Approximation Bounds for Some Sparse Kernel Regression Algorithms

Tong Zhang
IBM T.J. Watson Research Center,
Yorktown Heights, NY 10598 USA
tzhang@watson.ibm.com

Abstract

Gaussian processes have been widely applied to regression problems with good performance. However, they can be computationally expensive. In order to reduce the computational cost, there have been recent studies on using sparse approximations in Gaussian processes. In this paper, we investigate properties of certain sparse regression algorithms that approximately solve a Gaussian process. We obtain approximation bounds, and compare our results with related methods.

1 Introduction

Gaussian processes have recently attracted much attention since they are relatively simple and achieve good performance for regression problems (see [Wahba, 1990, Williams and Rasmussen, 1996, Williams, 1998] and references therein). From the convex analysis point of view, they can be written as dual forms of ridge regressions. By using kernel representations, Gaussian processes can solve ridge regression problems in infinite dimensional Hilbert spaces.

One disadvantage associated with Gaussian processes is that there is a parameter for every data point. This implies that the computation can still be very expensive when we have a large number of training data. In order to overcome this problem, various methods have been proposed in the literature to accelerate the computation. In this paper, we are interested in certain sparse Gaussian process algorithms such as those in [Csató and Opper, 2002, Gao et al., 2001, Luo and Wahba, 1997, Lin et al., 2000, Nair et al., 2001, Smola and Bartlett, 2001, Xiang and Wahba, 1998].

The problem of achieving sparse representation in a regression problem has drawn much attention in the literature. Some previous approaches are summarized in Section 2. In Section 3, we review the duality between ridge regression and Gaussian processes, and establish some notations and conventions used throughout the paper. Section 4 outlines some sparse Gaussian process algorithms. We then derive approximation bounds for these algorithms in Section 5. However, such results do not distinguish the performance of greedy algorithms from randomized algorithms.

Therefore from a closely related but different point of view, we derive additional approximation results that are only applicable to greedy algorithms. Section 7 discusses relationships among different methods. In Section 8, we use numerical examples to illustrate aspects of the proposed algorithms. Section 9 summarizes the paper. It is worth mentioning that this paper focuses on the quality of sparse approximation. In particular, we do not consider the generalization performance of the proposed procedures so that the paper can be concentrated on the main topic.

2 Previous work

We consider the standard linear regression model in the following setting. Assume we have a set of feature vectors $x_1, \ldots, x_n$, with corresponding real-valued output variables $y_1, \ldots, y_n$. Our goal is to find a linear weight vector w such that $y \approx w^T x$ for all future feature data x. The quality of this approximation is measured by the squared loss as:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - w^T x_i). \tag{1}$$

We would like to find a weight vector w that gives a small loss in (1). Many methods have been proposed over the years to achieve sparseness in w for regression problems. We shall only mention ideas that are directly related to this paper.

In (1), we may consider the output variables $[y_i]_{i=1:n}$ as a vector in a (finite dimensional) Hilbert space H, and consider the components of training inputs $[x_{ij}]_{i=1:n}$ as a library of basis vectors in H. We use $x_{i,j}$ to denote the j-th component of a training input x_i. In this regard, (1) can be viewed as the approximation of a vector (or function) in a Hilbert space by a linear combination of a library of basis vectors (or functions). Assume that the target vector is in the convex hull of the basis vectors, Jones observed in [Jones, 1992] that using greedy approximation, one can achieve an error rate of $O(1/k)$ if we approximate the target vector using k library vectors. The same idea has been used in [Barron, 1993] to analyze the approximation property of neural networks. In an algorithmic form it led to the *matching pursuit* method in [Mallat and Zhang, 1993].

It is also possible to directly impose a sparseness constraint into the problem formulation itself. However, a difficulty is that the resulting formulation is typically NP-hard (for example, see [Natarajan, 1995]). One way to remedy this problem is to include a 1-norm regularization condition in the ridge regression formulation, as in *basis pursuit* [Chen et al., 1999]. That is, we favor a weight vector w that has a small 1-norm, as in the following formulation:

$$\hat{w} = \arg\min_{w}[\frac{1}{n}\sum_{i=1}^{n}(y_i - w^T x_i)^2 + \lambda\|w\|_1]. \tag{2}$$

The positive parameter λ is used to control the complexity of w measured by its 1-norm. In basis pursuit, we still have a convex programming problem, which can be solved efficiently. Furthermore,

1-norm regularization leads to a sparse $\hat{w}$ solution.

The above mentioned methods directly deal with the primal form of regression formulation (1). They compute sparse coefficients for the primal weight vector w. As we have mentioned earlier, kernel methods such as Gaussian processes are expressed in the dual form of (ridge) regression. The dual representation has the advantage that a finite dimensional kernel formulation corresponds to a possibly infinite dimensional primal formulation. It is thus useful to consider an approximation of the solution using a sparse dual representation (the primal weight vector can still be dense). One way to achieve dual sparseness is through Vapnik's ϵ-insensitive loss function in a support vector machine [Vapnik, 1998]: its equivalent primal formulation can be expressed as:[1]

$$\hat{w} = \arg\min_{w}[\frac{1}{n}\sum_{i=1}^{n}\max(|y_i - w^T x_i| - \epsilon, 0)^2 + \lambda w^2], \tag{3}$$

where we use w^2 to denote $w^T w$. The relationship of this method and the primal basis pursuit regression has been discussed in [Girosi, 1998].

In this paper, we are mainly interested in achieving sparsity through greedy approximation. This approach has a number of advantages over a direct sparse convex programming formulation such as basis pursuit or support vector learning. One advantage is that the sparsity is directly controlled in a greedy approximation algorithm, rather than through some other quantities such as λ in (2) or ϵ in (3). Another advantage is that greedy approximation does not change the objective optimization function, while a direct method such as basis pursuit (or SVM) modifies the objective function by including a 1-norm regularization (or ϵ-insensitive loss). The third advantage, which is more important for this paper, is that in general, provably good approximation bounds can be obtained for greedy methods, but not for direct convex programming methods.

Methods studied in this paper are closely related to some recent works. One related method is the online update scheme in [Csató and Opper, 2002] which only deals with algorithmic issues. It can be regarded as a special form of rank-one matrix update algorithms widely studied in numerical linear algebra (for example, see relevant sections in [Golub and Van Loan, 1996] and references therein). The work does not contain theoretical results concerning the approximation rate which we are interested in here. In the statistical literature, sparse approximation has been widely used for kernel spline regression [Gao et al., 2001, Luo and Wahba, 1997, Lin et al., 2000, Xiang and Wahba, 1998]. Although some theoretical justifications were presented, they were different from techniques used in this paper. In addition, they didn't provide explicit approximation bounds which we are interested in.

Although some greedy approximation bounds were described in [Smola and Bartlett, 2001], they relied on results proved in [Natarajan, 1995]. These results specify the approximation quality of the proposed greedy procedure in terms of the best possible sparse approximation and quantities

[1] Originally, Vapnik's ϵ-insensitive loss was linear. We use a quadratic form here for compatibility with other formulations.

that are not always well-behaved. They are different from those in [Jones, 1992, Barron, 1993], where such quantities do not appear. Both type of bounds have advantages and disadvantages. In spite of their differences, we show in this paper that both types of bounds can be obtained using the same underlying technique.

In Section 5, we derive bounds in the form of [Jones, 1992, Barron, 1993] by using a randomization technique. One disadvantage of this analysis is that the resulting approximation bound has a relatively slow convergence rate of $O(1/k)$ where k is the number of sparse components. Note that such a rate is very typical for randomization or Monte Carlo methods. Another disadvantage is that due to the underlying randomization, the analysis does not distinguish greedy algorithms from randomized algorithms. Therefore in Section 6, we further investigate the behavior of greedy algorithms where we shall improve the main result of [Natarajan, 1995] (hence the bound given in [Smola and Bartlett, 2001]). In this analysis the convergence rate of certain greedy algorithms can be as fast as $O((1-\mu)^k)$ but μ depends on additional quantities that characterize the orthogonality of the underlying basis functions (which may not always be well-behaved). We outline our proofs in such a way that it becomes apparent that the analysis of Section 5 and that of Section 6 are based on the same underlying principle. Therefore this investigation establishes an fundamental relationship of [Natarajan, 1995] to [Jones, 1992].

Note that we study Gaussian processes in the dual representation, which has an objective function different from the primal form of matching pursuit. Therefore, in order to analyze the system, we also employ ideas from online learning. As we see later, this analysis leads to some interesting theoretical consequences and algorithmic implications.

3 Preliminaries

3.1 Duality in ridge regression

We have mentioned that Gaussian processes can be expressed in dual forms of regression problems. Since this duality is important in our discussion and in distinguishing primal greedy algorithms such as matching pursuit from their dual counterparts, we shall briefly review the basic ideas. This discussion also introduces notations which we use throughout the paper.

Consider the linear least squares regression problem in (1). In this paper, we assume that data x belong to a Hilbert space H that may be infinite dimensional. If the dimension of the input-vector space is larger than the training sample size, the problem of minimizing the least squares loss in (1) is singular. This is because there are infinitely many solutions of w such that $w^T x_i = \hat{w}^T x_i$ for all i, which implies that all these solutions have the same expected loss.

To eliminate this problem of uncertainty, one may consider a simple remedy called ridge regression [Hoerl and Kennard, 1970] which is widely used in statistics. In ridge regression, we obtain a

weight $\hat{w}$ by minimizing the following modified primal objective function:

$$\hat{w} = \arg\min_w \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(w^T x_i - y_i)^2 + \frac{\lambda}{2}w^2, \tag{4}$$

where λ is a regularization parameter that may either be pre-determined or computed from the data using various empirical methods such as cross-validation. As we shall later see from our results, the regularization parameter λ also influences the rate of sparse kernel approximation.

We would like to point out that sparse representation itself can also be regarded as a form of regularization. There is generally a trade-off between λ and k (where k is the number of basis functions in the sparse representation)x. However, there could be a broad region in the (k, λ) space which give very similar results. In such case, a smaller k is often preferred for its obvious computational advantage.

Equation (4) also has a natural Bayesian statistical interpretation. It is the MAP estimator corresponding to a probability model with Gaussian noise and Gaussian weight prior. Although such an interpretation can provide valuable insights, it is not essential in this work. Interested readers are referred to [Smola and Bartlett, 2001, Wahba, 1990, Williams, 1998].

Compared with the basis pursuit formulation (2), ridge regression in (4) employs a 2-norm regularization term rather than a 1-norm regularization. This 2-norm regularization is necessary in order to obtain kernel methods such as Gaussian processes. The conversion to kernel methods rely on a duality relationship between the feature-space representation and the sample-space representation, which we now derive.

By differentiating (4) with respect to w at the optimal solution $\hat{w}$, we obtain a representation of $\hat{w}$ in the following form

$$\hat{w} = \sum_{i=1}^{n}\hat{\alpha}_i x_i, \tag{5}$$

$$\hat{\alpha}_i = -\frac{1}{\lambda n}(\hat{w}^T x_i - y_i), \qquad i = 1, \ldots, n. \tag{6}$$

Equation (4) is called the *primal formulation*, which involves the *primal variable* w. $\hat{\alpha}$ is the *dual variable*. To obtain the corresponding *dual formulation* in the dual variable α, we can eliminate $\hat{w}$ and obtain

$$\hat{\alpha}_i = -\frac{1}{\lambda n}(\sum_{k=1}^{n}\hat{\alpha}_k x_k^T x_i - y_i), \qquad i = 1, \ldots, n. \tag{7}$$

It is easy to check that $\hat{\alpha}$ is the solution of the following dual optimization problem:

$$\hat{\alpha} = \arg\max_\alpha [\sum_{i=1}^{n}\lambda(-\frac{\lambda n}{2}\alpha_i^2 + \alpha_i y_i) - \frac{\lambda}{2}\sum_{i=1}^{n}\sum_{k=1}^{n}\alpha_i\alpha_k x_i^T x_k]. \tag{8}$$

The prediction $\hat{w}^T x$ can be expressed in the dual variable as:

$$\hat{w}^T x = \sum_{k=1}^{n} \hat{\alpha}_k x_k^T x. \tag{9}$$

An equivalent formulation of Gaussian processes can be obtained by directly inserting (5) into the primal formulation (4) to obtain a system involving the dual variable α:

$$\hat{\alpha} = \arg\min_{\alpha} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} (\sum_{k=1}^{n} \alpha_k x_k^T x_i - y_i)^2 + \frac{\lambda}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} \alpha_i \alpha_k x_i^T x_k. \tag{10}$$

The solution of (8) is equivalent to the solution of (10).

For all primal vectors w, we define

$$R(w) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} (w^T x_i - y_i)^2 + \frac{\lambda}{2} w^2, \tag{11}$$

and for any dual vector α, we define

$$Q(\alpha) = [\sum_{i=1}^{n} \lambda(-\frac{\lambda n}{2} \alpha_i^2 + \alpha_i y_i) - \frac{\lambda}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} \alpha_i \alpha_k x_i^T x_k]. \tag{12}$$

We also define

$$\Delta R(w) = R(w) - \inf_{w} R(w),$$
$$\Delta Q(\alpha) = \sup_{\alpha} Q(\alpha) - Q(\alpha).$$

The following facts are useful in our later analysis. Although these results follow from simple linear algebra, we shall include proofs for completeness.

Proposition 3.1 *At the optimal solution $\hat{w}$ of (4) and $\hat{\alpha}$ of (8), $R(\hat{w}) = Q(\hat{\alpha})$.*

Proof. Since $\hat{w}$ and $\hat{\alpha}$ satisfies (5), therefore

$$\begin{aligned}
Q(\hat{\alpha}) =& \sum_{i=1}^{n} \lambda(-\frac{\lambda n}{2} \hat{\alpha}_i^2 + \hat{\alpha}_i y_i) - \frac{\lambda}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} \hat{\alpha}_i \hat{\alpha}_k x_i^T x_k \\
=& [\sum_{i=1}^{n} \lambda(-\frac{\lambda n}{2} \hat{\alpha}_i^2 + \hat{\alpha}_i (y_i - \hat{w}^T x_i))] + [\lambda \hat{w}^T \sum_{i=1}^{n} \hat{\alpha}_i x_i - \frac{\lambda}{2} \sum_{i=1}^{n} \sum_{k=1}^{n} \hat{\alpha}_i \hat{\alpha}_k x_i^T x_k] \\
=& [\sum_{i=1}^{n} \lambda(-\frac{\lambda n}{2} (\frac{1}{\lambda n} (\hat{w}^T x_i - y_i))^2 - \frac{1}{\lambda n} (\hat{w}^T x_i - y_i)(y_i - \hat{w}^T x_i))] + [\lambda \hat{w}^T \hat{w} - \frac{\lambda}{2} \hat{w}^T \hat{w}] \\
=& R(\hat{w}) \qquad \square
\end{aligned}$$

Proposition 3.2 *Let $\hat{w}$ be the solution of (4), then for all w,*

$$\Delta R(w) = \frac{1}{2}(w - \hat{w})^T G(w - \hat{w}),$$

where G denotes the following operator (in matrix form): $G = \frac{1}{n}\sum_{i=1}^n x_i x_i^T + \lambda I$, where I is the identity operator.

Proof. Note that

$$R(w) - R(\hat{w}) = \frac{1}{2}[\frac{1}{n}\sum_{i=1}^n((w - \hat{w})^T x_i)^2 + \lambda(w - \hat{w})^2] + (w - \hat{w})^T[\frac{1}{n}\sum_{i=1}^n(\hat{w}^T x_i - y_i)x_i + \lambda\hat{w}].$$

Now, using the first order condition (5) at the optimal solution $\hat{w}$, we have

$$\frac{1}{n}\sum_{i=1}^n(\hat{w}^T x_i - y_i)x_i + \lambda\hat{w} = 0.$$

Combining the above two equalities, we obtain the proposition. □

Proposition 3.3 *Let $\hat{w}$ be the solution of (4), then*

$$\Delta Q(\alpha) \geq \frac{1}{2n}\sum_{i=1}^n(\hat{w}^T x_i - y_i + \lambda n\alpha_i)^2.$$

Proof. Let $\hat{\alpha}$ be the solution of (8). For any α, using (7), we have

$$\lambda\sum_{i=1}^n(\lambda n\hat{\alpha}_i - y_i + \sum_{k=1}^n \hat{\alpha}_k x_k^T x_i)(\hat{\alpha}_i - \alpha_i) = 0.$$

This implies that

$$\Delta Q(\alpha) = Q(\hat{\alpha}) - Q(\alpha) + \lambda\sum_{i=1}^n(\lambda n\hat{\alpha}_i - y_i + \sum_{k=1}^n \hat{\alpha}_k x_k^T x_i)(\hat{\alpha}_i - \alpha_i).$$

Now replace $Q(\cdot)$ by its definition in (12) in the above equation and combine relevant terms, we obtain

$$\begin{aligned}\Delta Q(\alpha) =& \lambda\sum_{i=1}^n[\frac{\lambda n}{2}(\hat{\alpha}_i - \alpha_i)^2 + \frac{1}{2}\sum_{k=1}^n(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_k - \alpha_k)x_i^T x_k] \\ \geq& \lambda\sum_{i=1}^n\frac{\lambda n}{2}(\hat{\alpha}_i - \alpha_i)^2 = \frac{1}{2n}\sum_{i=1}^n(\hat{w}^T x_i - y_i + \lambda n\alpha_i)^2.\end{aligned}$$

The above inequality follows from the semi positive definiteness of the Gram matrix $[x_i^T x_k]$. □.

3.2 Kernel formulation

Since in the dual formulation (8) and in the dual linear prediction representation (9), we only need to compute the data-space inner product of the form $x_k^T x$, we may replace it by any symmetric positive kernel function $K(x_k, x)$, which leads to the general form of Gaussian processes.

The idea of using kernels have been used in statistics for a long time (see [Wahba, 1990] for example). A kernel form of (10) can be obtained by replacing each inner product $x_k^T x$ with a symmetric positive kernel function $K(x_k, x)$ as:

$$\hat{\alpha} = \arg\min_{\alpha} \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}(\sum_{k=1}^{n} \alpha_k K(x_k, x_i) - y_i)^2 + \frac{\lambda}{2}\sum_{i=1}^{n}\sum_{k=1}^{n} \alpha_i \alpha_k K(x_i, x_k).$$

For technical reasons, we need to assume that a kernel function $K(x_1, x_2)$ considered in this paper can be decomposed as:

$$K(x_1, x_2) = \sum_j \phi_j(x_1)\phi_j(x_2). \tag{13}$$

Mercer's theorem [Mercer, 1909] guarantees the existence of such a decomposition for all (positive) kernels that satisfy some moderate regularity conditions. For some practical kernels such as the polynomial kernel $K(x_1, x_2) = (1 + x_1 x_2)^d$ and the exponential kernel $K(x_1, x_2) = \exp(x_1^T x_2)$, a decomposition in (13) can also be obtained directly from Taylor expansion.

Using (13), the kernel function $K(\cdot, \cdot)$ induces a feature representation of data x as:

$$x \to \phi(x) = \{\phi_j(x)\}_j.$$

$\phi(x)$ lies in a Hilbert space H_K(reproducing kernel Hilbert space) under the 2-norm inner-product. A kernel predictor in the form of

$$p(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x)$$

can be considered as a linear predictor $p(x) = w^T \phi(x)$ in H_K with weight w given by:

$$w = \sum_{i=1}^{n} \alpha_i \phi(x_i) \in H_K.$$

This implies that kernel methods can also be analyzed in the primal form by considering linear regression in H_K, where we simply replace a data point x by the corresponding feature space representation $\phi(x) \in H_K$.

For notational simplicity, analysis in this paper is represented under the special case $\phi(x) = x$. However, the above discussion implies that algorithms and analysis in this paper are also suitable

for general kernel methods for which we simply replace x by $\phi(x)$. Since kernel methods only require the inner product computation, we do not need to know the feature representation $\phi(x)$ to work with a kernel algorithmically.

4 Sparse regression algorithms

We have already proved that the two systems (10) and (8) are equivalent. One might think that it is possible to work with either formulation to obtain a sparse representation of the dual variable α. Unfortunately Proposition 3.3 implies that in general (8) is not appropriate for obtaining sparse representation. To see this, we note that for a regression problem containing noise, there can be a significant portion of data such that $\hat{w}^T x_i - y_i$ is bounded away from zero. In this case, for a sparse variable α, a significant portion of $(\hat{w}^T x_i - y_i - \alpha_i)^2$ is also bounded away from zero. Now the inequality in Proposition 3.3 implies that $\Delta Q(\alpha)$ has to be bounded away from zero. That is, a sparse dual variable α does not approximately solve (8). Because of the above mentioned reason, we shall work with (10) in order to obtain a sparse representation of the dual variable α.

Equation (10) is simply equation (4) with the primal variable w substituted by (5). In order to obtain a sparse dual representation in (10), we only need to approximately minimize (4) with w expressed in the form $w = \sum_i \alpha_i x_i$. It is easy to see that under this assumption, the objective function $R(w) = R(\sum_i \alpha_i x_i)$ can be expressed in a kernel form. In this paper, we consider the following algorithms that approximately solve a Gaussian process with a sparse dual variable.

Algorithm 1 *(Greedy sparse Gaussian process)*

let $w^0 = 0$
for $k = 1, 2, \ldots$
  find $i_k \in \{1, \ldots, n\}, \alpha_k$ and β_k that minimize
    $R(\beta_k w^{k-1} + \alpha_k x_{i_k})$
  let $w^k = \beta_k w^{k-1} + \alpha_k x_{i_k}$
end

Algorithm 2 *(Random sparse Gaussian process)*

Given sparseness k
Randomly draw k samples $x_{i_1}, \ldots, x_{i_k}$ uniformly
  from $\{x_1, \ldots, x_n\}$
find $\alpha_1, \ldots, \alpha_k$ that minimize $R(\sum_{j=1}^k \alpha_j x_{i_j})$
let $w^k = \sum_{j=1}^k \alpha_j x_{i_j}$

Algorithm 3 *(κ-greedy sparse Gaussian process)*

Given integer κ: $1 \leq \kappa \leq n$
let $w^0 = 0$
for $k = 1, 2, \ldots$
  Randomly draw a subset S_k of size κ uniformly
    from $\{1, \ldots, n\}$
  find $i_k \in S_k, \alpha_k$ and β_k that minimize
      $R(\beta_k w^{k-1} + \alpha_k x_{i_k})$
  let $w^k = \beta_k w^{k-1} + \alpha_k x_{i_k}$
end

Algorithm 1 is a greedy sparse approximation algorithm. In the statistical literature, this type of algorithms are sometimes referred to as adaptation methods. A work particularly relevant to this paper is [Luo and Wahba, 1997], where the authors investigated greedy approximation for (kernel-based) splines. A similar algorithm has also been proposed in [Smola and Bartlett, 2001]. However, there are two major differences. One difference is that unlike the algorithm proposed in [Smola and Bartlett, 2001], we do not utilize the dual functional $Q(\alpha)$. The reason has been explained earlier: in the most general situation, it is not possible to find a sparse solution α that approximately maximizes $Q(\alpha)$. The second difference is that at each step, in addition to α_k, we also seek a scaling factor β_k that shrinks the current weight w^{k-1}. As we shall see later, this scaling is important in our analysis.

Algorithm 2 is a random sparse approximation method appeared in [Williams and Seeger, 2000]. In our analysis this algorithm has similar worst case approximation properties as those of the greedy approximation algorithm. However, in practice, the greedy algorithm is usually superior in approximation quality. The full greedy algorithm is computationally quite expensive, therefore a compromise is to consider searching through i_k in a subset of $\{1, \ldots, n\}$ at each step as in Algorithm 3. This idea of choosing a random subset, suggested in [Smola and Bartlett, 2001], is a compromise between the full greedy sparse algorithm and the random sparse algorithm.

Note that in Algorithm 2, we use the more expensive approach of computing the exact optimal approximation in the randomly selected k-dimensional subspace. In the other two algorithms, we only optimize in a two-dimensional subspace at each step. However one may also consider using the full subspace optimization approach as in Algorithm 2. The resulting approximation bounds for the modified algorithms will be the same. Similar randomized algorithms for kernel methods have appeared in the literature, such as approximate smoothing spline methods [Gao et al., 2001, Xiang and Wahba, 1998].

Although in this paper we focus on the approximation aspect of different algorithms, it is useful to compare their computational costs. In the kernel formulation, we make the assumption that the inner product $K(x_i, x_k)$ can be computed in $O(1)$ flops (floating point operations).

In the full greedy and the κ-greedy algorithms, assume we have stored the results of $w^{(k-1)T} x_i$ for $i = 1, \ldots, n$ and $w^{(k-1)T} w^{(k-1)}$ at step k. Then with each fixed i_k, the minimization of $R(\beta_k w^{k-1} +$

$\alpha_k x_{i_k}$) requires $O(n)$ flops. Therefore the flops required to find i_k, α_k and β_k that minimize $R(\beta_k w^{k-1} + \alpha_k x_{i_k})$ is $O(n^2)$ for the full greedy algorithm and $O(\kappa n)$ for the κ-greedy algorithm. Using the recursion $w^k = \beta_k w^{k-1} + \alpha_k x_{i_k}$, the computation of $w^{(k)T} x_i$ for $i = 1, \ldots, n$ and $w^{(k)T} w^{(k)}$ requires another $O(n)$ operations. Overall we need $O(n^2)$ flops per step in Algorithm 1 and $O(\kappa n)$ flops per step in Algorithm 3. The computational costs after k steps are $O(kn^2)$ for Algorithm 1 and $O(\kappa kn)$ for Algorithm 3.

In order to solve the reduced problem in Algorithm 2, we need to rewrite the system in the form of $A\alpha = b$. Note that to form the (j_1, j_2)-th element in A $(j_1, j_2 = 1, \ldots, k)$, we need to evaluate $\sum_{i=1}^{n} k(x_{i_{j_1}}, x_i) k(x_{i_{j_2}}, x_i)$, which requires $O(n)$ flops. Therefore overall, we need $O(k^2 n)$ time to form the reduced system $A\alpha = b$. The solution of the reduced system $A\alpha = b$ can be then obtained in $O(k^3)$ flops.

Based on the above discussion, we summarize the computational costs of the three algorithms in Table 1.

Table 1: Computational costs (in flops) for k-term sparse approximation

algorithm	full greedy	random	κ-greedy
computation	$O(kn^2)$	$O(k^2 n)$	$O(\kappa kn)$

5 Approximation bounds for sparse regression algorithms

Let w be an approximate solution to (4), and $\hat{w}$ be the exact solution. We consider the following type of stochastic gradient update rule with a datum x_u:

$$w_u = w - \eta((\hat{w}^T x_u - y_u) x_u + \lambda w), \tag{14}$$

where $\eta > 0$ is a fixed variable which is to be determined later. Obviously, if w is of form (5), then w_u is also of form (5). Therefore the update rule is suitable for kernel methods. Our goal is to find u in (14) so that $R(w_u)$ is minimized.

For convenience we introduce the following short hand notation

$$\phi_u(w) = (\hat{w}^T x_u - y_u) x_u + \lambda w.$$

Using G in Proposition 3.2, we define the following quantities:

$$a_w = \frac{1}{n}\sum_{u=1}^{n} \phi_u(w)^T G \phi_u(w), \tag{15}$$

$$b_w = \frac{1}{n}\sum_{i=1}^{n} (w^T x_i - y_i)x_i + \lambda w = Gw - \frac{1}{n}\sum_{i=1}^{n} y_i x_i. \tag{16}$$

Our analysis starts with the following lemma.

Lemma 5.1 *Let $\eta = 2\lambda\Delta R(w)/a_w$ in (14), then the following equality holds:*

$$\frac{1}{n}\sum_{u=1}^{n} R(w_u) = R(w) - \frac{2\lambda^2 \Delta R(w)^2}{a_w}. \tag{17}$$

Proof. Note that

$$R(w_u) - R(w) = \frac{1}{2}[\frac{1}{n}\sum_{i=1}^{n}((w_u - w)^T x_i)^2 + \lambda(w_u - w)^2] + (w_u - w)^T[\frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)x_i + \lambda w].$$

Now, summing over u, we obtain

$$\begin{aligned}
&\frac{1}{n}\sum_{u=1}^{n} R(w_u) - R(w) \\
=&\frac{1}{2n}\sum_{u=1}^{n}[\frac{1}{n}\sum_{i=1}^{n}((w_u - w)^T x_i)^2 + \lambda(w_u - w)^2] + \frac{1}{n}\sum_{u=1}^{n}(w_u - w)^T[\frac{1}{n}\sum_{i=1}^{n}(w^T x_i - y_i)x_i + \lambda w] \\
=&\frac{a_w}{2}\eta^2 - \eta(b_{\hat{w}} + \lambda(w - \hat{w}))^T b_w.
\end{aligned}$$

The first order condition (5) implies that $b_{\hat{w}} = 0$. Using Proposition 3.2, we have

$$(b_{\hat{w}} + \lambda(w - \hat{w}))^T b_w = \lambda(w - \hat{w})^T(b_w - b_{\hat{w}}) = \lambda(w - \hat{w})^T G(w - \hat{w}) = 2\lambda\Delta R(w).$$

Therefore

$$\frac{1}{n}\sum_{u=1}^{n} R(w_u) = R(w) + \frac{a_w}{2}\eta^2 - 2\eta\lambda\Delta R(w).$$

This leads to (17) by picking η such that $\eta = 2\lambda\Delta R(w)/a_w$. □

In the above, we computed the average (over data points) of a specific gradient descent rule (14) with optimal choice of η. Clearly this gives an upper bound of the infimum over all such rules (such as used in a sparse greedy algorithm). We thus have the following result:

Lemma 5.2 *Let $M = \max_i \|x_i\|_2$. For all vectors w,*

$$\frac{1}{n}\sum_{i=1}^{n} \inf_{\alpha,\beta} R(\alpha x_i + \beta w) \leq R(w) - \frac{\lambda^2(\Delta R(w))^2}{(R(\hat{w})M^2 + \lambda\Delta R(w))(M^2+\lambda)}.$$

Proof. In order to bound the right hand side of (17), we bound a_w in (15) as:

$$a_w = \frac{1}{n}\sum_{u=1}^{n}[\frac{1}{n}\sum_{i=1}^{n}[\phi_u(w)^T x_i]^2 + \lambda\phi_u(w)^2] \leq \frac{1}{n}\sum_{u=1}^{n}\phi_u(w)^2(\frac{1}{n}\sum_{i=1}^{n}x_i^2 + \lambda).$$

The above inequality follows from the Cauchy-Schwartz inequality. Using $b_{\hat{w}} = 0$, we obtain

$$\begin{aligned}
\phi_u(w)^2 = \frac{1}{n}\sum_{u=1}^{n}((\hat{w}^T x_u - y_u)x_u + \lambda w)^2 =& \frac{1}{n}\sum_{u=1}^{n}((\hat{w}^T x_u - y_u)x_u)^2 - \lambda^2\hat{w}^2 + \lambda^2(w-\hat{w})^2 \\
\leq& \frac{1}{n}\sum_{u=1}^{n}(\hat{w}^T x_u - y_u)^2 M^2 + \lambda(w-\hat{w})^T G(w-\hat{w}) \\
=& \frac{1}{n}\sum_{u=1}^{n}(\hat{w}^T x_u - y_u)^2 M^2 + 2\lambda\Delta R(w).
\end{aligned}$$

The inequality follows directly from the assumption that $x^2 \leq M^2$ and the fact that $G - \lambda I$ is positive semi-definite. The last equality follows from Proposition 3.2. Now we have

$$a_w \leq [\frac{1}{n}\sum_{u=1}^{n}(\hat{w}^T x_u - y_u)^2 M^2 + 2\lambda\Delta R(w)](M^2+\lambda) \leq (2R(\hat{w})M^2 + 2\lambda\Delta R(w))(M^2+\lambda). \quad (18)$$

We obtain from (17):

$$\frac{1}{n}\sum_{u=1}^{n} R(w_u) \leq R(w) - \frac{\lambda^2\Delta R(w)^2}{(R(\hat{w})M^2 + \lambda\Delta R(w))(M^2+\lambda)}.$$

It is easy to see that this inequality implies the lemma. □

Now, by using induction and applying Lemma 5.2 repeatedly, we obtain the following approximation bounds on sparse Gaussian process algorithms.

Theorem 5.1 *Let $M = \max_i \|x_i\|_2$. Then in Algorithm 1,*

$$\Delta R(w^k) \leq \frac{\Delta R(0)(M^2+\lambda)}{\frac{\lambda^2\Delta R(0)}{R(\hat{w})M^2 + \lambda\Delta R(0)}k + (M^2+\lambda)}.$$

Furthermore, in Algorithm 2 and Algorithm 3, the expected approximation error satisfies:

$$E\,\Delta R(w^k) \leq \frac{\Delta R(0)(M^2+\lambda)}{\frac{\lambda^2 \Delta R(0)}{R(\hat{w})M^2+\lambda\Delta R(0)}k + (M^2+\lambda)},$$

where the expectation E is taken over all possible randomized instances.

Proof. We prove the theorem by induction. The bounds clearly hold for $k = 0$. Now assume the theorem holds for w^k at an integer $k \geq 0$. Let

$$s = \frac{\frac{\lambda^2 \Delta R(0)}{R(\hat{w})M^2+\lambda\Delta R(0)}(k+1) + (M^2+\lambda)}{\Delta R(0)(M^2+\lambda)}.$$

Then the induction hypothesis for w^k implies that

$$sE\,\Delta R(w^k) - \frac{\lambda^2 E\,\Delta R(w^k)}{(R(\hat{w})M^2+\lambda\Delta R(0))(M^2+\lambda)} \leq 1. \tag{19}$$

In the above inequality, the expectation E is deterministic for Algorithm 1, and is averaged over all possible randomized instances for Algorithm 2 and Algorithm 3.

Using Lemma 5.2, and note that $E\Delta R(w^k)^2 \geq (E\Delta R(w^k))^2$ always holds (Jensen's inequality), we obtain:

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n} E \inf_{\alpha,\beta} \Delta R(\alpha x_i + \beta w^k) \\
\leq & E\,\Delta R(w^k) - \frac{\lambda^2 E\,\Delta R(w^k)^2}{(R(\hat{w})M^2+\lambda\Delta R(0))(M^2+\lambda)} \qquad \text{(Lemma 5.2)} \\
\leq & \frac{1}{s}[sE\,\Delta R(w^k)(1 - \frac{\lambda^2 E\,\Delta R(w^k)}{(R(\hat{w})M^2+\lambda\Delta R(0))(M^2+\lambda)})] \qquad \text{(Jensen's inequality)} \\
\leq & \frac{1}{4s}[sE\,\Delta R(w^k) + 1 - \frac{\lambda^2 E\,\Delta R(w^k)}{(R(\hat{w})M^2+\lambda\Delta R(0))(M^2+\lambda)}]^2 \leq 1/s.
\end{aligned}$$

In the above, the third inequality follows from the algebraic inequality $ab \leq \frac{1}{4}(a+b)^2$. The last inequality follows from (19). This inequality implies that the theorem holds for w^{k+1}. □

We also have the following corollary:

Corollary 5.1 *Let $M = \max_i \|x_i\|_2$. Then in Algorithm 1,*

$$\Delta R(w^k) \leq \frac{R(0)(M^2+\lambda)^2}{\lambda^2 k + (M^2+\lambda)^2}.$$

Furthermore, in Algorithm 2 and Algorithm 3, the expected approximation error satisfies:

$$E\,\Delta R(w^k) \leq \frac{R(0)(M^2+\lambda)^2}{\lambda^2 k + (M^2+\lambda)^2},$$

where the expectation E is taken over all possible randomized instances.

Proof. Observe that $\Delta R(0) \leq R(0)$ and $R(\hat{w})M^2 + \lambda \Delta R(0) \leq R(0)(M^2 + \lambda)$. The corollary follows from Theorem 5.1 and simple algebra. □

For small noise problems, Theorem 5.1 can be significantly better than Corollary 5.1. In this case, we typically choose a small λ and we can assume that $R(\hat{w})/\lambda$ is small.[2] Theorem 5.1 implies an approximation bound of the order $O(1/\lambda k)$ while Corollary 5.1 implies an approximation bound of the order $O(1/\lambda^2 k)$.

One drawback of our analysis is that bounds described in Theorem 5.1 are the same for all three sparse kernel regression algorithms considered in this paper. In reality, Algorithm 1 usually gives better sparse approximation than Algorithm 3, which gives better sparse approximation than Algorithm 2. An experiment is included in Section 8 to demonstrate this phenomenon. One may also draw this conclusion from our analysis. Note that Theorem 5.1 relies on Lemma 5.2, which is an average bound for stochastic gradient update (14). However in reality, a greedy algorithm can decrease the objective function much more quickly than the average case when the variance corresponding to the average is large. Unfortunately, this difference is difficult to quantify in our theoretical analysis.

We have only considered the worst case behavior in Theorem 5.1 without any attempt to optimize the bounds. These results show that with fixed λ, the approximation error decreases at a rate of $O(1/k)$ where k is the number of non-zero coefficients in the sparse approximation. An interesting observation is that bounds in Theorem 5.1 do not depend on the sample size n.

For some simplified problems, it was shown by Xiang [Xiang, 1996] that as n increases, a sparse approximation with $k = O(n^a/\lambda^b)$ basis functions (where a and b are problem dependent constants) can convergence at the same rate to the true solution as that of using the full set of basis functions. Approximation bounds obtained in this section are complementary to such results. Although we do not have generalization error results, the approximation bounds imply the possibility of achieving a significant reduction in the number of basis functions, without reducing the accuracy of the solution by much.

Analysis given in this section has certain advantages over analysis given in Section 5 or that of [Smola and Bartlett, 2001]. In particular, the bound presented in [Smola and Bartlett, 2001] relies on some additional quantities that can be ill-behaved: in fact the bound can become trivialized when the Gram matrix $[x_i^T x_j]$ after appropriate column-normalization is nearly singular. Although the refined analysis given in Section 5 will address this problem to a certain degree, it still relies on similar quantities that are not always well-behaved. As a comparison, Theorem 5.1 does not suffer from this problem.

Bounds given in Corollary 5.1 are expressed in terms of the initial approximation $R(0)$, the maximum norm $M = \max_i \|x_i\|_2$, and the regularization parameter λ. The first two quantities,

[2] If the problem is noise-free: that is, we can find a parameter $\tilde{w}$ such that $\|\tilde{w}\|$ is small and $y = \tilde{w}^T x$, then it is easy to see that $R(\hat{w})/\lambda \leq \tilde{w}^2$ is also small.

which also appear in the matching pursuit approximation bound, are easy to understand. In addition, we show that the dependency of regularization parameter λ cannot be avoided in the worst case scenario. The following example demonstrates that if we allow $\lambda \to 0$ and $n \to \infty$, then any k-term (with k fixed) sparse approximation error can be bounded below by a positive number that is independent of k. Thus if we let $\lambda \to 0$, the best possible k-term sparse approximation error does not decrease to zero as $k \to \infty$.

We consider an orthonormal basis $\{e_i : i = 1, \ldots, d\}$ in a d-dimensional real vector space. Let $x_i = e_i$ and $y_i = 1$ for $i = 1, \ldots, n$ where $n = d$. For any $\lambda > 0$, the optimal solution $\hat{w}$ of (4) is $\hat{w} = \sum_{i=1}^{n} \frac{1}{1+\lambda n} e_i$. For any k, the best k-term approximation is given by $w^k = \sum_{j=1}^{k} \frac{1}{1+\lambda n} e_{i_j}$ where $\{i_j : j = 1, \ldots, k\}$ is any k-member subset of $\{1, \ldots, n\}$. We have $\Delta R(w^k) = R(w^k) - R(\hat{w}) = \frac{1}{2(1+\lambda n)}(1 - k/n)$. Clearly, for all k, if we allow $n \to \infty$ and $\lambda = o(1/n)$, then $\Delta R(w^k) \to 0.5$.

6 Additional performance bounds for greedy algorithms

Based on some statistical insights, it was argued in [Luo and Wahba, 1997] that a greedy algorithm acts differently than a random or stratified sample of the basis functions. A greedy algorithm may perform better than a random algorithm although bounds given in the previous section share the same form. Note that the analysis is based on a randomized average. If the variance with respect to this randomization is large, then greedy algorithms can perform much better than the average. However for technical reasons, it is difficult to estimate the variance directly and quantify the difference. In this section, we shall approach the problem from a slightly different point of view similar to that of [Natarajan, 1995]. Using the same underlying idea as that of Section 6 (but with a different stochastic gradient descent rule to focus on the high variance part of (14)), we are able to derive a bound that improves the main result in [Natarajan, 1995]. Therefore analysis in section establishes a relationship of the analysis given in Section 5 and that of [Natarajan, 1995].

For notational simplicity, we rewrite (11) as

$$R(w) = \frac{1}{2} w^T G w - w^T z + c, \tag{20}$$

where for kernel regression, G is given in Proposition 3.2, $z = \frac{1}{n}\sum_{i=1}^{n} x_i y_i$, and $c = \frac{1}{n}\sum_{i=1}^{n} y_i^2$. Our goal is to seek a sparse representation of w as

$$w = \sum_{i=1}^{n} \alpha_i x_i \tag{21}$$

to approximately minimize (20).

We would like to point out that most derivation given in this section does not require specific forms of G and z (unless explicitly mentioned). However, we do assume that G is a symmetric positive definite linear operator for simplicity (though semi-positive definiteness would have been

sufficient). This operator G induces a norm which will become useful in our analysis:

$$\|v\|_G = (v^T G v)^{1/2}.$$

It is useful to start with a relatively simple scenario and gain some insights into the performance of greedy algorithms:

Proposition 6.1 *If $x_i^T G x_j = 0$ when $i \neq j$, then each step of Algorithm 1 computes an optimal sparse solution to (20):*

$$R(w^k) = \inf_{(j_1,\alpha_1),\ldots,(j_k,\alpha_k)} R(\sum_{\ell=1}^{k} \alpha_\ell x_{j_\ell}).$$

Proof. Using the assumption $x_i^T G x_j = 0$ when $i \neq j$, it is easy to verify that

$$R(\sum_{i=1}^{n} \alpha_i x_i) = \sum_{i=1}^{m} [\frac{1}{2}\alpha_i^2 x_i^T G x_i - \alpha_i x_i^T z] + c.$$

The above is minimized at $\hat{\alpha}_i = x_i^T z/(x_i^T G x_i)$ (we will let $\hat{\alpha}_i = 0$ when $x_i^T G x_i = 0$), and it is easy to check:

$$\Delta R(\sum_{i=1}^{n} \alpha_i x_i) = \sum_{i=1}^{n} [\frac{1}{2}(\alpha_i - \hat{\alpha}_i)^2 x_i^T G x_i] \tag{22}$$

From (22) we can see that given any $w = \sum_{i=1}^n \alpha_i x_i$, if we want to minimize $\Delta R(\sum_{i=1}^n \alpha_i x_i)$ by modifying only one component α_u, then it is necessary to choose the component u with the largest value of $(\alpha_u - \hat{\alpha}_u)^2 x_u^T G x_u$, and we should change α_u to $\hat{\alpha}_u$ (unless $x_u^T G x_u = 0$).

Clearly, this implies that starting with $w^0 = 0$, we will each time select one component $\alpha_k x_{i_k}$ in Algorithm 1 with the largest $\hat{\alpha}_u^2 x_u^T G x_u$ value among all remaining components not selected so far. This implies that after k iterations we will find components $i_1, \ldots, i_k$ with the k largest values of $\hat{\alpha}_i^2 x_i^T G x_i$. Moreover, $\beta_j = 1$ and $(\alpha_j - \hat{\alpha}_{i_j})^2 x_{i_j}^T G x_{i_j} = 0$ for $1 \leq j \leq k$. This is clearly the optimal way of choosing w_k to minimize (22) among all weight vectors of form (21) with k nonzero components. □

The above result shows that the greedy algorithm works optimally when the basis vectors are G-orthogonal to each other. This is because basis vectors work independently from each other, and greedy search will result in the optimal component. For kernel regression problems considered in this paper, the orthogonality condition in Proposition 6.1 will be satisfied when $x_i^T x_j = 0$ $(i \neq j)$. Since in this case the greedy algorithm is optimal, it can perform significantly better than random sampling. In certain methods, basis functions are specifically chosen to be orthogonal (for example, Fourier or Wavelet basis). If such basis functions are used, greedy algorithms are superior to random sampling.

If the basis vectors are not G-orthogonal to each other, then the greedy algorithm may not perform optimally. However, if the basis vectors are nearly G-orthogonal, then near optimal performance can be achieved. For some technical reasons, it is necessary to modify Algorithm 1. In the following we use $\mathrm{span}\{x_{i_1}, \ldots, x_{i_k}\}$ to denote the subspace spanned by $x_{i_1}, \ldots, x_{i_k}$.

Algorithm 4 *(Subspace greedy sparse Gaussian process)*

let $w^0 = 0$
for $k = 1, 2, \ldots$
  find $i_k \in \{1, \ldots, n\}$ and $w^k \in \mathrm{span}\{x_{i_1}, \ldots, x_{i_k}\}$ to minimize $R(w^k)$
end

Clearly the analysis given in Section 5 still applies to Algorithm 4. In the following, we derive another bound for this algorithm that improves the main result of [Natarajan, 1995].

Definition 6.1 *Let V be a vector space, and consider $x_1, \ldots, x_n \in V$. Let V' be a subspace of V. Define*

$$\rho(x_1, \ldots, x_N; V') = \sup_{\alpha \in R^N, v \in V'} \frac{\sum_{i=1}^N (\alpha_i)^2 \|x_i\|_G^2}{\|\sum_{i=1}^N \alpha_i x_i + v\|_G^2}.$$

Note that $\rho \geq 1$, and the equality is achieved if we assume that $x_i^T G x_j = 0$ when $i \neq j$ and $x_i^T G v = 0$ when $v \in V'$. This means that the above quantity measures the degree of orthogonality of the underlying basis vectors.

Theorem 6.1 *Consider a permutation $(j_1, \ldots, j_n)$ of $(1, \ldots, n)$ and a number $N : 1 \leq N < n$. Let $V_1 = \mathrm{span}\{x_{j_1}, \ldots, x_{j_N}\}$ and $V_2 = \mathrm{span}\{x_{j_{N+1}}, \ldots, x_{j_n}\}$. Then for all $\bar{w} \in V_1$ and $\gamma > 0$, w^k obtained in Algorithm 4 satisfies:*

$$\Delta R(w^k) \leq \max\left((1+\gamma)^2 \Delta R(\bar{w}), \left[1 - \frac{\gamma^2}{N(2+\gamma)^2 \rho(x_{j_1}, \ldots, x_{j_N}; V_2)}\right]^k \Delta R(0)\right).$$

Proof. See Appendix A. □

From Appendix A, we can see that the proof of Theorem 6.1 is very similar to the analysis given in Section 5. The resulting bound improves the main result in [Natarajan, 1995], which if adapted to our notations, can be stated as follows. Under the assumptions of Theorem 6.1, $R(w^{\bar{k}}) \leq 4\Delta R(\bar{w})$ after at most

$$\bar{k} = \left\lceil 9N\rho(x_1, \ldots, x_n; 0) \ln \frac{4\Delta R(0)}{\Delta R(\bar{w})} \right\rceil$$

steps. As a comparison, the following bound can be obtained by setting $\gamma = 1$ in Theorem 6.1

$$\bar{k} = \left\lceil 9N\rho(x_{j_1}, \ldots, x_{j_N}; V_2) \ln \frac{4\Delta R(0)}{\Delta R(\bar{w})} \right\rceil.$$

Since

$$\rho(x_1, \ldots, x_n; 0) = \sup_{\alpha \in R^n} \frac{\sum_{i=1}^n (\alpha_i)^2 \|x_{j_i}\|_G^2}{\|\sum_{i=1}^n \alpha_i x_{j_i}\|_G^2} \geq \sup_{\alpha \in R^n} \frac{\sum_{i=1}^N (\alpha_i)^2 \|x_{j_i}\|_G^2}{\|\sum_{i=1}^n \alpha_i x_{j_i}\|_G^2} = \rho(x_{j_1}, \ldots, x_{j_N}; V_2),$$

our bound is always better. The difference can be significant when V_2 contains a large number of basis vectors (this always happens in the case of sparse approximation) that are not orthogonal.

It is worth mentioning that due to a some what different reasoning used in [Natarajan, 1995], it does not appear that their derivation can be easily refined to obtain improved results such as Theorem 6.1. We have achieved this improvement by identifying the essential element in their analysis and relating it to the randomization idea presented in Section 5. Consequently a number of major modifications are made in the approach presented in Appendix A. On one hand, this significantly simplifies their proof; on the other hand, we are also able to obtain an improved bound.

7 Relationship among different methods

Since our analysis of sparse Gaussian process algorithms is based on the stochastic gradient descent updates (14) and (24), it is closely related to online learning, which in general is based on a similar form of stochastic gradient descent update as follows:

$$w_u = w - \eta((w^T x_u - y_u)x_u + \lambda w).$$

Note that in both (14) and (24), we have used quantities that are not available to online algorithms. The usual mistake bound framework in online learning focuses on worst case upper bounds for stochastic gradient descent over a sequence of data. On the other hand, in Lemma 5.2, we consider upper bounds on the average improvement of stochastic gradient descent over the training data. This implies that we use different criteria to measure the performance of a stochastic gradient descent rule.

As we have mentioned earlier, the analysis in Section 5 is also closely related to matching pursuit type greedy approximations. However, in matching pursuit, one works with the primal variable w and seeks a sparse representation of $\hat{w}$. Techniques used in the proofs of approximation rates are closely related. Both employ the same averaging technique which results in a rate of $O(1/k)$. In matching pursuit, the averaging is with respect to a measure which is not known a priori. This means that the proof of matching pursuit is non-constructive, and one has to employ the full greedy approach to achieve the approximation rate of $O(1/k)$. The proof given in Section 5 is

constructive, and the averaging is with respect to a uniform distribution over the observed samples. As a comparison, the proof of Theorem 6.1 relies on a non-constructive randomization measure. It also implies a faster convergence rate when some additional quantities are well-behaved. From our analysis, a close relationship of [Natarajan, 1995] and [Jones, 1992] can be established.

Another related approach is to apply sparse approximation directly to the dual objective function $Q(\alpha)$. By Proposition 3.3, we know that this approach fails for noisy problems. As we have pointed out, an SVM with Vapnik's ϵ-insensitive loss slightly modifies the dual objective function $Q(\alpha)$ so that its solution may directly yield a sparse dual variable $\hat{\alpha}$. This is similar to basis pursuit which incorporates primal sparseness condition on $\hat{w}$ into the primal objective function. Our analysis implies that SVMs can only produce sparse coefficients for nearly noise-free problems. However, sparse approximation algorithms studied in this paper have provable approximation bounds both for noise-free and for noisy problems.

8 Experiments

In this section, we use two experiments to study some aspects of the proposed sparse Gaussian process algorithms as well as the corresponding theoretical bounds. Since we are mainly interested in bounds in Section 5, we will not include results on Algorithm 4. For easier control of the experimental design, we will use artificially generated data, which are sufficient for the purpose of this paper. Real data experiments for similar algorithms can be found in [Gao et al., 2001, Luo and Wahba, 1997, Nair et al., 2001, Smola and Bartlett, 2001, Xiang and Wahba, 1998].

The first experiment compares the three different sparse Gaussian process algorithms proposed in Section 4. Although in Section 5 we can only derive the same approximation bound for all three algorithms, the experiment shows that the greedy approach achieves better approximation quality in practice. We use "greedy" to denote Algorithm 1, "κ-greedy" to denote Algorithm 3, and "random" to denote Algorithm 2.

We consider a regression problem in $d = 1000$ dimension, with $n = 1000$ samples. Each input datum x is a sparse vector, where each of its components is a random number independently generated as: 0 with probability 0.99, and a uniform random number in $(0, 1)$ with probability 0.01. We also generate a random vector $\bar{w}$ with each component a uniform random number in $(0, 1)$. We then set $y_i = \hat{w}^T x_i + n_i$ where n_i is a uniform noise in $(-1, 1)$.

We set $\lambda = 0.1$, and compare the three algorithms considered in this paper. In this example, $R(\hat{w}) \approx 3.03$ and $\Delta R(0) \approx 0.90$. In Figure 1, we compare the sparse approximation performance of the algorithms, where sparseness k is the number of nonzero terms in the approximation, and approximation error is ΔR. For Algorithm 2 and Algorithm 3 that are random, the curves are obtained by averaging over 5 different random runs. In Figure 2, we plot the average (the "avg" curves), the best (the "min" curves), and the worst (the "max" curves) values of ΔR over the 5 different random runs. This result suggests that the variance caused by randomization is very

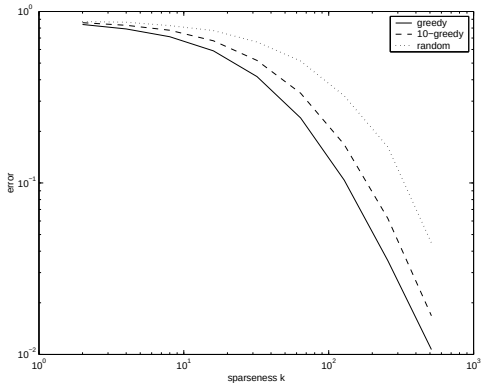


Figure 1: Approximation performance of different algorithms: ΔR as a function of sparseness k.

small.

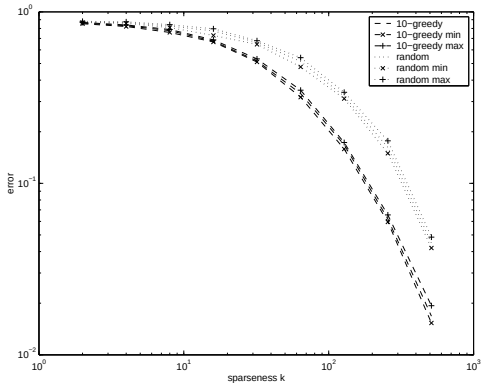
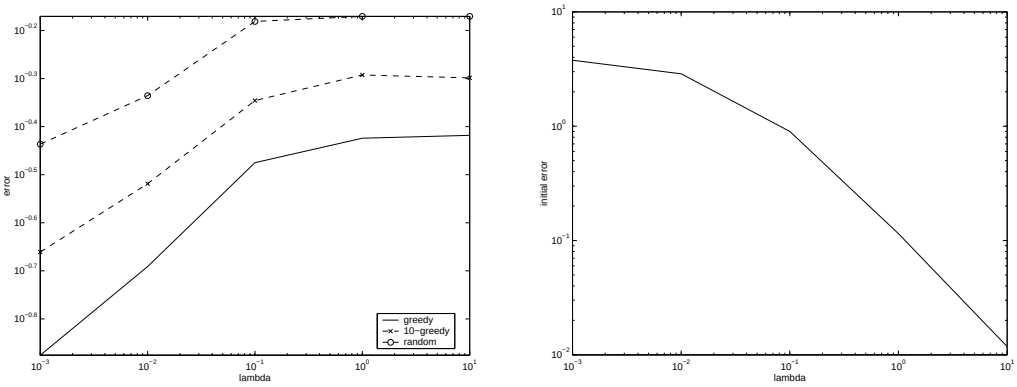


Figure 2: Variance in randomized algorithms: ΔR as a function of sparseness k.

In Figure 3, we compare the approximation performance for different values of λ. We also fix the sparseness k at $k = 50$ for illustration purpose. It is appropriate to measure the rate of approximation as $\Delta R(w^k)/\Delta R(0)$. Although the result does not show the behavior predicted by Theorem 5.1 which implies that the approximation rate degrades when $\lambda \to 0$, this does not suggest a contradiction since Theorem 5.1 only gives the worst case behavior.

To illustrate the phenomenon that the proposed sparse Gaussian process algorithms converge more slowly as $\lambda \to 0$, we consider another regression problem in $d = 500$ dimension with $n = 500$ samples. In this problem, each component of an input datum x is generated as one plus a random number uniformly distributed in $(-0.1, 0.1)$. Each output y is generated as a random number



$\Delta R(w^k)/\Delta R(0)$ $\Delta R(0)$

Figure 3: Approximation rates as functions of λ ($k = 50$).

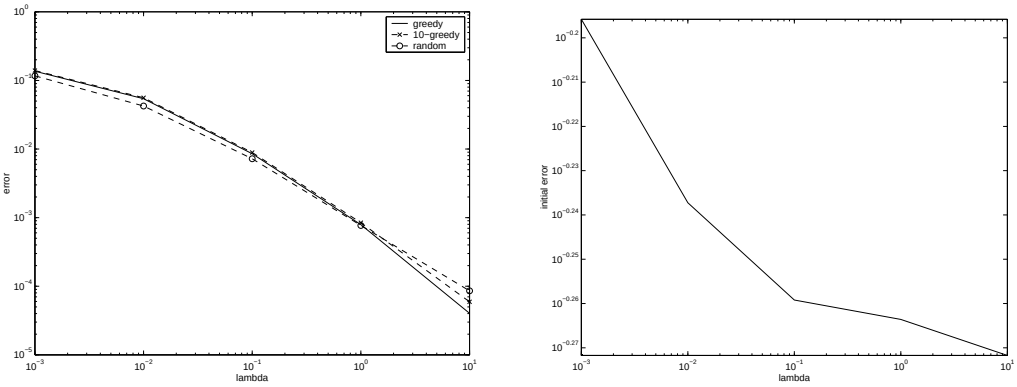
uniformly distributed in $(0, 2)$. The approximation rate of different algorithms at different λ are plotted in Figure 4. The example shows that the rate of approximation degrades at a rate of $1/\lambda$ as $\lambda \to 0$.

It is also interesting to observe that all three algorithms perform similarly in this case. This is not too surprising since data are all similar with each component has value one plus noise. Therefore random choices and greedy choices do not make much difference. The reason that the random algorithm is slightly better for small λ is due to the fact that in Algorithm 2, we use the more expensive approach of computing the exact optimal approximation in the randomly selected k-dimensional subspace. In the other two algorithms, we only optimize in a two-dimensional subspace at each step, which does not give the optimal approximation within the greedily selected k-dimensional subspace. However, it is interesting to observe that even in this case, greedy algorithms show advantages with large λ.

9 Summary

In this paper, we considered a few procedures for obtaining sparse approximations in a Gaussian process. In Section 5, we have obtained approximation error bounds of the form $O(1/k)$ where k is the number of non-zeros in the dual variable. Such a rate is typical for the randomization based analysis of [Jones, 1992]. In addition, we have also derived a bound of the form $O((1-\mu)^k)$ for a greedy algorithm, but μ relies on some additional quanitites that may be ill-behaved for certain problems. This bound improves a related result in [Natarajan, 1995].

We have also included some experimental comparisons for the proposed sparse Gaussian processes. We demonstrated that in practice, greedy search can usually give better sparse approximation than randomized sample selection. However, the difference can be small for some problems



$\Delta R(w^k)/\Delta R(0)$ $\Delta R(0)$

Figure 4: Approximation rates as functions of λ ($k = 50$).

and large for others. This trade-off between randomization and greedy search is consistent with our theory.

Although the analysis in this paper provides valuable theoretical insights into certain sparse approximation algorithms, the obtained bounds are not necessarily optimal. In particular, we do not have lower bounds that can match the derived upper bounds. It also remains an interesting issue that how tight and useful these approximation bounds are for practical kernel regression problems.

Acknowledgement

The author would like to thank anonymous referees for pointing out related works, and for constructive suggestions that helped to improve the paper.

A Proof of Theorem 6.1

The proof is similar to that of Theorem 5.1. For easier comparison, we use the same notations as those in Section 5.

Without loss of generality, we will assume that $j_\ell = \ell$ for $\ell = 1, \ldots, n$ throughout the proof. Assume that $w = w^k$ for some k in Algorithm 4. Under the assumptions of Theorem 6.1, we can decompose w as $w = v_1 + v_2$ where $v_1 \in V_1$ and $v_2 \in V_2$. We assume that v_1 is a linear combination of those basis vectors x_{i_j} found by Algorithm 4 that are in $\{x_1, \ldots, x_N\}$, and v_2 is a linear combination of those basis vectors x_{i_j} found by Algorithm 4 that are in $\{x_{N+1}, \ldots, x_n\}$. The algorithm implies that $R(v_1 + \beta v_2)$ is minimized at $\beta = 1$. Differentiate $R(v_1 + \beta v_2)$ in (20)

with respect to β, it is easy to check that we have the following first order condition

$$v_2^T(Gw - z) = 0. \tag{23}$$

Note that this equation is the only part in our analysis that differentiates Algorithm 4 from Algorithm 1. Following the same idea used in Section 5, we will use a stochastic gradient rule to derive an equality similar to (17). Observe that $v_1 - \bar{w} \in V_1$, we can thus represent it as follows:

$$v_1 - \bar{w} = \sum_{i=1}^{N} \beta_i x_i.$$

We consider the following type of stochastic gradient update rule with a datum x_u $(1 \leq u \leq N)$:

$$w_u = w - \eta \beta_u x_u, \tag{24}$$

We also replace (15) and (16) by the following quantities:

$$a_w = \sum_{i=1}^{N} \beta_u^2 x_u^T G x_u, \qquad b_w = (w - \bar{w})^T (Gw - z).$$

We start our analysis with the following lemma, which is similar to Lemma 5.1.

Lemma A.1 *Let $\eta = b_w/a_w$, then the following equality holds:*

$$\frac{1}{N} \sum_{u=1}^{N} R(w_u) = R(w) - \frac{b_w^2}{2N a_w}. \tag{25}$$

Proof. Using (24), we obtain from simple algebra that

$$R(w_u) - R(w) = \frac{\eta^2}{2} \beta_u^2 x_u^T G x_u - \eta \beta_u x_u^T (Gw - z).$$

Summing over u, we have

$$\sum_{i=1}^{N} R(w_u) - N R(w) = \frac{\eta^2}{2} \sum_{i=1}^{N} \beta_u^2 x_u^T G x_u - \eta (v_1 - \bar{w})^T (Gw - z).$$

Using (23), we have $(v_1 - \bar{w})^T (Gw - z) = b_w$. Therefore

$$\sum_{i=1}^{N} R(w_u) - N R(w) = \frac{\eta^2}{2} a_w - \eta b_w = -\frac{b_w^2}{2a_w}.$$

□

Clearly the above lemma will play the same role as that of Lemma 5.1 in Section 5. Since the averaging is only taken with respect to part of the basis $x_1, \ldots, x_N$, the analysis only applies to a greedy algorithm. Next we need to lower bound the quantity b_w.

Lemma A.2 *$\forall w, \bar{w}$ such that $\Delta R(w) \geq \Delta R(\bar{w})$, we have*

$$b_w \geq \sqrt{2}\Delta R(w)^{1/2}\|w - \bar{w}\|_G \frac{\Delta R(w)^{1/2} - \Delta R(\bar{w})^{1/2}}{\Delta R(w)^{1/2} + \Delta R(\bar{w})^{1/2}}.$$

Proof. Let $\hat{w} = G^{-1}z$, then similar to Proposition 3.2, we have

$$\Delta R(w) = \frac{1}{2}(w - \hat{w})(Gw - z) = \frac{1}{2}(w - \hat{w})G(w - \hat{w}).$$

This implies that

$$\|w - \bar{w}\|_G \leq \|w - \hat{w}\|_G + \|\bar{w} - \hat{w}\|_G = \sqrt{2}[\Delta R(w)^{1/2} + \Delta R(\bar{w})^{1/2}]. \tag{26}$$

We thus have:

$$\begin{aligned}
\frac{b_w}{2} =& \frac{1}{2}(w - \bar{w})^T(Gw - z) \\
=& \frac{1}{2}(w - \hat{w})^T(Gw - z) + \frac{1}{2}(\hat{w} - \bar{w})^T G(w - \hat{w}) \\
\geq& \Delta R(w) - \frac{1}{2}|(\hat{w} - \bar{w})^T G(w - \hat{w})| \\
\geq& \Delta R(w) - \frac{1}{2}\|\hat{w} - \bar{w}\|_G \|w - \hat{w}\|_G \\
=& \Delta R(w) - \Delta R(w)^{1/2}\Delta R(\bar{w})^{1/2} \\
\geq& \frac{1}{\sqrt{2}}\Delta R(w)^{1/2}\|w - \bar{w}\|_G \frac{\Delta R(w)^{1/2} - \Delta R(\bar{w})^{1/2}}{\Delta R(w)^{1/2} + \Delta R(\bar{w})^{1/2}}.
\end{aligned}$$

In the above, the second inequality follows from the Cauchy-Schwartz inequality, and the third inequality follows from (26). □

Proof of Theorem 6.1. From Lemma A.1 and Lemma A.2, if $\Delta R(w) \geq (1+\gamma)^2 \Delta R(\bar{w})$, then

$$\frac{1}{N}\sum_{i=1}^{N} R(w_u) - R(w) \leq -\frac{\gamma^2 \Delta R(w)\|w - \bar{w}\|_G^2}{N a_w (2+\gamma)^2} \leq -\frac{\gamma^2 \Delta R(w)}{N(2+\gamma)^2 \rho(x_1, \ldots, x_N; V_2)}.$$

This implies that if in Algorithm 4, $\Delta R(w^k) \geq (1+\gamma)^2 \Delta R(\bar{w})$, then

$$\Delta R(w^{k+1}) \leq \Delta R(w^k)\left[1 - \frac{\gamma^2}{N(2+\gamma)^2 \rho(x_1, \ldots, x_N; V_2)}\right].$$

Using induction and assuming $\Delta R(w^k) \geq (1+\gamma)^2 \Delta R(\bar{w})$, we obtain:

$$\Delta R(w^{k+1}) \leq \left[1 - \frac{\gamma^2}{N(2+\gamma)^2 \rho(x_1, \ldots, x_N; V_2)}\right]^{k+1} \Delta R(0).$$

This proves the theorem. □

References

[Barron, 1993] Barron, A. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945.

[Chen et al., 1999] Chen, S. S., Donoho, D. L., and Saunders, M. A. (1999). Atomic decomposition by basis pursuit. *SIAM J. Sci. Comput.*, 20(1):33–61.

[Csató and Opper, 2002] Csató, L. and Opper, M. (2002). Sparse on-line gaussian processes. *Neural Comp.*, 14:641–668.

[Gao et al., 2001] Gao, F., Wahba, G., Klein, R., and Klein, B. (2001). Smoothing spline ANOVA for multivariate Bernoulli observations, with applications to ophthalmology data. *J. Amer. Statist. Assoc.*, 96:127–160. with discussion.

[Girosi, 1998] Girosi, F. (1998). An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10(6):1455–1480.

[Golub and Van Loan, 1996] Golub, G. and Van Loan, C. (1996). *Matrix computations*. Johns Hopkins University Press, Baltimore, MD, third edition.

[Hoerl and Kennard, 1970] Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

[Jones, 1992] Jones, L. K. (1992). A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training. *Ann. Statist.*, 20(1):608–613.

[Lin et al., 2000] Lin, X., Wahba, G., Xiang, D., Gao, F., Klein, R., and Klein, B. (2000). Smoothing spline ANOVA models for large data sets with Bernoulli observations and the randomized GACV. *Ann. Statist.*, 28:1570–1600.

[Luo and Wahba, 1997] Luo, Z. and Wahba, G. (1997). Hybrid adaptive splines. *J. Amer. Statist. Assoc.*, 92:107–114.

[Mallat and Zhang, 1993] Mallat, S. and Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415.

[Mercer, 1909] Mercer, J. (1909). Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, A 209:415–446.

[Nair et al., 2001] Nair, P., Choudhury, A., and Keane, A. (2001). Some greedy algorithms for sparse nonlinear regression. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 369–376.

[Natarajan, 1995] Natarajan, B. K. (1995). Sparse approximate solutions to linear systems. *SIAM J. Comput.*, 24(2):227–234.

[Smola and Bartlett, 2001] Smola, A. J. and Bartlett, P. (2001). Sparse greedy Gaussian process regression. In *Advances in Neural Information Processing Systems 13*, pages 619–625.

[Vapnik, 1998] Vapnik, V. (1998). *Statistical learning theory.* John Wiley & Sons, New York.

[Wahba, 1990] Wahba, G. (1990). *Spline Models for Observational Data.* CBMS-NSF Regional Conference series in applied mathematics. SIAM.

[Williams and Rasmussen, 1996] Williams, C. and Rasmussen, C. (1996). Gaussian processes for regression. In *Advances in Neural Information Processing Systems 8.*

[Williams and Seeger, 2000] Williams, C. and Seeger, M. (2000). The effect of the input density distribution on kernel-based classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning.* Morgan Kaufmann.

[Williams, 1998] Williams, C. K. (1998). Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In Jordan, M., editor, *Learning and Inference in Graphical Models.* Kluwer.

[Xiang, 1996] Xiang, D. (1996). *Model Fitting and Testing for Non-Gaussian Data with a Large Data Set.* PhD thesis, University of Wisconsin.

[Xiang and Wahba, 1998] Xiang, D. and Wahba, G. (1998). Approximate smoothing spline methods for large data sets in the binary case. In *Proceedings of the 1997 ASA Joint Statistical Meetings, Biometrics Section*, pages 94–98.