Sparse Gaussian Process Regression via ℓ_1 Penalization

Feng Yan YAN12@PURDUE.EDU
Department of Computer Science, Purdue University, West Lafayette, IN 47907 USA

Yuan (Alan) Qi ALANQI@CS.PURDUE.EDU
Departments of Computer Science and Statistics, Purdue University, West Lafayette, IN 47907 USA

Abstract

To handle massive data, a variety of sparse Gaussian Process (GP) methods have been proposed to reduce the computational cost. Many of them essentially map the large dataset into a small set of basis points. A common approach to learn these basis points is evidence maximization. Nevertheless, evidence maximization may lead to overfitting and cause a high computational cost. In this paper, we propose a novel sparse GP regression approach, GPLasso, that explicitly represents the trade-off between its approximation quality and the model sparsity. GPLasso minimizes a ℓ_1-penalized KL divergence between the exact and sparse GP *posterior processes*. Optimizing this convex cost function leads to sparse GP parameters. Furthermore, we use incomplete Cholesky factorization to obtain low-rank matrix approximations to speed up the optimization procedure. Experimental results on synthetic and real data demonstrate that, compared with several state-of-the-art sparse GP methods and a direct low-rank matrix approximation method, GPLasso achieves a significantly improved trade-off between prediction accuracy and computational cost.

1. Introduction

Gaussian processes (GP) are powerful nonparametric Bayesian approach to model unknown functions. Exact learning with GP is, however, intractable for large datasets. For linear regression, training the exact GP model with N samples demands an $O(N^3)$ cost for inverting a $N \times N$ covariance matrix. Predicting the mean of a new instance requires an $O(N)$ cost.

In general, one can reduce the computational cost by using a sparse representation of the posterior distribution for the unknown function. This representation could be used to summarize training data for Bayesian learning, or it could be passed around as a message in order to do inference in a larger probabilistic model. One successful approach is to map the training data into a small set of basis points, and then compute the exact posterior that results from those points (Quiñonero Candela & Rasmussen, 2005). To learn these basis points, a sensible approach will be to treat them as hyperparameters and maximize the evidence over them (Snelson & Ghahramani, 2006). Nevertheless, maximized evidence can causes overfitting as a type-II maximum likelihood estimation.

In this paper we present a new framework, GPLasso, for sparse GP regression. Unlike the previous approaches, GPLasso explicitly represents the trade-off between the approximation quality of the sparse GP *posterior process* and the parameter sparsity, improving the balance between the computational cost and prediction accuracy. Specifically, GPLasso minimizes a *convex* cost function that combines the KL divergence between the exact and the approximate GP posterior processes and a ℓ_1 penalty over GP parameters. To minimize this cost function we present an efficient algorithm based on low-rank matrix approximations (Fine et al., 2001) and Least Angle Regression (Efron et al., 2002). In summary, the main contributions of this paper include the following:

- We propose GPLasso, a new framework for sparse GP regression, which minimizes the ℓ_1-penalized KL divergence in the *infinite* functional space between the exact and the approximate posterior processes.
- We combine the LAR algorithm, a powerful vari-

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

able selection method, with Incomplete Cholesky Factorization (ICF) to efficiently learn the sparse *nonparametric* Bayesian model. We also demonstrate that GPLasso, although based on low-rank matrix approximations, achieves much higher prediction accuracy than the direct application of the low-rank approximation to GP prediction.

- We provide an alternative view to GPLasso based on a generalization of the representor theorem: GPLasso defines a sparse kernel machine in a Reproducing kernel Hilbert Space (RKHS).

- Experimental results demonstrate that, compared with several state-of-the-art methods, GPLasso achieves a significantly improved trade-off between predictive performance and computational cost.

2. Gaussian Process regression

We denote N independent and identically distributed samples as $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}_N$, where $\mathbf{x}_i$ is a d dimensional input and y_i is a scalar output. We assume there is a latent function f that we are modeling and the noisy realization of f at $\mathbf{x}_i$ is y_i.

A Gaussian process places a prior distribution over the latent function f. Its projection at the samples $B = \{\mathbf{x}_i\}_N$ defines a joint Gaussian distribution:

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{m}^0, K)$$

where $m_i^0 = m^0(\mathbf{x}_i)$ is the mean function and $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$[1] is the covariance function, which encodes the prior notion of smoothness. Traditionally the mean function is set to zero and we follow this tradition in this paper.

For regression, we use a Gaussian likelihood function

$$p(y_i|f) = \mathcal{N}(y_i|f(\mathbf{x}_i), v_y) \tag{1}$$

where v_y is the observation noise.

Given the Gaussian process prior over f and the data likelihood, the posterior process is

$$p(f|\mathcal{D}) \propto GP(f|0, K) \prod_{i=1}^{N} p(y_i|f(\mathbf{x}_i)) \tag{2}$$

Since the Gaussian process is grounded on the N examples, they are called the basis points.

[1] We abuse the notation of K to represent both the kernel matrix and the kernel function.

The posterior process $p(f|\mathcal{D})$ has the mean function $m(\mathbf{x})$ and covariance function $V(\mathbf{x}, \mathbf{x}')$:

$$m(\mathbf{x}) = K(\mathbf{x}, B)\alpha \tag{3}$$

$$V(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') - K(\mathbf{x}, B)\beta K(B, \mathbf{x}') \tag{4}$$

where B represents all the input points of the training set, and

$$\beta = (K + v_y \mathbf{I})^{-1}, \qquad \alpha = \beta \mathbf{y} \tag{5}$$

For the predictive process, the mean function is the same as that of the posterior process and the covariance function is the covariance function of the posterior process plus the observation noise variance v_y.

To calculate the variable α, we need to invert the $N \times N$ matrix β, which takes $O(N^3)$. To make the prediction for a new sample, we need to store all the training data points and evaluate α and the kernel function at every training point. For massive datasets, therefore, it is too costly to perform the exact GP inference.

3. GPLasso: ℓ_1-Penalized KL minimization

To reduce the computational cost for GP inference, a common strategy is to reduce the parameters of the GP model. Instead of choosing sparse basis points that the GP model is grounded on (as in Snelson & Ghahramani (2006)), we directly sparsify the variable α in the GP posterior process, so that we can make fast predictions for new data samples. To obtain sparse α, we perform ℓ_1-penalized KL minimization, which balances the approximation quality and the sparsity level. Specifically, we propose the following cost function

$$Q(\alpha^\star) = 2\mathrm{KL}(p(f|\mathcal{D})||q(f)) + \lambda|\alpha^\star|_1 \tag{6}$$

where $|\alpha^\star|_1 = \sum_i |\alpha_i^\star|$ is the ℓ_1 norm of $\alpha^\star$. Note that this KL divergence $\mathrm{KL}(p||q)$ is defined in an *infinite* functional space that measures the difference between the the exact and the sparse GP *posterior processes*.

The exact posterior process $p(f|\mathcal{D})$ depends on α and β as shown in (3) and (4), so does the proposed cost function. To minimize 6, a simple approach will be to first solve α and β and then sparsify them. But this would be computationally expensive for large datasets. As shown below, we actually do not need to solve them directly to obtain the sparse $\alpha^\star$.

According to Chapter 2 of Csató (2002), we have

$$\begin{aligned} &2\mathrm{KL}(p(f|\mathcal{D})||q(f)) \\ =&(\alpha - \alpha^\star)^{\mathrm{T}}(-\beta^\star + K^{-1})^{-1}(\alpha - \alpha^\star) + \\ &\mathrm{tr}[(\beta^\star - \beta)(K^{-1} - \beta^\star)^{-1}] - \ln|(K^{-1} - \beta)(K^{-1} - \beta^\star)^{-1}| \end{aligned}$$

If we set the variable $\beta^\star$ of the sparse GP to be the same as that of the exact GP, i.e., $\beta^\star = \beta$, the above equation becomes

$$2\mathrm{KL}(p(f|\mathcal{D})||q(f)) = \quad (\alpha - \alpha^\star)^{\mathrm{T}} A(\alpha - \alpha^\star) + \text{constant}$$

where $A = (-\beta + K^{-1})^{-1}$ and the second term is a constant given β.

Applying the Woodbury matrix identity to A, we obtain $A = K + \frac{1}{v_y}KK$ and compute $Q(\alpha^\star)$ as follows:

$$Q(\alpha^\star) = \alpha^{\star\mathrm{T}}(K + \frac{1}{v_y}KK)\alpha^\star - \frac{2}{v_y}\alpha^{\star\mathrm{T}}K\mathbf{y} + \lambda|\alpha^\star|_1 \tag{7}$$

The above cost function does not depends on α and β and corresponds to a ℓ_1-penalized least square fitting problem. Solving this problem gives the sparse $\alpha^\star$. The coefficient λ controls how sparse $\alpha^\star$ will be.

We can interpret GPLasso from a "pseudo-output" (instead of "pseudo-input") perspective. Let us define the "pseudo-output" $\mathbf{y}^\star$ as

$$\mathbf{y}^\star = \beta^{-1}\alpha^\star. \tag{8}$$

Then the sparse posterior process $q(f)$ in (6) can be rewritten as

$$q(f) \propto GP(f|0, K)\prod_{i=1}^{N}\mathcal{N}(y_i^\star|f(\mathbf{x}_i), v_y). \tag{9}$$

So $q(f)$ is essentially grounded on *all* the data inputs, but with the pseudo-output $\mathbf{y}^\star$. So GPlasso is in sharp contrast to previous sparse GP models grounded on a small set of basis points.

For a new data sample $\mathbf{x}_{N+1}$, we can make the prediction

$$y_{N+1} = K(\mathbf{x}_{N+1}, B)\alpha^\star$$

where we only need to evaluate the kernel function at the training points corresponding to nonzero elements of $\alpha^\star$.

Since we do not solve $\beta^\star = \beta$ directly, we cannot provide the exact variance for a prediction (though for many applications, we may not need this quantity). Instead, we can obtain an estimate for the variance. One approach is to use the bounding technique proposed by Smola & Bartlett (2001). Another approach is to use efficient iterative linear system solver, such as the Conjugate Gradient, to avoid the matrix inversion in variance computation. In practice, a reasonable variance approximation can be obtained with only a small number of iterations.

4. GPLasso estimation

Now we present an efficient algorithm to solve the convex minimization problem (7) based on Least Angle Regression (LAR) (Efron et al., 2002) and low-rank matrix approximations.

4.1. Convex optimization by LAR

LAR was originally designed to solve ℓ_1-penalized linear least-square problems. To use it for our minimization problem, we need to convert it to an equivalent ℓ_1-penalized least-square problem. Our derivation is similar to that of the LAR-EN algorithm for the elastic net (Zou & Hastie, 2005). Let $K = R^T R$ be the Cholesky factorization of the kernel matrix K. We define $2N \times N$ matrix C and $2N \times 1$ vector d as

$$C = \begin{bmatrix} K/\sqrt{v_y} \\ R \end{bmatrix}, \quad d = \begin{bmatrix} \mathbf{y}/\sqrt{v_y} \\ 0 \end{bmatrix}. \tag{10}$$

Then we have the equivalent ℓ_1-penalized least-squared problem

$$\min_{\alpha^\star} \|C\alpha^\star - d\|^2 + \lambda|\alpha^\star|_1 \tag{11}$$

This minimization problem is equivalent to the following

$$\min_{\alpha^\star} \|C\alpha^\star - d\|^2, \quad \text{subject to } |\alpha^\star|_1 \le t \tag{12}$$

where the positive constraint value t depends on λ. LAR efficiently computes the lasso solution paths of (12) for all $t > 0$, which allows us to explore the full set of sparse solutions of $\alpha^\star$ from just one run of LAR. It brings us the computational advantage when we cannot decide the optimal level of the model sparsity and want to explore various levels of sparsity.

Roughly speaking, the LAR algorithm maintains an active set $\mathcal{A}$ of all the training data points and the corresponding elements of $\alpha^\star$. At each step, it either includes a new point to $\mathcal{A}$ or deletes a point from $\mathcal{A}$, and then updates $\alpha^\star$ until the constraint in (12) is violated. We can also track the size of $\mathcal{A}$ to obtain solutions with a desired number of nonzero elements in $\alpha^\star$. Suppose LAR stops at the l-th step, the time complexity is $O(M^3 + lN^2)$ and the space complexity is $O(N^2)$ (Efron et al., 2002). Now the computational bottleneck is the Cholesky factorization, which takes $O(N^3)$ time. The total time complexity is $O(lN^2 + N^3)$, in which we omit M^3 since it is dominated by the other terms.

4.2. Low-rank kernel matrix approximation

To reduce the cost of the Cholesky factorization, we can use low-rank kernel matrix approximations, among

which Incomplete Cholesky Factorization (ICF) is a popular choice(Fine et al., 2001). Specifically, we approximate the kernel matrix by

$$K \approx \tilde{R}^T \tilde{R}, \tag{13}$$

where $\tilde{R}$ is a $r \times N$ matrix and $r \ll N$. The rank r can be determined by either a target computational cost or an expected reconstruction quality of K.

We define $S = \tilde{R}\tilde{R}^T$ and let R_S be the low-rank Cholesky factor of S. Then we approximate the original minimization problem (11 & 12) as follows:

$$\min_{\alpha^\star} \|\tilde{C}\alpha^\star - \tilde{d}\|^2, \quad \text{subject to } |\alpha^\star|_1 \le t \tag{14}$$

$$\tilde{C} = \begin{bmatrix} R_S\tilde{R}/\sqrt{v_y} \\ \tilde{R} \end{bmatrix}, \quad \tilde{d} = \begin{bmatrix} R_S\tilde{R}\mathbf{y}/\sqrt{v_y} \\ 0 \end{bmatrix}, \tag{15}$$

where $\tilde{C}$ is a $2r \times N$ matrix and $\tilde{d}$ is a $2r \times 1$ vector. The time complexity of ICF is $O(Nr^2)$ and the space complexity is $O(Nr)$. With the ICF approximation, the time complexity of GPLasso is $O(M^3 + lNr + Nr^2) = O(\max\{lNr, Nr^2\})$ — where M^3 is dominated by the other terms — and the space complexity is $O(Nr)$. Our experimental results show that a relatively small rank r can lead to fast GPLasso training with high prediction accuracy, much higher than that of the full GP with the direct ICF approximation. Detailed results are given in Section 7.

5. Alternative view

We offer an alternative view of our sparse GP model in terms of functional regularization in the Reproducing Kernel Hilbert Space (RKHS). The Gaussian process regression can be viewed as Regularized Least Squared Regression (RLSR) in RKHS (Poggio & Girosi, 1990). Let $\mathcal{X}$ be the sample space where $\mathbf{x}_i \in \mathcal{X}$ and $\mathcal{H}$ be the RKHS on $\mathcal{X}$ induced by a kernel K. RLSR aims to find a regression function $f \in \mathcal{H}$ by minimizing the sum of squared errors plus a regularization term,

$$\min_{f \in \mathcal{H}} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}} (y_i - f(\mathbf{x}_i))^2 + v_y \|f\|_{\mathcal{H}} \tag{16}$$

The predictive value of RLSR for a new instance is the same as the predictive mean of Gaussian processes. Moreover, the celebrated representer theorem states that the solution of the above problem can be written as a linear combination of the kernel functions associated with the training points. A natural question is whether our sparse GP model has a kernel machine counterpart and a related representer theorem or not.

We consider the subset $\mathcal{H}_0 \subset \mathcal{H}$

$$\mathcal{H}_0 = \{f_0 \in \mathcal{H} | f_0(\cdot) = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}_i, \cdot), \alpha_i \in \mathbb{R}\}$$

and denote the projection operator onto $\mathcal{H}_0$ as P_0. Define the "ℓ_1 norm" $\|f\|_1 = \sum_{i=1}^{N} |\alpha_i|$ where $\mathrm{P}_0 f = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i$, $\forall f \in \mathcal{H}_0$. An extension of the original representer theorem involving the "ℓ_1 norm" regularization term is described as follows:

Theorem 1. *Given the training data $\mathcal{D} = \{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathbb{R} | i = 1, \ldots, N\}$, a cost function $\mathcal{L} : \mathcal{X} \times \mathbb{R} \times \mathcal{H} \to \mathbb{R}$, a nondecreasing function $g : \mathbb{R} \to \mathbb{R}$, and an arbitrary function $h : \mathbb{R} \to \mathbb{R}$. The risk minimization problem*

$$\min_{f \in \mathcal{H}_0} \sum_{i=1}^{N} \mathcal{L}[(\mathbf{x}_i, y_i, f(\mathbf{x}_i))] + g(\|f\|_{\mathcal{H}}) + h(\|f\|_1) \tag{17}$$

has minimizers taking the form

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}_i, \mathbf{x}), \alpha_i \in \mathbb{R} \tag{18}$$

The proof of Theorem 1 is very similar to the proof of the original representer theorem by showing the minimizer of each term in (17) lays in $\mathcal{H}_0$ (Schölkopf et al., 2001). Let $\mathcal{L}$ be the corresponding cost function in RLSR, $g(\mathbf{x}) = h(\mathbf{x}) = \mathbf{x}$. Applying the theorem, we recover an optimization problem that is equivalent to the optimization problem of GPLasso.

Therefore we can interpret GPLasso as a kernel machine defined in a RKHS with the additional "ℓ_1" regularizer (corresponding the ℓ_1 penalizer in GPLasso). Since our extension of the representer theorem is general, it also has the potential to be applied to other kernel machines to achieve sparse solutions.

6. Related works

Many sparse GP approaches select a representative subset of training points as basis points. For example, Seeger & Williams (2003) proposed a greedy basis-point selection method that efficiently chooses basis points from training data based on an information gain criterion. Snelson & Ghahramani (2006) removed the restriction that basis points are selected from the training set and used pseudo-inputs as basis points. They chose pseudo-inputs by evidence maximization. Quiñonero Candela & Rasmussen (2005) provided a unified framework to interpret various sparse GP regression models. More recently, Walder et al. (2008) generalized the pseudo-input approach to include basis-dependent length-scales for the kernel function and maximized evidence to learn the basis points

and hyperparameters. For the pseudo-input approach and its generalization, the computational cost is $O(gNM^2)$, where N is the number of training samples, M is the number of basis points, and g is the number of iterations for evidence maximization. According to our experiments, this maximization often requires a large number of iterations before convergence, resulting in a large computational cost. Furthermore, evidence maximization has the risk of overfitting, which has been observed by Titsias (2009) in the context of sparse GP learning and by Qi et al. (2004) in the context of variable selection.

Compared to the pseudo-input approach, GPLasso is more related to the approaches proposed by Smola & Bartlett (2001) and Keerthi & Chu (2006). Actually if we remove the ℓ_1 penalty term, our cost function becomes equivalent to the cost function proposed by Smola & Bartlett (2001), although ours is derived from a totally different perspective based on the penalized KL minimization. The approaches proposed by Keerthi & Chu (2006) and Smola & Bartlett (2001) use greedy procedures to solve a difficult integer programming problem for subset selection. As a result, these approaches suffer from local optimal solutions. By contrast, based on a convex cost function, GPLasso finds an (approximate) global optimal solution. This results in significantly higher prediction accuracy than these greedy approaches as demonstrated in Section 7. Also, using the ICF low-rank approximation, the time complexity of GPLasso is only $O(\max\{lNr, Nr^2\})$. Using a low rank r on the order of M and letting $l \approx g$ (which were true in all of our experiments), GPLasso is much more efficient than the evidence maximization approach, which was also empirically demonstrated in our experimental results.

Recently, Titsias (2009) proposed a variational method to learn sparse GP models. This method minimizes a KL divergence between $KL(q(\mathbf{f}, \mathbf{f}_S)||p(\mathbf{f}, \mathbf{f}_S|\mathbf{y}, \mathbf{X}))$, where $\mathbf{f}$ is the function values evaluated on all the training points and $\mathbf{f}_S$ is the inducing variables evaluated on all the pseudo-inputs (i.e., basis points) S. Again, a greedy approach is used to choose the basis points to avoid the prohibitive combinatorial search. By contrast, GPLasso minimizes the penalized KL divergence $\mathrm{KL}(p(f|\mathcal{D})||q(f))$ where f is in the *infinite* functional space instead of being a finite projection at the training points and the basis points. Furthermore, the KL divergence used by GPLasso has a different directionality compared to that used by the variational method, better preserving the posterior mean as discussed by Minka (2004).

7. Experiment

We evaluate the new sparse GP regression method, GPLasso, on both synthetic and real data and compare it with the pseudo-input sparse GP (SPGP) method (Snelson & Ghahramani, 2006)[2], the information vector machine (IVM) (Seeger & Williams, 2003)[3] and the kernel matching pursuit (KMP) method (Keerthi & Chu, 2006). In our experiments, we use the Gaussian kernel defined by $K(\mathbf{x}, \mathbf{x}') = \exp - \sum_j w_j (x_j - x'_j)^2$, where w_j represents the length-scale for the j-th dimension. We apply *Automatic Relevance Determination* (ARD) (Rasmussen & Williams, 2006) to learn the hyperparameters $\{w_j\}$ on a subset of training data using the full GP. For the full GP and SPGP, we use the optimization code, `minimize.m`, [4], which implements a conjugate gradient method. The optimization stops when the difference of two function values in consecutive updates is smaller than 10^{-5}. The prediction accuracy for every method is measured by Root Mean Squared Error (RMSE), $\sqrt{\frac{\sum_i (y_i - f(\mathbf{x}_i))^2}{N}}$.

7.1. Results on synthetic data

First, we test our method on a synthetic dataset previously used by Snelson & Ghahramani (2006). This dataset is small enough, so that we learn the kernel hyperparameters by maximizing the evidence of the full GP. Then we use these hyperparameters for all the other methods to have a fair comparison.

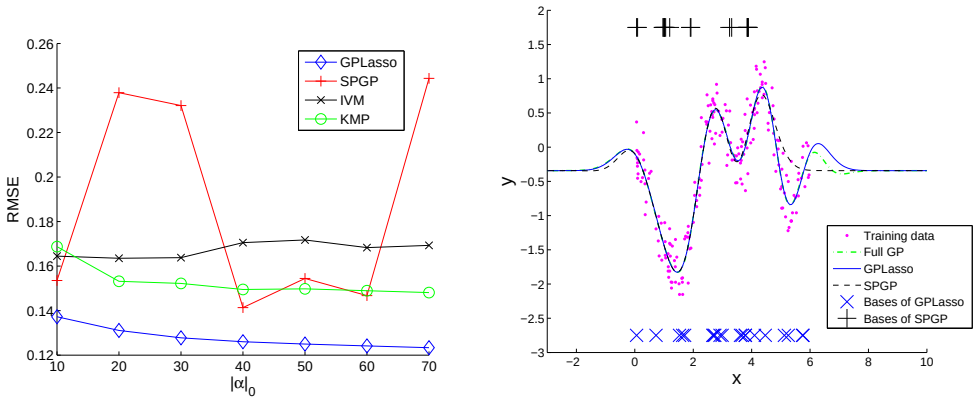
To examine how well each sparse GP method approximates the exact full GP, we compare the predictions of each method with those of the full GP on this synthetic data and compute the RMSE. Figure 1(a) shows the approximation error of each method. To mitigate the effect of random initializations, we randomly split the dataset 10 times, each time with 90% of the points as the training set and the remaining 10% as the test set. We report the RMSE averaged over the 10 runs for each method. For GPLasso, we set the rank of ICF r to 30. The figure shows that GPLasso significantly outperforms than all the other sparse GP methods.

The curve for SPGP is not smooth, since learning pseudo-inputs by evidence maximization leads to a nonlinear optimization problem and the solutions often get stuck in local optima. To visualize this local optimal problem, we examine the basis points selected

[2] `http://www.gatsby.ucl.ac.uk/~snelson/`

[3] `http://www.cs.manchester.ac.uk/~neil/ivm/`

[4] http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/minimize/



(a) (b)

Figure 1. Experiment results on synthetic data (a) Averaged RMSE of predictive mean for sparse GP methods compared to full GP. GPLasso with ICF achieves higher prediction accuracy in all cases. SPGP suffers from local optimal solutions which causes unstable prediction accuracy. (b) Predictive mean of full GP, SPGP and GPLasso with 20 basis points (nonzero elements of $\alpha^\star$). We use the same kernel hyperparameters learned from full GP.

by SPGP. In particular, figure 1(b) shows the locations of the learned basis points for SPGP and the corresponding data points for the nonzero elements of $\alpha^\star$ in GPLasso on one split of the dataset. Clearly the training points chosen by GPLasso are well spread the whole input space, while the basis points learned by SPGP are concentrated in the left side of the input space due to the local optimality of its solution. Accordingly, as shown in the figure, GPLasso gives predictions much closer to the predictions of the full GP than SPGP. While figure 1(b) shows the results from only one run, we repeat this experiment many times with different random initializations and find that the SPGP solutions often suffer from the local optimal solutions on this dataset. By contrast, GPLasso consistently finds a good sparse $\alpha^\star$, resulting in accurate predictive performance.

7.2. Results on real-world data

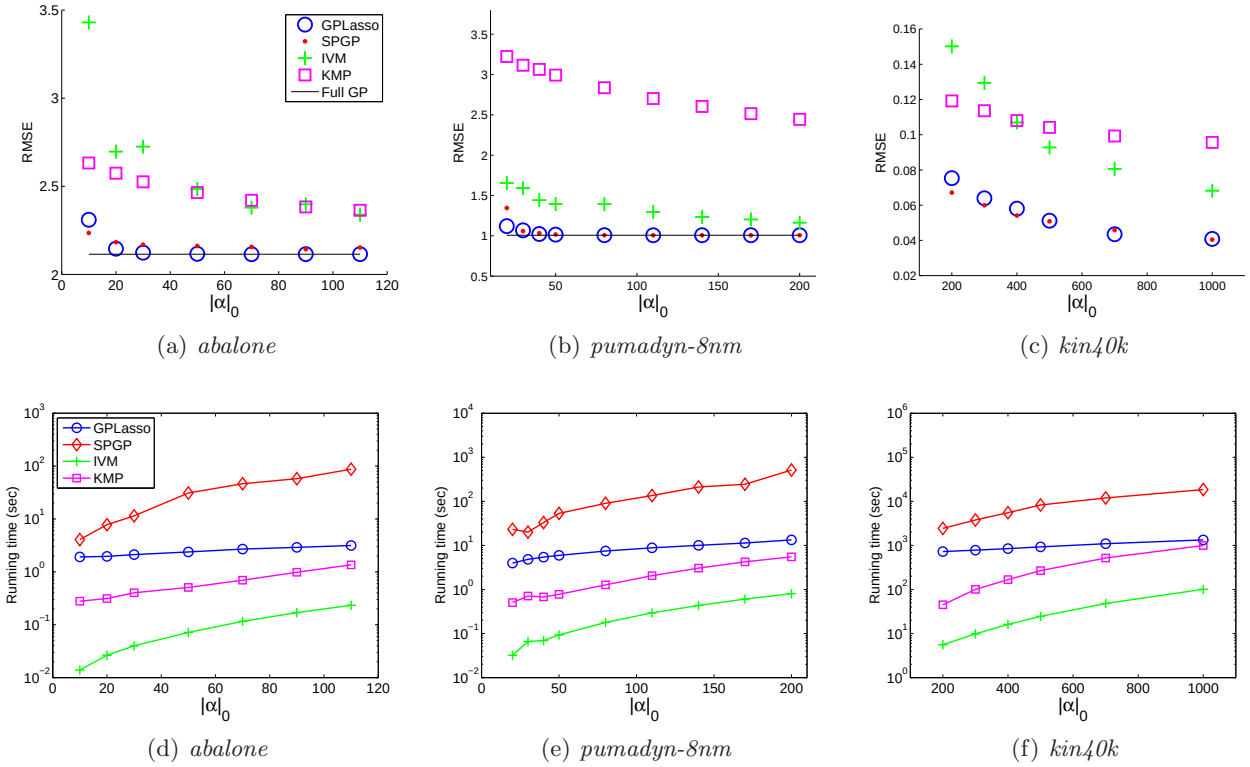
We test GPLasso, SPGP, IVM, and KMP on three real-world datasets *abalone*, *pumadyn-8nm* and *kin40k* [5]. We use a relatively large number of test data points to obtain reliable estimates of predictive performance. For *abalone* and *pumadyn-8nm*, the results are averaged over 10 randomly splits of the dataset. For *abalone*, each split has 1500 training and 2677 test points. For *pumadyn-8nm*, each split has 2000 training and 6192 test points For GPLasso, the ICF rank is set to $r = 250$ and $r = 300$ for these two datasets, respectively.

[5] *abalone*: http://archive.ics.uci.edu/ml/datasets/Abalone. *pumadyn-8nm* and *kin40k*: www.cs.toronto.edu/~delve/

For *kin40k*, due to the long running time, we use only one split for *kin40k* that contains 10000 training and 30000 test points. We do not run the full GP on the whole dataset. For GPLasso, we set the ICF rank $r = 2000$. We maximize the evidence of the full GP model trained on a subset of the training set to learn the model hyperparameters, and then keep them fixed for all the sparse GP methods.

We report the average RMSE based on the predictive mean and the labels of the test points for all these methods. The standard deviations of the prediction errors are very small, so that the error bars are not shown here to avoid cluttering. Figure 2 (a)-(c) show that GPLasso significantly outperforms IVM and KMP in terms of prediction accuracy. With the same hyperparameters, GPLasso starts off with the prediction accuracy similar to SPGP's. As the model sparsity decreases, the predictive performance of both methods converges rapidly to that of the full GP on *abalone* and *pumadyn-8nm*. Interestingly, if we jointly learn the basis points and the hyperparameters for SPGP, instead of using the hyperparameters learned by the full GP on the subset of the training data, the predictive performance of SPGP (not reported here due to space limitations) then becomes much worse than GPLasso. This demonstrates overfitting, which has been discussed by Snelson & Ghahramani (2006).

Figure 2 (d)-(f) show the average training time of each method in the logarithmic scale. All the methods are implemented in Matlab and we carefully optimize the code for each method to achieve the fastest running time. With the lower computational complexity as discussed in Section 6, GPLasso is much faster than



(a) *abalone* (b) *pumadyn-8nm* (c) *kin40k*

(d) *abalone* (e) *pumadyn-8nm* (f) *kin40k*

Figure 2. Prediction errors and training time on three real datasets. (a)-(c) show the mean RMSEs of both sparse GPs and the full GP. GPLasso is competitive with (sometimes better than) SPGP and much better than IVM and KMP in terms of prediction accuracy. (d)-(f) illustrate the training time of each method on these datasets. Note that the running time is plotted in the logarithmic scale. While GPLasso is slower than IVM and KMP, its predictive performance is much better than those of IVM and KMP based on the same number of data points. Furthermore, while maintaining the same level of predictive performance, GPLasso is significantly faster than SPGP by an order of magnitude.

SPGP (by an order of magnitude) in the training time. Although IVM and KMP are even faster, they trade predictive performance for speed. GPLasso keeps the comparable predictive performance as SPGP, but with much less computational time.

Low-rank kernel matrix approximation, e.g., ICFs, can be used directly as a tool to reduce the GP training time and it will be as efficient as GPLasso given the same low rank r. Therefore, a natural question is if that will lead to predictive performance similar to GPLasso. Figure 3 answers this question by comparing the prediction accuracy of the full GP with ICF approximation and GPLasso. We fix the number of nonzero elements in $\alpha^\star$ and keep r the same for the low-rank approximation of both the full GP and GPLasso. GPLasso achieves much higher prediction accuracy than the direct application of ICF to the full GP model, especially when the ICF ranks are low. This sharp difference reveals the great advantage of the posterior approximation in GPLasso.

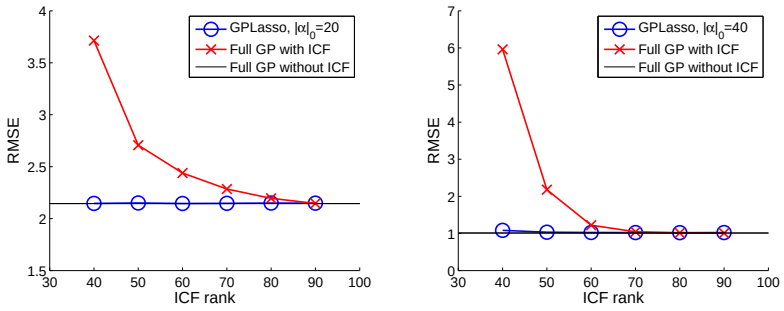
8. Discussion

In this work we have used ICF to reduce the computational cost. The Nyström method is an alternative to ICF for low-rank kernel matrix approximations (Williams & Seeger, 2001). Zhang et al. (2008) demonstrated that an improved Nyström method can provide much lower reconstruction error than ICF. We expect this method can be integrated into GPLasso to further improve its performance.

We have limited $\alpha^\star$ to the training samples. We can remove this limitation by minimizing the KL divergence over a sparse $\alpha^\star$ that corresponds to pseudo-inputs, instead of given training samples.

Acknowledgements

We would like to thank Mark Schmidt and Wei Chu for providing the LAR code and the kernel matching pursuit code respectively. F. Yan and Y. Qi were supported by NSF IIS-0916443 and NSF ECCS-0941533.



(a) *abalone* (b) *pumadyn-8nm*

Figure 3. Performance comparison between the full GP with an ICF approximation and GPLasso with with the same ICF rank r. These two methods are tested on *abalone* and *pumadyn-8nm*. For GPLasso, while r is varying, we fix $|\alpha|_0 = 20$ and $|\alpha|_0 = 40$ for the two datasets, respectively. With a similar computational cost, GPLasso significantly outperforms the full GP with the ICF approximation, especially when the rank of ICF is low.

References

Csató, Lehel. *Gaussian Processes - Iterative Sparse Approximations*. PhD thesis, Aston University, 2002.

Efron, Bradley, Hastie, Trevor, Johnstone, Lain, and Tibshirani, Robert. Least angle regression. *Annals of Statistics*, 32, 2002.

Fine, Shai, Scheinberg, Katya, Cristianini, Nello, Shawe-taylor, John, and Williamson, Bob. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2, 2001.

Keerthi, Sathiya and Chu, Wei. A matching pursuit approach to sparse Gaussian process regression. In Weiss, Y., Schölkopf, B., and Platt, J. (eds.), *Advances in Neural Information Processing Systems 18*. MIT Press, 2006.

Minka, Tom. Power EP. Technical report, Microsoft Research, 2004.

Poggio, T. and Girosi, F. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247(4945), 1990.

Qi, Yuan, Minka, Thomas P., Picard, Rosalind W., and Ghahramani, Zoubin. Predictive automatic relevance determination by expectation propagation. In *Proceedings of Twenty-first International Conference on Machine Learning*, 2004.

Quiñonero Candela, J. and Rasmussen, Carl E. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6, 2005.

Rasmussen, Carl E. and Williams, Christopher K. I. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

Schölkopf, B., Herbrich, R., Smola, A., and Williamson, R. A generalized representer theorem. In *COLT*, 2001.

Seeger, Matthias and Williams, Christopher K. I. Fast forward selection to speed up sparse Gaussian process regression. In *AISTATS*, 2003.

Smola, Alex J. and Bartlett, Peter. Sparse greedy Gaussian process regression. In *Advances in Neural Information Processing Systems 13*. MIT Press, 2001.

Snelson, Edward and Ghahramani, Zoubin. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*. MIT press, 2006.

Titsias, Michalis. Variational learning of inducing variables in sparse Gaussian processes. In *International confernece on AI and Statistics*, 2009.

Walder, Christian, Kim, Kwang In, and Schölkopf, Bernhard. Sparse multiscale Gaussian process regression. In *Proceedings of the 25th international conference on Machine learning*, 2008.

Williams, Christopher and Seeger, Matthias. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*. MIT Press, 2001.

Zhang, Kai, Tsang, Ivor W., and Kwok, James T. Improved Nyström low-rank approximation and error analysis. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, 2008.

Zou, Hui and Hastie, Trevor. Regularization and variable selection via the elastic net. *Journal Of The Royal Statistical Society Series B*, 67(2), 2005.