Journal of Machine Learning Research x (2009) x-xx Submitted 5/08; Published xx/09

Stable and Efficient Gaussian Process Calculations

Leslie Foster FOSTER@MATH.SJSU.EDU
Alex Waagen AWAAGEN@MAILBOLT.COM
Nabeela Aijaz NABBO_A@YAHOO.COM
Michael Hurley MHURLEY@GMAIL.COM
Apolonio Luis JPOLOROLU@GMAIL.COM
Joel Rinsky JOEL_RINSKY@YAHOO.COM
Chandrika Satyavolu CHANDRIKA_S84@YAHOO.COM
Department of Mathematics
San Jose State University
San Jose, CA 95192, USA

Michael J. Way MICHAEL.J.WAY@NASA.GOV
NASA Goddard Institute for Space Studies
New York, NY, 10025, USA

Paul Gazis PGAZIS@MAIL.ARC.NASA.GOV
Ashok Srivastava ASHOK@EMAIL.ARC.NASA.GOV
NASA Ames Research Center
Intelligent Systems Division, MS 269-4
Moffett Field, CA 94035, USA

Editor:

Abstract

The use of Gaussian processes can be an effective approach to prediction in a supervised learning environment. For large data sets, the standard Gaussian process approach requires solving very large systems of linear equations and approximations are required for the calculations to be practical. We will focus on the subset of regressors approximation technique. We will demonstrate that there can be numerical instabilities in a well known implementation of the technique. We discuss alternate implementations that have better numerical stability properties and can lead to better predictions. Our results will be illustrated by looking at an application involving prediction of galaxy redshift from broadband spectrum data.

Keywords: Gaussian Processes, Low Rank Approximations, Numerical Stability, Photometric Redshift, Subset of Regressors Method

1. Introduction

The use of Gaussian processes can be an effective approach to prediction in a supervised learning environment. For large data sets, the standard Gaussian process approach requires solving very large systems of linear equations and approximations are required for the calculations to be practical. We will focus on the subset of regressors technique which involves low rank approximations. The goal of the paper is to describe techniques that are fast – requiring $O(nm^2)$ operations where n is the number of data points available for

training and m is the rank of a low rank approximation – and have good numerical stability properties in the sense that the growth of computer arithmetic errors is limited.

The paper begins with a review of Gaussian processes and the subset of regressors approach. We then show that implementation of the subset of regressors method using normal equations can be inaccurate due to computer arithmetic errors. A key contribution of the paper is a discussion of alternative implementations of the subset of regressors technique that have improved numerical stability. Another valuable contribution of the paper is a discussion of how pivoting can be incorporated in the subset of regressors approach to further enhance numerical stability. We discuss the algorithm of Lucas (Lucas, 2004, pp. 4-5) for construction of a partial Cholesky factorization with pivoting and emphasize that with this algorithm the flop count, including subset selection, of the subset of regressors calculations is $O(nm^2)$.

In Section 2 we provide background about using Gaussian processes to facilitate prediction. In Section 3 we discuss how low rank approximations lead to the subset of regressors approach. In Section 4 we describe why a commonly used implementation of this technique may suffer from numerical instabilities and in Section 5 we propose two alternative implementations that have better numerical stability properties. In Section 6 we address the subset selection problem and indicate that a solution to this problem can enhance numerical stability. In Section 7 we discuss tools that aid in the choice of rank in the low rank approximation. In Section 8 we illustrate that the numerical stability issues addressed in Section 4 can lead to unacceptably large growth of computer arithmetic errors in an important application involving prediction of galaxy redshift from broadband spectrum data. Our alternative implementations of the subset of regressors method overcome these difficulties. Also in Section 8 we discuss code, available at dashlink.arc.nasa.gov/algorithm/stableGP, that implements our ideas. Finally, in Section 9 we summarize our results.

2. Gaussian Processes

Supervised learning is the problem of learning input-output mappings using empirical data. We will assume that a training dataset is known consisting of a $n \times d$ matrix X of input measurements and a n by 1 vector y of output or target values. The task is to use the training dataset to develop a model that can be used to make prediction with new data. We will assume the new data, called the testing data, is contained in an $n^* \times d$ matrix X^* of inputs. The $n^* \times 1$ vector y^* will represent the target values corresponding to X^*. The goal is to predict the value of y^* given X, y, and X^*.

In the Gaussian process approach the prediction of y^* involves selection of a covariance function $k(x, x')$, where x and x' are vectors with d components. It is required that the covariance function be positive semidefinite (Rasmussen and Williams, 2006, p. 80) which implies that the $n \times n$ covariance matrix K with entries $K_{ij} = k(x_i, x_j)$ where x_i and x_j are rows of X is symmetric positive semidefinite (SPS), so that $v^T K v \geq 0$ for any $n \times 1$ real column vector v. The covariance function can be used to construct K and also the n^* by n cross covariance matrix K^* where $K^*_{ij} = k(x^*_i, x_j)$ where x^*_i is the i^{th} row of X^*. The prediction $\widehat{y}^*$ for y^* is given by the Gaussian processes equation (Rasmussen and Williams, 2006, p. 17):

$$\widehat{y}^* = K^*(\lambda^2 I + K)^{-1} y \tag{1}$$

The parameter λ in this equation represents the noise in the measurements of y and, in practice, it is often selected to improve the quality of the model (Rasmussen and Williams, 2006).

It is often not clear how to choose the covariance function k. There exist many different covariance functions that apply broadly to many cases. Potential covariance function choices include the squared exponential (sometimes called the radial basis function), Matern, rational quadratic, neural network, polynomial or other covariance functions (Rasmussen and Williams, 2006, pp. 79-102). Most of these covariance functions contain free parameters that need to be selected. Such parameters and λ in (1) are called hyperparameters. We will not focus on the choice of a covariance function or alternative methods for selection of hyperparameters in this paper. In the examples discussed in Section 8 we tried out a variety of covariance functions and selected the one that provided the best predictions. Hyperparameters were selected using the Matlab routine minimize from (Rasmussen and Williams, 2006, pp. 112-116, 221) which finds a (local) maximum of the marginal likelihood function calculated using the training set data.

We should mention that the choice of the hyperparameter λ can affect the numerical stability of the Gaussian process calculations. Generally larger values of λ lead to reduced computer arithmetic errors but a large value of λ may be a poor theoretical choice – note that $\widehat{y}^* \to 0$ as $\lambda \to \infty$. One needs to select a value of λ that balances such competing errors. The choice of λ in Gaussian processes is closely related to the parameter choice in ridge regression in the statistics literature (Montgomery et al., 2006, pp. 344-355) and in the literature on regularization (Hansen, 1998, pp. 175-208). As mentioned above we select hyperparameters, including λ, using the routine minimize from (Rasmussen and Williams, 2006, pp. 112-116, 221). This technique worked well for the practical example presented in Section 8 when used with our algorithms with improved numerical stability.

We should note that Gaussian process approach also leads to an equation for C the covariance matrix for the predictions in (1). If the $n^* \times n^*$ matrix K^{**} has entries $K^*_{ij} = k(x^*_i, x^*_j)$ then (Rasmussen and Williams, 2006, pp. 79-102):

$$C = K^{**} - K^*(\lambda I + K)^{-1} K^{*T}. \tag{2}$$

The superscript T indicates transpose. The pointwise variance of the predictions is $\mathrm{diag}(C)$, the diagonal of the $n^* \times n^*$ matrix C.

3. Low Rank Approximation: the Subset of Regressors Method

In (1) the matrix $(\lambda^2 I + K)$ is an n by n matrix that, in general, is dense (that is has few zero entries). Therefore for large n, for example $n \geq 10000$, it is not practical to solve (1) since the memory required to store K is $O(n^2)$ and the number of floating point operations required to solve (1) is $O(n^3)$. Therefore for large n it is useful to develop approximate solutions to (1).

To do this, for some $m < n$, we can partition the matrices K and K^* as follows:

$$K = \begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{pmatrix} = \begin{pmatrix} K_1 & K_2 \end{pmatrix}, \; K^* = \begin{pmatrix} K^*_1 & K^*_2 \end{pmatrix} \tag{3}$$

Here K_{11} is $m \times m$, K_{21} is $(n-m) \times m$, $K_{12} = K_{21}^T$ is $m \times (n-m)$, K_{22} is $(n-m) \times (n-m)$, K_1 is $n \times m$, K_2 is $n \times (n-m)$, K_1^* is $n^* \times m$ and K_2^* is $n^* \times (n-m)$. Next we approximate K and K^* using

$$K \cong \widehat{K} \equiv K_1 K_{11}^{-1} K_1^T \tag{4}$$

and

$$K^* \cong \widehat{K}^* \equiv K_1^* K_{11}^{-1} K_1^T \tag{5}$$

and in (1) we replace K with $\widehat{K}$ and K^* with $\widehat{K}^*$. Therefore

$$\widehat{y}^* \cong \widehat{y}_N^* \equiv \widehat{K}^* (\lambda^2 I + \widehat{K})^{-1} y =$$

$$K_1^* K_{11}^{-1} K_1^T (\lambda^2 I + K_1 K_{11}^{-1} K_1^T)^{-1} y =$$

$$K_1^* K_{11}^{-1} (\lambda^2 I + K_1^T K_1 K_{11}^{-1})^{-1} K_1^T y, \text{ so that}$$

$$\widehat{y}_N^* = K_1^* (\lambda^2 K_{11} + K_1^T K_1)^{-1} K_1^T y. \tag{6}$$

Equation (6) is called the subset of regressors method (Rasmussen and Williams, 2006, p. 176) and was proposed, for example, in (Wahba, 1990, p. 98) and (Poggio and Girosi, 1990, p. 1489). As we discuss in the next section the subscript N stands for normal equations. We refer to use of (6) as the SR-N approach.

If $m << n$ then (6) is substantially more efficient than (1). For large n the leading order term in the operation count for the calculations in (6) is nm^2 flops or floating point operations (where a floating point operation is either an addition, subtraction, multiplication or division), whereas the calculations in (1) require approximately $2n^3/3$ flops. If $n = 180,000$ and $m = 500$, as in an example discussed later, the solution to (1) requires approximately 4×10^{15} flops which is five order of magnitudes greater than the approximately 4×10^{10} flops required to solve (6). Furthermore, to use (1) one needs to calculate all $n^2 + nn^*$ elements of K and K^* whereas (6) requires that one calculate only the $nm + n^*m$ elements in K_1 and K_1^*. This also improves the efficiency of the calculations and will reduce the memory requirements dramatically.

We should add that if in equation (2) we use the approximations (4), (3) and

$$K^{**} \cong \widehat{K}^{**} \equiv K_1^* K_{11}^{-1} K_1^{*T} \tag{7}$$

then, in (2) replacing K with $\widehat{K}$, K^* with $\widehat{K}^*$, K^{**} with $\widehat{K}^{**}$ and using algebra similar to that used in deriving (6), it follows that

$$C \cong \widehat{C}_N \equiv \lambda^2 K_1^* (\lambda^2 K_{11} + K_1^T K_1)^{-1} K_1^{*T}. \tag{8}$$

For an alternate derivation of (8) see (Rasmussen and Williams, 2006, p. 176). Also $\text{diag}(C) \cong \text{diag}(\widehat{C}_N)$ so that $\text{diag}(\widehat{C}_N)$ provides approximations for the variance of the predictions.

4. Numerical Instability

The sensitivity of a problem measures the growth of errors in the answer to the problem relative to perturbations in the initial data to the problem, assuming that that there are no errors in the solution other than the errors in the initial data. A particular algorithm

implementing a solution to the problem is numerically stable if the error in the answer calculated by the algorithm using finite precision arithmetic is closely related (a modest multiple of) the error predicted by the sensitivity of the problem. An algorithm is unstable if the error in the answer calculated by the algorithm is substantially greater than the error predicted by the sensitivity of the problem.

A straightforward implementation for the subset of regressors approximation using (6) has a potential numerical instability. To see this note that since K is SPS it follows that the $m \times m$ submatrix K_{11} is also. Therefore we can factor the matrix K_{11} with a Cholesky factorization (Golub and Van Loan, 1996, p. 148)

$$K_{11} = V_{11} V_{11}^T \tag{9}$$

where V_{11} is an $m \times m$ lower triangular matrix. Now let

$$A = \begin{pmatrix} K_1 \\ \lambda V_{11}^T \end{pmatrix} \text{ and } b = \begin{pmatrix} y \\ 0 \end{pmatrix}, \tag{10}$$

where 0 is an $m \times 1$ zero vector, A is an $(n+m) \times m$ matrix and b is a $(n+m) \times 1$ vector. Consider the least square problem:

$$\min_x ||Ax - b|| \tag{11}$$

where the norm is the usual Euclidean norm. The normal equations solution (Golub and Van Loan, 1996, p. 237) to this least squares problem is $x = (A^T A)^{-1} A^T b = (\lambda^2 V_{11} V_{11}^T + K_1^T K_1)^{-1} K_1^T y$ and so by (9)

$$x_N = (\lambda^2 K_{11} + K_1^T K_1)^{-1} K_1^T y. \tag{12}$$

Therefore the solution $\widehat{y}_N^*$ presented in (6) can also be written

$$\widehat{y}_N^* = K_1^* x_N. \tag{13}$$

The subscript N indicates the use of the normal equations solution to (10).

The potential difficulty with the above solution is that the intermediate result x_N is the solution to a least squares problem using the normal equation. It is well known that the use of the normal equation can, in some cases, introduce numerical instabilities and can be less accurate than alternative approaches. As discussed in (Golub and Van Loan, 1996, p. 236-245) the sensitivity of the least squares problem (11) is roughly proportional to $\text{cond}(A) + \rho_{LS}\,\text{cond}^2(A)$, where $\rho_{LS} = ||b - Ax||$ and $\text{cond}(A) = ||A||||(A^T A)^{-1} A^T||$ is the condition number of A. The problem with the normal equations solution to (11) is that the accuracy of the calculated solution is (almost always) proportional to $\text{cond}^2(A)$, the square of the condition number of A, whereas in the case that ρ_{LS} is small the sensitivity of the least squares problem is approximately $\text{cond}(A)$. To quote from (Golub and Van Loan, 1996, p. 245):

> We may conclude that if ρ_{LS} is small and $\text{cond}(A)$ is large, then the method of normal equations ... will usually render a least squares solution that is less accurate than a stable QR approach.

We will discuss use of the stable QR approach and another alternative to the normal equations in the next section.

5. Improving Numerical Stability

The calculation of $\widehat{y}_N^*$ as given by (6) is equivalent to the solution to (11) and (13) using the normal equations (12). We can reduce the computer arithmetic errors in the calculation of $\widehat{y}_N^*$ if we develop algorithms that avoid the use of the normal equations in the solution to (11) and (13) . We will present two alternative algorithms for solving (11) and (13). We should add that although these algorithms can have better numerical properties than use of (6), all the algorithms presented in this section are mathematically (in exact arithmetic) equivalent to (6).

5.1 The Subset of Regressors using a QR factorization

We first describe use of the QR factorization to solve (11). In this approach [Golub and Van Loan, 1996, p. 239] one first factors $A = QR$ where Q is an $(n+m) \times m$ matrix with orthonormal columns and R is an $m \times m$ right triangular matrix. Then

$$x_Q = R^{-1}Q^T b = R^{-1}Q^T \begin{pmatrix} y \\ 0 \end{pmatrix} \tag{14}$$

so that

$$\widehat{y}_Q^* = K_1^* x_Q = K_1^* R^{-1} Q^T \begin{pmatrix} y \\ 0 \end{pmatrix}. \tag{15}$$

With the above algorithm $\widehat{y}^*$ can still be solved quickly. Assuming that the elements of K_1 and K_1^* have been calculated, and that $m << n$, then the approximate number of operations for the QR approach is $2nm^2$ flops. Therefore both the QR and normal equations approach require $O(nm^2)$ flops.

We should also note that we can use the QR factorization to reduce computer arithmetic errors in the computation of the approximate covariance matrix in (8). If we let

$$\widehat{C}_{QR} \equiv \lambda^2 (K_1^* R^{-1})(K_1^* R^{-1})^T \tag{16}$$

then mathematically (in exact arithmetic) $\widehat{C}_N$ and $\widehat{C}_{QR}$ are the same. However, for reasons similar to those discussed in Section 4, the computer arithmetic errors in $\widehat{C}_{QR}$ will usually be smaller than those in $\widehat{C}_N$, assuming, for example, that $\widehat{C}_N$ is computed using a Cholesky factorization of $\lambda^2 K_{11} + K_1^T K_1$.

We will refer to the subset of regressors method using the QR factorization as the SR-Q method. We should add that the use of a QR factorization in equations related to Gaussian process calculations is not new. For example (Wahba, 1990, p. 136) discusses using a QR factorization for cross validation calculations.

5.2 The V method

If we assume the V_{11} is nonsingular we can define the $n \times m$ matrix V:

$$V = K_1 V_{11}^{-T} \tag{17}$$

where the superscript $-T$ indicates inverse transpose. Note that by (9) it follows that V is lower trapezoidal and that $V = \begin{pmatrix} V_{11} \\ V_{21} \end{pmatrix}$, where $V_{21} = K_{21} V_{11}^{-T}$. Substituting $K_1 = V V_{11}^T$

and (9) into (12) we get

$$x_V = V_{11}^{-T}(\lambda^2 I + V^T V)^{-1} V^T y \tag{18}$$

so that

$$\widehat{y}_V^* = K_1^* x_V = K_1^* V_{11}^{-T}(\lambda^2 I + V^T V)^{-1} V^T y \tag{19}$$

We should note that this formulation of the subset of regressors method is not new. It is presented, for example, in (Seeger et al., 2003) and (Wahba, 1990, p. 136) presents a formula closely related to (19). We will call the formula (19) for $\widehat{y}^*$ the V method. We should note, as will be seen in Section 6.2, that one can calculate V as part of a partial Cholesky factorization rather than using (17).

We will see in our numerical experiments and the theoretical analysis in Section 6 that the V method is intermediate in terms of growth of computer arithmetic errors between the normal equations and QR approach. Often, but not always, the accuracy of the V method is close to that of the QR approach.

Assuming that the elements of K_1 and K_1^* have been calculated, and that $m << n$, then the approximate number of operations for the V method is $2nm^2$ flops – approximately nm^2 flops to form V and another nm^2 flops to solve for x_V using (18). This is approximately the same as SR-Q and approximately twice the flop count for the SR-N method.

We can also compute the approximate covariance matrix with the V method approach:

$$\widehat{C}_V \equiv \lambda^2 K_1^* V_{11}^{-T}(\lambda^2 I + V^T V)^{-1} V_{11}^{-1} K_1^{*T}. \tag{20}$$

In exact arithmetic $\widehat{C}_N$, $\widehat{C}_{QR}$ and $\widehat{C}_V$ are identical but the computer arithmetic errors are often smaller in $\widehat{C}_V$ and $\widehat{C}_{QR}$ than $\widehat{C}_N$.

5.3 Examples illustrating stability results

We present two sets of examples that illustrate some of the above remarks.

Example 1 *Let the $n \times n$ matrices K be of the form $K = UDU^T$ where U is a random orthogonal matrix (Stewart, 1980) and D is a diagonal matrix with diagonal entries $s_1 \geq s_2 \geq \ldots s_n \geq 0$. Therefore s_1, s_2, ..., s_n are the singular values of K. We will choose a vector $w \in R^n$, where R^n is real n dimensional space, of the form $w = \begin{pmatrix} x \\ 0 \end{pmatrix}$ where $x \in R^m$ is a random vector and 0 indicates a zero vector with $(n-m)$ components. We let the target data be $y = Kw$. We will also assume for simplicity that $\lambda = 0$.*

Due to the structure of w each of x_N (12), x_Q (14), and x_V (18) will calculate x exactly in exact arithmetic. Therefore in finite precision arithmetic $||x - \dot{x}||$, with $\dot{x} = x_N$, x_Q or x_V will be a measure of the computer arithmetic errors in the calculation.

We carried out an experiment $n = 100$, $m = 50$, $s_i = 10^{-(i-1)/5}$, $i = 1, 2, \ldots, m$, and $s_i = 10^{-10}$, $i = m+1, m+2, \ldots, n$ using a set of one hundred random matrices of this type. For this class of matrices the singular values of K vary between 1 and 10^{-10}, cond$(K) = 10^{10}$ and cond$(K_1) \cong 10^{10}$. The results are:

$\dot{x}$ =	x_N	x_V	x_Q
min $\|\|x-\dot{x}\|\|/\|\|x\|\|$	9.3×10^{-1}	5.1×10^{-7}	2.7×10^{-8}
mean $\|\|x-\dot{x}\|\|/\|\|x\|\|$	9.1×10^{0}	3.6×10^{-6}	1.2×10^{-7}
max $\|\|x-\dot{x}\|\|/\|\|x\|\|$	9.6×10^{1}	9.9×10^{-6}	4.5×10^{-7}

Table 1: Min, mean and max errors, $||x-\dot{x}||/||x||$, for 100 matrices and various methods.

For this set of matrices x_Q and x_V have small errors. However x_N has large errors due to its use of normal equations.

Example 2 *This example will illustrate that, although the V method often greatly improves upon the stability of the SR-N method, this is not always the case. For $0 < s \leq 1$ let $C = \begin{pmatrix} s^2 & 10s \\ 10s & 200 \end{pmatrix}$, let the 4×4 matrix $K = \begin{pmatrix} s^2C & 10sC \\ 10sC & 200C \end{pmatrix}$, let $x = \begin{pmatrix} 1/3 \\ 1/3 \end{pmatrix}$, $w = \begin{pmatrix} x \\ 0 \\ 0 \end{pmatrix}$, $\lambda = 0$ let $y = Kw$.*

Due to the structure of w we again have each of x_N, x_Q and x_V will calculate x exactly in exact arithmetic. However, in finite precision arithmetic the calculated values will not be exact. For this example for small s the errors in both x_N and x_V can be significantly larger than the errors in x_Q. For example if $s = 10^{-4}$ we get the following results:

$\dot{x}$ =	x_N	x_V	x_Q
$\|\|x-\dot{x}\|\|/\|\|x\|\|$	8.8×10^{-1}	2.1×10^{-1}	7.7×10^{-11}

Table 2: Errors $||x-\dot{x}||/||x||$ for a 4×4 matrices and various methods.

In Section 6 and Appendix A we will discuss the reason that the V method performs poorly in this example and show that the numerical instability illustrated in this example can be cured by interchanging the columns and rows of K appropriately. Also we should note that although difficulties like the one illustrated here are possible for the V method, experiments like those in Example 1 suggest that such difficulties are not likely. As we discuss in Section 6, the method performed well when we applied it to real world applications.

6. Pivoting and Subset Selection

In Section 5 we discussed low rank approximations to K which involved the first m columns of K. However one can select any subset of the columns to construct a low rank approximation. The choice of these columns or the "active" set is the subset selection problem. This problem has been addressed by, for example, (Smola and Bartlett, 2001), (Seeger et al., 2003), (Csato and Opper, 2002) and (Fine and Scheinberg, 2001). The technique that we will use is the same as that in (Fine and Scheinberg, 2001). However we will focus on the effect of the resulting choice of the active set on the numerical stability of the resulting algorithm. This is a different motivation than the motivations in the above references.

6.1 The Singular Value Decomposition

To pursue this we will first discuss the singular value decomposition which, in a certain sense, produces an optimal low rank approximation to K. The singular value decomposition (SVD) of the symmetric semidefinite matrix K produces the factorization

$$K = UDU^T = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix} \begin{pmatrix} U_1 & U_2 \end{pmatrix}^T \tag{21}$$

where U is an $n \times n$ orthogonal matrix, D is an $n \times n$ diagonal matrix whose diagonal entries $s_1 \geq s_2 \geq \ldots \geq s_n \geq 0$ are the singular values of K, U_1 is $n \times m$, U_2 is $n \times (n-m)$, D_1 is an $n \times n$ diagonal matrix, and D_2 is an $(n-m) \times (n-m)$ diagonal matrix. We then can construct the truncated singular value decomposition (TSVD) low rank approximation to K:

$$\widehat{K}_{SVD} = U_1 D_1 U_1^T. \tag{22}$$

The TSVD approximation $\widehat{K}_{SVD}$ is the best low rank approximation (Golub and Van Loan, 1996, p. 72) to K in the sense that

$$\min_{\text{rank}(\widehat{K})=m} ||K - \widehat{K}|| = ||K - \widehat{K}_{SVD}|| = s_{m+1}. \tag{23}$$

Given an $n \times q$ matrix A with rank $m \leq \min(n, q)$ we will define (Björck, 1996, p. 28) the condition number of A to be $\text{cond}(A) = s_1/s_m$ where s_1 and s_m are singular values of A. This definition generalizes to singular matrices the definition of condition number that we used in Section 4 (where A had m columns). It then follows from (22) that

$$\text{cond}(\widehat{K}_{SVD}) = s_1/s_m \tag{24}$$

where s_1 and s_m are singular values of K (which are the same as the singular values of $\widehat{K}_{SVD}$). Thus the singular value decomposition provides two desirable properties:

- equation (23) indicates that $\widehat{K}_{SVD}$ will be close to K, if there exists a rank m approximation that is close to K, and
- equation (24) limits the condition number of $\widehat{K}_{SVD}$ which will limit the growth of computer arithmetic errors in the use of $\widehat{K}_{SVD}$.

However, for large n, it is not practical to calculate the SVD of K since the SVD requires $O(n^3)$ operations and is much more expensive than the algorithms described in Section 5 which require $O(nm^2)$ operations. We would like to construct an approximation that requires only $O(nm^2)$ operations and that produces low rank approximations with properties related to (23) and (24).

6.2 Cholesky factorization with pivoting

The algorithms describe in Sections 3 and 5 (which are mathematically but not numerically identical) do not satisfy relations related to (23) and (24) as is apparent from the following example.

Example 3 *For the matrix*

$$K = \begin{pmatrix} 1+\epsilon & 1-\epsilon & 0 \\ 1-\epsilon & 1+\epsilon & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{25}$$

if we let $m = 2$ *then by (4) and (22) we have*

$$\widehat{K} = \begin{pmatrix} 1+\epsilon & 1-\epsilon & 0 \\ 1-\epsilon & 1+\epsilon & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad and\ \widehat{K}_{SVD} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{26}$$

so that, for small ϵ*,*

$$||K - \widehat{K}_{SVD}|| = 2\epsilon \ << \ 1 = ||K - \widehat{K}||\ and \tag{27}$$

$$cond(\widehat{K}_{SVD}) = 2 \ << \ 1/\epsilon = cond(\widehat{K}) \tag{28}$$

For this example the low rank approximation $\widehat{K}$ *has two problems: (1) it does not provide a good approximation to* K *even though a good low rank approximation exists and (2) the condition number of* $\widehat{K}$ *can be arbitrarily large which potentially could lead to a large growth of computer arithmetic errors.*

To overcome the difficulties illustrated in this example we can use a Cholesky factorization, with pivoting, to insure that linearly independent columns and rows appear first. The Cholesky factorization with pivoting produces a decompostion

$$P^T K P = L L^T \tag{29}$$

where P is an $n \times n$ permutation matrix and L is an $n \times n$ lower triangular matrix. To produce our low rank approximations to the Gaussian process equations we do not need to factor all of K, rather it is sufficient to calculate a partial factorization that factors only m columns and rows of $P^T K P$. This is a partial Cholesky factorization with pivoting. If the pivoting is done using complete pivoting (that is the pivoting in the Cholesky factorization is equivalent to using complete pivoting in Gaussian elimination) then there are a variety of algorithms that determine the factorization (Higham, 2002, p. 202), (Golub and Van Loan, 1996, p. 149), (Lucas, 2004, pp. 4-5), (Fine and Scheinberg, 2001, p. 255). Here we will summarize the algorithm presented in (Lucas, 2004, pp. 4-5) since it is not as widely known as the algorithms in (Higham, 2002, p. 202), (Golub and Van Loan, 1996, p. 149) and is more efficient in our context. The algorithm below is also the same as that in (Fine and Scheinberg, 2001, p. 255) except for the stopping criteria.

Data: an $n \times n$ symmetric positive semidefinite matrix K
a stopping tolerance $tol \geq 0$
the maximum rank, $max\_rank \leq n$, of the low rank approximation
Result: m, the rank of the low rank approximation
an $n \times m$ partial Cholesky factor V
a permutation vector piv
Note: on completion the first m rows and columns of $P^T K P - V V^T$ are zero, where P is a permutation matrix with $P_{piv_i, i} = 1$, $i = 1, \ldots, n$
initialize:
$d_i = K_{ii},\ i = 1, \ldots, n$
$K_{\max} = \max_{i=1,\ldots,n} (d_i)$
$piv_i = i,\ i = 1, \ldots, n$
$m = max\_rank$
for $j = 1$ *to* max_rank **do**
  $[d_{\max}, j_{\max}] = \max_{i=j,\ldots,n} (d_i)$
  where $j_{\max}$ is an index where the max is achieved
  if $d_{\max} \leq (tol) K_{\max}$ **then**
    $m = j - 1$;
    exit the algorithm ;
  end
  if $j_{\max} \neq j$ **then**
    switch elements j and $j_{\max}$ of piv and d
    for $i = j + 1 : n$ let $u_i =$ element i of column $j_{\max}$ of $P^T K P$
    switch rows j and $j_{\max}$ of the current $n \times (j-1)$ matrix V
  end
  $V_{jj} = \sqrt{d_{\max}}$
  for $i = j + 1$ *to* n **do**
    $V_{ij} = (u_i - \sum_{k=1}^{j-1} V_{ik} V_{jk}) \,/\, V_{jj}$
    $d_i = d_i - V_{ij}^2$
  end
end

Algorithm 1: Algorithm for the partial Cholesky factorization

There are two choices of the stopping tolerance tol that have been suggested elsewhere. For the choice $tol = 0$ the algorithm will continue as long as the factorization determines that K is positive definite (numerically). This choice of tol is used in LINPACK's routine xCHDC (Dongarra et al., 1979) and also by Matlab's Cholesky factorization chol (which implements a Cholesky factorization without pivoting). The choice $tol = n \times \epsilon$ where ϵ is machine precision is suggested in (Lucas, 2004, p. 5) and in (Higham, 2002). The best choice of tol will depend on the application.

There are a number of attractive properties of the partial Cholesky factorization.

- The number of floating point operations in the algorithm is approximately $nm^2 - 2m^3/3$ flops. The calculations to determine the pivoting require only $O(nm)$ flops.
- The algorithm accesses only the diagonal entries of K and elements from m columns of K.

- The storage requirement for the algorithm is approximately $n(m+2)$ floating point numbers plus storage for the integer vector *piv* and any storage needed to calculate entries in K.

- The accuracy and condition number of the low rank approximation to K produced by the algorithm is related to the accuracy and condition number of the low rank approximation produced by the singular value decomposition. In particular

Theorem 1 *Let the $n \times m$ matrix V be the partial Cholesky factor produced by Algorithm 1 and let*

$$\widehat{K}_P = PVV^T P^T. \tag{30}$$

Also let $\widehat{K}_{SVD}$ be the rank m approximation (22) produced by the singular value decomposition. Then

$$||K - \widehat{K}_P|| \leq c_1 \, ||K - \widehat{K}_{SVD}|| \text{ and} \tag{31}$$

$$\text{cond}(\widehat{K}_P) \leq c_2 \, \text{cond}(\widehat{K}_{SVD}) \text{ where} \tag{32}$$

$$c_1 \leq (n-m)4^m \text{ and } c_2 \leq (n-m)4^m. \tag{33}$$

Proof *The theorem follows from results in (Gu and Eisenstat, 1996) for the QR factorization with pivoting. First we consider a Cholesky factorization, without pivoting, of K so that $K = LL^T$ where L is and $n \times n$ lower triangular matrix. Let $\sigma_i(A)$ represent the i^{th} singular value of a matrix A. Then, making use of the singular value decomposition, it follows easily that $\sigma_i(K) = \sigma_i^2(L)$, $i = 1, \ldots, n$. Consider a QR factorization of L^T with standard column pivoting (Golub and Van Loan, 1996, p. 249-250) so that $QR = L^T P_1$. The permutation matrix P_1 produced by this QR factorization will be identical, in exact arithmetic, to the permutation matrix produced by the Cholesky factorization with pivoting applied to K (Dongarra et al., 1979, p. 9.26). In addition, the Cholesky factorization, with pivoting, of K is $P_1^T K P_1 = R^T R$, assuming the diagonal entries of R are chosen to be nonnegative (Dongarra et al., 1979, p. 9.2). Now we partition the Cholesky factorization:*

$$P_1^T K P_1 = \begin{pmatrix} R_{11}^T & 0 \\ R_{12}^T & R_{22}^T \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \tag{34}$$

It follows from Theorem 7.2 of (Gu and Eisenstat, 1996, p. 865) that

$$\sigma_1(R_{22}) \leq c_3 \, \sigma_{m+1}(L) \text{ and } \frac{1}{\sigma_m(R_{11})} \leq c_4 \frac{1}{\sigma_m(L)} \text{ where } c_3, \, c_4 \leq \sqrt{n-m} \; 2^m. \tag{35}$$

Now the first m steps of Cholesky factorization, with pivoting, of K will produce identical results to the m steps of the partial Cholesky factorization described in Algorithm 1. Let

$$V = \begin{pmatrix} V_{11} \\ V_{21} \end{pmatrix} \text{ and } R_1 = \begin{pmatrix} R_{11} & R_{12} \end{pmatrix} \text{ so that } R_1^T = \begin{pmatrix} R_{11}^T \\ R_{12}^T \end{pmatrix}. \tag{36}$$

In the (complete) Cholesky factorization with pivoting of K, after the first m steps of the algorithm additional pivoting will be restricted to the last $n-m$ rows and columns

of P^TKP. Let P_2 be a $n \times n$ permutation matrix representing the pivoting in the last $n-m$ steps in the algorithm. Then it follows that

$$P_1 = PP_2,\ V_{11} = R_{11}^T \text{ and } R_1^T = P_2^T V. \tag{37}$$

Therefore

$$\widehat{K}_P = PVV^TP = P_1P_2^TVV^TP_2P_1^T = P_1R_1^TR_1P_1^T. \tag{38}$$

By (23), (34), (35), (36) and (38) we can conclude that

$$||K - \widehat{K}_P|| = ||R_{22}^TR_{22}|| = \sigma_1^2(R_{22}) \le c_3^2\ \sigma_{m+1}^2(L) = c_1\,\sigma_{m+1}(K) = c_1\,||K - \widehat{K}_{SVD}||. \tag{39}$$

Also, by (35), (38) and the interlace theorem (Björck, 1996, p. 15)

$$\sigma_m(\widehat{K}_P) = \sigma_m^2(R_1) \ge \sigma_m^2(R_{11}) \ge \sigma_m^2(L)/\,c_4^2 = \sigma_m(K)/\,c_4^2. \tag{40}$$

Next by (38) and the interlace theorem

$$\sigma_1(\widehat{K}_P) = \sigma_1^2(R_1) \le \sigma_1^2(R) = \sigma_1(K). \tag{41}$$

Finally, (24), (40) and (41) imply that

$$cond(\widehat{K}_P) = \sigma_1(\widehat{K}_P)/\sigma_m(\widehat{K}_P) \le c_4^2\,\sigma_1(K)/\sigma_m(K) = c_2\,cond(\widehat{K}_{SVD}). \tag{42}$$

■

The bounds in (33) on c_1 and c_2 grow exponentially in m and in principle can be large for larger values of m. In practice this appears to be very uncommon. For example the constants c_3 and c_4 in (35) are closely related to $||W||$ where $W = R_{11}^{-1}R_{22}$ (Gu and Eisenstat, 1996, p. 865). Numerical experiments indicate the $||W||$ is almost always small in practice (typically less than 10) (Higham, 2002, p. 207), (Higham, 1990). Therefore $c_1 = c_3^2$ and $c_2 = c_4^2$ will not be large in practice. We should add that there are choices of the pivot matrices P in (30) which guarantee bounds on c_1 and c_2 that are polynomials in n and m rather than exponential in m as in (33) (Gu and Miranian, 2004). However algorithms that produce such pivot matrices are more expensive than Algorithm 1 and, in practice, usually do not lead to an improvement in accuracy.

Prior to applying one of the methods – SR-N, SR-V and SR-Q – from Sections 3 and 5 one can carry out a partial Cholesky factorization of K to determine the permutation matrix P, and apply the algorithms of Sections 3 and 5 using the matrices $\widetilde{K} \equiv P^TKP$, $\widetilde{K}^* = K^*P$ and the vector $\widetilde{y} = P^Ty$. If pivoting is used in this manner, we will call the algorithms SR-NP, SR-VP and SR-QP corresponding, respectively, to the algorithms SR-N, SR-V and SR-Q without pivoting.

Since the algorithms SR-N, SR-V and SR-Q are all mathematically (in exact arithmetic) equivalent, then by (4) in all these algorithms the low rank approximation to $\widetilde{K}$ is $\widetilde{K}_1\widetilde{K}_{11}^{-1}\widetilde{K}_1^T$ where $\widetilde{K}_1$ is the first m columns of $\widetilde{K}$ and $\widetilde{K}_{11}$ is the first m rows of $\widetilde{K}_1$. Therefore the low rank approximation to $K = P\widetilde{K}P^T$ would be

$$\widehat{K}_P = P\widetilde{K}_1\widetilde{K}_{11}\widetilde{K}_1^TP^T. \tag{43}$$

We then have

Theorem 2 *In exact arithmetic the matrices $\widehat{K}_P$ in (30) and (43) are the same.*

Proof *Let V be the factor produced by a partial Cholesky factorization, with pivoting, of K. Then, as mentioned in Algorithm 1, the first m columns and rows of $P^TKP - VV^T$ are zero. Since $\widetilde{K} = P^TKP$ it follows that $\widetilde{K}_{11} = V_{11}V_{11}^T$ and $\widetilde{K}_1 = VV_{11}^T$, where V_{11} is the $m \times m$ leading principle submatrix of V. Therefore that $VV^T = \widetilde{K}_1\widetilde{K}_{11}^{-1}\widetilde{K}_1^T$. We conclude $PVV^TP^T = P\widetilde{K}_1\widetilde{K}_{11}^{-1}\widetilde{K}_1^TP^T$.* ■

A key conclusion of Theorems 1 and 2 is that for the algorithms SR-NP, SR-VP and SR-QP which use pivoting, the low rank approximation $\widehat{K}_P$ to K has the desirable properties (31-33) which show that the accuracy and condition number of $\widehat{K}_P$ is comparable to the accuracy and condition number of the low rank approximation produced by the singular value decomposition. Therefore if m is small, difficulties such as those illustrated in Example 3 are not possible since for small m the bound $(n-m)4^m$ for c_1 and c_2 is not large. Furthermore, such difficulties are unlikely for large m since, as mentioned earlier, for large m, the values of c_1 and c_2 are, apparently, not large in practice.

For the algorithm SR-VP one does not need to calculate V using (17) since, as shown in the proof of Theorem 2, V is calculated by the partial Cholesky factorization. Using this fact the floating point operation counts of the six algorithms that we have discussed are:

method	no pivoting	pivoting
SR-N / SR-NP	nm^2	$2nm^2$
SR-V / SR-VP	$2nm^2$	$2nm^2$
SR-Q / SR-QP	$2nm^2$	$3nm^2$

Table 3: Approximate flop counts, for n and m large and $n >> m$, for various algorithms.

We should note that flop counts are only rough measures of actual run times since other factors, such as the time for memory access or the degree to which code uses Matlab primitives, can be significant factors. This is discussed further in Section 8.

Also we should note that all the algorithms listed in Table 3 require memory for $O(mn)$ numbers.

Another advantage of the use of pivoting is that if pivoting is included in the V method then for small examples such as Example 2 the potential numerical instability illustrated in Example 2 cannot occur. We illustrate this in the next example. In Appendix A we describe the reason that the SR-VP method is guaranteed to be numerically stable for small problems and why numerical instability is very unlikely for larger real world problems.

Example 4 *This example illustrates that if one includes pivoting in the V method then the numerical instability illustrated in Example 2 does not occur in the V method. As in*

Example 2 for $0 < s \leq 1$ *let* $C = \begin{pmatrix} s^2 & 10s \\ 10s & 200 \end{pmatrix}$, *let the* 4×4 *matrix* $K = \begin{pmatrix} s^2C & 10sC \\ 10sC & 200C \end{pmatrix}$. *Now let* $x = \begin{pmatrix} 1/3 \\ 1/3 \end{pmatrix}$, $w = \begin{pmatrix} 0 \\ x_2 \\ 0 \\ x_1 \end{pmatrix}$, $\lambda = 0$ *let* $y = Kw$.

Due to the structure of w *(and since, in this example, a partial Cholesky factorization will move column 4 of* K *to the first column of* $\widetilde{K} = P^T K P$*) we again have each of* x_{NP}, x_{QP} *and* x_{VP} *will calculate* x *exactly in exact arithmetic. In finite precision arithmetic the calculated values will not be exact. For this example for small* s *the errors in both* x_{VP} *and* x_{QP} *are very small. For example if* $s = 10^{-4}$ *we get the results in Table 4.*

$\dot{x}$ =	x_{NP}	x_{VP}	x_{QP}
$\|\|x - \dot{x}\|\|/\|\|x\|\|$	1.7×10^{-1}	2.6×10^{-11}	9.7×10^{-12}

Table 4: Errors $||x - \dot{x}||/||x||$ for a 4×4 matrices and various methods.

Note that even with pivoting the error in the normal equations approach is large. With the normal equations approach the error in the calculated x *includes a term proportional to* $cond^2(K_1)$. *Even with pivoting* $cond^2(K_1)$ *can be large enough so the accuracy of the normal equations approach is poor.*

7. Rank Selection

In using low rank approximation the choice of rank will affect the accuracy of the approximation. It may be impractical to repeat the computations for a variety of different ranks and it is useful to have techniques to facilitate determination of the accuracy of a variety of low rank approximations.

We first consider the case that the true target values y^* corresponding to the testing data X^* are known. Then if $n^* < n$ the accuracy of the prediction for y^* can be calculated efficiently for all low rank approximations with rank less than a specified value m.

To illustrate this we first consider the QR implementation, (14) and (15), of the subset of regressors method. For the $(n + m) \times m$ matrix A in (10) let $A = QR$ where Q is an $(n + m) \times m$ matrix with orthonormal columns and R is an $m \times m$ upper triangular matrix and let $x = R^{-1}Q^T b$, as in (14) (where we omit the subscript Q on x to simplify our notation). Then by (15) the predicted values of y^* are

$$\widehat{y}^* = K_1^* x \tag{44}$$

where K_1^* is the $n^* \times m$ matrix defined in (3).

Now for some i, $1 \leq i \leq m$, consider the construction of a prediction for y^* using a rank i low rank approximation. Let $\widetilde{A}$ consist of the first i columns of A. It then follows from (11), (15) and the fact that the last $m - i$ rows of b and $\widetilde{A}$ are zero that the rank i prediction, which we call $\widetilde{y}^*$, for y^* is given by solving

$$\min_{\widetilde{x}} ||\widetilde{A}\,\widetilde{x} - b|| \text{ and letting} \tag{45}$$

$$\widetilde{y}^* = K_1^* \begin{pmatrix} \widetilde{x} \\ 0 \end{pmatrix} \tag{46}$$

where $\widetilde{x} \in R^i$ and the 0 in (46) indicates a vector of $m - i$ zeros. Since $A = QR$ it follows that $\widetilde{A} = Q \begin{pmatrix} \widetilde{R} \\ 0 \end{pmatrix}$ where the 0 here indicates $m - i$ rows of zeros. Therefore if $c =$ (the first i elements of $Q^T b$) it then follows (Golub and Van Loan, 1996, p. 239) that we can construct $\widetilde{x}$ using

$$\widetilde{x} = \widetilde{R}^{-1} c. \tag{47}$$

We can use (47) to construct predictions for y^* for every low rank approximation of rank less than or equal to m. To do this we let C be a $m \times m$ upper triangular matrix whose i^{th} column consists of the first i elements of $Q^T b$ and is zero otherwise. Let $\widetilde{Y}$ be the $n^* \times m$ matrix whose i^{th} columns consists of the prediction for y^* using a rank i approximation. Then, for the reasons described in the last paragraph,

$$\widetilde{Y} = K_1^* R^{-1} C. \tag{48}$$

If y^* is known (48) can be used to calculate, for example, the root mean square error of the prediction for y^* for all low rank approximations of rank less than or equal to m.

After the rank m low rank prediction for y^* is constructed, the above calculations require $O(m^3 + n^* m^2)$ floating point operations. If n^* is less than n, this is less than the $O(nm^2)$ operations required to construct the initial rank m prediction. Although we will not present the details here similar efficiencies are possible when using the normal equations approach or the V method.

If the true value y^* for the test set are not known, one can use the subset of regressors approach to estimate the known y values in the training set (by replacing K_1^* with K_1 in (13), (15) or (19)). Again one can calculate the accuracies in estimating y for every low rank approximation of rank less than a given rank m and this can be done relatively efficiently after the initial rank m low rank approximation is constructed. These accuracies will give some indication of the relative difference in using low rank approximations of different ranks.

Finally, we should note that our algorithms provide a limit on the largest rank that can be used. For example in SR-NP, SR-VP and SR-QP Algorithm 1 is used to determine the subset selection. Algorithm 1 returns a rank m where the factorization is stopped and m can be used as the maximum possible rank. For the SR-V and SR-Q algorithms a Cholesky factorization of K_{11} is required in (9). If Matlab's Cholesky routine chol is used for this factorization there is an option to stop the factorization when it is determined that K_{11} is not positive definite (numerically). The size of the factor that successfully factors a positive definite portion of K_{11} sets a limit on the rank that can be effectively used. Finally, SR-N and SR-NP require solving a system of equations (6) involving the symmetric semidefinite system $\lambda^2 K_{11} + K_1^T K_1$. A good way to solve this system is to use Matlab's chol, which again has an option that can be used to determine a limit on the rank that can be effectively used. As discussed in the next section if these rank limits are exceeded then the calculated answers are often dominated by computer arithmetic errors and are not accurate.

8. Practical Example

In the Sloan Digital Sky Survey (York et al., 2000) broadband u, g, r, i, z photometric measurements will be made for 100s of million galaxies but only approximately 1 million galaxies will have careful spectroscopic measurements of redshifts. Therefore the estimation of redshift from broadband photometric measurements is important since it can lead to much better constraints on the formation and evolution of large-scale structured element in cosmological models (Way and Srivastava, 2006).

We illustrate our earlier remarks by using a training set of 180045 galaxies, each with five measured u, g, r, i, z broadband mesurements. The training set consists of a 180045×5 matrix X of broadband measurements and the 180045×1 vector y with the corresponding redshifts. The testing set will consist of a 20229×5 matrix X^* of broadband measurements and the 20229×1 vector y^* of redshifts. This data is from the SDSS GOOD data set discussed in (Way and Srivastava, 2006).

To determine a good choice for a covariance function we calculated the root mean square (RMS) error for the prediction $\widehat{y}^*$ for y^* using the Matern (with parameter $\nu = 3/2$ and with parameter $\nu = 5/2$), squared exponential, rational quadratic, quadratic and neural network covariance functions from (Rasmussen and Williams, 2006, Chap. 4). As mentioned earlier we selected the hyperparameters for each covariance function using the Matlab routine minimize from (Rasmussen and Williams, 2006, pp. 112-116, 221). The covariance function which produced the smallest RMS error for the prediction of y^* was the neural network covariance function (Rasmussen and Williams, 2006, p. 91). For example, for low rank approximations of rank 500 with bootstrap resampling runs (described below) of size 100 the neural network median RMS error was .0204. The next smallest median RMS error was .0212 for the Matern covariance function with $\nu = 3/2$ and the largest median RMS error was .0248 for the quadratic covariance function. Therefore in the experiments below we will use the neural network covariance function.

To compare, experimentally, the efficiency of our implementations of the subset of regressors method we choose a training set size of 90023 (consistent with the bootstrap resampling runs describe below) and low rank approximations of rank $m = 150$ and $m = 1500$. On a computer with 2.2 GH Core Duo Intel processor we timed the SR-N, SR-V, SR-Q, SR-NP, SR-VP and SR-QP methods. On all the calculations in this section that use Algorithm 1 we set the stopping tolerance *tol* to 0. We ran each of the methods with the additional calculations required to determine the "history" of the accuracy of all low rank approximations less than the specified rank (either 150 or 1500) and also without these extra calculations. The results are summarized in Figure 1.

As can be seen in Figure 1, without pivoting the normal equations approach is the fastest, the QR factorization the slowest and the V method in between. With pivoting all the methods take similar amounts of time (the V method is slightly faster). The reason that all the methods require about the same time when using pivoting is that the code for SR-N, SR-V and SR-Q is written so that the key calculations are done almost entirely with Matlab primitives whereas our implementation of the partial Cholesky factorization contains loops written in Matlab code. The Matlab primitives make use of BLAS-3 (Anderson et al., 1999) routines and will make effective use of cache memory. Therefore, even though the big-O operation counts are similar, the partial Cholesky factorization takes longer to run than

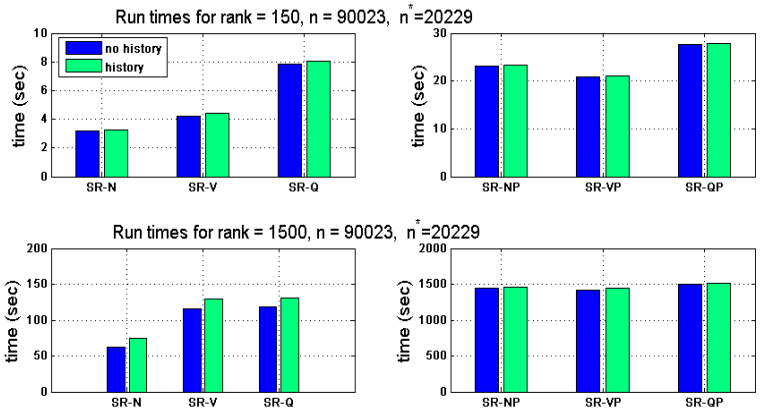

Figure 1: Comparison of run times for implementations of the subset of regressors method.

SR-N, SR-V or SR-Q and the partial Cholesky factorization dominates the run times in the SR-NP, SR-VP and SR-QP code. We should add that the times for the partial Cholesky factorization would be reduced if a partial Cholesky factorization with pivoting could be implemented using BLAS-3 operations. We are not aware of such an implementation. Finally, we should note that the calculations required to determine the accuracy of all low-order approximations adds only a modest amount to the run times.

To determine the accuracy of the algorithms for different choices of the training set we carried out bootstrap resampling (Efron and Tibshirani, 1993). For each of 100 samples we randomly selected half or 90023 of the 180045 galaxies in the original training set and used this smaller training set to predict the redshift for the 20229 galaxies in the testing set. We considered such resampling with replacement as well as without replacement. For SR-N, SR-V and SR-Q we selected the indices in the active set randomly. Following this we selected the hyperparameters using the minimize routine in (Rasmussen and Williams, 2006, pp. 112-116, 221). For SR-NP, SR-VP and SR-QP the active set was determined by the partial Cholesky factorization with pivoting. To illustrate the variation in the calculated accuracies, after carrying out a bootstrap resampling run we sorted the 100 RMS errors in increasing order and plotted these errors versus the sample number. The results for low rank approximations of rank 1500, using resampling without replacement, are pictured in Figure 2.

Note that mathematically (in exact arithmetic) SR-N, SR-V and SR-Q will produce identical results; as will SR-NP, SR-VP and SR-QP. Therefore the differences illustrated in Figure 2 between SR-N and SR-V or SR-Q and the differences between SR-NP and SR-VP or SR-QP are due to computer arithmetic and, in particular, the numerical instabilities in using a normal equations approach to solve the least squares problem (11). Also note that although pivoting reduces the numerical instability in using the normal equations approach, still in SR-NP the instability is evident for approximately half of the bootstrap resampling runs. Also we should remark that the $\widehat{y}^*$ predictions calculated using SR-V and SR-Q are essentially identical – they agree to at least seven significant digits in this example – as

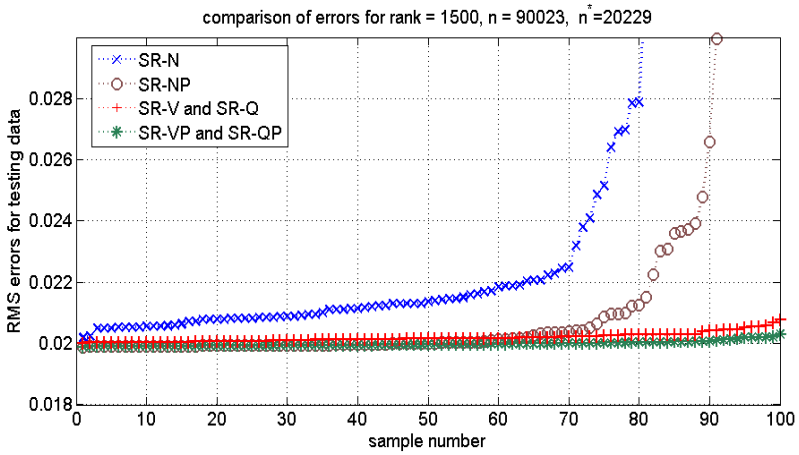

Figure 2: Bootstrap resampling: Comparison of RMS errors for implementations of the subset of regressors method.

are the $\widehat{y}^*$ predictions calculated using SR-VP and SR-QP. Finally we should note that for this example the methods that avoid normal equation and use pivoting – SR-VP and SR-QP – are a small amount better than their counterparts, SR-V and SR-Q, that do not use pivoting.

As mentioned earlier, the parameter λ in the Gaussian process computations was selected while optimizing the hyperparameters using the routine minimize from (Rasmussen and Williams, 2006, pp. 112-116, 221). The values of λ varied over a small range, $.0176 \leq \lambda \leq .0214$, for the 100 samples illustrated in Figure 2. For our stable algorithms these values of λ were good values as can be seen by the accuracy of the results of SR-V, SR-VP, SR-Q and SR-QP pictured in Figures 2, 3 and 4. For SR-N and SR-NP we experimented with a variety of choices of λ but did not reliably achieve accurate predictions for any of our choices.

We might also add that we tried other types of resampling. We obtained results similar to those illustrated in Figure 2 when using bootstrap resampling with replacement and also when we choose a number of galaxies in the sample size other than 90023.

We can also illustrate the ability to efficiently calculate the accuracy of low rank approximations lower than a specified rank. For the same runs picture in Figure 2 we calculated the mean RMS error of the 100 samples for each rank less than 1500 for each of the six implementations of the subset of regressors method. This is pictured in Figure 3.

As one increases the rank of the low rank approximation the condition number of the matrix A in (11) will tend to increase. This will increase the computer arithmetic errors in the calculated results. The ranks where significant computer arithmetic errors arise are illustrated in Figure 3 by the jumps in the mean errors calculated for the SR-N and SR-NP methods. The ranks where this occurs and the magnitude of the jumps is dependent on the particular data chosen for a bootstrap resampling run and will vary for different bootstrap resampling runs. For the SR-N method the ranks where numerical difficulties were first substantial varied between a rank of 46 to a rank of 839. For the SR-NP method the ranks

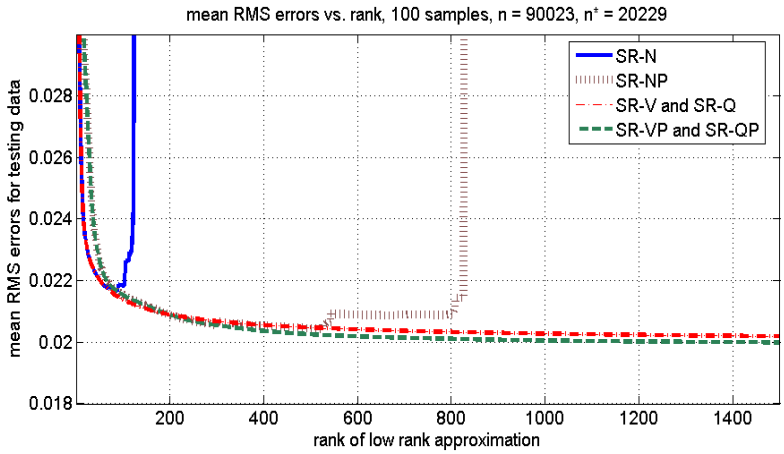

Figure 3: Mean RMS errors versus rank for implementations of the subset of regressors method.

where numerical difficulties were first substantial varied between ranks of 325 and 1479. For the SR-V, SR-Q, SR-VP and SR-QP methods we did not encounter significant numerical difficulties with these runs and the graphs for these methods smoothly decrease.

The SR-VP and SR-QP method, which use pivoting, are somewhat more accurate than the corresponding methods without pivoting after a rank of approximately 200 but prior to this SR-V and SR-Q are more accurate. Our motivation for subset selection using the Cholesky factorization with pivoting is based on controlling the condition number and improving numerical stability. For smaller ranks it appears that this choice of the active set is good but not optimal. Finally, we should note that Figure 3 indicates for the stable methods the mean RMS errors decrease rapidly for smaller ranks but are only slowly decreasing for larger ranks.

As we mentioned earlier all of our algorithms may limit the rank so that the effective rank can be less than the desired rank. This did not occur on the above runs for SR-V, SR-Q, SR-VP or SR-QP but did occur for SR-N and SR-NP due to our use of the Cholesky factorization to solve the linear system (6). It is possible to solve the linear system in (6) using Gaussian elimination, rather than using a Cholesky factorization, for ranks up to 1500. However the Cholesky factorization in (6) will fail only if the matrix $\lambda^2 K_{11} + K_1^T K_1$ is very ill conditioned. In this case solving the system of equations in (6) by any method will be prone to large computer arithmetic errors. Indeed, for these runs, if we used Gaussian elimination to solve (6) for large ranks the errors became larger than when we limited the rank as we have described earlier. Also when the Cholesky factorization failed in the solution to (6) we tried perturbing K_{11} a small amount following a suggestion in the code provided with (Rasmussen and Williams, 2006). For our runs this did not improve the calculated results in a significant manner.

In (Way and Srivastava, 2006) there is a comparison of a variety of methods for predicting redshift with data from the SLOAN digital sky survey. The methods compared in (Way and Srivastava, 2006) include linear regression, quadratic regression, artificial neural

networks (label ANNz in Figure 4), E-model and Gaussian processes using a quadratic covariance function (labeled GP in Figure 4). In Figure 4 we have compared these methods with our predictions using the SR-VP and SR-QP implementations of the subset of regressors Gaussian processes method with a neural network covariance function. Other than the SR-VP and SR-QP predictions the results in Figure 4 are from (Way and Srivastava, 2006). As seen in Figure 4, in this example either SR-VP or SR-QP provides overall the best predictions. The E-model approach is also quite good.

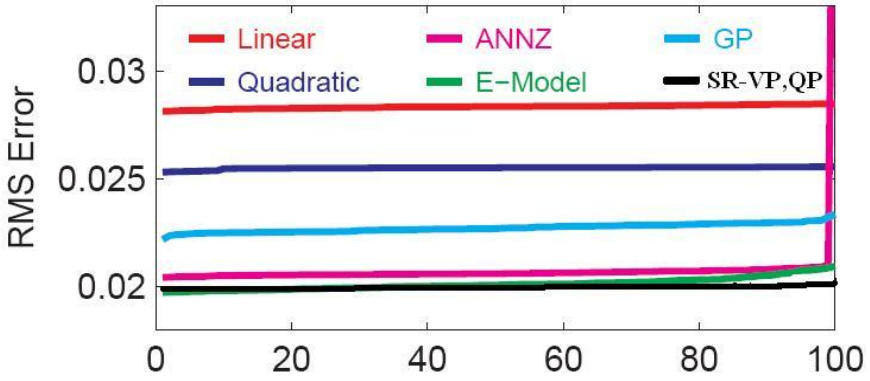

Figure 4: Bootstrap resampling: comparison of RMS errors for six methods of predicting redshift.

We should add that in addition to the data set which was used to generate the results in Figures 1 to 4 we have also carried out experiments using other data sets described in (Way and Srivastava, 2006) (for example redshift prediction using photometry properties in addition to broadband measurements) and using the SARCOS robot arm inverse dynamics (Rasmussen and Williams, 2006; Vijayakumar et al., 2002). For the other redshift data sets significant computer arithmetic errors in the predictions were common for the SR-N and SR-NP algorithms. For some data sets, for example the SARCOS robot arm, computer arithmetic errors were not significant and all the algorithms worked well. Also we might note that although prediction using Gaussian processes was more accurate than alternatives approaches in some cases, in other cases the E-model or artificial neural network approaches provided better accuracy.

Finally, we should note that Matlab code which implements the SR-N, SR-NP, SR-V,SR-VP, SR-Q and SR-QP methods and can produce graphs such as those in Figures 2 and 3 is available at dashlink.arc.nasa.gov/algorithm/stableGP. Our code makes use of the code from (Rasmussen and Williams, 2006, p. 221) and the syntax is modelled on that code. We should also note that (Foster et al., 2008) and (Cayco et al., 2006) discuss additional results related to redshift prediction.

9. Conclusions

An important conclusion of our results is that with the subset of regressors approach to Gaussian process calculations use of normal equations can be unstable and should, in some important practical examples, be avoided. We expect that this principle is also applicable to other approaches to Gaussian process calculations. For example when using approximations based on sparse Gaussian processes with psuedo-inputs (Snelson and Ghahramani, 2006) which is called the FITC approximation in the framework of (Quinonero-Candela and Rasmussen, 2005) the predicted values are calculated using

$$\widehat{y}^*_{FITC} = K_1^*(\lambda^2 K_{11} + K_1^T(\Lambda + I)^{-1}K_1)^{-1}K_1^T(\Lambda + I)^{-1}y. \tag{49}$$

where

$$\Lambda = diag(K - K_1 K_{11}^{-1} K_1^T)/\lambda^2. \tag{50}$$

Our results suggest that it may be more accurate to carry out these calculations using a QR factorization of $\begin{pmatrix} DK_1 \\ \lambda V_{11}^T \end{pmatrix}$ where $D = (\Lambda + I)^{-1/2}$ rather than, for example, using a Cholesky factorization of $\lambda^2 K_{11} + K_1^T(\Lambda + I)^{-1}K_1$.

To summarize our results, we have presented different implementations of the subset of regressors method for solving, approximately, the Gaussian process equations for prediction. An implementation of the subset of regressors method which uses the normal equations is the fastest approach but also can have poor numerical stability and unacceptable large growth of computer arithmetic errors. An implementation using orthogonal factorization is somewhat slower but in principle has better numerical stability properties. A third approach, which we call the V method, is intermediate between these other two approaches in terms of accuracy and stability. We can use the partial Cholesky factorization to select the active set prior to implementation of any of the above methods. This also will tend to reduce the growth of computer arithmetic errors and can, in some cases, improve the accuracy of the predictions. All of these implementations require $0(nm^2)$ operations where n is the number of data points in the training set and m is the size of the active set or the rank of the low rank approximation used. In this sense all these implementations are efficient and can be much faster than implementation of the full Gaussian process equations. Finally, we have illustrated these result with an important practical application – redshift prediction from broadband spectral measurements. Code implementing our algorithms is available at dashlink.arc.nasa.gov/algorithm/stableGP.

Appendix A. Numerical Stability of SR-VP

Here we explain why, even though there is a potential numerical instability in SR-V, as illustrated in Example 2, this difficulty cannot occur with the SR-VP method for small problems and is very unlikely to occur for larger problems from real world applications.

Let P be the $n \times n$ permutation matrix determined by the partial Cholesky factorization with pivoting applied to K, let $\widetilde{K} = P^T K P$ and let $\widetilde{K}_1$ be the first m columns of $\widetilde{K}$. In the SR-VP method we apply equations (17)-(19) to $\widetilde{K}$ and $\widetilde{K}_1$ rather than K and K_1.

We will begin by considering the special case where $\lambda = 0$ and later consider the more general case. In the case that $\lambda = 0$ the least square problem (11), with K_1 replaced by $\widetilde{K}_1$

since we are incorporating pivoting, is equivalent to

$$\min_x ||\widetilde{K}_1 x - y||. \tag{51}$$

and, by (18), we have

$$x = V_{11}^{-T}(V^T V)^{-1} V^T y. \tag{52}$$

where $\widetilde{K}_1 = V V_{11}^T$. There is a potential concern in using (52) since to construct x the linear system of equations

$$(V^T V)z = V^T y \tag{53}$$

must be solved. Forming $V^T V$ squares the condition number of V which, potentially could lead to the introduction of undesirable computer arithmetic errors. However we will argue that the matrix $B = V^T V$ is diagonally equivalent to a matrix that is guaranteed to be well conditioned for small problems and, in practice, is almost always well conditioned for larger problems. This will limit the growth of computer arithmetic errors. We should add that without pivoting one cannot prove such results, as is illustrated by Example 2.

Now V is formed by a partial Cholesky factorization with pivoting of the symmetric positive semidefinite matrix K. Since pivoting is included in the partial Cholesky factorization of the SPS matrix it follows, for each $i = 1, \ldots, m$, that the i^{th} diagonal entry of $\widetilde{K}_1$ is at least as large in magnitude as any off diagonal entry in row i or column i of $\widetilde{K}_1$ (Trefethen and Bau III, 1997, p. 176) and that the lower trapezoidal matrix V has the property that, for each $i = 1, \ldots, m$, the i^{th} diagonal entry in V is at least as large in magnitude as any entry in column i (Higham, 2002, p. 202). Therefore we can write V as $V = LD$ where D is an $m \times m$ diagonal matrix and L is an $n \times m$ lower trapezoidal matrix with all entries one or less in magnitude and with ones on the diagonal. Indeed this matrix L is identical to the lower trapezoidal matrix produced if Gaussian elimination with complete pivoting is applied to $\widetilde{K}_1$ (Higham, 2002, p. 202). Also since the pivoting has already been applied in forming $\widetilde{K}_1$ Gaussian elimination with complete pivoting will not pivot any entries in $\widetilde{K}_1$ and this implies that Gaussian with partial pivoting will not pivot any entries in $\widetilde{K}_1$ and will produce the same lower trapezoidal factor L. Now it follows from (Higham, 2002, p. 148) that

$$cond(L) \leq \sqrt{nm}\ 2^{m-1} \tag{54}$$

and therefore for n and m small, as in Example 2, L is well conditioned. More generally, according to (Björck, 1996, p. 73), if partial pivoting is used in the factorization of $\widetilde{K}_1$ then L is usually well conditioned and, indeed, the discussion in (Trefethen and Bau III, 1997, p. 169) indicates that, for matrices from applications and for random matrices $\widetilde{K}_1$, the matrix L is almost always well conditioned, in the sense, for example, that cond(L) is far from being exponentially large.

Thus V is a diagonal rescaling of a matrix L that is well conditioned in practice. Now define $U = DV_{11}^T$. It then follows from (52) that

$$x = U^{-1}(L^T L)^{-1} L^T y. \tag{55}$$

Equation (55) is precisely the Peters-Wilkinson method (Peters and Wilkinson, 1970), (Björck, 1996, p. 73) to the least square problem (52). Since L is usually well conditioned

then the calculation of $(L^TL)^{-1}L^Ty$ can be computed without substantial loss of accuracy and the calculation of x using (55) is more stable than using the normal equation solution to (52) (Björck, 1996, p. 73).

The SR-VP method uses (52) rather that (55). However, since V is a diagonal rescaling of L and U is a diagonal rescaling of V_{11}^T the SR-VP method will also have good numerical stability properties in practice. To demonstrate this we can write $V = LD_1D_2$ where the entries of the diagonal matrix D_1 are between 1 and 2 and where entries in D_2 are exact powers of 2. Since L will be well conditioned in practice then so is $W = LD_1$ (since $\text{cond}(LD_1) \leq \text{cond}(L)\text{cond}(D_1) \leq 2\ \text{cond}(L)$). Now, by (52), we have

$$x = (D_2V_{11})^{-T}(W^TW)^{-1}W^Ty. \tag{56}$$

Since W is well conditioned in practice it follows, for the same reasons that (55) has good numerical stability, that (56) will have good numerical stability properties.

To finish the analysis of numerical stability of the SR-VP method in the case that $\lambda = 0$ note that since D_2 has entries that are exact powers of 2, it follows by the discussion in (Higham, 2002, p. 200) and (Forsythe and Moler, 1967, 37-39), for any computer using base 2 computer arithmetic, that the x calculated by (56) will be precisely the same, even in floating point arithmetic (as long as there is no overflow or underflow), as the x calculated by (52). Therefore we may conclude that in practice x calculated when using the SR-VP method will have good numerical stability properties and the SR-VP method will usually have smaller computer arithmetic errors than will the SR-N or SR-NP methods.

To consider the case that $\lambda \neq 0$ we note that in this case the condition number of $B = (\lambda^2 I + V^TV)$ will be important in solving

$$(\lambda^2 I + V^TV)z = V^Ty.$$

However we have

Theorem 3 *For any $\lambda \geq 0$, $cond(\lambda^2 I + V^TV) \leq cond(V^TV)$.*

Proof If V^TV has eigenvalues $\alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_m \geq 0$ then the eigenvalues of $(\lambda^2 I + V^TV)$ are $(\lambda^2 + \alpha_i)$, $i = 1, \ldots, m$. Therefore $cond(V^TV) = \alpha_1/\alpha_m$ and $cond(\lambda^2 I + V^TV) = (\alpha_1 + \lambda^2)/(\alpha_m + \lambda^2)$. However it follows easily that $\alpha_1/\alpha_m \geq (\alpha_1 + \lambda^2)/(\alpha_m + \lambda^2)$. ■

Since $cond(\lambda^2 I + V^TV) \leq cond(V^TV)$ we expect that solving $(\lambda^2 I + V^TV)z = V^Ty$ with $\lambda \neq 0$ will be more accurate than solving this equation with $\lambda = 0$. Since we have argued that the error growth in solving this equation for $\lambda = 0$ should be limited we expect that this should also be true when $\lambda \neq 0$.

Acknowledgments

We would like to acknowledge support for this project from the Woodward Fund, Department of Mathematics, San Jose State University.

M.J.W. acknowledges funding received from the NASA Applied Information Systems Research Program and from the NASA Ames Research Center Director's Discretionary Fund. M.J.W also acknowledges Alex Szalay, Ani Thakar, Maria SanSebastien and especially Jim Gray for their help with the Sloan Digital Sky Survey.

A. N. Srivastava wishes to thank the NASA Aviation Safety Program, Integrated Vehicle Health Management Project for supporting this work.

Funding for the SDSS has been provided by the Alfred P. Sloan Foundation, the Participating Institutions, the National Aeronautics and Space Administration, the National Science Foundation, the U.S. Department of Energy, the Japanese Monbukagakusho, and the Max Planck Society. The SDSS Web site is http://www.sdss.org/.

The SDSS is managed by the Astrophysical Research Consortium for the Participating Institutions. The Participating Institutions are The University of Chicago, Fermilab, the Institute for Advanced Study, the Japan Participation Group, The Johns Hopkins University, Los Alamos National Laboratory, the Max-Planck-Institute for Astronomy, the Max-Planck-Institute for Astrophysics, New Mexico State University, University of Pittsburgh, Princeton University, the United States Naval Observatory, and the University of Washington.

Finally, we wish to thank the anonymous referees for carefully reading the manuscript and offering excellent suggestions.

References

E. Anderson, Z. Bai, C. H. Bischof, S. Blackford, J. W. Demmel, J. J. Dongarra, J. J. Du Croz, A. Greenbaum, S. J. Hammarling, A. McKenney, and D. C. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, third edition, 1999. ISBN 0-89871-447-8.

Åke Björck. *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996. ISBN 0-89871-360-9.

Bem Cayco, Wasin So, Miranda Braselton, Kelley Cartwright, Michael Hurley, Maheen Khan, Miguel Rodriguez, David Shao, Jason Smith, Jimmy Ying, and Genti Zaimi. Camcos project – Fall 2006: Improved linear algebra methods for redshift computation from limited spectrum data. At www.math.sjsu.edu/~foster/camcos07/redshift.html, 2006.

Lehel Csato and Manfred Opper. Sparse on-line gaussian processes. *Neural Computation*, 14:641–668, 2002.

J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart. *LINPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1979. ISBN 0-89871-172-X.

B. Efron and R. J. Tibshirani. *An Introduction to the Bootsrap*. Chapman and Hall, New York, 1993.

Shai Fine and Katya Scheinberg. Efficient svm training using low-rank kernel representations. *J. of Machine Learning Research*, 2:243–264, 2001.

George E. Forsythe and Cleve B. Moler. *Computer Solution of Linear Algebraic Systems*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1967.

Leslie Foster, Alex Waagen, Nabeela Aijaz, Michael Hurley, Apolo Luis, Joel Rinsky, Chandrika Satyavolu, Ashok Srivastava, Paul Gazis, and Michael Way. Improved linear algebra methods for redshift computation from limited spectrum data - II. NASA Technical Report NASA/TM-2008-214571, NASA Ames Research Center, Moffett Field, CA, 2008. Available at ntrs.nasa.gov and at www.math.sjsu.edu/~foster/camcos07/redshift.html.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, USA, third edition, 1996. ISBN 0-8018-5413-X, 0-8018-5414-8.

Ming Gu and Stanley C. Eisenstat. Efficient algorithm for computing a strong rank-revealing qr factorization. *SIAM J. Sci. Comput.*, 17(4):848–869, 1996.

Ming Gu and L. Miranian. Strong rank-revealing Cholesky factorization. *Electronic Transactions on Numerical Analysis*, 17:76–92, 2004.

Per Christian Hansen. *Rank-Deficient and Discrete Ill-Posed Problems*. SIAM, Philadelphia, PA, USA, 1998.

Nicholas J. Higham. Analysis of the Cholesky decomposition of a semi-definite matrix. In M. G. Cox and S. J. Hammarling, editors, *Reliable Numerical Computation*, pages 161–185. Oxford University Press, 1990.

Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edition, 2002. ISBN 0-89871-521-0.

Craig Lucas. LAPACK-style codes for level 2 and 3 pivoted cholesky factorizations. Numerical Analysis Report No. 442, Manchester Centre for Computational Mathematics, Manchester, England, 2004. LAPACK Working Note 161.

Douglas C. Montgomery, Elizabeth A. Peck, and G. Geoffrey Vining. *Introduction to Linear Regression Analysis*. John Wiley and Sons, Hoboken, NJ, USA, fourth edition, 2006.

G. Peters and J. H. Wilkinson. The least squares problem and pseudo-inverses. *Comput. J.*, 13(3):309–316, 1970.

Tomaso Poggio and Fedirico Girosi. Networks for approximation and learning. *Proceedings of IEEE*, 78:1481–1497, 1990.

Joaquin Quinonero-Candela and Carl E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *J. of Machine Learning Research*, 6:1939–1959, 2005.

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

Matthias Seeger, Christopher Williams, and Neil D. Lawrence. Fast forward selection to speed up sparse gaussian process regression. In C. M. Bishop and B. J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, San Francisco, 2003. Morgan Kaufmann.

Alex J. Smola and Peter Bartlett. Sparse greedy gaussian process regression. In T. Leen, T. Diettrich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press, 2001.

Edward Snelson and Zoubin Ghahramani. Sparse gaussian process using pseudo-inputs. In Y. Weiss, B. Scholkpf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1257–1264. MIT Press, 2006.

G. W. Stewart. The efficient generation of random orthogonal matrices with an application to condition estimators. *SIAM J. Numer. Anal.*, 17(3):403–409, 1980.

Lloyd N. Trefethen and David Bau III. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997. ISBN 0-89871-361-7.

S. Vijayakumar, A. DSouza, T. Shibata, J. Conradt, and S. Schaal. Statistical learning for humanoid robots. *Autonomous Robot*, 12:55–69, 2002.

Grace Wahba. *Spline Models for Observation Data*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1990.

Michael J. Way and Ashok Srivastava. Novel methods for predicting photometric redshifts from broadband data using virtual sensors. *The Astrophysical Journal*, 647:102–115, 2006.

Donald G. York, J. Adelman, and John E. Anderson et. al. The sloan digital sky survey: Technical summary. *The Astronomical Journal*, 120:1579–1587, 2000.