mathcal{O}(n^2/2)$.

It is interesting to notice that as a result of computing L_i one obtains "for free" the determinant of A_i (an additional cost of $\mathcal{O}(m)$ to th e$\mathcal{O}(nm)$ cost of the incremental Cholesky). In Sect. A.4 we give a general expression of incremental determinants.

A.4 Incremental Determinant

Consider a square matrix A_i that has a row and a column more than square matrix A of size $n \times n$:

$$A_i = \begin{bmatrix} A & \mathbf{b}_i \\ \mathbf{c}_i^\top & d_i \end{bmatrix} \ . \tag{A-51}$$

The determinant of A_i is given by

$$|A_i| = |A| \cdot (d_i - \mathbf{b}_i^\top A^{-1} \mathbf{c}_i) \ . \tag{A-52}$$

In the interesting situation where A^{-1} is known, the new determinant is computed at a cost of $\mathcal{O}(m^2)$.

A.5 Derivation of (29)

We give here details of the needed algebra for computing the predictive distribution of the Reduced Rank Gaussian Process. Recall that at training time we use a finite linear model approximation, with less weights than training inputs.

Each weight has an associated support input possibly selected from the training inputs. The linear model and prior on the weights are:

$$\begin{bmatrix}\mathbf{f}\\ f_*\end{bmatrix} = \Phi_{nm}\cdot\begin{bmatrix}\boldsymbol{\alpha}\\ \alpha_*\end{bmatrix}\ , \qquad p\left(\begin{bmatrix}\boldsymbol{\alpha}\\ \alpha_*\end{bmatrix}\middle|\, \boldsymbol{x}_*, X, \theta\right) \sim \mathcal{N}\left(0, A^{-1}\right)\ .$$

where we have defined

$$\Phi_{nm} = \begin{bmatrix}K_{nm} & \mathbf{k}_*\\ \mathbf{k}(\boldsymbol{x}_*)^\top & k_{**}\end{bmatrix}\ , \qquad A = \begin{bmatrix}K_{mm} & \mathbf{k}(\boldsymbol{x}_*)\\ \mathbf{k}(\boldsymbol{x}_*)^\top & k_{**}\end{bmatrix}\ . \tag{A-53}$$

The induced prior over functions is Gaussian with mean zero and covariance matrix C:

$$p\left(\begin{bmatrix}\mathbf{f}\\ f_*\end{bmatrix}\middle|\, \boldsymbol{x}_*, X, \theta\right) \sim \mathcal{N}\left(0, C\right)\ , \qquad C = \Phi_{nm}\, A^{-1}\, \Phi_{nm}^\top\ . \tag{A-54}$$

We use inversion by partitioning to compute A^{-1}:

$$A^{-1} = \begin{bmatrix}K_{mm}^{-1} + K_{mm}^{-1}\mathbf{k}(\boldsymbol{x}_*)\,\mathbf{k}(\boldsymbol{x}_*)^\top K_{mm}^{-1} & -K_{mm}^{-1}\mathbf{k}(\boldsymbol{x}_*)/c_*\\ -\mathbf{k}(\boldsymbol{x}_*)^\top K_{mm}^{-1}/c_* & 1/c_*\end{bmatrix}\ ,$$
$$c_* = k_{**} - \mathbf{k}(\boldsymbol{x}_*)^\top K_{mm}^{-1}\mathbf{k}(\boldsymbol{x}_*)\ ,$$

which allows to obtain C:

$$C = \begin{bmatrix}C_{nn} & \mathbf{k}_*\\ \mathbf{k}_*^\top & k_{**}\end{bmatrix}\ , \qquad C_{nn} \equiv K_{nm}\, K_{mm}^{-1}\, K_{nm}^\top + \mathbf{v}_*\mathbf{v}_*^\top/c_*\ , \tag{A-55}$$

where $\mathbf{v}_* \equiv \mathbf{k}_* - K_{nm}\, K_{mm}^{-1}\, \mathbf{k}(\boldsymbol{x}_*)$. We can now compute the distribution of f_* conditioned $\mathbf{f}$:

$$p(f_*|\mathbf{f}, \boldsymbol{x}_*, X, \theta) \sim \mathcal{N}\left(\mathbf{k}_*^\top C_{nn}^{-1}\mathbf{f}, k_{**} - \mathbf{k}_*^\top C_{nn}^{-1}\mathbf{k}_*\right)\ . \tag{A-56}$$

The predictive distribution, obtained as in (5), is Gaussian with mean and variance given by (29). We repeat their expressions here for convenience:

$$m_*(\boldsymbol{x}_*) = \mathbf{k}_*^\top\left[K_{nm}\, K_{mm}^{-1}\, K_{nm}^\top + \sigma^2\, I + \mathbf{v}_*\mathbf{v}_*^\top/c_*\right]^{-1}\mathbf{y}\ ,$$
$$v_*(\boldsymbol{x}_*) = \sigma^2 + k_{**} + \mathbf{k}_*^\top\left[K_{nm}\, K_{mm}^{-1}\, K_{nm}^\top + \sigma^2\, I + \mathbf{v}_*\mathbf{v}_*^\top/c_*\right]^{-1}\mathbf{k}_*\ .$$

B Matlab Code for the RRGP

We believe that one very exciting part of looking at a new algorithm is "trying it out"! We would like the interested reader to be able to train our Reduced Rank Gaussian Process (RRGP) algorithm. Training consists in finding the value of the hyperparameters that minimizes the negative log evidence of the RRGP (we give it in Sect. 4.1). To do this we first need to be able to compute the negative log evidence and its derivatives with respect to the hyperparameters. Then we can plug this to a gradient descent algorithm to perform the actual learning.

We give a Matlab function, `rrgp_nle`, that computes the negative log evidence of the RRGP and its derivatives for the squared exponential covariance function (given in (1)). The hyperparameters of the squared exponential covariance function are all positive. To be able to use unconstrained optimization, we optimize with respect to the logarithm of the hyperparameters.

An auxiliary Matlab function `sq_dist` is needed to compute squared distances. Given to input matrices of sizes $d \times n$ and $d \times m$, the function returns the $n \times m$ matrix of squared distances between all pairs of columns from the inputs matrices. The authors would be happy to provide their own Matlab MEX implementation of this function upon request.

Inputs to the Function `rrgp_nle`:

- `X`: $D + 2 \times 1$ vector of log hyperparameters, $\mathtt{X} = [\log \theta_1, \ldots \log \theta_{D+1}, \log \sigma]^\top$, see (1)
- `input`: $n \times D$ matrix of training inputs
- `target`: $n \times 1$ matrix of training targets
- `m`: scalar, size of the support set

Outputs of the Function `rrgp_nle`:

- `f`: scalar, evaluation of the negative log evidence at `X`
- `f`: $D + 2 \times 1$ vector of derivatives of the negative log evidence evaluated at `X`

Matlab Code of the Function `rrgp_nle`:

```
function [f,df] = rrgp_nle(X,input,target,m)

% number of examples and dimension of input space
[n, D] = size(input);
input = input ./ repmat(exp(X(1:D))',n,1);

% write the noise-free covariance of size n x m
Knm = exp(2*X(D+1))*exp(-0.5*sq_dist(input',input(1:m,:)'));
% add little jitter to Kmm part
Knm(1:m,:) = Knm(1:m,:)+1e-8*eye(m);

Cnm = Knm/Knm(1:m,:);
Smm = Knm'*Cnm + exp(2*X(D+2))*eye(m);
Pnm = Cnm/Smm;
wm = Pnm'*target;

% compute function evaluation
invQt = (target-Pnm*(Knm'*target))/exp(2*X(D+2));
logdetQ = (n-m)*2*X(D+2) + sum(log(abs(diag(lu(Smm)))));
f = 0.5*logdetQ + 0.5*target'*invQt + 0.5*n*log(2*pi);
```

```
% compute derivatives
df = zeros(D+2,1);

for d=1:D
  Vnm = -sq_dist(input(:,d)',input(1:m,d)').*Knm;
  df(d) = (invQt'*Vnm)*wm - 0.5*wm'*Vnm(1:m,:)*wm+...
      -sum(sum(Vnm.*Pnm))+0.5*sum(sum((Cnm*Vnm(1:m,:)).*Pnm));
end
aux = sum(sum(Pnm.*Knm));
df(D+1) = -(invQt'*Knm)*wm+aux;
df(D+2) = (n-aux) - exp(2*X(D+2))*invQt'*invQt;
```

References

Cressie, N. A. C. (1993). *Statistics for Spatial Data*. John Wiley and Sons, Hoboken, New Jersey.

Csató, L. (2002). *Gaussian Processes – Iterative Sparse Approximation*. PhD thesis, Aston University, Birmingham, United Kingdom.

Csató, L. and Opper, M. (2002). Sparse online gaussian processes. *Neural Computation*, 14(3):641–669.

Gibbs, M. and MacKay, D. J. C. (1997). Efficient implementation of gaussian processes. Technical report, Cavendish Laboratory, Cambridge University, Cambridge, United Kingdom.

Lawrence, N., Seeger, M., and Herbrich, R. (2003). Fast sparse gaussian process methods: The informative vector machine. In Becker, S., Thrun, S., and Obermayer, K., editors, *Neural Information Processing Systems 15*, pages 609–616, Cambridge, Massachussetts. MIT Press.

MacKay, D. J. C. (1994). Bayesian non-linear modelling for the energy prediction competition. *ASHRAE Transactions*, 100(2):1053–1062.

Mackay, D. J. C. (1997). Gaussian Processes: A replacement for supervised Neural Networks? Technical report, Cavendish Laboratory, Cambridge University, Cambridge, United Kingdom. Lecture notes for a tutorial at NIPS 1997.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer, Heidelberg, Germany.

Press, W., Flannery, B., Teukolsky, S. A., and Vetterling, W. T. (1992). *Numerical Recipes in C*. Cambridge University Press, Cambridge, United Kingdom, second edition.

Rasmussen, C. E. (1996). *Evaluation of Gaussian Processes and Other Methods for Non-linear Regression*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Ontario.

Rasmussen, C. E. (2002). Reduced rank gaussian process learning. Unpublished Manuscript.

Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels*. MIT Press, Cambridge, Massachussetts.

Schwaighofer, A. and Tresp, V. (2003). Transductive and inductive methods for approximate gaussian process regression. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 953–960, Cambridge, Massachussetts. MIT Press.

Seeger, M. (2003). *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations*. PhD thesis, University of Edinburgh, Edinburgh, Scotland.

Seeger, M., Williams, C., and Lawrence, N. (2003). Fast forward selection to speed up sparse gaussian process regression. In Bishop, C. M. and Frey, B. J., editors, *Ninth International Workshop on Artificial Intelligence and Statistics*. Society for Artificial Intelligence and Statistics.

Smola, A. J. and Bartlett, P. L. (2001). Sparse greedy Gaussian process regression. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 619–625, Cambridge, Massachussetts. MIT Press.

Smola, A. J. and Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning. In Langley, P., editor, *International Conference on Machine Learning 17*, pages 911–918, San Francisco, California. Morgan Kaufmann Publishers.

Tipping, M. E. (2001). Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244.

Tresp, V. (2000). A bayesian committee machine. *Neural Computation*, 12(11):2719–2741.

Wahba, G., Lin, X., Gao, F., Xiang, D., Klein, R., and Klein, B. (1999). The bias-variance tradeoff and the randomized GACV. In Kerns, M. S., Solla, S. A., and Cohn, D. A., editors, *Advances in Neural Information Processing Systems 11*, pages 620–626, Cambridge, Massachussetts. MIT Press.

Williams, C. (1997a). Computation with infinite neural networks. In Mozer, M. C., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems 9*, pages 295–301, Cambridge, Massachussetts. MIT Press.

Williams, C. (1997b). Prediction with gaussian processes: From linear regression to linear prediction and beyond. Technical Report NCRG/97/012, Dept of Computer Science and Applied Mathematics, Aston University, Birmingham, United Kingdom.

Williams, C., Rasmussen, C. E., Schwaighofer, A., and Tresp, V. (2002). Observations of the nyström method for gaussiam process prediction. Technical report, University of Edinburgh, Edinburgh, Scotland.

Williams, C. and Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 682–688, Cambridge, Massachussetts. MIT Press.

Acknowledgements

The authors would like to thank Lehel Csató, Alex Zien and Olivier Chapelle for useful discussions.

This work was supported by the Multi-Agent Control Research Training Network - EC TMR grant HPRN-CT-1999-00107, and the German Research Council (DFG) through grant RA 1030/1.